WINNIPEG 99/4 USERS GROUP

Newsletter

January's Newsletter

The Winnipeg 99/4 User Group is a non-profit organization formed to meet the needs of Manitoba based Texas Instruments users. The content of this publication does not necessarily represent the view of the Winnipeg 99/4 User Group. This newsletter is one form of communication to keep Manitobans up on Texas Instruments Computers and its clones.

Next General Meeting - Date : Febuary 6th, 1986
Time : 7:00 P.M.
Place: Winnipeg Centennial Library
2nd Floor, Assembly Room

Mailing Address: NEWSLETTER EDITER
WINNIPEG 99/4 USERS GROUP
P.O.B. 1715
WINNIPEG, MANITOBA
CANADA, R3C 2Z6

BBS #: (204)-889-1432 SYSOP: Charles Carlson
SETTINGS: 300/1200 baud, 7 bits, 1 stop bit, odd parity
HOURS: 7:00 p.m. to 8:00 a.m. 7 days/week.

EDITORIAL COMMENTS:

Hope all have had a good Christmas and are looking forward to another swing around ol' Sol. This month Paul Degner, our contributing editor, seems to be exercising his literary talents as he gives us his contribution, Quid Pro Quo. A hefty five pages. Schematics and description of an in-console 32K memory expansion are also in this issue. These were submitted by Rick Lumsden, and are much appreciated. The Programmer's Helpfile Basic section takes a look at using 'character strings' (or text) in programs. Reviews takes a look at the MYARC 128K expansion card. Thanks go to Minn and Dakota Home Users Group, who meet in Grand Forks, for this review. While I'm on the subject I'd like to wish all the users groups that we are in contact with a Happy New Year, and thank them for the many newsletters that we have received and enjoyed reading. That about does it. See you all next month.

If you have a review, user hints, or helpful programming tips, get them to me for the next newsletter. The deadline that I have set for submissions is one week before the date of the group's meeting. If you can't contact me by phone, mail your submission to the POB address on the front cover or get them to Paul Degner, contributing editer. Thanks to all who have submitted items for this issue of our newsletter.

MISCELLANIA:

Miscellaneous news and reminders.

Paul Degner has received a video tape of the Chicago TI-Faire. Portions of the tape will be viewed at meetings over the next few months.

The software packages sold at the last meeting seem to have been a hit. Paul will continue to bring these packages to meetings as long as they are popular. I don't know why no one tried doing this sort of thing before.

For more news and some interesting mail from the Timeline BBS turn to Quid pro Quo.

READER RESPONSE:

Have a comments or questions that you would like to express to the other readers of our club's newsletter? Space is available, here, for this purpose. Please use it!

REVIEWS:

This column presents reviews of materials that may be of interest to the user. The views expressed are the opinions of the reviewers, exclusively.

HARDWARE:

The following was taken from MAD HUG's December 1985 newsletter.

???MEMORY EXPANSION???

Is the MYARC 128K expansion card really worth it? I'd say yes! It has 32K of computer accessible RAM and 96K (96K + 32K = 128K) for RAM-disk and/or PRINT SPOOLER.

Let me explain what a RAM-disk and PRINT SPOOLER is for those who really want to know.

A RAM-disk is a block of memory that can be utilized as if it were a floppy diskette and is ultra-fast.

A PRINT SPOOLER is a buffer that holds text or data to be printed. It gives the user control of the computer faster. With the PRINT SPOOLER the computer dumps the text or data to the buffer then gives control to the user and continues to print until the buffer is empty. You can do much more computing in the same amount of time instead of watching your printer print.

The user also gets a few more commands.

CALL PART(X,Y) partitions memory. Where X equals Kbytes allocated to RAM-disk, Y equals Kbytes allocated to the PRINT SPOOLER, and the sum of X and Y is equal to 96K. This is how you "INITIALIZE" the memory and must be done every time the power is turned on.

The device name for RAM-disk is RD. CALL EMDK(X) instructs the RAM-disk to emulate a floppy diskette or turn off the emulation. X=1 instructs RD to emulate DSK1. X=0 turns off the emulation.

The user can emulate disk 1-5. Since it is emulating a disk drive the user can use OLD DSKX.NAME or OLD RD.NAME (same with SAVE) to access programs in RAM-disk.

The device name for PRINT SPOOLER is PS/1, PS/2, or PIO. Which is RS232/1, RS232/2, and PIO respectively. To abort the PRINT SPOOLER, ENTER: CALL ABPS/1 or /2.

To get a directory listing of all files in the (RAM-disk) directory from BASIC or EXTENDED BASIC, ENTER: CALL RDDIR. It acts and looks like the the directory the DISK MANAGER 2 produces. The user can also add a DISK NAME to the RAM-disk by using CALL VOL("volume-name").

MYARC also has a power pack that plugs into the back of the card. This keeps the card in a power up state and the user can retain all the data in the card, and so the user doesn't have to 'PARTITION' (INITIALIZE) the memory on every power up.

This card is expandable to 512K. The only drawback I have found is this expansion card is not compatable with the CorComp RS232 card. The PRINT SPOOLER feature won't work with this RS232.

Once you use it, you won't give it up.

- M.A.G. -

SOFTWARE:

Acquire some software for your TI lately and feel good, bad, or indifferent about it. This space is available for your comments.

HELPFUL HINTS AND TIPS! (FOR THE USERS, BY THE USERS!)

This column features tips brought to my attention from members of this group, other user group's newsletters, and various other sources. **WARNING:** These hints and tips are to be used at your own risk!

SPEECH:

Ever wonder how to get all of the words listed for the speech synthesizer to work from XBASIC? There are several multiple word combinations that don't quite do what they should unless you know the syntax that the speech subroutine recognizes. Enclosing multiple word phrases within number signs will solve this problem.
e.g. #GOOD WORK#, #TEXAS INSTRUMENTS#, #THAT IS CORRECT#, and so on.

EXTENDED BASIC

This tip was taken from the December 1984 edition of MICROpendium.

Dan Parrott, president of SMAUG of Grand Bay, Alabama, writes: "Smooth, curvilinear sprite motion can be achieved with sine and cosine functions. The trick is to store the factors derived from the trigonometric functions in an array that can be called much faster than having to recalculate each time. For example:"

```
100  CALL CLEAR
110  CALL SCREEN(2)
120  DIM  A1(28),A2(28)
130  FOR  A=1 TO 28
140  CALL SPRITE(#A,46,16,98,128)
150  A2(A)=SIN(A/4.456)*40 :: A2(A)=COS(A/4.456) :: NEXT A
160  FOR  A=1 TO 28 :: CALL MOTION(#A,-A1(A),-A2(A)) :: NEXT A
170  GOTO 130
```

Parrott recommends that you vary the constants in line 150 to produce different results. This requires Extended BASIC.

SUBMITTED BY RICK LUMSDEN.

THIS ARTICLE IS REPRINTED FROM AN ARTICLE APPEARING IN THE R/D COMPUTING NEWSLETTER FROM RYTE DATA IN HALIBURTON ONTARIO

IMPORTANT:
Any modifications to your computer should be attempted by users who have some expereince in electronic work and will void any warranty in effect.

32K MEMORY EXPANSION INSIDE THE TI-99/4A CONSOLE.

NOTE: This 32K version does not run off the 16 Bit Buss that was reported in the last newsletter. This is an econo version that will run most popular software(e.g. TI-Writer,Multiplan,TI-Logo,TI-Forth,Editor Assembler) but may not work with machine language programs where CRITICAL timing is involved. An example is programs that use the Speech Synthesizer.

The circuit is shown on the attached wiring schematic and can be built on a piece of stripboard and mounted on the RF sheild on the computer motherboard. The schematic is very straightforward and the only thing that needs clarification is the fact that in the drawing it only shows to HM6264LP-15 chips. This is because the chips are "piggybacked" one on top of the other.

This article originally comes from the International Society of Almalgamted Dodo Users and Dead Ducks (ISADUDD) from Australia. If this circuit is built on a large enough stripboard, as mentioned in the parts list, there are more circuits on the drawing board. These include:

1. Provision of a CRU selectable 8K CMOS RAM chip in the DSR area of CPU RAM from >4000->5FFF
2. Console ROM to be selectable between ROM and battery backed CMOS RAM.
3. An EPROM copier-progammer with a zero insertion force socket mounted on the cooling slots on the top of the console.

This modification uses the Hitachi Static RAM chips that do not need the refresh circuitry of a Dynamic RAM and thus accounts for the small parts count required. Cassete users should be advised that the 32K is not fully usable since you can still only save 12K to tape, however, the programs will have a much larger operating space.

PARTS LIST

1. 4 HM6264LP-15 Hitachi CMOS RAM chips
2. 2 or 4 28 pin IC sockets(depending if you "piggyback" the chips)
3. 1 peice of copper stripboard 32 strips wide by 23cm. long
4. 2 or 4 22uf Tantalum capacitors
5. Insulating stand-offs to mount the board on the RF sheild
6. Wire, solder, and soldering gun.

A	REDRAW SCHEMATIC 25-2-85.	[illegible]
B	FORMAL RELEASE ✓	[illegible]

REVISIONS

FIG. 3C.

EXISTING TI-99/4(A) "MOTHER BOARD"

* REAR VIEW OF GROM "EXTENDER" SHOWING PIN NUMBERING AND FUNCTION OF THE WIRES REQUIRED BY THE MEMORY CHIPS.

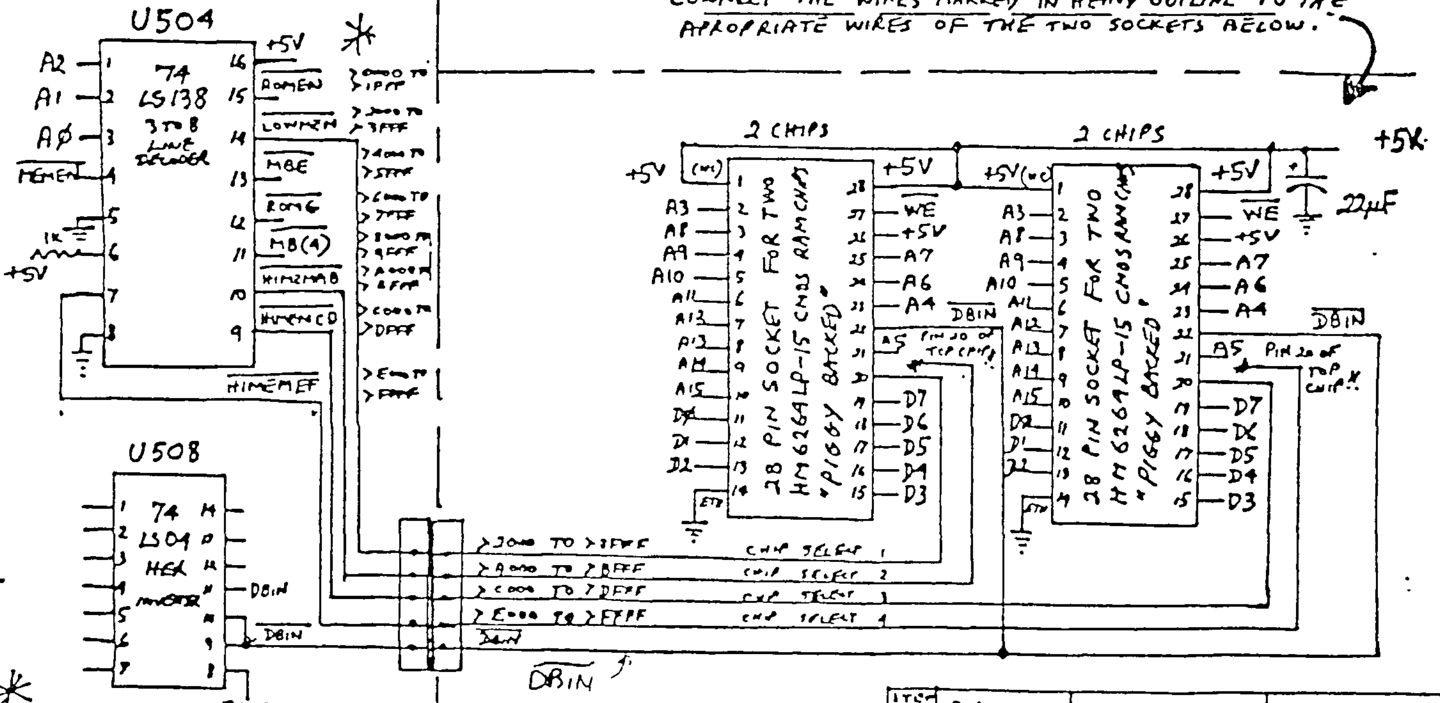
36	34	$\overline{WE}$ 32	A4 30	A5 28	A6 26	A3 24	A7 22	A8 20	A9 18	A10 16	A11 14	A12 12	A13 10	A15 8	6	4	+ 2
35	33	31	29	27	25	A14 23	21	+5V 19	DØ 17	D1 15	D2 13	D3 11	D4 9	D5 7	D6 5	D7 3	1

* REAR VIEW OF GROM EXTENDER. A-INDICATES ADDRESS BUS D-INDICATES DATA BUS.

CONNECTIONS TO REAR OF GROM EXTENDER

FIG. 3A.

CONNECT THE WIRES MARKED IN HEAVY OUTLINE TO THE APPROPRIATE WIRES OF THE TWO SOCKETS BELOW.



32K EXPANSION

FIG. 3B.

FOUR CHIPS ARE IN TWO SOCKETS W/ PINS 20 ON EACH CONNECTED TO ENABLE LINES ON THE 74LS138 DECODER INSIDE THE RF CASE.

*

$\overline{ROMEN}$	ENABLES CONSOLE ROM.
$\overline{ROMG}$	ENABLES MODULE PORT.
$\overline{MBE}$	ENABLES DSR ROMS
$\overline{MB(4)}$	ENABLES PAD.

ITEM No	PART No	DESCRIPTION	MATERIAL	REMARKS

PHIL WEST
BERNIE ELSNER
T.I.U.P. - W.A.
21-3-85

WESTRALIAN INSTRUMENTS
PROJ-01 TI-99/4 AT
32K, MATCHBOX EXP.
D 10853971 6842 3 OF 3

Quid Pro Quo

by Paul Degner

Another year is upon us as well as the cold weather and the colds. Nineteen eighty five was a quiet year for our computer and hopefully not this year too! We expect marvelous things such as seeing the 'Noah' in action. 'Noah' is the current name for the fictional 99/128 personal computer the baby of Myarc Industries. Locally our users group this year hopes to be more 'user friendly' for the interests of its members. The group is trying new methods to increase the body count at our meetings because we found out that some members just don't bother to show up. We want to know why and what we can do to correct this situation. If we don't see a significant member turnout before the end of the year there is always a possibility of a rumour that the group might fold may have some truth. In order to circumvent this rumour is to try to show up at these meetings no matter how boring they can be and present some feedback to the group on the various features the group offers you such as the program libraries, newsletter, and program tutorials. I hope I have enlightened you on your responsibilities for the upcoming months and hopefully we will have a place to come every first thursday of the month.

Briefs:

- (Ottawa NL)\(Yvan Breton) Subject: Display Variable 80 File Recovery

The extensive use of a Terminal Emulator program (TeII or Fast-Term) or any program that creates a Display/Variable 80 file could lead some day to the loss of a valuable file: the Disk Manager shows you a directory where your file is listed, for example: "FILENAME 1(sector) DISPLAY/FIXED 0" but such a file can't be read by Ti-Writer. This article deals with a way of recovering a "lost" file using a program like Disk Fixer or Disk+Aid and was made possible thanks to Bruce Caron's articles on the disk organization. This important series of articles appeared in the Ottawa T.I.99/4 Users' Group Newsletter from November 1984 to March 1985.

Since the following is the result of only a few trials, it would be more prudent to work on a duplicated diskette copied sector by sector rather than using the bit map. For more information on this process see the Disk Manager 1000 manual.

The "lost" file, as it seems, was probably sent to the disk and a Directory entry sector was created for it but, for no known reason, it wasn't closed. Since the exact location of the file is printed on that sector during the closing process, the file doesn't seem to be anywhere although its proper name figures on the directory. If you use the File Recovery option from DM1000, you'll find that it recovers exactly what the directory screen was showing just before. This option won't work in such a case probably because this program rebuilds the file directory entry and not from the file itself. Let's bear in mind that when a file is deleted, the Disk Manager erases the "pointer" of the file directory entry (on sector 01) but not the file directory entry itself.

A disk fixer program like DISK+AID allows you to read or modify information contained on each sector of the disk. If you read the directory entry sector corresponding to your file, you'll see that its name is the only information printed there. If your disk doesn't contain too many files, the directory entry should be located between sector 01 and sector 22, otherwise somewhere close to 22. In order to recover that file, the task will be to write in the same sector the necessary information to be able to read the file from Ti-Writer or another similar program. Here are the operations:

1: SCAN THE DISK: to find the exact location of the file;
2: SEARCH THE [illegible] OF FILE MARKER;
3: WRITE TO DI[illegible]ORY ENTRY SECTOR;
4: READ AND RE-SAVE YOUR FILE.

SCAN THE DISK

To find the file location, it is necessary to scan the disk sector by sector. This process shouldn't be too difficult since the file we're looking for is the DISPLAY type, so we should be able to read it as a text on the screen. Scan the disk until you find a sector that you recognize as belonging to your file. You can scan the disk skipping a few sectors at your convinience. What is then needed are the numbers of the first and last sectors used by the file. With that information, we'll be able later to calculate its size. If your file is not fractured, that is in more than one piece of consecutive sectors, this will be an easy operation. If not, the scanning will then take longer since we'll have to find the number of the starting and ending sectors of every portion of the fractured file.

SEARCH THE END OF FILE MARKER

Once the file has been found, we have to find the position of its end on its last sector. If we're dealing with a fractured file, its location will be on the last -: tor of the last portion of the fractured file. In all the cases I've examined, at the end of a file there is a marker "[illegible]" called the End of File Offset. Depending from which program you've save the file with (TI-Writer or an emulator program), the marker may or may not be preceded by some other characters, not always printable. In the case of the TI-Writer files, these characters refer to the file's tabulation. You can easily check this by comparing last sector of a file saved by any program and the last sector of the same file saved by TI-Writer. What we need to remember is the exact position, in hexadecimal, of the marker for our file.

WRITING TO DIRECTORY ENTRY SECTOR

We now have almost everything we need to write on the file's directory entry sector (a minimum of nine items). Ask your disk fixer program to print on your screen the file's directory entry sector. If you toggle from ASCII to HEX notation, you'll see the corresponding codes of the letters of the file name between position 00 to 09. The file name can occupy a maximum of 10 of the 256 bytes available in that sector (from 00 to FF). Remaining in the hexadecimal notation will make our work easier since the numbers we have to write don't always have a corresponding and printable ASCII character. Here's what we should write:
(postion=POS; value=VAL, both in hex)

POS ØC : VAL 8Ø : This position is for the file status flag so 8Ø indicates that the file is the DISPLAY/VARIABLE type.

POS ØD ; VAL Ø3 : This position refers to the maximum number of records on the sector. Since the size of the sector is 256 bytes, we can only have a maximum of (256/8Ø) 3 or Ø3 records on a sector.

POS ØE and ØF ; VAL X : The value X represents the total number of sectors occupied by the file. This number will represent one sector less than what you would get with a disk manager which counts the directory entry sector. For our purpose the value X will be the number of the last sector of the file minus the number of the first plus one so if, for example, the first sector is 22 and the last is 25, X will be (25)-(22)+1 = 4 sectors (22, 23, 24 and 25). The directory entry uses two bytes to store that value; if your file is longer than FF sectors (255 in dec) it'll need two bytes like Ø1 and 3F for a length of 13F. If your file is shorter than FF it will use only one byte so, in this case, the value X should be written at position ØF only.

POS 1Ø ; VAL Y : The value Y represents the End of File Offset. We have to write here the position in hex of the marker FF seen in the last sector of the file.

POS 11 ; VAL 5Ø : This value stands for the maximum length of each record. Since we are using Variable 8Ø we'll write 5Ø which equals 8Ø, but in hexadecimal notation.

POS 12 and 13 ; VAL X : This position stores the number of sectors used by variable length records. In the files I examined, all had less than 255 sectors, the number of sectors used (X) was repeated at the position 12. Even if this information uses two bytes, the bytes are put in reversed order. If the file would have been ABC sectors long, the value BC would have been located at 12 and the value ØA at 13.

That's it for the file description. And now for another trick: the last thing we must do is indicate the specific location of the file on the disk. This is done on the directory entry in the block link starting at position 1C (28).

The location of each continuous portion of a file uses three bytes. Lets take three of them as an example: AB, CD, EF found at position 1C, 1D, 1E. Say the D goes before the A and the C goes after the F, we now have the two values DAB and EFC which would mean that the starting sector of the file is DAB and the file continues for EFC additional sectors (fictitious of course). Since EFC means the number of the sector we will be at when reaching the last sector of the file, the size of the file would be EFD sectors since ØØ is the first sector.

To reverse the process, let's say that our file is 1Ø sectors long and starts at 35. The first value to take would be the starting sector Ø35 and the second would be ØØ9 because the 1Øth sector of the file would be sector number ØØ9 (starting at ØØØ). We now have to do the magic trick; take these two values and form the three bytes for directory entry. Since we have Ø35 and ØØ9, we take the last character of the second value which becomes the first of the second byte and the first character of the first value which becomes the last of the second byte; Ø35 and ØØ9 now becomes 35, 9Ø, ØØ that will be written at position 1C, 1D, and 1E.

Tricky isn't it; just like Bruce said, they must have been high on something when they decided it was going to be organized like that. If your file was not fractured, the writing part is over, you can skip what's following and go to the Read File section. If it's not the case the difficulty rises a little bit but is still sort of trivial. The same process described above will be applied to other portions of the file.

Since the value that goes with the starting sector of every portion of the file is the number of the sector we're at when reaching each last sector of the portions, we'll have to calculate that number for every portion with numbering starting at sector ØØ. As an example, lets say that our file is 15 sectors long, that is ØE sectors. If the first portion goes from sector 51 to 59, then the first sector will be at 51 and the offset will be Ø8 since when we'll reach the ninth sector of the file will be at sector number Ø8, ØØ being the first. If the second portion of the file is located from sector 72 to 77, the second starting sector will be 72 and this time, it's offset will be ØD. Even though ØD equals 14 in decimal it'll be the 15th sector of the file, still because ØØ is the first. It's quite a mind-boggler, but you'll get used to it after a while. So writing the first portion's location to the directory entry sector: Ø51 and ØØ8 would become 51 Ø8 and ØØ. The second portion's location would be, according to Ø72 and ØØD: 72 DØ and ØØ. That's all there is to it, if you have a third portion, then the same process is repeated.

READING THE FILE

When you've done all that, the last thing to do is to read the file and re-save it with a program similar to TI-Writer. This is important if you want to keep the file on disk for future use. Since we didn't write anything on the bit map of sector Ø, a disk manager program would recognize the file sectors as free and would write the next program you'd save over them. So re-saving it with TI-Writer would take care of writing on sector Ø which sectors are actually used by the file.

This article was presented for a specific purpose; if ever you would like more information on any of the particularities mentioned here, you could find it in Bruce Caron's articles or in the Advanced Diagnostics manual. If you would like more details on the process described, I'd be happy to try and help you. You can reach me through the TIOUG's BBS or in Hull at (819) 777-7686.

- At the last meeting a new service of the public domain program library was introduced and was well recieved by all members. This indication of acceptance has made the executive decide to offer a wide range of public domain software in the diskette and cassette format at a reasonable cost of three dollars each at every general meeting. At the last meeting all of the software packages on disk were sold but only one monthly cassette and no monthly diskettes were sold! I will keep producing the monthly diskettes until the end of the February meeting unless there is a demand for it. All new software titles will be announced at the beginning of the meeting to provide some clarity. Demonstrations of the software will be provided to those interested!

- Timeline has some new changes to the system! First there is a monthly minimum charge of $2 cdn to inactive accounts or to the use of less than $2 a month in order to keep your personal files active. The lifetime membership is going up to $25 cdn as of January one but old membership forms will still be honored with a postdate bearing the year 1985. The connect fee will now be 3ØØ bps at nine cents a minute and 12ØØ bps at ten cents a minute. New membership forms can be attained from me when time permits! The TCON function has now been expanded to handle 4Ø channels of teleconferences. Texas Instruments users have reserved channel 33 for weekly TCONs.

- (T.U.G)<(IAN.A15C1) Subject: System Sale
A TI-99/4A console, peripheral expansion box, TI disk controller card, TI memory expansion card, TI SSDD disk drive, and all manuals for $7ØØ cdn. All parties please contact the contributing editor if interested.

- (T.U.G)((LUIGI.A14FD) Subject: P System
A Pascal U.S.C.D System including the P Code Card, Editor/Filer, Compiler, Utilities, and all books for $2ØØ cdn. All parties please contact the contributing editor if interested.

- Jane LaFlamme, Vice Chairperson of the Ottawa T.I.99/4 Users' Group, recently announced on T.U.G they are sponsering a 1986 National TI Fest to be held in Ottawa on April 26. More details as they appear!

- I have come across a little gem of a program that all owners of a speech synthesizer and extended basic should be interested in.

```
1ØØ Rem *** Weird Sounds ***
11Ø Rem by David Huggett
12Ø Rem
13Ø Rem 9T9er Users Group, Toronto.
14Ø Rem The word INSTRUCTIONS in line 16Ø can be changed to any word in the resident vocabulary for different effects
15Ø For Y = 1 To 88 :: If Y = 4 Then Y = 7
16Ø Call Spget("INSTRUCTIONS",D$)
17Ø D = Len(D$) :: Print Y
18Ø D$ = Seg$(D$,1,2) & Chr$(D) & Seg$(D$,Y,D)
19Ø For X = 1 To 6 :: Call Say(,D$) :: Next X :: Next Y
```

- (Sher-TI-Bune)((Berry Minuk) Subject: Forth
De temps en temps, des gens demandent des questions concernant les differentes versions de FORTH. Malgre qu'il y ait plusiurs differentes versions du BASIC, il n'y a que 2 versions principales de FORTH.
Pour le TI, il existe deux formes differentes de FORTH quoiqu'ils sont semblables. Le premier a sortir fut le WYCOVE FORTH version 1, et maintenant le version 2 amelioree. Et bein sur, le TI FORTH. Les deux sont bases sur le 'fig-forth' souvent appele "78-FORTH" qui fut donne public par le FORTH INTEREST GROUP, d'ou le nom, fig-forth. D'autres versions furent promulges par le FORTH STANDARDS COMMITEE, 79,83...
Il contient alors des additions a la version 'fig'. Et il est plutot offert par des vendeurs prives pour les plus gros ordinateurs.
Mais puisque FORTH est un language extensible, il est possible d'ajouter tous les mots dont vous auriez de besoin. Et on peut alors composer un programme de traduction qui nous permettrait d'utiliser une autre version que la notre (fig) tel le 79 ou autre.

- (Popular Computing)((Les Cowan and Larry McClain) Subject: Talking With Your Computer
The NBC television show *Knight Rider* has risen to the top of the Nielsen ratings because of a simple premise: that the real achievers of the 198Øs are the people who can converse with their computers. The computer that controls the dashing Mr. Knight's all-but-indestructable car is a marvel, all right: it has enough memory and intelligence to comprehend its owner's every spoken command. And after analyzing the input, the computer utters its responses in what can only be described as a mellifluous voice.
But *Knight Rider* is televised fiction. Those astounding speech-synthesis and voice-recognition features won't be standard on next year's cars or home computers. In the here and now, most speech synthesizers still sound like Robby the Robot or the telephone company's canned messages, such as "The (pause) number (pause) you (pause) dialed (pause) has (long pause) been changed." Similarly, today's speech-recognition software is still in the todler stage. Some systems can recognize simple commands like "run" and "print," but no product on the market can decipher sentences like "Yeah, this is Fritz Mondale--give me all your files or John Glenn, will ya, pal?"
In this article we'll examine the principles involved in speech synthesis and recognition and look at some speech-related hardware and software currently available for personal computers. Although still in its infancy, speech capability is proving genuinely useful in many business and educational applications.
Most people have too much fun talking and singing to even think about the physiological aspects of how words are created, much less converted to their digital equivalents. And the physical process is indeed complex. Audible communication--whether it's a baby's first word or a majestic aria--involves far more than sheer lung-power.
As air is pushed by the diaphragm through what linguists call the "vocal tract," sound resonates through the chest cavity, throat, and mouth. The difference between voices is attributable to three variables of sound: volume, frequency, and timbre.
Volume is self-explanatory (unless you've never noticed that baseball's Billy Martin bellows a bit louder than Alistair Cooke). The basic frequency of a person's voice is usualy called "pitch," which simply means that Linda Ronstadt's vocal cords vibrate considerably faster than Neil Diamond's do. Timbre--just a fancy term for vocal tone--results from the sonic resonances within a person's chest and throat. When the word-to-be bounces around in your vocal tract, secondary frequencies (usually called overtones) are also created that give the spoken word its tone and texture. Timbre, not pitch, determines that Andy Rooney's nasal voice is better suited to *6Ø Minutes* than to a classical FM station.
The smallest unit of speech is the *phoneme*--the short "a" in cat, the "ch" sound in chocolate, and so on. Speech software denoting the word "fire" would break the word into four phonemes: F consonant, short A vowel, long E vowel, and a guttural R consonant.
When you use a microphone to enter words into a computer's speech-storing system, the input gets converted into an electric current that can be displayed as a *waveform* on the monitor. On screen, a waveform looks like a two-dimensional (sometimes three-dimensional) representation of a mountain range; each peak and valley in the waveform represents a different combination of frequency, volume, and timbre.
Computers use two basic approaches to store speech digitally. One method, called time-domain synthesis, involves a rapid sampling of the data as it's being input. Sampling is a process in which special circuitry in the computer checks the strength, or *amplitude*, of the electrical signal--that is, the peaks and valleys of the waveform--at regular intervals. This information is then converted to Øs and 1s by an analog-to-digital converter and stored on floppy disk. To replicate the sound, you access those numbers in memory and create another signal that gets turned back into a spoken command via a digital-to-analog converter.
This kind of rapid sampling demands a sampling rate twice that of the highest frequency of speech signal to avoid losing crucial data. In other words, because 4ØØØ Hz (hertz) is the highest frequency needed for speech storage, the time-domain approach requires at least 8ØØØ samples per second. The procedure gobbles up memory voraciously: 8ØØØ bytes

per second of stored speech.

If you need to be more frugal with computer memory, a linear-predictive coding (LPC) system is far preferable to a time-domain synthesis system. The LPC approach uses what's sometimes called "memory compression," a means of sampling only selected bits, skipping the rest, so that memory space is conserved. The missing data gets filled in by computer-generated estimates--predictions, if you will--of the amplitude represented by the various bits. The mathematical coefficients needed to concoct these estimates are stored on special ROM (read-only memory) chips. The eventual playback can't match Rich Little in the art of mimicry, but the tradeoff is memory conservation: the LPC method consumes only about 100 bytes per second of stored speech.

Strictly speaking, both time-domain synthesis and LPC are not speech-synthesis procedures per se, merely speech-storage procedures. Cheaper still--not to mention stingier with memory--are systems that do away with microphones and voice input altogether. Products like the Votrax Type 'n' Talk are keyboard-based. You type the letter approximating the phoneme you want to hear, the program quickly fetches a block of memory containing a text-to-speech algorithm. After digital-to-analog conversion, the speech produced is jerky and mechanical. But the real beauty of these bona fide speech-synthesis systems is that they eat up only about 10 bytes of memory per second of synthe-babble.

In 1973, rock singer David Bowie had a hit song called "Space Oddity" in which the protagonist, an astronaut many years in the future, goes half mad from isolation. The song's only oversight was that computer technology will eventually provide interactive talking gizmos as chatmates for space explorers.

Alas, computer owners who'd like to turn their machines into ersatz Dick Cavetts will have to be patient. Although we do have voice-recognition systems now that let people call up spreadsheet files a continent away to vocally change a cell's data, the days of human-to-computer repartee are several decades away.

Sue Charbonneau, a linguist and computer scientist at American Microsystems Inc. of Santa Clara, California, explains why she doesn't think that the dawning of full-scale speech recognition is imminent. "The day is very far away," she maintains, "when computers can recognize every nuance of what you say. That's due mostly to the way a sound changes in the context of a sentence. For example, we don't know how to program a computer to tell the difference between 'this guy' and 'the sky' which sound exactly the same when most people pronounce them."

Victor Zue, assistant professor of electrical engineering and computer science at MIT and an expert in speech reproduction and recognition, agrees. "Developing true speech recognition will take decades," he says.

However, Dr. Zue thinks that the next couple of years will see a blossoming of useful, albeit limited, speech applications involving small vocabularies. "These will encompass almost any kind of numerical transaction," he predicts, "as with the cardless teller machines where you speak an I.D. code into the computer and say how much money you want."

The speech-recognition products to which Dr. Zue alludes come in two main varieties: speaker-dependent and speaker-independent. In the former category, the user creates a vocabulary by speaking into a microphone and making several "passes"--repeating each word a few times so that the computer can register variations in his or her voice. The passes create what's known as a *template*, the speech pattern against which the computer compares al future voice commands. The quality of the user's microphone and the noisiness of the room at template-making time significantly affect how accurately a speaker-dependent system works. Most manufactures claim an accuracy rate better than 90 percent: if you tell the system "run" or "list," more than 9 times out of 10 it will recognize your voice and follow the instruction.

Speaker-dependent systems, though, have a limited usefulness. The vocabularies of such systems rarely exceed 100 to 200 words, and there's always the 10 percent chance that a head cold will prevent the computer from recognizing its master's voice. The main drawback, however, is that entire departments cannot use these systems--they respond to just the one employee who trained the system and created the template.

Speaker-independent systems boast a slightly higher recognition rate. Many claim to have 95 percent accuracy rates because their templates cover everything from a Mississippi drawl to a Cape Cod lockjaw reminiscent of Ted Kennedy's. But the main shortcoming of a speaker-independent system is its paltry vocabulary. In some cases, the template consists of only the words yes and no and the numbers 0 through 9.

Mountain Computer of Scotts Valley, California, is the leading manufacturer of speech-storage hardware for the nation's schools. Mountain makes a board called Supertalker SD 200 for the Apple II, II Plus, and IIe ($199) and a board called Supertalker II for the IBM Personal Computer ($565); the IBM PC board carries 32K bytes of extra RAM (random-access memory). Bob Bryne, a Mountain Computer customer service representative, describes the method of memory compression used in these boards as "slope modulation." When the slope of the waveform being sampled is rising, a 1 bit is stored; when the slope is falling, a 0 bit gets stored. The high sampling rate of 2K bytes per second, however, limits the duration of sound that can fit in a typical micro's memory. For instance, 16K bytes of free RAM can hold only 8 seconds of speech.

The Mountain Computer boards have their own ROM chips that hold routines through which the speech capabilities can be accessed by any software. One company already tapping into the Mountain-compatible market is American Educational Computers of Palo Alto, California, which markets a language-skills system called Micro-Read. After the computer pronounces a vowel sound and displays words in which that vowel appears, the student must identify those words in which the vowel is actually pronounced. For example, the computer might say the sound "e," as in beet, and show the words seek and hole. Seek would be a correct answer, hole would not.

The Mountain boards have also been used in an educational project called Writing to Read, which is based on the premise that students will learn to read more easily if they are first taught to write. Using 300 IBM PCs equipped with Supertalker II, the project has dramatically improved the students' reading scores.

These boards have also been put to more personal applications. According to Debbie Aldridge, marketing administrator at Mountain Computer, one customer records bird calls on a portable tape recorder and then digitizes and stores them in his database along with the more traditional details of a siting, such as date, time, location, and the color and markings of the bird.

Until it recently pulled out of the home computer market, Texas Instruments led the pack in speech synthesis with add-ons for the TI 99/4A. Although the company will no longer manufacture the 99/4A, its speech-related peripherals will continue to be sold.

The Speech Synthesizer attachment costs $99.95 and provides a vocabulary of approximately 350 words. With appropriate software, the system can be programmed to speak these words in any order you desire. Furthermore, by using the TI Terminal Emulator software, the 99/4A can be programmed to speak with an unlimited vocabulary.

One use to which the TI Speech Synthesizer can be put is to teach children spelling. A TI program called Speak & Spell pronounces a word, the computer checks the spelling. Speak & Spell also lets the child play a number of spelling games such as Hangman.

TI expects to add simple speech recognition to the 99/4A by year's end with the MBX voice-recognition device. The MBX will be manufactured by Milton Bradley, the toy maker, and marketed by TI. Consisting of a 64-position membrane keypad, a microphone, and a joystick, the MBX will plug into the computer's joystick port. Dave Leonnig, of TI public relations, says that with this peripheral, users will be able to issue simple voice instructions to some TI software, including games and educational applications. The MBX will cost about $13Ø.

Most businesspeople have two things in mind when they purchase hardware and software for voice recognition: maximizing the efficiency of workers involved primarily in hands-on tasks and accessing or creating files from remote locations. Votan's Inc.'s V8ØØØ and Supersoft's Voicedrive (used in conjunction with a Tecmar Voice Recognition board) are representative of the latest group of voice-recognition products geared to the business and scientific communities.

About the size of a normal disk drive, the V8ØØØ can be used with any computer--micro or mini--equipped with an RS-232C interface. The unit contains its own microphone, speaker, and circuitry. Despite its $9ØØØ price tag, the V8ØØØ can be especially useful in businesses with employees involved in hands-on work. One California company, for example, uses V8ØØØ to issue voice commands to a computer that drives both an X-ray spectrometer and an electron microscope. When an employee sits hunched over the microscope in a darkened lab, he doesn't have time to answer a phone or type in keyboard commands to control the equipment.

Voicedrive, speech recognition software for the IBM PC, is tailor-made for the businessperson who always seems to be two or three time zones away from vital spreadsheet data. Voicedrive and its companion hardware, the Tecmar Voice Recognition board, sell together for $995 and enable the distant businessperson to access and operate the Scratchpad spreadsheet program verbally. Essentially, the jet-lagged executive places the phone call, pulls a file by voice, vocally transmits the numbers to be entered in various cells, and waits for the computer to spit back a synthesized answer.

Another product designed for excutives is the Texas Instruments Speech Command system for the TI Professional Computer. This $26ØØ system includes a plugin board as well as software that lets a user set up speaker-dependent voice recognition for entering commands. This "transparent keyboard" can be used with any software to set up multiple vocabularies of 5Ø words each, in which each word replaces up to 4Ø keystrokes. The number of vocabularies is limited by the system's memory. The system also allows the computer to record incoming phone messages, dial numbers, deliver outgoing phone messages, and play back messages from a remote telephone. The speech system can store up to 16 minutes of speech on a standard 32ØK-byte disk or up to 4 hours on a 5-megabyte hard disk.

Edward Jacklitch, director of marketing for Centigram Corporation's voice-output products, feels that the slowly evolving speech-recognition market is inhibiting sales of speech-storage and speech-synthesis devices. "State-of-the-art speech recognition is lagging far behind speech reproduction," he notes, "and many would-be buyers feel that one application isn't useful without the other." Moreover, Jacklitch stresses that many businesspeople ignore speech-synthesis developments because of the trivial applications they've observed (such as the American-built luxury cars that pipe up and let the driver know that all systems are functional). "Talking cars are a gimmick that few people take seriously," says Jacklitch.

Another objection to speech reproduction and recognition is that English, a language already in disrepair, will decay even faster once computers get involved. Tecmar's Val Matula scoffs at the idea. "I've found that my speech has become *more* precise," he says. "After you use this board for a while, you won't say 'uh' very often between words because you'll know what it sounds like in playback. I honestly don't think machines will cause human speech to become stilted."

It's likely that speech synthesis and speech recognition won't truly catch on until computer users realize low blessedly interruptible these systems are. Unlike an irate umpire or chattering fiver-year-old, a computer can be quickly silenced.

Thomas Edison never touched an integrated circuit, but his marvelous Victrola was society's first "voice reproducer." When a turn-of-the-century admirer fell all over himself praising Edison's contraption. Tom leaned back and said, "Friend, I didn't invent the first talking machine--just the first one that could be turned off."

```
1ØØ REM APR 84, DUMPTEST2 FO
R OKIDATA, TERRY ATKINSON, 2
8 SAVONA CT. DARTMOUTH, NS.
B2W 4R1 PH 434-3121 OR 434-1
346
11Ø DIM K(112,8),B(1,7),MSB(
32,8):: SP=24
12Ø FOR I=1 TO 8 :: K(Ø,I)=2
55 :: NEXT I
13Ø FOR I=2 TO 112 :: K(I,1)
=-1 :: NEXT I
14Ø HEX$="123456789ABCDEF" :
: BLANK$="ØØØØØØØØØ[illegible]"
:: FF$="FFFFFFFFFFFF[illegible]"
15Ø OPEN #1:"PIO.CR.LF"
16Ø PRINT #1:CHR$(13)&CHR$(1
Ø)&CHR$(1Ø)&CHR$(1Ø)&CHR$(3)
;
17Ø FOR ROW=1 TO 24
18Ø PRINT #1:CHR$(3)&CHR$(2)
&CHR$(13)&CHR$(27)&[illegible]37)&
CHR$(57)&CHR$(2)&RPT[illegible] ",SP
)&CHR$(3);
19Ø FOR COL=1 TO 32
2ØØ CALL GCHAR(ROW,COL,X)::
IF X>32 THEN Q=X ELSE Q=95
21Ø CALL SPRITE(#1,Q,16,ROW*
8-7,COL*8-7)
22Ø X=X-31 :: IF X<1 THEN X=
1 :: GOSUB 52Ø :: GOTO 42Ø
23Ø IF K(X,1)<>-1 THEN GOSUB
 52Ø :: GOTO 42Ø ELSE CALL C
HARPAT(X+31,KAR$)
24Ø IF KAR$=BLANK$ THEN X=1
:: GOSUB 52Ø :: GOTO 42Ø
25Ø IF KAR$=FF$ THEN X=Ø ::
GOSUB 52Ø :: GOTO 42Ø
26Ø BYTE=7
27Ø FOR A=15 TO 1 STEP -2
28Ø B(Ø,BYTE)=POS(HEX$,SEG$(
KAR$,A,1),1)
29Ø B(1,BYTE)=POS(HEX$,SEG$(
KAR$,A+1,1),1)
3ØØ BYTE=BYTE-1
31Ø NEXT A
32Ø FOR D=Ø TO 1
33Ø FOR E=3 TO Ø STEP -1
34Ø FOR F=Ø TO 7
35Ø IF (B(D,F)AND 2^E)THEN H
=H+2^F
36Ø NEXT F
37Ø Y=Y+1 :: K(X,Y)=H :: H=Ø
38Ø NEXT E
39Ø NEXT D
4ØØ Y=Ø
41Ø GOSUB 52Ø
42Ø FOR Q=1 TO 8 :: IF K(X,Q
)=3 THEN PRINT #1:CHR$(3);CH
R$(3);ELSE PRINT #1:CHR$(K(X
,Q));
43Ø NEXT Q
44Ø NEXT COL
45Ø PRINT #1:CHR$(3)&CHR$(2)
&CHR$(13)&CHR$(27)&CHR$(37)&
CHR$(57)&CHR$(15)&RPT$(" ",S
P)&CHR$(3);
46Ø FOR CO=1 TO 32 :: FOR I=
1 TO 8
47Ø PRINT #1:CHR$(MSB(CO,I))
;
48Ø NEXT I :: NEXT CO
49Ø NEXT ROW
5ØØ PRINT #1:CHR$(3);CHR$(2)
&CHR$(1Ø)&CHR$(1Ø)&CHR$(1Ø)&
CHR$(13):: CALL SOUND(1ØØØ,5
23,Ø):: CLOSE #1 :: END
51Ø REM CONVERT LOWER BYTE T
O NEW CHAR IN BINARY
52Ø FOR I=1 TO 8
53Ø IF K(X,I)>127 THEN MSB(C
OL,I)=1 :: GOTO 55Ø
54Ø MSB(COL,I)=Ø
55Ø NEXT I
56Ø RETURN
```

TI99/4A Machine language program deloader copies program in computer memory back to disc in a format useable by the E/A module. Send $10.00 to: **Kings Marketing**, P.O.Box 591, Station "J", Calgary, Alberta, Canada TZA 4X8. 11-71

COMPUTER SHOPPER, Users Groups

TI99/4A EXPANDED: Box 128K Ram, complete Pascal System including diskettes, XBasic, disk manager, multiplan TI writer, RS232, 100s/programs, disk drive, controller, more $800. **Gardner Controls Inc.,** 66 Goodturn Rd., Levittown, PA 19057. (215)946-8004. 12-72

MANITOBA
WINNIPEG 99/4A USERS GROUP, 1105 CHURCH AVENUE
WINNIPEG, MANITOBA CANADA R2X-1G1

TI99 4A-12 Cartridges, 50 other programs on tape, speech synthesizer, manuals $119. **C.L. Tomek,** 1141 Columbian, Oak Park, IL 60302. 312-848-0353.

PROGRAMMING HELP FILE:

The purpose of this column is to present, to the user, techniques that will be useful in the writing of programs for the TI-99/4A home computer. As not all readers will have the same programming skills I will present the material at a fundamental level but in a way that tries to be stimulating to the more experienced. Not everyone has knowledge about assembly language and I hope that what is presented here will get some of those people started with it. I hope that there is something, in what follows, for everyone. If you can provide some prgramming insight that might be useful to someone, please, feel free to pass it on to me, and I'll get it into the next newsletter.

BASIC/EX-BASIC:

Character strings have got to one of the more tricky items that one has to deal with when programming. Text is used throughout a program to inform or to prompt for a response. Last month we had a sample of how text can be used in the input and output statements. This month my discussion will take a closer look at character strings, character string variables and functions that operate on character strings and string variables.

What is a character? A character, in computer terminology, is a symbol that is represented inside a computer by a byte value (8 bits). Common symbols that are used as characters are the letters of the alphabet (both upper and lowercase), punctuation and monetary symbols, numbersand arithmetic operators. The space or blank character must also be represented as a byte value. Other single byte characters that are used in computers are used to control spacing on printers or in communications, and to signal various devices. The carriage return and linefeed are two very common characters used in communicating with computer devices.

A byte value is associated with a character symbol, in the TI99/4A, by what is known as the American Standard Code for Information Interchange or ASCII representation. Since a byte can have a numeric range of 0-255 each of these 256 byte values may represent a different character. Because a computer works with bytes and not the actual symbols that an ASCII values may represent, but must display the symbols for human interpretation, the computer must have some means of transforming the internal ASCII value of a character into a graphic image on your computer screen. You may, if you wish, change a graphic image (symbol) associated with a particular ASCII value to another character image, but I will leave the discussion of that exercise till another newsletter.

So its not too much more difficult to understand, now, that a character string is merely a series of characters that follow one another. A character string, inside a computer, would be a number of consecutive bytes which contain ASCII values. Right now we are not worried much about the internal representation as we are about how strings are represented in TI BASIC/XBASIC.

In BASIC/XBASIC character strings must be enclosed in quotation marks.
ie. "this is a character string!"
All symbols within the quotes are then part of the character string. A character string variable is used to hold a character string, and is different from a numeric variable in that the last symbol of the variable name is a '$'. eg. STRING$, LETR$, M$. Here are a few examples of how strings and string variables are used.

```
     100  A$="hello"
     110  B$="how are you?"
     120  PRINT "this is a string test"
     130  PRINT A$,B$:A$;B$
     140  PRINT "mix of strings and string variables."
     150  PRINT A$;", ";B$:".<>,:;-/+=)(*&^%$#@!|~[]_?"
     160  PRINT "end of string test."
```

If you try above example you will see that only what is within the quotes of a string is printed and that where a PRINT statement contains a string variable only the exact value of the string assigned to the variable is printed.

A character string must not be more than 255 characters long since only one byte is used to keep track of a string's length. The LEN() function will return the number of characters that are in a character string or assigned to a string variable.

eg. LEN("hello") returns then number 5 since "hello" is five characters long and LEN(V$) would return the length of the string assigned to V$.

There are some other usefull functions that assist in converting an ASCII value to a single character, and a character to its ASCII value. You can also change numbers to strings and strings consisting of numbers to number values (8 byte internal number). Since numeric values cannot be assigned to string variables and strings cannot be assigned to numeric variables these functions are often usefull.

ASC() is used to get the ASCII value of a single character. CHR$() given an ASCII value will return a string character. STR$() given a numeric value will return a character string where each character is a digit of the number. VAL() given a character string consiting only of numerics will return the number value.

Other operations that can be done with strings and that are useful when you whant to change a character string in some way are the SEG$() function and the concatenation operator, '&'.

The concatenation operator joins two strings together to make a longer string. eg. A$="hello! "&"how are you?" , and A$ when printed will give the string, "hello! how are you?". Notice that a blank was placed after the '!' in the string "hello! " so that the joined strings would be readable. The 'null' string or zero length character string ("") is often used to start a concatention. eg. A$="", A$=A$&B$.

SEG$() is used to extract parts of a string, usually for the purpose of deleting characters, making the original string shorter or building a different string. eg. SEG$("hello hi bye goodbye",1,5) will return 5 characters starting from the first character of the string "hello hi bye goodbye". Usefull if you want a choice of greetings in your program. With some experimentation you can build all kinds of strings and even have your computer generate some random sentences or poetry.

RPT$() is also usefull at times. This function allows you to repeat a character string a specified number of times. eg RPT$("*",42) will return a string that consists of 42 "*"'s. RPT$("XO",6) will return a string that is 12 characters long and consisting of alternating Xs and Os.

A final string function is POS(). This function is usefull when you want to find characters or a sequence of characters within a string, it works as follows. eg. POS("arsonist","son",1), the second string "son" is compared with characters of the first string, "arsonist", starting at the position of character 1. If the string "son" is found in the string "arsonist" then the position of the first character of the first match will be returned. If no match is made then a zero value will be returned. Functions like this are usefull in word processing programs.

Thats the end of my discussion about character strings. The best way to learn more about these functions and using characters strings is to sit down at your computer and try some experiments. Who knows what stories you can get your computer to tell.

ASSEMBLY:

The following tutorial comes from the MAD HUG newsletter, which was taken from Lehigh 99'ers December newsletter. It should be short enough for most to enter and assemble.

From Lehigh 99'er, December, 1985

LEHIGH 99'ER

```
(XXXXXXXXXXXXXXXXXXXX)
()                  ()
()  BEGINNERS ASSEMBLY  ()
()       Part I       ()
()                  ()
(XXXXXXXXXXXXXXXXXXXX)
```

```
****************************************
*                                      *
* THIS IS THE FIRST PROGRAM FOR THE    *
* BEGINNER ASSEMBLER TUTORIAL.         *
* IT CLEARS THE SCREEN, ADDS TWO NUMBERS *
* TOGETHER AND DISPLAYS THE SUM IN THE *
* CENTER OF THE SCREEN.                *
* HERE'S WHAT IT WOULD LOOK LIKE IN    *
* EXTENDED BASIC:                      *
* 10 CALL CLEAR                        *
* 20 X=10                              *
* 30 Y=27                              *
* 40 X=X+Y                             *
* 50 DISPLAY AT(12,15):X               *
* 60 END                               *
****************************************

* THIS PART OF THE PROGRAM IS THE INITIALIZATION *

       DEF  START  THE PROGRAM NAME IS START
       REF  VSBW,VMBW CONSOLE ROUTINES WE ARE GOING TO USE
WSREG  BSS  >20    SETS ASIDE A BLOCK OF 32 BYTES FOR USE AS WORKSPACE REGISTERS
X      DATA 10     COULD HAVE USED >A INSTEAD OF 10 (LIKE X=10)
Y      DATA 27     (Y=27) COULD HAVE ALSO SAID Y EQU 0027
TEN    DATA 10
ANS    DATA 0        WORD TO PUT ANSWER IN. INIT TO 0.

* PROGRAM BEGINS HERE
*---------------------------*
START  LWPI WSREG    LOAD WORKSPACE POINTER IMMEDIATE. POINT TO OUR WORKSPACE.

* CLEAR THE SCREEN
       CLR  R0       CLEARS R0 TO ZERO (BEGINNING OF SCREEN IMAGE TABLE)
       LI   R1,>2000 LOAD IMMEDIATE R1 WITH >2000. VSBW ROUTINE WRITES THE LEFT
LOOP   BLWP @VSBW    BYTE IN R1 ALWAYS. IN THIS CASE >20 OR 32 OR SPACE CHR
       INC  R0       ADD 1 TO R0
       CI   R0,767   COMPARE IMMEDIATE R0 TO 767
       JLE  LOOP     IF ITS LESS THAN OR EQUAL JMP (GOTO) LOOP

* ADD THE NUMBERS TOGETHER AND CONVERT TO ASCII
*---------------------------*
       A    @Y,@X    ADDS X TO Y AND PLACES RESULT IN X
       MOV  @X,R6    MOVE WHATS AT X TO R6
       CLR  R5       CLEAR R5
       DIV  @TEN,R5  DIVIDES 10 INTO 37. QUOTIENT IN R5. REMAINDER R6.
       AI   R6,>30   ADD IMMEDIATE ASCII OFFSET TO R6
       MOV  R6,@ANS  MOVE CONTENTS OF R6 TO THE WORD ANS
       MOV  R5,R6    MOVE CONTENTS OF R5 TO R6
       CLR  R5       CLEAR R5
       DIV  @TEN,R5  DIVIDE 10 INTO R5, R6
       AI   R6,>30   ADD IMMEDIATE ASCII OFFSET >30 TO R6
       SLA  R6,8     SHIFT LEFT ARITHMETIC R6 8 BITS.
       MOVB R6,@ANS  MOVE MSBYTE R6 TO R1
  --------------------------*

* DISPLAY ON THE SCREEN AT ROW 12 COLUMN 15

       LI   R0,366   POSITION ON THE SCREEN IS 366
       LI   R1,ANS   LOAD R1 WITH THE ADDRESS OF ANS
       LI   R2,2     TWO BYTES TO WRITE
       BLWP @VMBW

       JMP  $        PREVENTS THE PROGRAM FORM ENDING SO YOU MAY SEE THE RESULT

*---------------------------*
* RETURN TO THE CALLING PROGRAM

       CLR  @>837C   CLEAR THE STATUS BYTE
       LWPI >83E0    LOAD GPL WORKSPACE REGISTERS
       B    @>0070   BRANCH TO THE CALLING PROGRAM
       END
```

FORTH:

This section is postponed till next month.

PASTIMES & PUZZLES

Solution to December:
who is the Engineer? Smith.

	SMITH	JONES	ROBINSON
BRAKEMAN	NO	YES	NO
FIREMAN	NO	NO	YES
ENGINEER	YES	NO	NO

	Mr. Smith	Mr. Jones	Mr. Robinson
CALGARY	No	YES	No
TORONTO	No	No	YES
BETWEEN	YES	No	NO

Sorry! No puzzle this month. I was slow in getting started this issue.

```
FILE:  SWAP SELL
DATE:  1/[illegible]86
TITLE: SWAP SELL

  INDEX
  0 = PAGE #
  1 = ITEM
  2 = TYPE
  3 = DSCRPTION
  4 = PRICE
  5 = PHONE


 0        1                2             3              4        5

-- ---------------- ----------- ---------------- ------- ---------
 1 BEST SOFTWARE    CASSETTE    SOFTWARE         $10.00 586-6889
 2 BRISTOL CSSTT    PLAYER      HARDWARE         $20.00 586-6889
 3 CAR WARS         MODULE      SOFTWARE         $20.00 668-1781
 4 DOW-4 GAZELLE    CASSETTE    SOFTWARE         $20.00 586-6889
 5 EGYPT GRPH ADV   CASSETTE    SOFTWARE         $10.00 586-6889
 6 HUNT THE WUMPUS  MODULE      SOFTWARE         $20.00 668-1781
 7 MDVL GRPH ADV    CASSETTE    SOFTWARE         $10.00 586-6889
 8 PUBLICATIONS     MAGAZINES   BACK ISSUES ECH  $1.00  586-6889
 9 SIGNALMAN MARK3  MODEM       HARDWARE         $100.0 586-6889
10 SUNDIAL ISL 1&2  CASSETTE    SOFTWARE         $10.00 586-6889
11 TEACH SELF XBAS  CASSETTE    SOFTWARE         $15.00 668-4804
12 TEACH SELF XBAS  CASSETTE    SOFTWARE         $18.00 888-1346
13 TEACH SELF XBAS  CASSETTE    SOFTWARE         $20.00 668-1781
14 TEACHSELF BASIC  CASSETTE    SOFTWARE         $17.00 668-4804
15 TEACHSELF BASIC  CASSETTE    SOFTWARE         $20.00 632-4987
16 TI FORTH PKG     DISK        SOFTWARE         $40.00 895-7067
17 TREASURE HUNT    DISK        SOFTWARE         $10.00 586-6889
18 WIZARD&PRINCESS  DISK        SOFTWARE         $30.00 586-6889
19 LAST WORD ON TI  BOOK        INFORMATION      $10.00 895-7067
20 DUAL CASSETTE    CABLES      HARDWARE         $18.00 668-4804
21 TI-99/4A         COMPUTER    HARDWARE         $60.00 669-4804
22 MINI-MEMORY      MODULE      SOFTWARE         $30.00 668-1781
23 TI LOGO II       MODULE      SOFTWARE         $90.00 489-3127
24 ADVENTURE        MODULE      SOFTWARE         $55.00 489-3172
25 TI SYSTEM        UNIT        HARDWARE         $775.  489-3172
```

13

[illegible]

NEWSLETTER EDITOR
WINNIPEG 99/4 USERS GROUP
P.O.B. 1715
WINNIPEG, MANITOBA
[illegible]

DAVID WOOD
54 SANDHAM CRES
WINNIPEG MAN
R3R 1M7

EDMONTON TIERS
PO BOX 11983
EDMONTON ALBERTA
T5J 3L1