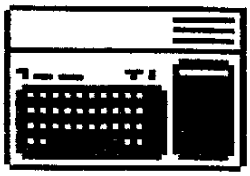
TI-D-BITS

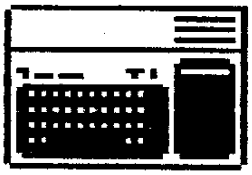
PHILADELPHIA AREA USERS GROUP NEWSLETTER
COVERING THE TI99/4A
AND MYARC 9640 COMPUTERS

JANUARY 1989

DONT FORGET TO
RENEW YOUR
MEMBERSHIP

HAPPY NEW YEAR





RENEW NOW

PHILADELPHIA AREA

TI 99

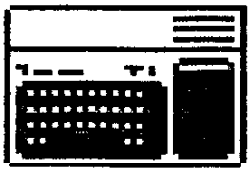
USERS GROUP

1989 DUES

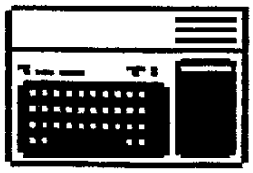
#12

SEND TO

DONALD ARSENAULT
1290 BUTTONWOOD DR
LANSDALE PA
19446



MAKE



CHECK PAYABLE TO PATIUG

****

IMPACT/99

By JACK SUGHRUE

****

IMPACT/99 BLUE RIBBON 1989 WINNER

If this annual award could be given to the same company two years in a row, ASGARD SOFTWARE (with its incredibly varied and impressive catalog) would certainly be very much in contention again. So I'm glad I didn't have to make that decision this year.

Instead, it was a clear choice: MYARC is the winner of the 1989 IMPACT/99 BLUE RIBBON AWARD.

MYARC is one of the few companies still making anything for TI owners on a steady basis. But it isn't just ANYTHING that they are making; they have given us the most powerful hardware and software that exists for us. They haven't just provided enhancements; they have given us a future.

MYARC (the vision, the dream, of former TI employee LOU PHILLIPS) has been around a long time. Since 1982, actually, when Lou developed Winchester Hard-Disk capabilities which sold better in other countries than here (as we were mostly all fledglings at the time). Later he produced a not-very-successful competitor to the TI PE box (still flooding the interested market at the time). So he moved into the card development. And there MYARC (which is a mutilated acronymic form of "Microcomputer Architects") began to blossom.

From a personal viewpoint (as this column has always been - for better or worse), MYARC and I have had a perfect relationship. I own lots of their products, and I have never had to speak to or write to anyone about them. They have been easy to use and have never broken down. And they have made my computing life much richer.

A few years ago my TI disk controller card was behaving erratically. Lots of my friends recommended the MYARC card. Got it.

Loved it from the minute I pulled out my old card and plugged in the new. It immediately made my original Shugart SSSD into a SSDD drive, so I doubled my potential on every disk and no longer had to "flippy" anything.

Not only did the MYARC controller work smoothly, but it was faster than my old controller, and it had inside a built-in disk cataloguer which could be accessed from anywhere by CALL DIR(n). I forget how wonderful this is until I get to someone else's non-MYARC TI.

And it had MYARC's legendary disk management system. Still my first choice among a pile of excellent systems and one that remains constantly configured in FUNNELWEB on my RAM. (But I'm getting a bit ahead of myself.) Lots of programmers learned a lot of techniques from this OM, but for users like myself it opened up a large world (particularly within its futuristic utility menu).

Now my drive was old, so I thought I'd geta new DSDD one and add a power supply for my old one. I did. Again, the controller took everything in stride. Switched from one kind of drive to another with no heavy breathing.

As my computer madness grew I knew I'd never be happy without a RAMdisk or some extended memory. MYARC had just come out with their 512 card to go along with their 256 and 128 cards.

As I had such great fortune with MYARC, I bought their 512. Took out my 32K card, plugged in the new. Just like the controller, it worked perfectly from that moment.

I had a LARGE RAMdisk that I could partition as a buffer for my printer and have lots of options available. But did I really need all that space? I didn't think so at the time. I wondered why I hadn't purchased the smaller cards with my hard-earned pennies.

However, within a couple weeks, I had all the FUNNELWEB and PLUS! files I use regularly (and some other very specific utilities and games) all on a RAM load with an automatic 80K set aside for buffering (which turned out to be one of the greatest enhancements I ever added to my TI).

The RAM portion is wonderful to operate. Everything I need is THERE at the moment I want it. All the word processing tools. All the assembly tools. All the utilities, in short, that I always used to load one-by-each as needed. In those days the thing NOT inmemory was the thing needed most at any given time.

And my controller? Well, I just designated my 512 card as drive 3, and it went about its business as if I had hardly given it an adult task. Its "HO-Hum" manner showed me that the design of the thing was ingenious. No fuss. No muss. No bother. I like things that way.

Now, here I was with a MYARC-stuffed full-blown system when my extra SSSD original drive (in the power supply box) died after much faithful service. Six years is a long time, I've been told. Particularly for the kind of use I give the drives. So I bought a couple DSDD half- heights on sale, ut them in the P-Box, put the DSDD from the box into the added power supply, and ran my software. But all my software had been geared to making drive 3 as my RAMdisk. My controller winked at me.

"Call the extra drive 4," it said, "and keep the RAM at 3." I took its advice. Now I have all four drives (with 512 at 3) operating quickly and flawlessly and wondered how I ever did with three drives or two. Can't even imagine how I survived with one.

(There's something very obsessive about this kind of behavior.)

Although I am the ultimate non-techie, even I can plug in cards and (as a last resort) read manuals. MYARC makes it so easy, you don't have to read the manuals in most cases, though they warn the user NEVER do anything without first reading the manual completely.

After a few years of bliss with MYARC, I was pleased as punch to learn that they were developing a NEW COMPUTER that would be compatible with the TI. Not just an upgrade. But a NEW COMPUTER.

Well, like ALL (without exception) new products in the computer industry world wide, the announcements of its coming dragged on and on. But each stage was publicized to the point of annoyance. Probably what was the most annoying were the doomsayers. They dumped all over MYARC for the delays. It's too bad, really. The kinds of stuff coming out for still-manufactured computers does not raise the ire with the endless delays because there is so much else being manufactured and released. With MYARC, it was the only show in town. So it got spotlighted. And, in some people's minds, got a bad rep. Not deserved. Not deserved at all.

If you're the ONLY company making a compatible upgrade for an orphaned computer, you are taking a great risk to begin with. You get no support to continue with. And you get to live with what you have created to end with.

What MYARC ended with is a minor miracle. The GENEVE (9640) costs about twice what the keyboards sold separately costs. Less than twice the different RAMdisks cost. For under $500 99ers can now buy a computer that was almost 100% compatible with every piece of software they own. It has 640K built in. It has a full-size enhanced keyboard. It can partition a huge buffer for those novels of yours. It has the best graphic resolution in the business. It comes with some pretty impressive software and ports for mouse, printer, modem, etc.

The GENEVE is the ONLY answer for TI upgrading. Thank goodness it's a great answer. In addition to the powerful DOS, the software includes MYWORD (an excellent 80-column processor), Advanced BASIC (that goes far beyond Extended BASIC), Pascal, GPL, and a cartridge downloader.

Early owners (like myself) have been receiving updates of all the software free. So our machine keeps getting better and better. As a matter of fact, there is another whole package being sent out by MYARC this month. I can't wait. What a service this is!

This computer has so much speed that you have to set most software on slower modes in order to handle the difference.

And, like all the other stuff from MYARC, this computer is on a card that just plugs right into your P-Box. (the manual is huge and includes quite a section on the superb Advanced BASIC.) it will take quite a bit of time and effort on the user's part to use the GENEVE to its full potential (if one can ever reach the full potential of any computer). There are also many options (such as a 512 card) that can be added to the GENEVE. There is also a growing software support. MYART is a mouse-served, high-resolution package. Most TI software makers are creating GENEVE compatibility right at the start.

And, NOW!!! Before I even get a chance to start to master the GENEVE, MYARC has done it again!

They have just released the first Hard and Floppy Disk Controller with Streamer Tape Backup Support with MYARC OM-V, the most intuitive OM on the market.

The controller includes a real built-in time clock for file stamping; interfaces with standard floppy, hard and streamer drives; support of up to four 5 1/4 and/or 3 1/2 drives in any configuration; provides RAMdisk speed of a hard-drive transfer rate of 5Mbit per second. And so on.

I have no plans in the immediate future for hard-driving, but it is sure nice to know that MYARC is providing the options if I do. It is also nice to know that some of the best minds in the TI World Community have participated in the creation of these great MYARC advances.

It is a real pleasure to present this annual award to a company that has the TI owners in mind and who has brought us into the high-tech age enjoyed by so many other computers. Their continued support in the face of a lot of adversity is not just commendable but outstanding. MYARC doesn't deserve the bum rep given to it by the loud (but fortunately small in number) complainers who seem to need a scapegoat for their own self esteem.

CONGRATULATIONS, MYARC! You're doing a great job, Lou! Keep it up.

****

HOME PUBLISHING ON THE 99/4A
FROM BCS NEWSLETTER
NOV. 1988

****

The Western NewYork 99ers has for sale a package called "Home Publishing on the 99/4A" by Harry Thomas Brashear. It consists of a disk and an 80-page manual on how to use the supplied programs to create banners, pictures, and creative text. According to the group's flyer:

The manual "will show in detail what clips and instances are and how to use them." It also catalogues over 150 fonts and shows how they may be used.

"Home Publishing talks about the many graphics programs, just how they relate to each other, or how to put them together to work for you."

"But it doesn't stop there. The manual you get is 80 pages long... and it just wasn't long enough. For the cost of the original manual you will also receive two 20-page supplement issues during the coming year. This is so your manual will always be up to date with information on the latest graphics software."

"You will also receive a disk of programs to help you make your documents look their best. The programs include a TI-Artist instance printer that will not leave any skip lines in the printout, a program to organize your fonts and graphics, and the world's best columnizer."

The cost of the complete package is $15.00, which includes postage and handling. User groups can but a minimum of 10 packages for $12.50 each. To order, or for more details, write to:

Western New York 99ers
c/o Harry T. Brashear
2753 Main Street
Newfane, NY 14108

><*><*><*><*><*><*><*><*><*><*><

****

DRACULA LIVES ! !
By David Fink
NUTMEG TI-99ers User Group

****

England has recently shipped us a treasure in the form of CARFAX ABBEY, a fantasy game in the style of TUNNELS of DOOM or more recently LEGENDS. The scope of CARFAX is not as great as either of the aforementioned games as it can be played easily in an hour or two. However the graphics are good, the monsters abound, animation is used well, and the plot is interesting and simple enough to hold the attention of an adult or child.

Basically the plot consists of the player being inside Carfax Abbey which you may recall is Dracula's home in England. A crucifix and 200 spellpoints are required before one may face down Dracula and then escape from the abbey.

The game itself is written in Extended BASIC with assembly language enhancements. In its present form 18K of program space and 6K of stack space are used by the program. Play is smooth and room changes are at least as fast as T.O.D. Status screens are available with every move without cost, games can be saved, and 1,000 variants of the game can be generated.

My major gripe about the game has to do with the speed of loading and set up. However this is a minor matter and there are speed up methods available to the end user which I hope to describe later in the NON PROGRAMMER'S NOTEBOOK

Hardware required:
32K
1 or more disk drive
Extended BASIC

This program is FAIRWARE. If you try it and like it, send David Vincent the Yankee dollars and comments that he requests in the documentation.

><*><*><*><*><*><*><*><*><

****

THE NONPROGRAMMER'S NOTEBOOK

By David Fink
NUTMEG TI-99ers

****

This is the first of what I hope will be a series of non technical articles aimed at users who don't want to program but who would benefit from learning more about some of the enhancement utilities that are available for Extended BASIC. These utilities are often no more difficult to use than DM-1000.

For our first example let's use Barry Boone's SYSTEX to improve the loading speed of the fantasy game CARFAX ABBEY. CARFAX ABBEY is a fairware game by David Vincent of Kent, UK. This game uses 7 assembly language graphic routines that take over 1 minute to load. After these programs are in memory, the main program begins to load. From start to finish it takes about 3 minutes to get the program booted. Now this is a short time in geological terms, but it is forever and a day to a computer user soooo let's improve it.

First you'll need one disk drive, a back-up copy of CARFAX ABBEY,and a copy of SYSTEX. Now turn on the TI.At the ready prompt type:

```
CALL INIT
CALL LOAD("DSK1.GRAPH1","DSK1.GRAPH2","DSK1.GRAPH3",
"DSK1.GRAPH4","DSK1.GRAPH5","DSK1.GRAPH6","DSK1.GRAP
H7")
```

Place the disk containing CARFAX ABBEY in drive 1 and press ENTER. Now read the paper until the load is done. Place the SYSTEX disk in drive 1 and type:

```
11 RUN "DSK1.CARFAX"
```

Press "ENTER". Now type:

```
SAVE DSK1.GRAPHICS
```

Place a formatted disk in drive 1 and press "ENTER". Believe it or not we have just performed a very delicate operation, 7 assembly language routines have just been disguised as 1 fast loading Extended BASIC program.There will also be a saving of about 47 sectors of disk space. Place the copy of CARFAX ABBEY in drive 1 and type:

```
OLD DSK1.CARF/LOAD
```

Press "ENTER". At the READY prompt type 440 and press the down arrow (Fctn x), you should see:

```
440 CALL INIT :: CALL LOAD("
DSK1.GRAPH1","DSK1.GRAPH2","
DSK1.GRAPH3","DSK1.GRAPH4","
DSK1.GRAPH5","DSK1.GRAPH6","
DSK1.GRAPH7")
```

As you can see this is a step that we want to eliminate, so press Fctn 3, followed by Fctn x. The display should now say:

```
450 RUN "DSK1.CARFAX"
```

Rewrite the line to read:

```
450 RUN "DSK1.GRAPHICS"
```

Now press "ENTER", place the disk containing the file we just created into drive 1, and type:

```
SAVE DSK1.LOAD
```

Press "ENTER". The final step consists of transferring the needed files from the old CARFAX ABBEY disk to the new one. The files that must be moved are:

```
CARFAX

CARFAX/DOCS

HELP (if your disk has a file called CHEAT/NOTES,
     copy it and change its name to HELP)
```

I hope you have tried the steps listed above and don't forget to pay the authors if you make these programs a part of your library. SYSTEX can be applied to other programs, like Steve McWatty's GRAPHIC LABELER, or many others that load in assembly programs. Try it out, you'll get an error message if it doesn't work and you'll lose about 3 to 4 minutes of your time.

P.S. Let me know if the instructions are too detailed. I am aiming at the tyro and devout non programmer. Feedback from any source will be accepted and acted upon as time and ability allow.

)(*)(*)(*)(*)(*)(*)(*)(*)(

****

Converting GRAPHX To TI-ARTIST

By Chris Bobbitt

Taken from LA 99ers CIN-DAy

****

I recently read an article in the Hoosier User's Group's excellent user group newsletter on converting GRAPHX to TI-ARTIST with interest. I too had faced this dilemma some time back in trying to transfer our popular GRAPHX Companion series of products over to TI-Artist (a project which regrettably still has never been completed due to time limitations and a rather low priority). In any case, I thought the procedure we worked out may be of interest as well.

After playing with both programs for a while, we hit upon the solution offered by Mr. Robert Coffey. As a matter of fact, on our now discontinued Artist companion disk there is a font that was converted over in just such a manner. It was so time consuming that we soon gave up.

The matter stayed dropped until 9 months or so ago when we were preparing Font Writer for release. I asked Peter Hoddie, the author if he knew of a way to convert files over, and he said that Font Writer could be used.

Later that week, I sat down, and 4 hours later I had my first font. I chose a very elegant Times Roman, with a complete upper and lower case alphabet from GRAPHX Companion IV (which was then in the editing phase) for the experiment.

The process was rather simple, actually. Step one involves getting the fonts to TI-Artist format. If they are stored in a clipboard, as our GRAPHX Companion series fonts are, this involves first pasting them onto a screen (leaving plenty of room between characters) and then saving the screen to disk. As mentioned in Mr. Coffey's article, it is a very good idea to use an empty disk for this.

Next, enter TI-Artist and select the conversions section. Convert the screen from GRAPHX to TI-Artist format (load it as a GRAPHX screen and save it under TI-Artist).

Next, enter the TI-Artist section from the menu, and select disk options. Load the screen into memory. Leave TI-Artist and go to the Enhancements option. The screen you loaded in TI-Artist will be in memory. Next, enter the Slides menu and select the Save Instance option. The file name you give should be the ASCII character that the picture you are saving represents (i.e., if the picture is of an "A" the filename should be A). Next, the screen will appear. Move the cursor to the upper left corner of the character picture you are saving, press the fire button, and move to the lower right side of the picture, boxing in the character. Press the fire button again and the picture will be saved to disk. Do this over and over until the whole font is saved as individual instances.

When you are done, exit TI-Artist and load your copy of our Font Writer program. Enter the Font Editor option.

When that portion of the package is loaded, go to the menu options, and select the option for opening a font for output. Do NOT select the Append font option. Next, enter the Instances selection from the same menu, and load in the first character (A or whatever). After it loads, you will be dropped to the graphics window.

At this point, it is a good idea if you establish a "baseline" first. The baseline is the bottom line that you will use (not physically) for placing the characters. Characters with descenders (which are the hardest to center), can be easily lined up if you adjust them according to that line (remember such characters may look odd, with all that empty space above them, but that is the way they should look - TI-Artist and Font Writer look at a font from the upper left hand corner).

Using the Move Picture keys of the editor, you can easily move the picture left, right, up, and down to center it on that imaginary line (use the block boundary markers to avoid confusion). After the picture of the character is centered, enter the menus again, and again select the font options. Then select the option to save a picture in a font. The Editor will ask you what ASCII characters it represents, then a white cursor will appear on the screen showing the picture. Position this cursor, with the arrow keys on the lower right corner and press Enter. The program will automatically save it to disk in the font file you specified as the ASCII character you specified. Do this over and over until all your Instances are converted to a font.

The advantages of this system over the one mentioned by Mr. Coffey (which, while still a good system and not requiring Font Writer) is that it is much faster, and you can center the characters in the font a lot easier since Font Writer has the tools for it.

Converting regular clipart from GRAPHX to TI-Artist, of course, is a much simpler procedure since all you really have to do is paste it on the screen, convert the screen to TI-Artist, and save each individual picture as an Instance or Slide.

The reverse process, converting TI-Artist to GRAPHX, is very easy. All you have to do is get whatever you are converting into a screen, save it to disk, convert the

screen to GRAPHX (again using TI-Artist's conversion utility) and then paste it to the clipboard from a GRAPHX SCREEN.

Regarding the legality of converting art between GRAPHX to TI-Artist; I'm not sure what the policy of other manufacturers is, but ours is that once you buy the stuff, it's yours. You can convert it to any format that you like. However, remember that the works in our GRAPHX Companions and Artist Instances series ARE copyrighted (they are in fact the product of literally thousands of hours of work - a single font may take up to 10 hours to draw with GRAPHX!), and you can't give them out to anyone else. You can convert them for your own use, but no one elses.

Font Writer is also copyrighted to J. Peter Hoddie and is manufactured and distributed by Asgard Software. The use of it as described here is only one of the many functions of the product.

<*><*><*><*><*><*><*><*><*><*><*><*>

****

EXTENDED BASIC
STILL A GOOD CHOICE!

By Art Byers
Fm Central Westchester 99'ers

****

There are new guys in the 99/4A neighborhood. Among them are such stars as FORTRAN, FORTH, PILOT and SMALL C. They have lots of adherents who talk about "like Basic" (FORTRAN), "Freedom and Exceptionally flexible" (FORTH), "Simplicity" (PILOT), and "Speed and structure" ('c'). They are Compiled languages which means they certainly run much faster than old friend XBasic. So, why bother with Extended Basic at all? Why not go with the New? The Better? The Faster?

One of the great things about our beloved 99/4A is that even with its limited memory, it CAN support FORTH and C and PILOT. I consider any of the computer languages that will accomplish what is needed to be fine! For me, however, Extended Basic still remains the EASIEST and BEST, most especially when coupled with Assembly Language subroutines that speed up often used important areas.

Let me try to lead you through a discussion of the pros and cons of Extended Basic without 'putting down' in the slightest ANY other language for the 99/4A (including Pascal -However Pascal requires a special PEB card and those are hard to find and some early versions have bugs).

Extended Basic has many advantages from a programmer's viewpoint, not the least of which is that it is an interpreted language with a plethora of error debugging routines built in. One of the real swift pains in the neck of a compiled language is that if it is compiled containing errors or bugs, these are extremely difficult to find. This does not mean they cannot be found or that good programmers cannot produce error free compiled code. It is just that debugging, added to, subtracting from, changing code, etc. is much easier with XB. It is a shame that TI chose to make XB a "double" interpreted language by writing it in GPL, TI's "secret" proprietary language, also interpreted, (Which to the best of my knowledge TI has NEVER released and should they have chosen to take legal action, they could make trouble for those who have violated their rights by selling GPL programs, books explaining GPL, etc. and etc.). It would have been better if the interpreter had been written in Assembly a la MYARC's XB. The added speed of MYARC's XB is a big improvement over TI,s XBasic.

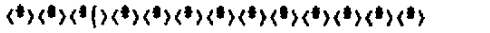
One of the biggest advantages of XB is its EASE OF USE AND UNDERSTANDING. BASIC Itself was written just for that purpose. BASIC is supplied with such popular computers as Apple, Atari, Commodore, and IBM. This ease of use was most important in bringing better understanding of computers and use of computer languages to large numbers of Americans. For no other reason, the Basic language continues to survive.

As far as the 99/4A goes, another advantage is that the language itself resides outside the RAM areas. It is in ROM and GROM. The cover of the XB manual states that the module contains "32k bites of preprogrammed memory". Most of the RAM is free. Additionally, XB accesses, again with simplicity, clarity and ease, the built in ROM routines such as Device Service - printers, cassette, disk drives -, screen access and display, setting up of buffers, graphics and sprites, mathematics, etc. Many of the "new" languages save RAM memory by also accessing these same ROM routines, running at the same speed for all! Now lets talk about available memory. Because support for Forth and 'C', for examples, must be loaded into the main 32k memory area, they do not have as much memory available as some programmers feel is absolutely necessary. This problem has been solved by using virtual memory - that is disk storage of Forth screens (blocks) or C support routines. XB support resides in console ROM and the module itself, the full 24k upper Ram is available for programs and the 8k low memory for Assembly support routines, and most of VDP RAM for string storage

etc. For example, I recently purchased a Disassembler which was written in Forth. The author plainly stated that because of the memory used by Forth itself plus the program, it was not feasible to disassemble programs from RAM. It did its disassembly right off the disk!. Since Basic resides in ROM, a disassembler written for E/A or MM modules can be written in plain old BASIC, and can disassemble programs that use the 24k upper and 8k lower memory, because it resides in VDP RAM, and not overwrite the program.

Some last points! let us look at what we have to work with. We have a machine designed as a HOME computer. For almost every purpose or use at home, memory and speed available through XB are more than sufficient. We are not tracking satellites, doing high order lengthy math, searching a database the size of the national Social Security register. We have a hundred or so names on our phone list. We do not require massive spread sheets. For our normal practical purposes XB and the 99/4A can suit our needs. In fact I may be accused of HERESY, but I did almost everything with only the XB module and cassette - NO memory expansion or disk!!!

What is more, when I need a special program written to fill a personal need, I wrote it, debugged it and am using it in a matter of a few minutes to at most an hour. This is possible because the most frequently used XB GOSUB routines and CALL SUBs are saved on disk as MERGE files ready to be placed into a program, easily and quickly. Many programmers overlook this useful feature of XB.



1989 FAIRS

MARCH

TICOFF (TI Computer Owners Fun Faire), March 18, Roselle Park, New Jersey. For information, write TICOFF '89 c/o Roselle Park High School, 185 West Webster Ave., Roselle Park, NJ 07204, or call Robert Guellnitz at (201) 241-4550 or (201) 382-5963, or the TICOFF BBS at (201) 241-8902.

MAY

Multi User Group Conference, May 20, Reed Hall/ Student Activities Building, Ohio State University, Lima Ohio. For further information, write Lima Users Group, P.O. Box 647, Venedocia, OH 45894, or call Dave Szipple evenings at (419)228-7109.

PRESIDENT'S COLUMN

1989 heralds a new era for the Philadelphia Area TI Users Group. Hopefully we will become larger and more able to provide our members with information and help than before. In order for us to be able to meet this lofty goal, we need the cooperation of YOU, the membership. We need your attendance at the meetings, we need your input to our newsletters, and we need to know about your needs and problems so that we may be able to help.

Our numbers have been steadily shrinking, mainly due to the fact that our computers have been out of production for some 5 years. Another reason is probably that some people have moved on to newer computer systems. I think that another reason for our dwindling numbers is that we are not providing some of the things that the membership expects from the Group. In this vein, we have provided an area on the renewal form for you to express just what YOU would like to get from the Group. Please take time to let us know what it is that we could be doing for you. Also on the form is an area for you to let us know what YOU would be willing to do for the Group. Take time to fill out this section also, because without the participation of our members, we have no group.

At our January La Salle meeting, we demonstrated a new program from England called CARFAX ABBEY. This is an adventure type game with graphics which takes us to the legendary castle of Count Dracula. The object of the game is to journey through the many rooms on the four floors of the Abbey, garnering magical powers and gold pieces, and acquiring the weapons necessary to slay the evil Count Dracula. The program allows an almost endless number of room layout variations, and also allows you to save and restore a partially completed game. It features the ability to constantly monitor ones status regarding where you are and how much power and gold you possess. I was never very much of an adventure game buff, but this one has been very intriguing to me. This program is available in our library, along with many other new entries.

Our "official" meeting will now be the YDC meeting, which is held on the 1st Saturday of each month, at the Youth Development Center in Bensalem, PA, next to the Neshaminy Mall. The La Salle meeting on the 3rd Saturday of each month will continue to held for those members of the Group who are members of PACS, and for those members of PACS who are not members of the PATIUG.

TIPS FROM THE TIGERCUB

#47

TIGERCUB SOFTWARE
156 Collingwood Ave.
Columbus, OH 43213

Distributed by Tigercub Software to TI-99/4A Users Groups for promotional purposes and in exchange for their newsletters. May be reprinted by non-profit users groups, with credit to Tigercub Software.

Over 120 original programs in Basic and Extended Basic, available on cassette or disk, NOW REDUCED TO JUST $1.00 EACH!, plus $1.50 per order for cassette or disk and PP&M. Minimum order of $10.00. Cassette programs will not be available after my present stock of blanks is exhausted. The Handy Dandy series, and Color Programming Tutor, are no longer available on cassette.
Descriptive catalogs, while they last, $1.00 which is deductable from your first order.

Tigercub Full Disk Collections, reduced to $5 postpaid. Each of these contains either 5 or 6 of my regular catalog programs, and the remaining disk space has been filled with some of the best public domain programs of the same category. I am NOT selling public domain programs - they are a free bonus!
TIGERCUB'S BEST, PROGRAMMING TUTOR, PROGRAMMER'S UTILITIES, BRAIN GAMES, BRAIN TEASERS, BRAIN BUSTERS!, MANEUVERING GAMES, ACTION REFLEX AND CONCENTRATION, TWO-PLAYER GAMES, KID'S GAMES, MORE GAMES, WORD GAMES, ELEMENTARY MATH, MIDDLE/HIGH SCHOOL MATH, VOCABULARY AND READING, MUSICAL EDUCATION, KALEIDOSCOPES AND DISPLAYS

NUTS & BOLTS DISKS
These are full disks of 100 or more utility subprograms in MERGE format, which you can merge into your own programs and use, almost like having another hundred CALLs available in Extended Basic. Each is accompanied by printed documentation giving an example of the use of each.
NUTS & BOLTS (No. 1) has 100 subprograms, a tutorial on using them, and 5 pp. documentation. NUTS & BOLTS No. 2 has 108 subprograms, 10 pp. of documentation. NUTS & BOLTS #3 has 140 subprograms and 11 pp. of documentation. NOW JUST $15 EACH, POSTPAID.

TIPS FROM THE TIGERCUB
These are full disks which contain the programs and routines from the Tips from the Tigercub newsletters, in ready-to-run program format, plus text files of tips and instructions.
TIPS (Vol. 1) contains 50 original programs and files from Tips newsletters No. 1 through No. 14. TIPS VOL. 2 contains over 60 programs and files from Nos. 15 thru 24. TIPS VOL. 3 has another 62 from Nos. 25 through 32. TIPS VOL. 4 has 48 more from issues No. 33 through 41. NOW JUST $10 EACH, POSTPAID.

```
*****************************
*        NOW READY          *
* TIPS FROM TIGERCUB VOL.5  *
* Another 49 programs and   *
* files from issues No. 42  *
* through 50. Also $10 ppd  *
*****************************
```

TIGERCUB CARE DISKS #1,#2,#3 and #4. Full disks f text files (printer required).
No. 1 contains the Tips newsletters #42 thru #45, etc. Nos. 2 and 3 have articles mostly on Extended Basic programming. No. 4 contains Tips newsletters Nos. 46-52. These were prepared for user group newsletter editors but are available to anyone else for $5 each postpaid.

If you bought my C11 disk, Kid's Games, please check line 100 of the Butterfly and Flowers program and, if necessary, change it to -
1000 CALL CLEAR :: CALL SCREEN(4).
If you bought my C12 disk, More Games, and have trouble loading Lost Plane and Andromedan Invasion, please go to line 1000 of the LOAD program and change *TC-18* to *TC-18 and *TC-23* to *TC-23. Or, return the disks to me and I will fix them.

Thanks to Ollie Hebert for this fix to the Gordian Knot in Tips #36. This will keep it from running off the edge and crashing in the automatic mode.

```
270 GOSUB 480 :: R=R-24*(R<1)+24*(R>24):: C=C-28*(C<3)+28*(C>30):: CH=128-(D=1)-(D=3):: CALL GCHAR(R,C,G):: IF G<>32 THEN IF INT(2*RND+1)<>1 THEN CH=G
```

The trouble with me is that, before I finish one program I've thought of another that I want to try writing - and so I don't take time time to test completed programs as well as I should. The Decompactor in Tips #35 was one that should have been tested more thoroughly. I think this version will work. It will break an XBasic program into single-statement lines to make it easier to modify. Then, John Dow's Compactor or a similar program will put it back together.

```
100 !DECOMPACTER V.1.1 by Jim Peterson fixed 12/87
110 DISPLAY AT(3,1)ERASE ALL :"TIGERCUB DECOMPACTER V.1.1 ": :" Program must first be -": :"RESequenced to greater in-":"crements than the number"
120 DISPLAY AT(9,1):"of statements in any one":"line.": :"SAVEd by":" SAVE DSK(filename),MERGE"
130 DISPLAY AT(16,1):"INPUT FILENAME?":"DSK" :: ACCEPT AT(17,4):IF$
140 DISPLAY AT(16,1)ERASE ALL:"OUTPUT FILENAME?":"DSK" :: ACCEPT AT(17,4):OF$
150 OPEN #1:"DSK"&IF$,INPUT,VARIABLE 163 :: OPEN #2:"DSK"&OF$,OUTPUT,VARIABLE 163
160 LINPUT #1:M$ :: LN=ASC(SEG$(M$,1,1))*256+ASC(SEG$(M$,2,1)):: IF LN>LN2 THEN 180
170 DISPLAY AT(12,1)ERASE ALL BEEP:"BRROR! RESEQUENCE PROGRAM TO":"GREATER INCREMENTS AND TRY":"AGAIN." :: CLOSE #1 :: CLOSE #2 :: STOP
180 LN2=LN
190 P=POS(M$,CHR$(130),3):: IF P=0 THEN PRINT #2:M$ :: GOTO 260
200 A$=SEG$(M$,1,P-1):: R=POS(A$,CHR$(132),3):: S=POS(A$,CHR$(201),3)
210 IF R=0 THEN PRINT #2:A$&CHR$(0):: GOTO 250
220 IF S=0 AND R<>0 THEN PRINT #2:M$ :: GOTO 260
230 IF S<>0 THEN IF S-R<3 THEN PRINT #2:A$&CHR$(0):: GOTO 250
240 PRINT #2:M$ :: GOTO 260
250 LN=LN+1 :: LN2=LN :: GOSUB 270 :: M$=LM$&SEG$(M$,P+1,255):: GOTO 190
260 IF EOF(1)<>1 THEN 160 ELSE CLOSE #1 :: CLOSE #2 :: DISPLAY AT(12,1)ERASE ALL:"Enter NEW": :"Then Enter":" M
```

```
ERGE DSK"&OF$ :: END
270 LN$=CHR$(INT(LN/256))&CH
R$(LN-256*INT(LN/256)):: RET
URN
```

If you have my BXB routine from Tips #40 (corrected in Tips #42) or from my TIPS disk Vol. 4 or NUTS & BOLTS #3, or Genial Traveller Vol. 1 No. 6, here is a neat improvement that Barry Traver thought of. Key this in, run it to create a merge file on a disk. Then clear memory with NEW, merge in BXB, then MERGE DSK1.LINEZERO, and now save BXB again in merge format and it will CALL itself from line zero (and do something else that I'm not going to tell you about!

```
100 OPEN #1:"DSK1.LINEZERO",
VARIABLE 163,OUTPUT
110 M$=CHR$(0)&CHR$(0)&CHR$(
157)&CHR$(200)&CHR$(3)&"BXB"
&CHR$(130)&CHR$(157)&CHR$(20
0)&CHR$(4)&"CHAR"&CHR$(183)&
CHR$(200)&CHR$(2)&"30"
120 M$=M$&CHR$(179)&CHR$(199
)&CHR$(16)&"81C37EA58199663C
"&CHR$(182)&CHR$(0):: PRINT
#1:M$ :: PRINT #1:CHR$(255)&
CHR$(255)
```

And if you have merged in BXB, the edge character (ASCII 31) can be redentified and colored (set 0) to give the screen an ornamental border.

```
100 CALL CHAR(31,"0"):: CALL
 CLEAR :: FOR J=1 TO 24 :: P
RINT :: NEXT J :: CALL CHAR(
31,"1824429999422418"):: CAL
L COLOR(0,5,16)
```

Here is an improved version of the CATWRITER program to create the Tigercub QUICKLOADER, which is intended for disks of programs which you have filled and do not plan to change. It will read the directory, display each filename, and ask you for the complete program name of each one. Then it prepares a program which displays one or more menu screens of complete program names, and auto-loads whichever one you select.

First, key in this part and save it to disk by SAVE DSK1.CAT1,MERGE. If you want, you can change the screen and character colors in line 10. Don't change the line numbers!

```
10 CALL CLEAR :: DIM M$(127)
:: CALL SCREEN(5):: FOR S=0
TO 14 :: CALL COLOR(S,16,1):
: NEXT S :: CALL PEEK(8198,A
):: IF A<>170 THEN CALL INIT
11 REM  (leave this in!)
12 ON WARNING NEXT :: GOSUB
21
13 X=X+1 :: READ M$(X):: IF
M$(X)<>"END" THEN 13
14 R=3 :: FOR J=1 TO X-1 ::
READ X$ :: DISPLAY AT(R,1):S
TR$(J);TAB(4);X$ :: R=R+1 ::
 IF R<23 THEN 17
15 DISPLAY AT(24,1):"Choice?
 or 0 to continue 0" :: ACCE
PT AT(24,26)VALIDATE(DIGIT)S
IZE(-3):N :: IF N>X-1 THEN 1
5
16 IF N<>0 THEN 10000 :: R=3
17 NEXT J
18 DISPLAY AT(24,1):"Choice?
" :: ACCEPT AT(24,9)VALIDATE
(DIGIT):N :: IF N=0 OR N>X-1
 THEN 18
19 CALL CHARSET :: CALL CLEA
R :: CALL SCREEN(8):: CALL P
EEK(-31952,A,B):: CALL PEEK(
A*256+B-65534,A,B):: C=A*256
+B-65534 :: A$="DSK1."&M$(N)
:: CALL LOAD(C,LEN(A$))
20 FOR J=1 TO LEN(A$):: CALL
 LOAD(C+J,ASC(SEG$(A$,J,1)))
:: NEXT J :: CALL LOAD(C+J,0
):: GOTO 10000
21 CALL LOAD(8196,63,248)
22 CALL LOAD(16376,67,85,82,
83,79,82,48,8)
23 CALL LOAD(12288,129,195,1
26,165,129,153,102,60)
24 CALL LOAD(12296,2,0,3,240
,2,1,48,0,2,2,0,8,4,32,32,36
,4,91)
25 CALL LINK("CURSOR"):: RET
URN
10000 RUN "DSK1.1234567890"
```

Next, key in this little routine and run it to create a file called CAT2. If you added or deleted any lines in the CAT1 file, change the J-loop accordingly.

```
100 OPEN #1:"DSK1.CAT1",VARI
ABLE 163,INPUT
110 OPEN #2:"DSK1.CAT2",VARI
ABLE 163,OUTPUT
120 FOR J=10 TO 26 :: LINPUT
 #1:M$ :: PRINT #2:CHR$(0)&C
HR$(J)&CHR$(156)&CHR$(253)&C
HR$(200)&CHR$(1)&"2"&CHR$(18
1)&CHR$(199)&CHR$(LEN(M$))&M
$&CHR$(0):: NEXT J
130 PRINT #2:CHR$(255)&CHR$(
255):: CLOSE #1 :: CLOSE #2
```

Finally, key in CATWRITER. Leave the line numbers as they are, we need that space after line 5.

Then MERGE in DSK1.CAT2 to combine the two, and SAVE.

```
1 CALL CLEAR :: CALL TITLE(1
6,"CATWRITER"):: CALL CHAR(1
24,"3C4299A1A199423C"):: DIS
PLAY AT(2,10):"Version 1.3":
:TAB(8);"; Tigercub Softwar
e"
2 DISPLAY AT(15,1):"For free
":"distribution":"but no pri
ce or":"copying fee":"to be
charged." :: FOR D=1 TO 500
:: NEXT D :: CALL DELSPRITE(
ALL)
3 DISPLAY AT(2,3)ERASE ALL:"
TIGERCUB CATWRITER V.1.3":::
" Will read a disk directory
,":"request an actual progra
m":"name for each program-ty
pe"
4 DISPLAY AT(7,1):"filename,
 and create a merg-":"able Q
uickloader which dis-":"play
s full program names and":"r
uns a selected program."
5 OPEN #2:"DSK1.CATMERGE",VA
RIABLE 163,OUTPUT
100 OPEN #1:"DSK1.",INPUT ,R
ELATIVE,INTERNAL :: INPUT #1
:N$,A,J,K :: LN=1000 :: FN=1
100
110 DISPLAY AT(12,1):"Disk n
ame?":::N$ :: ACCEPT AT(14,1
)SIZE(-28):N$ :: LX$=STR$(14
-LEN(N$)/2):: LXLEN=LEN(LX$)
120 PR$=CHR$(0)&CHR$(11)&CHR
$(162)&CHR$(240)&CHR$(183)&C
HR$(200)&CHR$(1)&"1"&CHR$(17
9)&CHR$(200)&CHR$(LXLEN)&LX$
130 PR$=PR$&CHR$(182)&CHR$(1
81)&CHR$(199)&CHR$(LEN(N$))&
N$&CHR$(0):: PRINT #2:PR$
140 X=X+1 :: INPUT #1:P$,A,J
,B :: IF LEN(P$)=0 THEN 180
 :: IF ABS(A)-5 OR ABS(A)-4 A
ND B=254 THEN 150 ELSE X=X-1
 :: GOTO 140
150 DISPLAY AT(12,1):P$;"
 PROGRAM NAME?" :: ACCEPT AT
(14,1)SIZE(25):F$
160 PRINT #2:CHR$(INT(FN/256
))&CHR$(FN-256*INT(FN/256))&
CHR$(147)&CHR$(200)&CHR$(LEN
(F$))&F$&CHR$(0):: FN=FN+1
170 M$=M$&CHR$(200)&CHR$(LEN
(P$))&P$&CHR$(179):: IF X<11
 THEN 140
180 IF M$="" THEN 200
190 PRINT #2:CHR$(INT(LN/256
))&CHR$(LN-256*INT(LN/256))&
CHR$(147)&SEG$(M$,1,LEN(M$)-
1)&CHR$(0):: LN=LN+1 :: M$="
" :: X=0 :: IF LEN(P$)<>0 TH
EN 140
200 PRINT #2:CHR$(INT(LN/256
))&CHR$(LN-256*INT(LN/256))&
CHR$(147)&CHR$(200)&CHR$(3)&
"END"&CHR$(0)
210 PRINT #2:CHR$(255)&CHR$(
255):: CLOSE #1 :: CLOSE #2
220 DISPLAY AT(8,1)ERASE ALL
:"Enter -":::"  NEW":::"  ME
RGE DSK1.CATMERGE":::"  DELE
TE ""DSK1.CATMERGE""":::"  S
AVE DSK1.LOAD"
230 SUB TITLE(S,T$)
240 CALL SCREEN(S):: L=LEN(T
$):: CALL MAGNIFY(2)
250 FOR J=1 TO L :: CALL SPR
ITE(#J,ASC(SEG$(T$,J,1)),J+1
-(J+1-S)+(J+1-S+13)+(J>14)*1
3,J*(170/L),10+J*(200/L)):
NEXT J
```

```
260 SUBEND
```

Mike Staufill and Ed Machonis and others have been publishing some neat little "tinygram" programs which can be listed on a single screen, so here is my contribution. It's not only a one-screener, it's a one-liner!

```
1 RANDOMIZE :: PRINT : : : :
 : :: A=INT(RND*7):: B=INT(R
ND*9+1):: FOR X=1 TO 5 :: Y=
A*X^2 B*X+B :: PRINT Y;:: NE
XT X :: Y=A*X^2-B*X+B :: PRI
NT : : :: INPUT "GUESS NEXT
NUMBER":N :: IF N=Y THEN PRI
NT :"RIGHT" :: GOTO 1 ELSE P
RINT :"CORRECT IS":Y :: GOTO
 1
```

MEMORY FULL! - Jim Peterson

<(*)(*)(*)(*)(*)(*)(*)(*)>

SPRITES, PART 1
by Jim Peterson

The sprites of TI Extended Basic are mostly used in fast-action arcade-type games, but they have other uses as well.

Up to 28 sprites can be placed on the screen at one time, but there is one very serious limitation - if more than 4 of them are in a line horizontally, only the 4 lowest-numbered ones will be visible. That is why, if you have numerous sprites moving about the screen, one of them will occasionally disappear and reappear, or a horizontal slice of a magnified sprite will become transparent.

A sprite is placed on the screen by the statement

CALL SPRITE(#N,ASC,COL, DOTROW,DOTCOL)

N is the sprite number, between 1 and 28, and it must be preceded by the # sign. ASC is the ASCII code of the character that you wish the sprite to have. It must be between 32 and 143 - the ASCII characters 33 through 126 are the keyboard characters, the others will be blank unless you redefine them. COL is the color you wish the sprite to have, using the same color codes, 1 to 16, as are used for CALL SCREEN or CALL COLOR.

DOTROW and DOTCOLUMN are the dot row and dot column at which you wish the sprite to appear. You know that the monitor screen consists of 24 rows and 32 columns. Using HCHAR or VCHAR, you can place a character on any one of those 768 spaces (PRINT and DISPLAY start at column 3 of the graphics screen). Each of those spaces consists of a grid of 8 x 8 dots, totaling 64. By turning various of those dots off (blank) or on (colored), a character is displayed on the screen. Therefore the screen is 8 x 32 or 256 dotcolumns wide and the visible screen is 8 X 24 or 192 dotrows deep. Actually dotrow can be anything up to 256; dotrows 193 through 256 are hidden below the bottom of the screen, and sprites can be hidden there.

The upper left hand corner of your sprite will be at whatever dotrow and dotcolumn you specify.

To convert an graphics screen (HCHAR) position into dotrow and dotcolumn, use DOTROW=ROW*8-7 and DOTCOL=COL*8-7; to convert a PRINT/DISPLAY position, you must use DOTCOL=(COL+2)*8-7. So, CALL SPRITE(#1,42,16,89 ,121) will place sprite #1, in the form of the asterisk (ASCII 42), colored white (16) in the middle of the screen. If you want, you can give it motion when you create it, by giving it a row-velocity and a column-velocity. These velocities can be from -128 to 127. A positive row velocity moves the sprite down, negative moves it up; a positive column velocity moves it right, negative moves it left. Velocity 0 is a standstill, and speed increases from 1 upwards and from -1 downwards.

So, CALL SPRITE(#1, 42 ,16, 89, 121, 5, 5) will place that white asterisk in the middle of the screen and start it moving slowly at a 45 degree angle downward to right (since the values 5 and 5 are positive and equal). It will continue moving at that direction and speed until you tell it to do otherwise, all by itself and without program control. When it reaches the right edge of the screen, it will "wrap around" and appear at the left. When it reaches the bottom, it will disappear briefly while it passes through those hidden dotrows, and "wrap around" to appear at the top.

If you want to change the pattern of the sprite, there are three ways to do so. You can CALL SPRITE again with the same sprite number but a different ASCII character - but if the existing sprite is not in the position of the dotrow and dotcolumn you specify, it will disappear and reappear in the new position. Or you can reidentify a character by CALL CHAR, and any sprite having that character will change accordingly, without affecting its color, position or movement. Or you can use CALL_PATTERN(#N,ASC) to change the pattern of sprite #N to the pattern of the specified ASCII character, without affecting color, position or motion.

There are also two ways to change the color of a sprite. CALL SPRITE with the same sprite number and ASCII but a different color code will recreate the sprite with the new color, but in whatever position is specified. CALL COLOR(#N,COLOR) will recolor sprite #N to the specified color code without affecting its pattern, position or motion.

If you want to change the position of a sprite, CALL LOCATE(#N,DOTROW,DOTCOL) will make it disappear at its old location and appear at the new location. The pattern and color will be unchanged, and if it was in motion the same motion will continue from the new position.

To change the motion of a moving sprite, or to start a stationary sprite into motion or vice versa, use CALL_MOTION(#N,RV,CV) - RV and CV being the same row velocity and column velocity optionally used in CALL SPRITE. CALL MAGNIFY will change the size of your sprite. You do not specify

a sprite number with this CALL, because it affects all sprites that are on the screen or are subsequently placed on the screen. CALL MAGNIFY(2) enlarges the sprite 4 times so that it fills 4 of the graphic screen spaces, 256 dot spaces. CALL MAGNIFY(3) causes the sprite to consist of 4 characters, occupying 4 graphic screen positions. The upper left of these characters will be the ASCII specified in the CALL SPRITE or CALL PATTERN, provided that the ASCII is evenly divisible by 4 otherwise, it will be the next smaller ASCII evenly divisible by 4. The next higher ASCII will be in lower left, the next in upper right, the next in lower right. In other words, if you use CALL MAGNIFY(3) and CALL SPRITE(#1,64,2,10,10) you will get a sprite looking like this - @B
AC

- and if you CALL SPRITE(#1,65,2,10,10) you will get exactly the same thing, because the computer will substitute the next lower number, 64, which is evenly divisible by 4.

Naturally, you will not have much use for sprites consisting of four characters, unless you redefine them into a single pattern, and in that case you must remember that they will appear in that upper left/lower left/upper right/lower right sequence. Fortunately, there are sprite editor programs to take care of this for you.

CALL MAGNIFY(4) will enlarge that 4-character sprite so that it fills 16 graphic screen positions. Note that magnification options 2 and 4 actually enlarge each dot to fill 4 dot positions, so that the sprites have a more angular, blocky appearance.

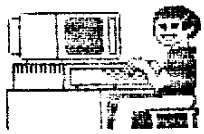
And finally, CALL MAGNIFY(1) will return magnified sprites to their normal single-space size.

Programming with sprite motion is unlike any other programming, because you do not control the program execution step-by-step. When you set a sprite in motion, it continues in motion while the program goes on to do whatever it is supposed to do next. When you want to control the sprite again, you must catch up with it and find out where it is. There are three ways to do this.

CALL COINC(ALL,C) will give a value of -1 to C if any two sprites on the screen are overlapping, even slightly, or 0 if they are not. CALL COINC(#1,#2,TOL,C) will give C a value of -1 if the upper left left hand corners of sprites #1 and #2 are within TOL dotrows and dotcolumns of each other. TOL may be any number you want, depending on whether you want to catch them only when they are right on top of each other, or just getting close. If not within tolerance, C will equal 0.

CALL COINC(#1, DOTROW, DOTROW, DOTCOL, TOL, C) will give C a value of -1 if the upper left corner of sprite #1 is within TOL dotrows and dotcolumns of the specified DOTROW and DOTCOL.

CALL COINC is not foolproof. If you give the sprites a fast motion, a coincidence may not be caught. And when you alternate your CALL COINC with other statements such as CALL JOYST, a coincidence will be missed if the program is executing some other statement at the time. CALL POSITION(#N,DOTROW,DOTCOL) will give the dotrow and dotcolumn that the upper left corner of the sprite is occupying at the instant it is called. This one again is not foolproof because the sprite will have moved from that position before another statement can be executed to do anything with the information.



CALL DISTANCE(#1,#2,D) or CALL DISTANCE(#1,DOTROW,DOTCOL,D) will give to D a value depending on the distance between the two sprites, or between the sprite and the location. The value, as I understand it, is the square root of the total of the squares of the difference between the dotrows added to the squares of the differences between the dot columns. I'm not sure how useful all that is, and I have rarely seen this CALL used by programmers.

Finally CALL DELSPRITE(#N) will delete sprite #1 from the screen and CALL DELSPRITE(ALL) will delete them all.

Those are just the basics of sprite programming. What can be done depends solely on your ingenuity.