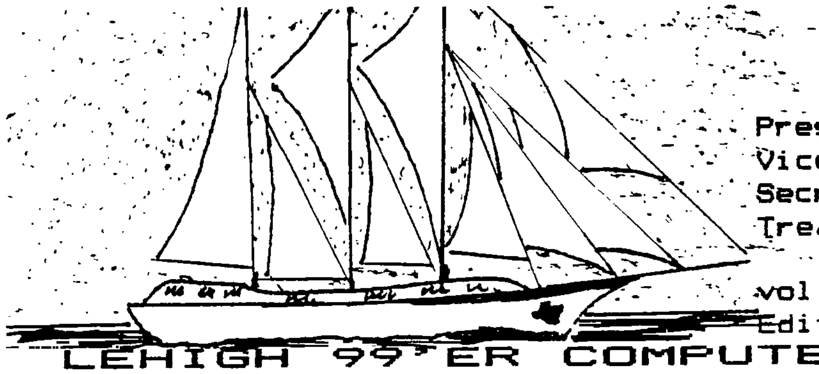
8503 057



President Rowland Buckwalter 264-5790
Vice-pres Joe Birchak 759-4052
Secretary Ann Halko 262-8206
Treasurer Mark DeNardo 791-1015

vol. III, no. 3 March 1985
Editor Frederick Hawkins 432-5913

LEHIGH 99'ER COMPUTER GROUP

Next meeting: Monday, Apr 15th 7:30 PM as usual, we hope.
Community Room, First Nat'l Bank 7th and Hamilton, Allentown

an adventure update, literally

I'm in a crowded community room in Allentown. Visible items are: a slightly disgruntled president, a frazzled newsletter editor, and a sign reading:
"ANY I/O PORT IN A STORM."

I'm carrying a blank disk, a pencil and last month's newsletter.
<--->
WHAT SHALL I DO?

Sound farfetched?? Not if you have a copy of "The Adventure Editor" currently being sold by TEX-COMP, P.O. Box 33084, Granada Hills, CA 91344.

Following the lead of Dave Hendricks in the December newsletter, I wrote to Markus Weiand in West Germany, about his "write-your-own-adventure" program. Here are the highlights of the letter I received:

The Adventure Editor is in fact a program package that permits you to analyze, change and write both old and new games for the Adventure Command Module.

The Adventure Editor requires either the MiniMemory or the Editor/Assembler cartridges. The ED/ASM version needs the 32K card and a disk drive. The MiniMem can get along with either a cassette or disk. The Adventure module is not necessary, but it's nearly impossible to write an error-free adventure game without the opportunity to check it. A printer is also supported, but not a must.

The program package consists of three files:

1. The adventure editor program for editing or printing games.

2. An adventure conversion program for converting existing programs into a form which can be used by the adventure editor (and which will still run with the Adventure module.)

3. An adventure template, a kind of "empty" adventure program to start new adventure games.

An adventure program consists of both lists and program lines in a special language. There are three kinds of lists:

1. Lists of pointers for the Adventure module, which the adventure editor will generate automatically.

2. Lists of text (messages, objects, locations, verbs and nouns), which the programmer must enter.

3. Lists correlating the lists of text (i.e. where are the objects in the beginning, how are the locations connected with each other, etc.) which the programmer must enter also.

You don't need a knowledge of assembly langauge or GPL to write an adventure with the Adventure Editor. The special language you need is quite simple: each line consists of some conditions and some actions that must be taken if the corresponding conditions are fulfilled. The only difficulty you will have writing an adventure is that you are never sure that you have thought of and planned for all the odd situations that a player can and most often will land in.

Mr. Weiand went on to say that he recently sold the U.S. Copyright for his program to TEX-COMP. They are offering the Adventure Editor for $29.95.

>>> Terry Staph

cruising at the io port

Some forthright definitions:
Sieve -- a net or device to separate the kernels.
Seive -- an editor's nemisis.
Elections -- what L99er's are up to this month.
Adventure -- what cruises are for.
DSDD -- a topical amulet to make big fishes from small.
P-system -- one whale of a series. It'll take a couple trips to haul it in.
Otherwise, one albatross on one turkey. SOUNDs fishy?
Evermore, quoth the Raven.

perils of PASCAL: episode one

PASCAL — an introduction

by Ronald J. Hartranft

There are three ways in which the name of Blaise Pascal (1623-1662), a French mathematician and philosopher, is used in computing. The most widely known use is as the name of a computer language developed by Niklaus Wirth. PASCAL is a structured language which is very similar to more recently developed languages such as MODULA, Ada, and C. The language will be discussed in later articles of this series. The name is also used somewhat loosely to refer to UCSD-PASCAL, which is a version of the language suitable for microcomputers. But UCSD-PASCAL is also a complete operating system capable of supporting other languages. This article contains an introduction to the UCSD-PASCAL operating system. The third use takes only the first letter of Pascal to name a pseudo machine language called p-code. Some sense of the three sides of PASCAL will help clarify the process of program development.

To develop a new program in the UCSD p-system, the editor is used to prepare a text file consisting of the PASCAL program commands. (See the end of this article for an elementary program.) The PASCAL compiler is then invoked to read the program text file from disk and create a p-code version of the program on disk. The linker is used to join the p-code program to any required system routines. The resulting complete executable program is then loaded into the computer and can be executed.

Most compilers transform commands written in languages such as PASCAL into machine language commands. There are several PASCAL compilers of this type for various microcomputers. The resulting machine language can be executed only on the identical kind of computer. The text of the program can be carried to another type of machine and compiled into its language (provided that disk formats are compatible). The UCSD p-System compiler does not compile directly to machine language. Instead it compiles a p-code (pseudo machine code) version. In operation, a p-code program is somewhat like TI BASIC in that it is interpreted. Each line of p-code is read by the p-code interpreter in the p-code card in your expansion box. One by one, each line is read, interpreted, and executed. The interpretation replaces p-code program instructions by the appropriate sequence of machine language instructions. But the interpretation occurs for each line of p-code each time it is encountered. A good deal of execution time can be taken by the interpreter, particularly when several branches to a subroutine require reinterpretation every time.

The p-code is intended to be a kind of universal assembly language. Somewhat faster than interpreted BASIC, it is slower than a machine language program. If a PASCAL program has already been compiled into p-code, it can be run on any other computer which has a p-code interpreter. In fact, page 18 of the p-code manual describes how a p-code program can be read from a cassette tape and executed with only the p-code card--no disk drives necessary. The portability of p-code programs between machines, however, depends on the type of program. Programs making use of the TI 99/4A sound and graphics (compiler manual, pages 116-155) would not be compatible with most other computers. But most programs manipulating text and numbers should be portable. But we like our TI sound and graphics, don't we? TI gave us routines like SET_PATTERN and SET_SPRITE, which are like the BASIC commands, CALL CHAR and CALL SPRITE. But in order to give us these familiar subroutines, TI had to leave out the turtle graphics subroutines which are part of the standard UCSD-PASCAL language.

the UCSD p-System

The manual for the p-code card contains brief descriptions of the commands available immediately after system initialization: Edit, Run, File, Compile, Link, Xecute, Assemble, Halt, Initialize, User restart, and Monitor. Any of these commands can be given by just pressing the first letter of the command. Pressing the "?" will allow you to step through all choices in case you don't remember a particular command. But you can press "F" for file even when it is not listed on the system promptline. For the remainder of this article, I want to concentrate on the file-related commands. These are similar to those available in the disk manager module, but there is a great deal of flexibility in the UCSD p-System.

Disks to be used with the UCSD p-System may be initialized with the disk manager module or by using the utility program, DFORMAT. The procedure is given at the end of this article. Note that disks used in the UCSD p-System appear to the disk manager module to contain a single file called PASCAL which completely fills the disk. Whereas the disk manager module measures file size in sectors (256 bytes per sector, 9 sectors per track, 40 sectors per side), the UCSD p-System uses blocks (512 bytes per block, 180 blocks per side). File names may be as long as 15 characters in the UCSD p-System. Only 10 are allowed in TI BASIC.

pascal: acting blaise 'mid the pitfalls

The Zero command removes the old directory entries on a disk and allows you to start fresh. The UCSD p-System can maintain two copies of the directory so that a backup is available. If you take advantage of this feature, your first file will start at block 10. List ("L") shows the files on a PASCAL disk; "E" is similar but gives more information. Change ("C") can be used to change either disk names or file names. Note that disk names include a colon (:) as the last character, but that it's not always displayed.

If you haven't already made yourself a backup copy of the PASCAL disks, you can do that now with the Transfer command. The simplest command after pressing "T" and getting the prompt, "Transfer ?", is "#4,#5". This destroys the contents of the disk in drive 2 (#5) and copies the contents of the disk in drive 1 (#4) to drive 2. If you have only one drive, the appropriate command is "#4,#4". The system prompts you to shuffle the disks in and out when necessary. Disk names can also be used as in "DN1:,DN2:", which destroys the contents of the disk named DN2: and copies the contents of the disk named DN1: to DN2:. You can add the contents of one disk to the contents of another by entering "DN1:=,DN2:$". The "=" sign is a wild card representing any number of characters (any file name) and the "?" sign means "use the same file names on DN2:. Drive numbers may also be used in place of DN1: and DN2:.

The Transfer command can also be used to make a copy of a file under a different name using "DN1:OLD,DN2:NEW". DN1 and DN2 may be the same, but if they are, OLD and NEW must be different file names. Transfer can also be used to print a file to either device #6 or #8, the RS232 ports. It can also be used to send files from one computer to another.

Files can be deleted using the Remove command. Note that the prompt after typing the file name asks if the directory should be updated. The file is not deleted until the directory is updated. So if you reply "no", the file (or files) is not deleted. In fact, the file is not erased at all. Only the directory entry is deleted. If the blocks occupied by the file are not overwritten, it is possible to recreate the directory entry and retrieve the file. More on this in the next article.

Some other file commands will be discussed next month. For now, I'd like to expand on the subject of file names. A file typically has a name of the form, DN:FILE.EXT, or #n:FILE.EXT. The part before the colon (:) is the volume name. #n is device number n (4,5,9 for disks; 6,8 for the RS232) and DN: is the device name (disk name for disks; PRINTER: or REMOUT: for the RS232). Note that a given disk device can hold any disk name. Referring to a file with a DN: prefix causes a search of the disk named DN where ever it is located. Referring to a file with a #n prefix causes a search of whatever disk is in drive #n (4,5, or 9). The system permits you to use a default prefix; "Prefix" allows you to set the default prefix. The default prefix can be either a disk name or a device name. If you enter a file name without a prefix, the default prefix will be assumed by the system. Even single drive systems are affected, for the prefix names are used to prompt you to switch disks. Note that "*" (without a ":") can be used as the prefix for the root disk volume -- the disk in drive 1 (#4) at startup -- usually the disk containing the "SYSTEM." files.

The file name proper, FILE.EXT, is divided for convenience into a name, FILE, and an extension, EXT. The period between the name and the extension is part of the file name. The system attaches ".TEXT" to a file created by the editor unless you force a different extension. My advice is don't force unless you have a very good reason. The system attaches ".CODE" to compiled programs. There can be good reasons to change the names of code files. A file called SYSTEM.STARTUP would have been a code (FILE.CODE) file to begin with, but the name is changed so that the system will automatically run it at startup. I'll get to the way the system treats files with these extensions next time in the discussion of the concept of the workfile.

The UCSD p-System has a wild card feature in its file name system. The wild card characters are "?" and "=". They can stand for any number of characters in a file name. Wild cards can save time, but they can also wipe you out. Be careful. Until you've practiced, use "?" instead of "=". It does the same thing but it gives you a chance to change your mind. I'll discuss the commands using "?"; when you feel brave you can substitute "=". In any command requiring file names (T,C,R, etc.), you always need the volume prefix (or the default prefix will be used). The rest of the name can be any combination of characters and one wild card. The "?" after a prefix is all files on the device or disk indicated. ".TEXT" is all files on the default device which end with ".TEXT". "A" is all files beginning with "A". "A?T" is all files beginning with "A" and ending with "T". In commands using two file names (C,T, etc.), the second name can also contain wild cards. Usually the second name uses the special character, "$", which means "use the same file name found for the first file". "#4:?,#5:$" in a Transfer command transfers all of the disk in device #4 to the disk in device #5 and uses the same names. "#4:A?,#5:$" transfers only those files beginning with "A" and uses the same names. "#4:?,#5:A$" transfers all but appends an "A" at the beginning of each name. Now watch this: "C" and "#4:A?,#5:=" changes all file names beginning with "A" and drops the "A" for the new name. The "=" here is a symbol which represents the string which the "?" stood for (the part following the "A".

(wait while we put on the next reel...))

pearls of PASCAL, continued...

PROCEDURE TO FORMAT A DISK

(The procedure is described for a system with two disk drives. Refer to the manuals if you only have one drive.)

Note: Go to step 3 if you've initialized the disk with the disk manager module.

1. Turn on drives, put UTILITY disk in drive 1 (#4) and your unformatted disk in drive 2 (#5). Turn on system. If already on, press "I".

2. Press "X", "#4:DFORMAT<enter>","5", "40<enter>", "S", "S", "Y", "N", "<space>".

3. Remove UTILITY disk and insert ED-FILR disk in drive 1 (#4). Press "F", "Z", "#5<enter>", "Y", "180<enter>", "diskname<enter>", "Y", "Q".

EXERCISE: A Simple Program.

(The instructions are for a two-drive system. If you have three drives, put the compiler disk, COMPILR, in drive 3 (#9). If you have only one drive, refer to the compiler manual, pages 12-14. The major difficulty is that the program text file must be saved on the disk COMPILR.)

1. Turn on drives, put ED-FILR disk in drive 1 (#4), put your formatted disk in drive 2 (#5), and turn on computer. If already on, press "I".

2. Press "F", "P", "#5<enter>", "D", enter the date, "Y", "Q".

3. Press "E". Editor should respond, "No workfile".

   a. If so, press "<enter>".

   b. If not, press "Q", "E", "F", "N", "Q", and try "E" again (go to 2).

4. Press "I" and insert (type) the program, "EVENINGALL", found below. Spacing and indentation are not critical, but be sure to use the same punctuation.

5. Press "<CTRL C>" (which means <etx> or "end text").

Note that "<CTRL .>" means <esc>.

6. Press "Q", "W", "HELLO<enter>", "E".

7. Remove ED-FILR disk from drive 1 and insert COMPILR disk.

8. Press "C" (and wait), "HELLO<enter>", "HELLO<enter>".

9. Remove COMPILR and insert ED-FILR.

10. Press "X", "HELLO<enter>","<space>", "U", "<space>", "U".

11. Congratulations.

Here is the program, "EVENINGALL".

```
PROGRAM EVENINGALL;
BEGIN
  WRITELN;
  WRITE('HELLO');
  WRITE(' ','HELLO HELLO');
  WRITELN('.  ','GOOD EVENING ALL.');
  WRITELN('AND WHAT DO WE HAVE HERE?');
  WRITELN('HELLO HELLO HELLO');
  WRITELN;
  WRITELN('GOODBYE');
  WRITELN
END. (* EVENINGALL *)
```

>Ronald J Hartranft

When DFORMAT don't --

Mike Bruss, writing in the excellent Central Valley 99/4 UG Newsletter (c/o RATCH, Genetics Dept., UCD, Davis, CA 95616), notes how to format disks with the CorComp controller for the p-system:

If you have the p-code card, you will be happy to hear that it works very well with the CorComp disk controller card. To use a disk in double density mode, you will have to format it, and this is where a warning is necessary. If you have the p-system utilities, you may know that it contains a disk formatting program called DFORMAT. One its options is double density formatting. DFORMAT unfortunately ignores the option and the disk will be formatted to single density. No error message is issued and the p-system will let you transfer up to 720 blocks (for DSDD) even though only 180 are present.... The bottom line is

do not use DFORMAT for double-density.

The correct way is a two step process. First, format the disk as double-density using either the CorComp or TI disk manager. Do not put any files on the disk -- not even the CorComp manager. Second, fire up the p-system and call up the FILER. Use the Zero command to "zero" the directory, and you're ready to go...

hardware: doubling drives..

CorComp's card review

The CorComp DSDD Disk Controller card is without a doubt the biggest improvement I have made to my system. This card should have been the original equipment! Let me explain my statement: The drives that came with my box were Shugart 400L single-side double-density (SSDD) drives. The controller card from TI is double-sided single-density (DSSD). So, we can access 90K per disk even though the drive is capable of 180K and the controller is capable of 180K. Why would any company sell such a system? The CorComp card ends this problem with its ability to read and write either SSSD, DSSD, SSDD, and of course DSDD.

Performance of the card is excellent. Speed was the first improvement to surface. Yes, double-density is faster to read and write. The card allows the head-step rates to be set for each individual head. I set the step rate for both of my drives to 3ms which also increased the drive access speed.

Powering-up the 99/4A with the CorComp card allows more options then the standard system. This is done with the use of a new menu that allows up to four selections. The first option is the Disk Manager, the second is for TI Basic, the third and fourth are Module selections -- if your module has multiple selections, the first and second will be presented with the menu at this time.

The Disk Manager is installed on disk and is 98 sectors long. But the Disk Manager loads in 8 seconds using double-density! The Disk Manager does all the things the TI manager does plus some extras -- a sector by sector copy ('Turbo copy') for backup, selection of sector interlacing, choosing foreground and background colors and the ability to call the Manager from other environments. The Manager should be saved on each disk for your convenience, since 98 sectors is not much when you have 1440 sectors per disk.

Other goodies include a set new BASIC commands; named the Tool Shed, these commands come complete with several example programs. There are 9 new commands:

* MPEEK, MPOKE allows reading and writing values into CPU memory.

* VPEEK, VPOKE for reading and writing to VDP memory.

* WRTRG allows writing values to the Video Registers.

* MOVEM allows you to move blocks of memory around.

* EXEC for executing assembly language subprograms by address.

* MGR for loading and running the disk manager program.

* DELETE "LD-CMDS" allows use of low memory expansion with Extended BASIC.

The Manager Disk also includes TI-Forth files which will allow you to load TI-Forth without the ED/ASM module. FORTH on a disk is 90 sectors. In DSDD with two drives you will have 630 sectors left for screens.

Faults found: I have been unable to operate Pascal in DSDD* and unformatted sectors cause complete stoppage when using the copy function. If your module does not come up on the title screen, press the space bar twice for the second menu and make your selection.

Faults aside, I believe the CorComp card is worth the $159-$199 purchase price and is the best enhancement per dollar I'v seen.

>Jack Schreiber

*(Since his writing, Jack has used the procedure noted on page 4; "it works", says he. I've left in his comment as it implies something about CorComp's documentation. -ed)

better DISPLAY AT's

Early on, many xtended users probably have thoroughly learned to use the SIZE(num) qualifier and may have missed and/or forgotten the following technique. (SIZE limits how much of the screen line will be erased. Coupled with the IMAGE, USING, and ACCEPT AT commands, quite sophisticated screens are made possible.) Jerry Glaze, a member of the Southern Nevada UG* has turned up this friendly XBASIC quirk:

xbasic ;=doing without

```
100 DATA SEMICOLONS,DON'T NE
ED,THE SIZE(-NN),SYNTAX,,UNL
IKE,DISPLAY AT'S,WITHOUT,SEM
ICOLONS,
110 FOR A=1 TO 24 :: PRINT "
this is a line to fill it up
"; :: NEXT A
120 FOR A=0 TO 8 :: CALL COL
OR(A,5,16,A+4,5,1):: NEXT A
130 B=B+1 :: READ A$ :: IF A
$>"" THEN DISPLAY AT(B+3,8):
A$;:: GOTO 130
140 B=B+1 :: READ A$ :: IF A
$>"" THEN DISPLAY AT(B+3,8):
A$ :: GOTO 140 !no ; after!
150 GOTO 150
```

*(SNUG, PO BOX 26301, Las Vegas, NV 89126-0301)

xperimenter's XBASIC:
sweat (WHEW! —remember) pays off

Previously, the I/O PORT has published some direct sound chip LOADing programs. First we had the BLACK BOX MEMORY LOADER (from Atlanta) and in December the symphonic WHEW6.* Both created sounds that were "out-of-this-world", or at very least, the corner arcade. One wondered then if this wrinkle could be made useful -- and where in the world this info would turn up.

One answer comes from the ends of the earth. Stephan Shaw, writing for the Autumn TI*MES in England and reprinted in T.I.U.P's TIT-BITS*, pointed out the path. (The TI*MES reprint also included Sydney Michel on MINIMEM screen scolling.) The rest was merely following 'long the different research threads:

1. I finally READ the SOUND chapter in the ED/ASM manual. AND hit the notepad with pencil and worked out TI's obtuse notation. (When in doubt, use your computer to crunch numbers.)
2. The manual won't mean a thing unless you disassemble the console ROM's sound interrupt routine. And when you do, you'll find out things TI didn't bother to document.

SIDEBAR: Long term villian, not sufficiently maligned and saved by me for the proper audience: HERB SHANZER. Manager (circa March '83) Calculator & Compact Computer Division, TI INC.
Said Herb:
"It would be impossible for us to document those (AL features) in sufficiently sanitary condition for them also to be totally useful to the...user. And it is our value judgement that, rather than restrict the flexiblity of those features, or hide the capabilities that we can't explain IN VERY SIMPLE TERMS, WE WOULD PREFER...A RELATIVELY CURSORY LEVEL OF INFORMATION AND DOCUMENTION...."
(from 99'er HCM, April 1983. Caps are mine.)
Old Herb was talking about the CC40, but he evidences a peculiar TI mind-set.

Anyway, I'm going to break with I/O PORT tradition and include a long program. But judging by other newsletters, long music listings don't seem to bother other editors, so... But first, a quick demonstration.

Stephen's article was about using the MINIMEM speech access and sound list processing and sundry approaches to using direct sound.
He references a direct sound loader by Neil Lawson in SMART PROGRAMMER, something that I couldn't find. Embarrassingly, what I DID find was Neil's program for 40 column XBASIC display in the Feb '84 SMART PROGRAMMER. THAT has been an ongoing topic in the last two I/O PORT issues, and mistakenly credited to Gary Noel. Humbug! Just goes to show you're as good as your sources.... It says something to me about BBS personalities as well.

*(About WHEW6: change line 3's FOR B=10 TO -10 to B=53 TO -50. WHEW6 then achieves a sort of ABBA song form which becomes more or less recognizable.
*T.I.U.P, PO BOX 246, MOUNT LAWLEY, WESTERN AUSTRALIA 6050. The U.P. stands for Users of Perth. Their quarterly newsletter is first rate. It ain't cheap to exchange with them either -- about 50 cents per half ounce. I don't know how they afford the postage AND the printing. Advice: don't bother them with a two-page newsletter of Views from the Pres and Last Month's Minutes with Directions to the Next Meeting.)

These SUBs really belong at the end of the program, but they're used by both programs and their understanding is integral. The details are at the right, on page 7-->

```
SUB ENABLE
CALL LOAD(-31806,0)::SUBEND

SUB DISABLE
CALL LOAD(-31806,32)::SUBEND

SUB XOFF(VOICE)
A=(VOICE-1)*32+159
CALL LOAD(-31744,A)::SUBEND
```

CRASH DEMO
from ED/ASM manual

```
100 CHIP=-31744 :: CALL INIT
110 SRC$(0)="GROM" :: SRC$(1
)="VDP"
120 IMAGE Slist ### ###  Tri
gger ###
130 IMAGE SYSSTA  ### (####)

140 CALL DISABLE

150 CALL PEEK(-31796,A,B,C):
: PRINT USING 120:A,B,C
160 CALL LOAD(-31796,7,0,1)

170 CALL PEEK(-31747,A):: B=
1+(A=0):: PRINT USING 130:A,
SRC$(B)
180 CALL LOAD(-31747,1)

190 CALL CHAR(128,"039FE4F20
502E4F00C02E4F20A02E4F40802E
4F60602E4F60602E4F80402E
4FA")
200 CALL CHAR(132,"0201FF00"
)

210 CALL ENABLE
220 PRINT : : : :"CRASH"
230 CALL SOUND(100,440,2)
240 PRINT "BEEP"
```

sound fundamentals

SUBprogram thumbnails:

The first two set the SOUND interrupt bit on and off. The PAD location in hex is >83C2. The high nybble (top four bits) are tested in order by the interrupt routines. The technique is simple enough, just a SLA, JOC pair. That's on a par with name-dropping -- Shift Left Arithmetic sets the carry status bit and the Jump On Carry skips the appropriate routine. In other words, if the bit is equal to zero, the appropriate routine is executed.
The bits, in order and from the MSBit down are: all user interrupts, sprite auto motion, SOUND list processing, and finally the QUIT key function. Since these are all in the high nybble, the decimal values required to set each are: 128, 64, 32 (Aha! that's DISABLE), and 16. Just add up the ones you'd like to stop.

setting voices w/ clarity

One would be hard-pressed to make the How-To of setting the SOUND chip's three voices plus noise any more obtuse than the ED/ASM manual makes it. Try these rules-of-thumb:

THE HIGH NYBBLE OF THE FIRST BYTE SENT SPECIFIES THE VOICE.
Each one has a 'name': Voice1 is named >8, Voice2 is >A, Voice3 is >C and the Noise is >E.

TO SET A TONE:
The first nybble will be the 'name'. Tones require two bytes.

TO SELECT A TYPE OF NOISE:
The first nybble will be >E. The second nybble ranges from 0 to 7.

TO SET ANY VOLUME:
Add one to the appropriate 'name': thus >9, >B, >D, and >F for Voice1 through Noise.
Maximum volume is 0. Turn off a voice with a low nybble of >F.

Since the SOUND chip gets its values just from the high byte of the data ~~address~~ bus, you have to pass the settings in two parts, a byte at a time. The first byte always contains the name of the voice; if the high nybble is even you'll set the frequency/noise type. Notice the volume settings pair up: 8 and 9 mean voice 1, A and B mean voice 2 and so on.

IMPORTANT: Because only the high byte is connected, that means only even bytes on CALL LOADs get sent to the SOUND chip. To LOAD from BASIC put a dummy value for the uneven bytes: CALL LOAD(-31744,A,0,B). A more elegant method might be CALL LOAD(-31744,A,A,B,B,C,C,D,D) etc, etc. Every second instance won't do anything. The SOUND chip will respond within the range of >8400 to >87FE or -31744 to -30722, so you don't need to reset the LOAD address often.

TO CALCULATE ANY GIVEN FREQUENCY:
Divide the frequency into 111860.8, then transform into low nybble and high six bits. It's easier to show the calculations than explain them:

```
A=111860.8/FREQ :: HI=(A AND 1008)/16 :: LO=(A AND 15)
```

HI is masked by 1008, hex 3F0, then dividing by 16 drops the 0 nybble. This value gets sent second. LO must have the 'name' added: 128, 160, 193 for Voice1 through Voice3. (Go ahead and try it: CALL LOAD(-31744,128+5,0,63,144)
The 144 sets voice1 to 0 or maximum volume. ↳ 0,

TO CALCULATE WESTERN TONAL SCALES:
A musical half-step increase can be found by multiplying by (2^(1/12)). A half-step down divides the frequency by the same value. A full-step is acchieved with two multiplications or two divisions.

---

fly, you blackbirds

a ragtime cakewalk

```
100 DATA 03AC1AB010
110 DATA 03860D9210
120 DATA 08908F08AD11C315D01
0
130 DATA 028A0A10
140 DATA 07D291BFCB238F0720
150 DATA 03B0C31510
160 DATA 028A0A10
170 DATA 05BF8F08CC1A10
180 DATA 02860D10
190 DATA 05B1C3158A0A10
200 DATA 028F0710
210 DATA 03BFCB2310
220 DATA 028F0810
230 DATA 058A0AB2C31510
240 DATA 02860D10
250 DATA 049FBFCC1A10
260 DATA 03860D9110
270 DATA 05B1C3158F0810
280 DATA 038A0A9010
290 DATA 058F07BFC41920
300 DATA 04B1CE0F9110
310 DATA 028A0A10
320 DATA 05DF800AAD1710
330 DATA 028E0B10
340 DATA 05D1A014800A10
350 DATA 03908F0710
360 DATA 04DFAC1F9110
370 DATA 02800A10
380 DATA 05D28A0AA01410
390 DATA 028E0B11
400 DATA 039FBFDF0101B20F
410 DATA 07B08107C00A91D210
420 DATA 03AD11D20F019401
430 DATA 0290D10F0295D301
440 DATA 06AB23D1918F0720
450 DATA 0295D2010491D0AD111
0
460 DATA 049FA014DA0101D10F
470 DATA 03C20ED010
480 DATA 05D1CE0BAD1110
490 DATA 02C00A10
500 DATA 04DFB1AB2310
510 DATA 01D210
520 DATA 07AD11B08F0891DF10
530 DATA 028F0710
540 DATA 039FA01410
550 DATA 07B08107C00A91D210
560 DATA 03AD11D20F019401
570 DATA 0290D10F0295D301
```

listing continues -->

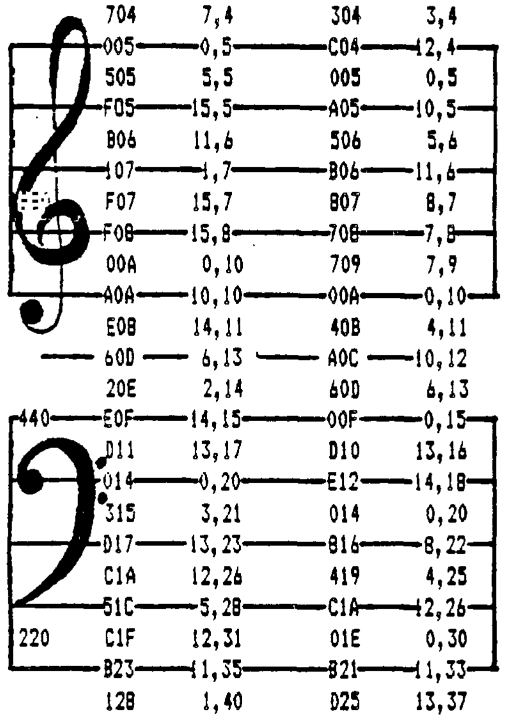
Frequency requires two bytes: / Volume needs one byte

by the nybbles:	BYTE ONE		BYTE TWO	decimal LOAD values		BYTE ONE		LOAD values
	voice 'name'	low 4 LLLL	most signif. OOHH HHHH	Base	Maximum	voice 'name'	volume	Base
VOICE 1	>8		00	128,	63	>9		144
VOICE 2	>A	0 to F	00 0 to >3F	160,	63	>B		176
VOICE 3	>C		00	192,	63	>D		208
							0 to F	

Noise needs one byte:

		0TTT					
NOISE	>E	0 to 7	→	224	>F		240

The scale that should have been



note/ freq	hex code (naturals)	dec. LOAD	hex code (sharps)	dec. LOAD
C	503	5,3	203	2,3
B	903	9,3	503	5,3
A 1760	004	0,4	C03	12,3
G	704	7,4	304	3,4
F	005	0,5	C04	12,4
E	505	5,5	005	0,5
D	F05	15,5	A05	10,5
C	B06	11,6	506	5,6
B	107	1,7	B06	11,6
A	F07	15,7	807	8,7
G	F08	15,8	708	7,8
F	00A	0,10	709	7,9
E	A0A	10,10	00A	0,10
D	E0B	14,11	40B	4,11
C	60D	6,13	A0C	10,12
B	20E	2,14	60D	6,13
A 440	E0F	14,15	00F	0,15
G	D11	13,17	D10	13,16
F	014	0,20	E12	14,18
E	315	3,21	014	0,20
D	D17	13,23	816	8,22
C	C1A	12,26	419	4,25
B	51C	5,28	C1A	12,26
A 220	C1F	12,31	01E	0,30
G	B23	11,35	821	11,33
F	128	1,40	D25	13,37
E	72A	7,42	128	1,40
D	A2F	10,47	F2C	15,44
C	735	7,53	732	7,50
B	A38	10,56	735	7,53
A 110	93F	9,63	03C	0,60

This scale puts middle A, 440, one octave lower than usual -- making entering low notes on a printed F Clef possible w/out transposition.

TO USE: Find the appropriate note and add the voice nybble from the table at the top.

Thus, for Voice3 set to the D note between Clefs, one adds >C to >E0B: >CE0B.

For direct BASIC LOAD, 192 is added to the first byte value, 14, giving decimal LOAD pair of 206,11. The command will look like this:

```
 CALL LOAD(-31744,206,0,11)
```

Remember to put a dummy number between each half, so that the values are passed to the SOUND chip only on even addresses.

FLATS: Take the sharp value from the next lower line, excepting F and C, which take the naturals E and B. Both B to C and E to F are legitimate half-steps. (Check E# and F.)

ABOUT BLACKBIRDS (and CRASH demo)

This program uses the SOUND list processing interrupt to play the song. The significant advantages include: true three voice melodies are possible and the volume can be changed independently from the frequency. The list is processed automatically so your program may continue doing other things. Optionally, when the song/sound list is finished, you may either stop processing, start a new list or even start over. (Important: this is determined by how the list

'blackbirds' continued

```
580 DATA 06AB23D1918F0720
590 DATA 0295D2010491D0AD110
F
600 DATA 06DF8A0AAC1F9010
610 DATA 02860D10
620 DATA 078E0BAD11C315D110
630 DATA 03918A0A10
640 DATA 04BFCB239210
650 DATA 03918E0B10
660 DATA 03DF8A0A10
670 DATA 02800A10
680 DATA 03AC1AB010
690 DATA 03860D9210
700 DATA 08908F08AD11C315D01
0
710 DATA 028A0A10
720 DATA 07D291BFCB238F0720
730 DATA 03B0C31510
740 DATA 028A0A10
750 DATA 05BF8F08CC1A10
760 DATA 02860D10
770 DATA 05B1C3158A0A10
780 DATA 028F0710
790 DATA 03BFCB2310
800 DATA 028F0810
810 DATA 058A0AB2C31510
820 DATA 02860D10
830 DATA 06CC1A8B06BFD010
840 DATA 01980101900F
850 DATA 03CD179110
860 DATA 01990101900F
870 DATA 0698A807B2C31501019
01F
880 DATA 05D1CC1A98B7010291B
11F
890 DATA 0598AF07C0140101900
F
900 DATA 03800A8010
910 DATA 04CC1A8F0710
920 DATA 028B0610
930 DATA 0391C01410
940 DATA 02810710
950 DATA 03DF8B0610
960 DATA 02810710
970 DATA 098B06A40BCE2190B0D
020
980 DATA 039FBFDF010390B0D03
F
990 DATA 099FBFDF8F08AA0ACD1
120
1000 DATA 0390B0D020
1010 DATA 039FBFDF010390B0D0
3F
1020 DATA 03DF8FBF20
1030 DATA 01D210
1040 DATA 019110
1050 DATA 05B192DFAB2310
```

Keep going →

ends -- your program doesn't have to keep track. The closest TI comes to documenting this feature is in PASCAL's SOUND UNIT. Stick it in yer ear, Herb.)

! CRASH demo. Line 200 contains "0201FF00". Ignore the >02, letting us break down the rest: 01 - bytes to follow, FF - noise off, 00 - command duration. The 00 tells the interrupt routine to STOP processing. Change line 200 to read: CALL CHAR(132,"02000700") and RUN. Notice any difference?

Both BLACKBIRDS and the CRASH demo use the CALL CHAR command to load a sound list into the VDP. (This is a headache if you'd like graphics with your songs. XBASIC users may use a CALL POKEV program which is not supplied herewithin to use some other part of the VDP for their list. Of course, XBASIC FORTH avoids all of that...) However, we know where any character pattern is located and can readily calculate the address -- A=(C+96)*8. So in CRASH demo, (128+96)*8=1792. Converting to hex we get >700 and splitting into bytes 7,0.

These last two are what you use to set the the SOUND list pointer at -31796 (>83CC). Additionally, both programs LOAD >83CE, the SOUND list trigger, with the list's address LOAD. We get away with that because we've CALLed DISABLE; otherwise one must LOAD the SYSTEM STATUS BYTE at >83FD with 1 first which tells the interrupt we've a list in the VDP. Another undocumented feature is that a 0 in >83CE signifies that the list is in GROM. Neglecting this LOAD or given the wrong order of events, we'll process GROM as a random sound list. Mostly harmless because the SOUND routine is smart -- it saves the current GROM address before (>83CE) that specifies the list is in the VDP.

(The CRASH program can show where HONK and BEEP are located. Make a syntax error -- type KK(enter), put a BREAK at line 160 and RUN. The GROM address shown is AFTER the sound list, so subtract 3 or 4. In short order, you'll get the correct address. Now you can zap 'em with your own HONK!)

That's about it. Remember to ENABLE the sound interrupt by resetting >83C2 (-31806.) ('Reset' means setting a bit off or equal to 0. 'Set', obviously, is its inverse of turning a bit on.) One final thing: a flow chart of the SOUND interrupt. Notice how the duration and command length bytes get handled.

>Frederick Hawkins

```
     ?ENABLED? (sound bit in >83C2=0?)     NO: SNDEXIT

yes  ?TRIGGERED? (>83CE greater than 0?)  NO: SNDEXIT

yes   DECREMENT DURATION (>83CE minus byte at >83FE. This is the byte that
                                  adjusts sound, cursor flash rate...(-31746))
     ?DURATION=0? (notice EQUAL ZERO!)     NO: SNDEXIT

yes   FETCH NEW COUNT (from VDP or GROM depending on bit in >83FD)
     ?COUNT=0? (zero stops this list but not processing!)    YES: MORSND
no   ?COUNT=>FF? (do we want to change source of list?)     YES: SNDSRC

no    SEND DATA TO SOUND CHIP
      FETCH NEW DURATION
     ?DURATION=0? (a zero here will stop sound processing)  YES: SNDOFF
      NO: NEWSND
--------------------------------------------------------------------------
SNDSRC:  Toggle the VDP/GROM bit at >83FD.       falls thru  MORSND
MORSND:  Next word = new SOUND LIST Address. SET DURATION=1. GOTO NEWSND.
--------------------------------------------------------------------------
SNDOFF:  DURATION=0 (yeah, TI's code is redundant!  It's already zero.  fall thru
NEWSND:  Save list pointer in >83CC and DURATION in >83CE (also used to TRIGGER)
SNDEXT:  quit key check is next, then user interrupt routine.....
```

```
1060 DATA 02870810
1070 DATA 0790B0D1A00A8F0720
1080 DATA 06A107800ACB2320
1090 DATA 07D0CC1A8B06AA0A22
1100 DATA 039FBFDF20
1110 DATA 0390B0D020
1120 DATA 04FF9FBFDF2E0
1130 DATA 000400,
1140 CALL INIT
1145 !"fly, you blackbirds"
from Brainard's Ragtime
   Collection (1899)
1146 ! arranged by
      DENES AGAY
(from The Joy of Ragtime
  selected & edited by
      Denes Agay
Yorktown Music Press, Inc)
1147 ! pgm by Fred Hawkins
1150 FOR VOICE=0 TO 3 :: CAL
L XOFF(VOICE):: NEXT VOICE
1160 X=32 :: CALL DISABLE
1170 CALL LOAD(-31796,4,0,1)
!set sound list and trigger
1180 CALL LOAD(-31747,1)!set
 sound source pointer
1190 READ A$ :: PRINT A$ ::
B$=B$&A$
1200 IF LEN(B$)>64 THEN GOSU
B 1240
1210 IF A$>"" THEN 1190
1220 IF LEN(B$)>1 THEN PRINT
 LEN(B$);B$ :: GOSUB 1240
1230 GOTO 1260
1240 C$=SEG$(B$,1,64):: B$=S
EG$(B$,65,64):: PRINT : :B$:
:
1250 PRINT :"LOADING";X;LEN(
C$):: CALL CHAR(X,C$):: X=X+
4 :: RETURN
1260 ACCEPT AT(4,4):C$ :: IF
 LEN(B$)>64 THEN CALL CHAR(1
32,SEG$(B$,65,255))
1270 CALL ENABLE
1280 PRINT "PLAY"
1290 FOR A=1 TO 12 :: CALL C
OLOR(A,3,13):: NEXT A :: CAL
L MAGNIFY(4)
1300 CALL SPRITE(#1,32,16,40
,50,1,3)
1310 FOR A=0 TO 80 :: CALL P
ATTERN(#1,32+A):: CALL COLOR
(#1,13*RND+3):: NEXT A :
: GOTO 1310
```

don't forget SUBs from page 6.

the asynchronous sieve

First off the line this month are some updates on last issue's travels around the bulletin boards. The Washington DC area board has a new phone number. If you have had some trouble getting through try this number: 703-631-8772. Also noticed along the way was a charge to access the download files of the Tampa TIBBS. It will cost you $15 to initially gain access and they request that you upload something for each download. Long distance callers can mail a disk of programs to save on their phone bill charges. This may seem like a high price to pay but a recent look into the download section revealed the following programs: 24 hour clock; SOUND-4TH, a FORTH music program; COMM99, a terminal emulator program with 16K buffer; and VFILER, a program to edit, print, view and show disk directories. This doesn't sound like the average offerings found on a BBS. YOU decide if it's worth the money!

Another BIG item that everyone seems to be talking about is the news from RYTE DATA of a TI-99/4A compatible computer. The manufacturer's name has not been disclosed as yet (we have a pretty good idea of who it might be) but more details are forthcoming. The machine sounds like a dream come true and the initial release mentions too many extras to even begin to cover in this column. In fact the release is a newsletter four pages long. To get a copy to read for yourself write:

RYTE DATA
BOX 210 MOUNTAIN ST
HALIBURTON, ONTARIO
KOM 1S0 CANADA

RYTE DATA and the manufacturer seem very interested in what you as the current TI user have to say and are looking for written responses in the form of ideas, suggestions and *just good* input on the final version of this new computer. Just a few tidbits to get your mouth watering: the proposed computer should have 128K RAM expandable to 512K and will use the current peripheral expansion box and built around the TM9995 micro processor chip (fully compatible with the 9900 in your 99/4A).

Has anyone out there had any dealings with the International Software Club? I recently ordered 20 programs from them and when my order arrived I found out that four programs had been randomly substituted and another two were not usable. The one looked incomplete and the other was so garbled that I couldn't figure it out. Even allowing for minor bugs, this program shouldn't have been listed for sale. I'm really disappointed with this group and would not advise using their services.

Interested in FORTH? The LA 99er's have compiled a 20 page booklet of articles and screens from various TI newsletters around the world. Coupled more recent articles that have appeared and there is almost enough information to fill a small book. Who said that there was no information available for TI-FORTH! (I recall a $5 fee, contact: LA 99ers CG, PO BOX 3547, GARDENA, CA 90247-7247)

There's just enough space left for a DEAD TURKEY AWARD. Two months ago I ordered the Hitchhiker's Guide to the Galaxy directly from INFOCOM. At that time they informed me that I should receive it in 2-3 weeks. Nearly six weeks later I received a notice that it was out of stock and would not be available until sometime in March. This didn't bother me until my Visa bill came with the charge from INFOCOM dated 1-10-85. March First arrived on time; the game didn't! A phone call to INFOCOM revealed that it was still backordered. I cancelled the order, called TENEX and had the game in my grubby little hands on March 7th at six bucks less than INFOCOM had charged me! So much for going right to the source. Question; Since I had "paid" for the program on 1-10 and did not receive it from INFOCOM did that entitle me to "pirate" a copy from another source? After all I did pay for it. Try that one on for size INFOCOM. In fact I could have made 100 copies in one day and INFOCOM couldn't get me ONE in TWO months time!

That's all for this month; I hope to survive long enough to write next month's column. Remember: DON'T PANIC.

> Dave Hendricks

P.O.Box 4837 * 1501 Lehigh St.
Allentown, Penna. 18103

```
------------------
! stamp target    !
!   put it        !
!   here, pal     !
Allentown, PA.18102
! PERMIT NO.2018  !
------------------
```

The Real Programmer suspects that the cutting edge of technology is like baby teeth: one is amazed when they appear but both have their fallings out.