

YESTERDAY'S NEWS

VOLUME 3 NUMBER 12 Established 2016

DECEMBER 2018

30 Years Ago...



Historical Information taken from Bill Gaskills TIMELINE DECEMBER 1988:

In a teleconference on Genie's TI-SIG, Asgard Software owner Chris Bobbitt outlines the features of the new Press word processor from Charles Earl. Features include;

- o on-screen formatting of text.
- o document size limited only by disk space as virtual memory is used.
- o on the fly spell checking.
- o multiple column layouts.
- o printer drivers.
- o a 4K Keyboard buffer.
- o multiline headers and footers.
- o macro support.
- o un-delete and undo features.
- o text blocking with inverse video.

In the News & Views Column of the December 1988 issue of the Western New York 99ers' Interface newsletter, columnist and newsletter editor Harry Thomas Brashear writes the following about Asgard's Press word processor,

"The Chicago fair for 1988 is history. Something over 500 people paid to attend the show and saw thirty-two vendors hawking their products. Most of the products were "old hat" and a couple that were supposed to make big splashes never showed. Darn !

Never the less, Asgard had a working copy of Press to show off and took orders by the gross. Being quite close to that product, I can tell you that it will be here soon. The problem is that C. E. (programmer Charles Earl) got sick in the middle of things and his equipment screwed up besides. The last hangup was due to a bug in the Myarc disk controller. Press will be on your doorstep by December, I promise."

TI-Base v2.0 is released on December 15th, a letter from Insebot Inc's Dennis Faherty dated 12/7/88 notifies users of a patch required to fix a bug found in the RECALL directive.

INSIDE	INFORMATION
ELEMENTS OF BASIC #14	Page 1
SPACE STATION I	Page 2
DIABLO	Page 4
XB DETECTIVE	Page 4
PROGRAM COMPACTOR	Page 5

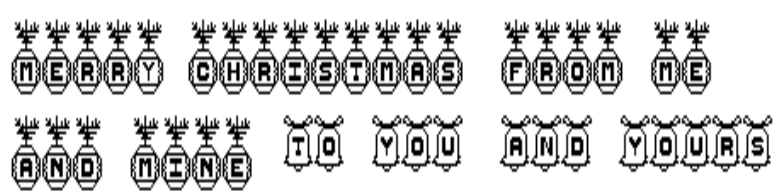
Myarc announces that work on the tape streamer back up software for the HFDC hard disk controller is scheduled to begin. John Birdwell, author of the MDMS Disk Manager for the HFDC, will be the programmer for the project.

Eric Zeno of Pittsburgh, Pennsylvania announces the Internal Board , a board designed to be used inside the 4/A console, that will provide 32K, a clock and more. It is ultimately dubbed the Zeno Board when it goes into production in 1989.

Donaldson Software, of Quebec, Canada, announces several new cassette games for the 99/4A, including;

- Swordfish Patrol
- Monster Castle
- War of the Netherworlds
- Sun and Planets
- Speed Rally
- TI Pac-Man

Dick Altman turns the management of his Fairware List over to Steve Mehr of Thousand Oaks, California.





ELEMENTS OF BASIC

By DAVE HOWELL

COURTESY OF THE EARLY 99'ERS

PART 14

FUNCTIONS

It has often been said that mathematics and computers go hand in hand. Some may argue that one doesn't have to understand mathematics or the physical sciences to master the use of computers or write programs. In fact, the experts are now saying that knowledge of programming is no longer necessary to operate a computer system or to use the huge inventory of software now available to the consumer.

Historically, however, computing devices have long been used to speed up complicated and tedious mathematical calculations required in business and modern technology. Because of this, most of the older programming languages provide for special calculations found in math and science. Such special calculations are known as FUNCTIONS in BASIC. Functions that are included in TI BASIC include logarithms, square roots, random numbers, integers, sines, cosines, tangents, etc.

INTEGERS and RANDOM NUMBERS

A function in computer language may be thought of as a command or statement. The `INT(x)`, the integer function, where `x` is the argument or the number to be acted upon, is often used to convert decimals into whole numbers. By definition, an integer is a positive or negative whole number. It contains no fractions or decimals. The numbers 1,2,3,4 and 5 are integers but 2.349 and 14.9 are not. The `INT` function is used to chop off the fractional or decimal portion of the number, leaving just the whole number - in this case, the integers 2 and 14 respectively. This is how the function is used in a program:

```
10 X=45.3896
20 A=INT(X)
30 PRINT A
```

When this program is run, the screen will display the number 45, which is the integer of 45.3896. The decimal portion has been truncated or chopped off. The above program may be written in a shorter way:

```
10 PRINT INT(45.3896)
```

Incidentally, the number inside the parentheses is called the "argument." Most functions require an argument upon

which to act.

The `INTeger` function is often used to round off decimal numbers. For example, to round off 12.394617 to the nearest hundredths (2 decimal places), the following program may be used:

```
10 LET A=12.394617
20 PRINT INT(100*A+.5)/1000
```

First, the number is multiplied by 100, the resulting decimal is 1239.4617. Since 0.5 is the midpoint between 2 whole numbers, 0.5 is added producing 1239.9617 - not quite enough to reach the next whole number, 1240. Therefore, when the `INTeger` of 1239.4617 is found (1239) and the result divided by 1000, the number 12.39 becomes the final rounded number.

The same arrangement can be used to round off numbers to the nearest tenths or thousandths. Just multiply and divide respectively by 10 or 1000 in line 20.

Another function in TI BASIC is `RND` or random number. In its simplest form, `RND` provides a random decimal between 0 and 1. Enter and run this line:

```
10 PRINT RND
```

Run the line over and over again, several times. Note that the decimal produced is the same number each time the line is run. To be truly random, each decimal number should have been different. Now add this line and run the program again:

```
5 RANDOMIZE
```

When the `RANDOMIZE` statement is used at the beginning of a program using `RND` functions, the random number generator in the computer produces a series of numbers selected completely at random. There should be no detectable pattern in the numbers appearing on the screen - even if the program is re-run over and over again!

Decimals, however, are not the most convenient numbers to work with. Generating whole numbers is much more useful. The first step to generate a series of whole numbers, for example, from 0 to 9 is to multiply the `RND` by 10. Change line 10 in the above program to read as follows and add line 20:

```
10 PRINT 10*RND
20 GOTO 10
```

Run the program. Line 20 will cause the program to repeat printing out a series of random numbers. Use the `FCTN-4` Keys to stop the program. All the numbers appearing on the screen should fall between 0 and 9. Now, let's get rid of the unwieldy decimals. Retype line 10 as follows and run the program again:

```
10 PRINT INT(10*RND)
```

The INTEger function truncates (chops off) the decimals leaving whole numbers between 0 and 9. If a series of random numbers from 1 to 10 is desired, add +1 to line 10 so that the entire program reads as follows:

```
5 RANDOMIZE
10 PRINT INT(10*RND)+1
20 GOTO 10
```

Similarly, changing the 10 to 100 in line 10 will produce random numbers from 1 to 100. This combination of the random and integer functions has wide application in predicting out-comes, simulations and games of chance. The following program simulates the tossing of a coin 1000 times and prints the results.

```
10 RANDOMIZE
20 FOR X=1 TO 1000
30 LET A=(2*RND)+1
40 IF A<2 THEN 80
50 PRINT "HEADS ";
60 LET H=H+1
70 GOTO 100
80 PRINT "TAILS ";
90 LET T=T+1
100 NEXT X
110 PRINT "HEADS =" ;H;"TAILS
    =" ;T
```

Lines 10 and 30 generate random decimal numbers between 1 and 3 with 2 being the midway point. Line 40 picks out the numbers less than 2 and sends them to line 80 as "TAILS." The rest of the numbers go on to line 50 as "HEADS." Lines 60 and 90 keep track of the number of HEADS and TAILS passing through their respective segments of the program. Lines 20 and 100 repeat the process 1000 times. When it's all done, line 110 prints the number of respective flips accumulated by lines 60 and 90 - which should be nearly 50-50 split.

The random statement can be used to generate numbers for arithmetic drills. Consider the following multiplication drill:

```
5 RANDOMIZE
10 LET A=INT(100*RND)+1
20 LET B=INT(100*RND)+1
30 PRINT "WHAT IS";A;" *";B;
40 INPUT G
50 IF G=A*B THEN 80
60 PRINT "WRONG. TRY AGAIN!"
70 GOTO 30
80 PRINT "CORRECT! ^HERE'S A
    NOTHER."
90 GOTO 10
```

A very popular use of the random routine is gaming. The following game of Russian Roulette uses a six-shooter revolver containing one bullet. That's why line 100 checks to see if the RND number is less than 1 out of 6. If so, then the gun goes off in line 150. Lines 30, 110 and 130 keep track of the number of "clicks." When the number reaches six without a "bang", the player wins in line 180.

```
10 RANDOMIZE
20 PRINT "RUSSIAN ROULETTE"
30 LET N=0
40 PRINT "TYPE 1 TO SPIN, 0
    TO QUIT"
50 INPUT C
60 IF C=1 THEN 90
70 PRINT "CHICKEN!!!"
80 GOTO 230
90 PRINT "LOTSALUCK!"
100 IF RND<1/6 THEN 150
110 LET N=N+1
120 PRINT "--CLICK--"
130 IF N=6 THEN 180
140 GOTO 40
150 PRINT "BANG!! I'M SORRY.
    YOU LOSE!"
160 PRINT "NEXT VICTIM, PLEA
    SE."
170 GOTO 20
180 PRINT "YOU DID IT!! 6 MI
    SSES!--YOU WIN!"
190 FOR K=1 TO 6
200 PRINT "YEA! ";
210 NEXT K
220 GOTO 240
230 PRINT "GET SOMEONE WHO I
    SN'T SO SMART!"
240 END
```

SPACE STATION I DATA FORCE Dominic J. Melfi

99ER - VOLUME 1, NUMBER 5 REVIEW BY SAM PINCUS

The latest TI arcade games are written in 9900 Assembly Language, and demonstrate the tremendous speed that the TI-99/4A is really capable of. But Assembly Language Programming on the home computer is not the sole domain of TI, at least, not any longer, with two assemblers presently available and a third one on its way [Mini-memory]. I was therefore extremely interested in seeing the first Assembly Language game program written by someone outside Texas Instruments.

The game is called Space Station I and was written by Dominic J. Melfi. It is presently available on disk from Data Force Inc. for \$34.95, and because it's in Assembly Language, it requires Extended BASIC and the 32K Memory Expansion peripheral.

Loading the game is quite simple. Just place the disk in Drive 1 and turn on your TI-99/4A. The BASIC "boot" program is automatically called in; it in turn calls a special assembly load program. This special load program then loads the actual program.

Why all this rigamarole just to load the program? Because, as the author says, It wasn't fair to make anyone wait the 3 or 4 minutes that TI's load program would require for a program this size. How fast is Mr. Melfi's program loader? To find out, I ran a time comparison of the two

loaders. Using the Extended BASIC CALL LOAD routine written in GPL (Graphics Programming Language), it took 3 minutes and 25 seconds to load the game after the Extended BASIC program started. Using the special load routine, it took just 20 seconds from powerup to the start of the game! This is indicative of the pride and care taken by the author in providing the best possible product for the user.

The game disk comes with 5 pages of documentation. The first 3 pages present the space game's scenario-providing the history for the years 2000-2020, and then explaining what the purpose of the game is and why it reacts as it does. The last two pages explain how to play the game. The basic idea of the game is quite simple: Your job is to protect a space station which is shown revolving in the middle of your screen.

Missile-firing alien ships attack in groups of three. You have four torpedoes with which to shoot down the aliens, by using a joystick or the arrow keys on the keyboard to control a target cross-hair. Pressing the fire button or the ENTER key launches a torpedo. As soon as your torpedoes are used up, they are replenished. After each wave is destroyed, another wave of aliens attack. They come from various directions to the left, right, above and below the space station. In addition, a mother ship (usually invisible) runs across the top of the screen, and at certain times, it is subject to attack.

To tell you the truth, I skipped very quickly over the documentation, and instead went right to the game. This was a big mistake because I immediately got lost trying to follow the game. I suggest that you do read the documentation before you play. And read it carefully! It's quite possible to miss some important details that will affect your understanding of the game. This is really the only complaint I have with an otherwise enjoyable product. An improvement in documentation would definitely be in order.

The actual program is well thought out. For example, you press the ENTER key to start it up. At the same time, pressing the key has told the program whether you have the TI-99/4 or 99/4A. As soon as you are ready to go, just press the ENTER key again (if you are using the keyboard) or the fire button (if you use the joysticks). This starts the game and tells the computer whether to read the keyboard or joysticks for the rest of the game.

Once you get the hang of the game, one fact stands out: Assembly Language games are fast! The graphics are very smooth, and there is no hesitation no delays while the program is running. The game speed is even fast enough for you to lay down a spread of torpedoes. This is possible because you don't have to lift up your finger from the fire button for the program to obey the cross-hair movement commands. In TI BASIC, of course, you cannot get

your program to read two keys from the keyboard simultaneously.

The game becomes very intriguing as the pace speeds up. It develops the feel of a real arcade game. In fact, if I had better joysticks, this game would have become addictive to me. (Here at 99er headquarters, we noticed quite an improvement with both the new TI joysticks, and Atari joysticks with the Denali Data adapter Ed.) I also had a hard time controlling the direction keys when I used the keyboard instead of joysticks; it seemed to require more manual dexterity than I'm personally capable of especially when the action began to really speed up.

The graphic characters are well done. The game screen is slightly cluttered with all the combatants moving around i.e., the space station, alien attackers, mother ships, comets, target cross-hairs etc. It takes a while until you learn to ignore the unimportant clutter. Until you do it is rather difficult to react to the alien attackers quickly enough to survive very long.

To summarize, this product has some very good points. For one thing, it really shows what the 9900 Assembly Language in the hands of a good programmer is capable of doing. The graphics are well done and the program is well thought out. It has the speed needed to make this kind of game very challenging. The only real complaint I had was with the documentation. Overall, this is a very good arcade-type product. If you like these kind of games, give it a try and see what a joy it is to play an original game instead of just another rehash of Space Invaders.

COMPUTE! - August 1983

REVIEW BY TONY ROBERTS

Space Station I mixes the sprite movement and sound abilities of the TI-99/4A with an interesting space-attack scenario to produce a fluid and challenging arcade-quality game.

The program, available on disk or cassette from Data Force, requires that your TI be equipped with Extended BASIC and extra memory.

The action takes place in the year 2020. An invisible alien force has attacked and defeated a secret military outpost orbiting Saturn, and has turned its attention to Earth, which you must defend. The battle at Saturn, however, took its toll on the alien force, weakening its fire-power, damaging its tactical computers, and making its drones visible 99 percent of the time.

Once the battle began, the Saturn outpost lasted only 34 seconds, but during that time, the station's tactical defense computer was able to transmit information back to Earth. The computer's report, which is printed in the instruction pamphlet, includes clues for developing the strategy you'll need to stave off the attackers.

DIABLO

Extended Software Company (ESC)

Programmer: Manuel Constantinidis

Watch Baffle On Scanner

On your scanner screen, you see Space Station I, orbiting quietly. Two green boxes are drawn around it. Press ENTER, and the sprite display begins. The alien drones, attacking in groups of three, swoop in; misguided missiles and bombs fly past; an orange alien command ship may appear from out of nowhere.

Using the Keyboard or a joystick, you bring your target beam into play. Place it over an alien ship or missile and fire a torpedo. The torpedo, which is released from the bottom of the screen, flies to the point designated by the target beam and detonates. The beam can be moved to a new target before the first torpedo detonates.

Most of the alien missiles are harmless. Those released by the drone ships or the command ship, however, are not. Your main concern is stopping the drones. They attack in groups of three, and sometimes hide off the edges of the screen. You'll learn to listen for the characteristic sound that tells you the drones are nearby.

The drones will fire only from within the inner green boundary, and once a missile is fired, the drones are helpless until the missile hits Space Station I or flies past the boundary area. If a missile is off course, it is best to attack the drones while they are helpless, then drop back on defense. Your station can survive five hits before the game ends.

The Command Ship

Your other concern, the command ship, has neither lost its invisibility nor its long-range firing ability. It must become visible to launch an attack, but after it fires, it disappears again. The command ship's foghorn-like sound, however, is its weakness. When you hear it coming, search for it with your targeting beam (you'll see its shadow if you find it), and fire.

Space Station I starts out rather slowly, giving you a chance to find your way around. But with each 10,000 points you accumulate, the aliens step up the attack. If you manage to accumulate 100,000 points, your hit counter will be reset to one, giving you four chances to play at high speed.

To play the game successfully, you'll have to develop a sound strategy, and you'll have to be capable of reacting to assaults from all parts of the screen. Challenging!

(Continued from page 5)

```

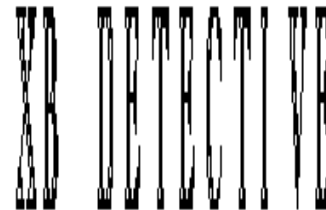
290 PRINT #7:A$ :: A$=B$ ::
IF SEG$(B$,1,1)=CHR$(255)AND
SEG$(B$,2,1)=CHR$(255)THEN
320 ELSE 180
300 V=0 :: FOR X=1 TO NV ::
IF X$=V$(X)THEN V=X :: RETUR
N
310 NEXT X :: RETURN
320 PRINT #7:CHR$(255)&CHR$(
255):: CLOSE #1 :: CLOSE #7
:: PRINT : : : "ENTER 'NEW'
THEN MERGE THE": "COMPRESSED
FILE INTO MEMORY.PROGRAM MA
Y THEN SAVED IN NORMAL MAN
NER."
```

Depending upon how well you plan your moves, Diablo, from Extended Software Company, could keep you at the console for hours! Derived from the European game, DIABLOTIN, your screen is filled with 232 sections of track, two tracks on each of 116 movable panels. The purpose of the game is to keep the ball moving on a continual course. It's much harder than it sounds!

You have one blank section with which to work. When you move your section UP, the section you replace moves DOWN, and so on. Each section has a different pattern, and moving the sections creates a new pattern each time. You must plan ahead and position your track to enable the ball to continue to roll. The game involves strategy and intense concentration for serious players.

Each time a section is traversed by the ball, the section disappears, thereby eliminating track footage. Should your ball roll off the end of the screen, or travel to the end of a section with no continuous path in sight, the ball will roll down to the bottom of the screen and the game will be over. Once 60 sections have been eliminated, the track turns green and a wraparound screen may be utilized. When the ball goes off one end of the screen it will appear at the opposite border, provided there is a compatible track to receive it. You only get one chance per screen, so stay on your toes!

Simplistic strategic challenge and excellent graphic resolution make this game truly worth its \$19.95 price.



By Warren Agee MICROPENDULUM 10/85

Programming in Extended BASIC can be confusing at times. Since BASIC is not a structured language, GOTOs, GOSUBs, and variables are often scattered throughout programs, making things messy and hard to read. The only help an Extended BASIC programmer has had in the past was Programming Aids III, a cross-reference program that was s-s-slow and very cumbersome to use, and Neatlist, a freeware offering which does make listings easier to read but with few features.

Utilitee Software, a brand new company from the Detroit area, has just come out with XB Detective, which is simply the most useful utility a BASIC or Extended BASIC programmer could have in his toolbox. It is written

entirely in assembly language, and auto-loads from Extended BASIC.

Once the program is loaded, you may load in any BASIC or Extended BASIC program, or begin to type in your own program. Detective resides in memory with your program. The following is a list of options which you may perform on the BASIC program once the utility is loaded into the machine:

- 1) List Variables
- 2) Find Variables
- 3) Find Reserved Words
- 4) Delete Lines
- 5) String Search
- 6) Return to BASIC

For instance, if you select option 1, XB Detective will list all of the variable names in your program within seconds. With option 2, you may choose the variable name for which you want to search, and the line number with which to start the search, and XB Detective will instantaneously list to the screen (or printer) all of the line numbers containing that variable. Find Reserved Words works in exactly the same way. Delete Lines allows you to quickly and easily delete an entire block of line numbers, instead of typing: line number to delete, (enter), line to delete, (enter).

Lastly, with String Search, you may choose to search for a string that is enclosed in quotes, in a data statement, or in a CALL statement. For example, suppose you want to find all the occurrences of RS232/2 in a program; you want these changed to your printer specification, say P10. All you do is select String Search, then search for a string in quotes, namely RS232/2. XB Detective will list to the screen all occurrences of RS232/2 in your program. You may optionally print out the results. Now, perhaps, you want to find all the occurrences of the CALL LOAD statement. CALL LOAD is not available under the Find Reserved Words option, but it is under String Search. Just select In a CALL statement, then enter LOAD for the string. Similarly, you may search for CALL CLEAR, CALL COLOR, CALL SPRITE, etc.

Performance: XB Detective operates flawlessly. I could honestly find no bugs whatsoever. It does exactly what it is supposed to do. It is very fast and efficient.

Ease of Use: Here is the best part: this program is INCREDIBLY easy to use. With practically all of the assembly utilities up till now, one had to use cumbersome CALL LINK statements to access the new functions. Not here. All that you need do is hit FCTN-7 (AID) and you are presented with the menu of options. From here on everything is menu driven. Once you select option 6 "Return to BASIC", the screen is restored to the condition it was in when you hit FCTN-7.

Finding a deficiency in this program is very difficult, though I believe one minor drawback is that not every reserved word can be searched with XB Detective. Most of the major ones are present on the menu, but not all. For instance, RANDOMIZE and RND are not available for the search. A function that I found missing (now I am getting picky) is an Oops Key. This would recover a program lost by typing NEW. I am told that this program uses up every bit of the 8 Kilobytes of space allotted for assembly language in the expansion memory; if more memory were used, it would infringe on the size of the Extended BASIC program you are editing: Having that limitation would be worse than not having an Oops Key.

Value: For \$19.95, this program can not be beat. In addition, it is designed to be fully compatible with the TI, CorComp, and Myarc disk controllers. If you are an avid BASIC or Extended BASIC programmer and you have a fully configured system, you will find yourself reaching for XB Detective time and time again. And even if you are not heavily into programming, just think of how many times you wanted to modify someone else's program to fit your system. XB Detective makes the entire process much easier.

```

10 DISPLAY AT(4,6)ERASE ALL          160 LINPUT #1:A$ :: IF GLN(A
BEEP:"PROGRAM COMPACTOR":          $)=65535 THEN PRINT #7:A$ ::
: : "USES A PROGRAM 'SAVED' I      GOTO 320
N MERGE FORMAT": : "USE A D       170 IF POS(A$,CHR$(147),1)<>
IFFERENT NAME FOR THE 'COMPRE     0 THEN PRINT #7:A$ :: GOTO 1
SSED' FILE!" !COMPRESSL          60 ! DATA
20 !BY J R DEW                    180 LINPUT #1:B$ :: P=POS(A$
30 ! COMPRESSES THE LINE NUM     ,CHR$(132),1):: IF P<>0 THEN
BERS FROM A MERGE FILE PROGR     290 ! IF
AM                                 190 IF POS(A$,CHR$(154),1)<>
40 DEF GLN(X$)=ASC(SEG$(X$,1)     0 THEN 290 ELSE IF POS(A$,CH
,1))*256+ASC(SEG$(X$,2,1))       R$(131),1)<>0 THEN 290 ! REM
50 R$=CHR$(201)! LINE NUMBER     OR !
REFERENCE                          200 IF GLN(B$)=65535 THEN PR
60 DIM V$(200)                    INT #7:A$ :: GOTO 320
70 INPUT "FILENAME: DSK1.":F     210 X$=SEG$(B$,1,2):: GOSUB
$ :: OPEN #1:"DSK1."&F$,DISP     300 :: IF V<>0 THEN 290
LAY ,INPUT ,VARIABLE 163         220 IF POS(B$,CHR$(150),1)<>
80 LINPUT #1:L$ :: LN=GLN(L$     0 THEN 290 ! NEXT
)                                   230 IF POS(B$,CHR$(161),1)<>
90 IF LN=65535 THEN 150          0 THEN 290 ! SUB
100 SS=1                           240 IF POS(B$,CHR$(154),1)<>
110 P=POS(L$,R$,SS):: IF P=0     0 THEN 290 ! REM
THEN 80 ELSE X$=SEG$(L$,P+1     250 IF POS(B$,CHR$(131),1)<>
,2)                                0 THEN 290 ! !
120 GOSUB 300 :: IP=V :: IF      260 IF POS(B$,CHR$(147),1)<>
V=0 THEN NV=NV+1 ELSE 140       0 THEN PRINT #7:A$ :: PRINT
130 V$(NV)=X$                      #7:B$ :: GOTO 160 ! DATA
140 SS=P+2 :: GOTO 110           270 IF LEN(A$)+LEN(B$)>=162
150 RESTORE #1 :: PRINT "COM     THEN 290
PRESSED" :: INPUT "FILENAME:     280 A$=SEG$(A$,1,LEN(A$)-1)&
DSK1.":C$ :: OPEN #7:"DSK1.     CHR$(130)&SEG$(B$,3,LEN(B$)-
"&C$,OUTPUT,VARIABLE 163


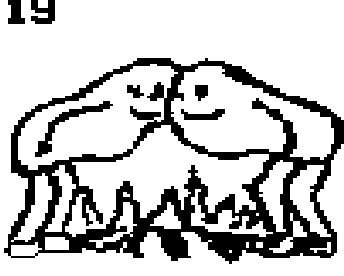
```

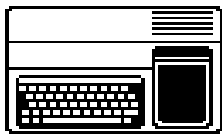
(Please turn to page 4)

NAME THAT TUNE

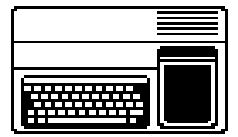
Cleveland Area Users Groups
December 1992

Christmas Carol/Song Titles.
Answers next issue.
Can you name them all?

<p>1</p> 	<p>2</p> 	<p>3 HO HO HO</p> 	<p>4</p> 																																							
<p>5</p> 	<p>6</p> 	<p>7</p> 	<p>8</p> 																																							
<p>9</p> 	<p>10</p> 	<p>11</p> 	<p>12</p> 																																							
<p>13</p> 	<p>14</p> <p>A B C D E F G H I J K M N O P Q R S T U V W X Y Z</p>	<p>15</p> 	<p>16</p> <table border="1"> <tr> <td colspan="5">EVEREMBER</td> </tr> <tr> <td></td> <td>1</td> <td>2</td> <td>25</td> <td>4</td> <td>5</td> </tr> <tr> <td>25</td> <td>7</td> <td>25</td> <td>9</td> <td>10</td> <td>25</td> <td>12</td> </tr> <tr> <td>13</td> <td>25</td> <td>15</td> <td>25</td> <td>17</td> <td>25</td> <td>19</td> </tr> <tr> <td>25</td> <td>21</td> <td>25</td> <td>23</td> <td>25</td> <td>25</td> <td>24</td> </tr> <tr> <td>25</td> <td>25</td> <td>25</td> <td>25</td> <td>25</td> <td></td> <td></td> </tr> </table>	EVEREMBER						1	2	25	4	5	25	7	25	9	10	25	12	13	25	15	25	17	25	19	25	21	25	23	25	25	24	25	25	25	25	25		
EVEREMBER																																										
	1	2	25	4	5																																					
25	7	25	9	10	25	12																																				
13	25	15	25	17	25	19																																				
25	21	25	23	25	25	24																																				
25	25	25	25	25																																						
<p>17</p> <p>OO</p> 	<p>18</p> 	<p>19</p> 	<p>20</p> 																																							
<p>21</p>  <p>1 2 3</p>	<p>22</p> <p>SHHH!</p> 	<p>23</p> 	<p>24</p> <p>HI HO</p> 																																							



Yesterday's News Information



Yesterday's News is a labor of love offered as a source of pleasure & information for users of the TI-99/4A and Myarc 9640 computers.

TI-99/4A HARDWARE

TI99/4A COMPUTER
MODIFIED PEB
WHT SCSI AND SCSI2SD
MYARC DSQD FDC
MYARC 512K MEMORY
HORIZON 1.5 MEG HRD
TI RS232
CORCOMP TRIPLE TECH
1 360K 5.25 DRIVE
1 360K 3.50 DRIVE
1 720K 5.25 DRIVE
1 720K 3.50 DRIVE

TI-99/4A SOFTWARE

PAGEPRO 99
PAGEPRO COMPOSER
PAGEPRO FX
PAGEPRO HEADLINER
PAGEPRO GOFER
PAGEPRO FLIPPER
PAGEPRO ROTATION
PIXPRO
PICASSO PUBLISHER
BIG TYPE
TI ARTIST PLUS
GIF MANIA

PC HARDWARE

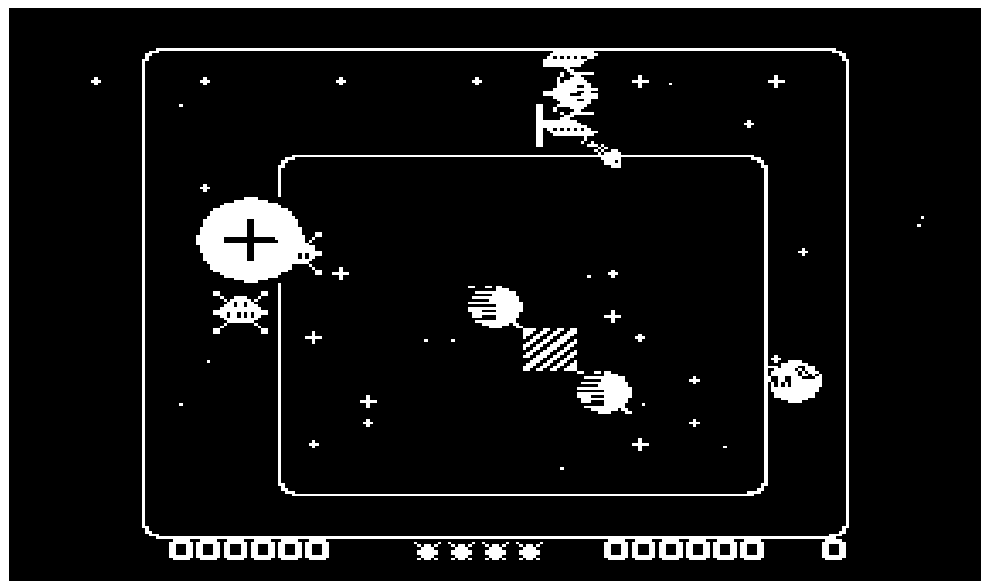
COMPAG ARMADA 7800
COMPAG ARMADASTATION
SAMSUNG SYNCMASTER

PC SOFTWARE

DEAD WINDOWS 98SE
FILECAP
PRN2PNS
IRFANVIEW
ADOBE DISTILLER
ADOBE ACROBAT

Yesterday's News is composed entirely using a TI-99/4A computer system. It consists of 11 PagePro pages which are "printed" via RS232 to PC to be published as a PDF file.

NOW PLAYING



Texas Instruments

color monitor

