RAMBLINGS
by
JIM COX

CREATIVE FILER - This is a data base program which has received some excellent reviews. We have entered into an agreement with the author which allows us to sell it at $9.95 and earn some money for the group. This disk-based, Extended BASIC program will be available at the March 17th meeting and we expect to also sell a number of copies at the April "fayah". (This is even cheaper to MUNCHees than if you sent off yourself. In addition to the $10, you would also have to send 3 disks and mailers and postage both ways. This way will be a savings of about $5 or $6 for this superb program.

++++++++++++++++++++++++++

HELP WANTED - We need people to help out with the MUNCH booth at the April 4th Fair. It will run from 10 to 6 at the JFK High School in Waltham. If you can help out for an hour (preferably in the afternoon) see Bruce Willard at the March meeting or send him a note at the group address. This is *really* important. Most of the active members of the club went to the Lexington Faire last year and had a great time. Come again this year and have a great time, too. But give the club an hour of your time that day. You'll enjoy that part of the day, too!

++++++++++++++++++++++++++

SOFTWARE/HARDWARE SALE - If you have any used equipment or original (no copies) software you would like sold at the April fair, your club will be happy to do it for you at our table there. We are charging a 10% fee on anything sold which will go to the group treasury. You can bring anything you want sold to the March meeting, give it to an officer or brin it to the fair. Make sure to give us your name and address, and the price you want for the item. I will send checks to you for items sold the week after the fair and you can pick up unsold items at the April 21st meeting. (Or put it on sale that night for our big super sale.)

++++++++++++++++++++++++++

MEETING SALE TABLE - There will be no sales table at the March meeting. People wishing to sell or buy come to the April meeting. Super Sales Table that night! Cartridge, disks, tapes. Utilities, games, tutorials, construction sets, graphics designers, artworks, and more. Many items never before offered for direct sale. Most are "one of a kind, first come first serve." All are original. Prepare for the April Super Sales. Sell what you no longer need; buy what you might never get another chance to (and a great savings).

++++++++++++++++++++++++++

See (or call) Don Mason about the following:

6X-100 PRINTER / dot matrix and graphics $75

ELECTRONIC MARVEL! Eliminate all system lockups <u>forever</u>! Your own console will be completely modified to prevent such frustrations. All parts and labor included. $79. Protect your investment. Guaranteed.

TV GENIES / Wireless remote control for TV or VCR or computer combinations / multiple units (as many as you want) at one time / operates on a transmitter. $79.

++++++++++++++++++++++++++

ADS - If you are a fully-paid member and wish to advertise your services or materials, there is no charge. For commercial ads or non-member ads please see the rates on the back inside page.

++++++++++++++++++++++++++

FUNNELWEB FARM EXTENDED BASIC TUTORIAL PART 1

EXTENDED BASIC TUTORIALS

from

FUNNELWEB FARM (1049) 52 3162)

I. INTRODUCTION

In this series of notes on TI Extended Basic for the TI-99/4a we will concentrate on those features which have not received due attention in User-group newsletters or commercial magazines. In fact most of the programs published in these sources make little use of that most powerful feature of XB, the user defined sub-program, or of some other features of XB. Worse still is to find commercially available game programs which are object lessons in how to write tangled and obscure code. The trigger for this set of tutorial notes was a totally erroneous comment in the TI.S.H.U.G Newsdigest in Jun 1983. Some of the books I have seen on TI Basic don't even treat that simpler language correctly, and I don't know of any systematic attempts to explore the workings of XB. The best helper is TI's Extended Basic Tutorial tape or disk. The programs in this collection are unprotected and so open for inspection and it's worth looking at their listings to see an example of how sub-programs can give an easily understood overall structure to a program.

Well, what are we going to talk about then ? Intentions at the moment are to look at:

(1) User-defined sub-programs
(2) Prescan switch commands
(3) Coding for faster running
(4) Bugs in Extended Basic
(5) Crunching program length
(6) XB and the Peripheral Box
(7) Linking in Assembler routines

Initially the discussion will be restricted to things which can be done with the console and XB only. Actually, for most game programming the presence of the memory expansion doesn't speed up XB all that much as speed still seems to be limited by the built-in sub-programs (CALL COINC, etc) which are executed from GROM through the GPL interpreter. The real virtue of the expansion system for game programming, apart from allowing longer programs, is that GPL can be shoved aside for machine code routines in the speed critical parts of the game, which are usually only a very small part of the code for a game. Even so careful attention to XB programming can often provide the necessary speed.

As an example, the speed of the puck in TEX-BOUNCE is a factor of 10 faster in the finally released version than it was in the first pass at coding the game.

Other topics will depend mainly on suggestions from the people following this tutorial series. Otherwise it will be whatever catches our fancy here at Funnelweb Farm.

II. SUB-PROGRAMS in OVERVIEW

Every dialect of Basic, TI Extended Basic being no exception, allows the use of subroutines. Each of these is a section of code with the end marked by a RETURN statement, which is entered by a GOSUB statement elsewhere in the program. When RETURN is reached control passes back to the statement following the GOSUB. Look at the code segments

```
290 ....
300 GOSUB 2000
310 ....
2000 CALL KEY(0,X,Y):: IF Y=1 THEN
RETURN ELSE 2000
```

This simple example waits for and returns the ASCII code for a fresh keystroke, and might be called from a number places in the program.. Very useful, but there are problems. If the line number of the subroutine is changed, other than by RESequencing of the whole program (and many dialects of Basic for microcomputers aren't even that helpful) then the GOSUBs will go astray. Another trouble, which you usually find when you resume work on a program after a lapse of time, is that the statement GOSUB 2000 doesn't carry the slightest clue as to what is at 2000 unless you go and look there or use REM statements. Even more confusingly the 2000 will usually change on RESequencing, hiding even that aid to memory. There is an even more subtle problem -- you don't really care what the variable "Y" in the subroutine was called as it was only a passing detail in the subroutine. However, if "Y" is used as a variable anywhere else in the program its value will be affected. The internal workings of the subroutine are not separated from the rest of the program, but XB does provide four ways of isolating parts of a program.

(1) Built-in sub-programs
(2) DEF of functions
(3) CALL LINK to machine code routines
(4) User defined BASIC sub-programs

The first of these, built-in sub-programs, are already well known from console Basic. The important thing is that they have recognizable names in CALL statements, and that information passes to and from the sub-programs through a well defined list of parameters and return variables. No obscure Peeks and Pokes are needed. The price paid for the power and expressiveness of TI Basic and XB is the slowness of the GROM/GPL implementation.

DEF function is a primitive form of user defined sub-program found in almost all BASICs. Often its use is restricted to a special set of variable names, FNA,FNB,... but TI Basic allows complete freedom in naming DEFed functions (as long as they don't clash with variable names). The "dummy" variable "X" is used as in a mathematical function, not as an array index

```
100 DEF CUBE(X)=X*X*X
```

doesn't clash with or affect a variable of the same name "X" elsewhere in the program. "CUBE" can't then be a variable whose value is assigned any other way, but "X" may be. Though DEF does help program clarity it executes very slowly in TI Basic, and more slowly than user defined sub-program CALLs in XB.

CALL LINK to machine code routines goes under various names in other dialects of Basic if it is provided (eg USR( ) in some). It is only available in XB when

the memory expansion is attached, as the TI-99/4a console has only 256 bytes of CPU RAM for the TMS9900 lurking in there. We will take up this topic later.

You should have your TI Extended Basic Manual handy and look through the section on SUB-programs. The discussion given is essentially correct but far too brief, and leaves too many things unsaid. From experiment and experience I have found that things work just the way one would reasonably expect them to do (this is not always so in other parts of XB). The main thing is to get into the right frame of mind for your expectations. This process is helped by figuring out, in general terms at least, just how the computer does what it does. Unfortunately most TI-99/4a manuals avoid explanations in depth presumably in the spirit of "Home Computing". TI's approach can fall short of the mark, so we are now going to try to do what TI chickened out of.

The user defined sub-program feature of XB allows you to write your own sub-programs in Basic which may be CALLed up from the main program by name in the same way that the built-in ones are. Unlike the routines accessed by GOSUBs the internal workings of a sub-program do not affect the main program except as allowed by the parameter list attached to the sub-program CALL. Unlike the built-in sub-programs which pass information in only one direction, either in or out for each parameter in the list, a user sub-program may use any one variable in the list to pass information in either direction. These sub-programs provide the programming concept known as "procedures" in other computer languages, for instance Pascal, Logo, Fortran. The lack of proper "procedures" has always been the major limitations of BASIC as a computer language. TI XB is one of the BASICs that does provide this facility. Not all BASICs, even those of very recent vintage are so civilised. For example the magazine Australian Personal Computer in a recent issue (Mar 84) carried a review of the IBM PCjr computer just released in the US of A. The Cartridge Basic for this machine apparently does not support procedures. Perhaps IBM don't really want or expect anyone to program their own machine seriously in Basic. You will find that with true sub-programs available, that you can't even conceive any more of how one could bear writing substantial programs without them (even within the 14 Kbyte limit of the unexpanded TI-99/4a let alone on a machine with more memory).

The details of how procedures or sub-programs work vary from one language to another. The common feature is that the variables within a procedure are localised within that procedure. How they communicate with the rest of the program, and what happens to them when the sub-program has run its course varies from language to language. XB goes its own well defined way, but is not at all flexible in how it does it.

Now let's look at how Extended Basic handles sub-programs. The RUNning of any XB program goes in two steps. The first is the prescan, that interval of time after you type RUN and press ENTER, and before anything happens. During this time the XB interpreter scans through the program, checking a few things for correctness that it couldn't possibly check as the lines were entered one by one, such as there being a NEXT for each FOR. The TI BASICs do only the most rudimentary syntax checking as each line is entered, and leave detailed checking until each line is executed. This is not the best way to do things but we are stuck with it and it does have one use. At the same time XB extracts the names of all variables, sets aside space for them, and sets up the procedure by which it associates variable names with storage locations during the running of a program. Just how XB does this is not immediately clear, but it must involve a search through the variable names every time one is encountered to trade off speed for economy of storage.

XB also recognizes which built-in sub-programs are actually CALLed. How can it tell the difference between a sub-program name and a variable name? That's easy since built-in sub-program names are always preceded by CALL. This is why sub-program names are not reserved words and can also be used as variable names. This process means that the slow search through the GROM library tables is only done at pre-scan, and Basic then has its own list for each program of where to go in GROM for the GPL routine without having to conduct the GROM search every time it encounters a sub-program name while executing a program. In Command Mode the computer has no way provided to find user defined sub- program names in an XB program in memory even in BREAK status. XB also establishes the process for looking up the DATA and IMAGE statements in the program.

Well then, what does XB do with user sub-programs? First of all XB locates the sub-program names that aren't built into the language. It can do this by finding each name after a CALL or SUB statement, and then looking it up in the GROM library index of built-in sub-program names. You can run a quick check on this process by entering the one line program

```
100 CALL NOTHING
```

TI Basic will go out of its tiny 26K brain and halt execution with a BAD NAME IN 100 error message, while XB, being somewhat smarter, will try to execute line 100, but halts with a SUBPROGRAM NOT FOUND IN 100 message.

The XB manual insists that all sub-program code comes at the end of the program, with nothing but sub-programs after the first SUB statement (apart from REMarks which are ignored anyway). XB then scans and establishes new variable storage areas, starting with the variable names in the SUB xxx(parameter list), for each sub-program from SUB to SUBEND, as if it were a separate program. It seems that XB keeps only a single master list for sub-program names no matter where found, and consulted whenever the interpreter encounters a CALL during program execution. Any DATA statements are also thrown into the common data pool. Try the following little program to convince yourself.

```
100 DATA 1
110 READ X :: PRINT X :: READ X :: PRINT
X
120 SUB NOTHING
```

```
130 DATA 2
140 SUBEND
```

When you RUN this program it makes no difference that the second data item is apparently located in a sub-program. IMAGEs behave likewise. On the other hand DEFed functions, if you care to use them, are strictly confined to the particular part of the program in which they are defined, be it main or sub. During the pre-scan DEFed names are kept within the allocation process separately for each subprogram or the main program. Once again try a little programming experiment to illustrate the point.

```
100 DEF X=1 :: PRINT X;Y :: CALL SP(Y)
:: PRINT X;Y
110 SUB SP(Z) :: DEF X=2 :: Z=X :: DEF
Y=3
120 SUBEND
```

This point is not explicitly made in the XB manual and has been the subject of misleading or incorrect comment in magazines and newsletters. A little reflection on how XB handles the details will usually clear up difficulties.

TI BASICs assign nominal values to all variables mentioned in the program as part of the prescan, zero for numeric and null for strings, unlike some languages (some Basics even) which will issue an error message if an unassigned variable is presumed upon. This means that XB can't work like TI LOGO which has a rule that if it finds an undefined variable within a procedure it checks the chain of CALLing procedures until it finds a value. However, unlike Pascal which erases all the information left within a procedure when it is finished with it, XB retains from CALL to CALL the values of variables entirely contained in the sub-program. The values of variables transferred into the sub-program through the SUB parameter list will of course take on their newly passed values each time the sub-program is CALLed. A little program will show the difference.

```
100 FOR I=1 TO 9 :: CALL SBPR(0):: NEXT
I
110 SUB SBPR(A):: A=A+1 :: B=B+1 ::
PRINT A;B
120 SUBEND
```

The first variable printed is reset to 0 each time SBPR is called, while the second, B, is incremented from its previous value each time. Array variables are stored as a whole in one place in a program, within the main program or sub-program in which the DIMension statement for the array occurs. XB doesn't tolerate attempts to re-dimension arrays, so information on arrays can only be passed down the chain of sub-programs in one direction. Any attempt by a XB sub-program to CALL itself, either directly or indirectly from any sub-program CALLed from the first, no matter how many times removed, will result in an error. Recursive procedures, an essential part of TI LOGO, are NOT possible with XB sub-programs , since CALLing a sub-program does not set up a new private library of values.

All of this discussion of the behaviour of TI Extended Basic comes from programming experience with Version 110 of XB on a TI-99/4a with 1981 title screen. Earlier Versions and consoles are not common in Australia, but TI generally seems to take a lot of trouble to keep new versions of programs compatible with the old. On the other hand TI has also been very reticent about the details of how XB works. The Editor/Assembler manual has very little to say about it, less by far even than it tells about console Basic. I am not presently aware of any discussion of the syntax of the Graphics Programming Language (GPL), let alone of the source code for the GPL interpreter which resides in the console ROM of every 99/4a.

Another simple programming experiment will demonstrate what we mean by saying that XB sets up a separate Basic program for each sub-program. RUN the following

```
100 X=1 :: CALL SBPR :: BREAK
110 SUB SBPR :: X=2 :: BREAK :: SUBEND
```

When the program BREAKs examine the value of variable X by entering the command PRINT X, and then CONTinue to the next program BREAK, which this time will be in the main program, where you can once again examine variable values.

We will now summarize the properties of XB sub-programs as procedures in complete XB programs, leaving the details of joining up the various procedures to the next section.

(a) XB treats each sub-program as a separate program, building a distinct table of named (REFed) and DEFed variables for each.

(b) All DATA statements are treated as being in a common pool equally accessible from all sub-programs or the main program as are also IMAGE statements, CHARacters, SPRITEs, COLORs, and File specifications.

(c) All other information is passed from the CALLing main or sub- program by the parameter lists in CALL and SUB statements. XB does not provide for declaration of common variables available on a global basis to all sub-programs as can be done in some languages.

(d) Variable values confined within a sub-program are static, and preserved for the next time the sub-program is CALLed. Some languages such as Pascal delete all traces of a procedure after it has been used.

(e) XB sub-programs may not CALL themselves directly or indirectly in a closed chain. Subject to this restriction a sub-program may be CALLed from any other sub-program.

(f) The MERGE command available in XB with a disk system (32K memory expansion optional) allows a library of XB sub-programs to be stored on disk and incorporated as needed in other programs.

> by Tony McGovern and reprinted from the LEHIGH 99 Newsletter #9<

IMPACT-99!

TEXTWARE, SOFTWARE, and ELSEWHERE
Happenings in the T.I. World Community
by JACK SUGHRUE

Some of my user-group friends ask me often about the graphic designs and fonts I use in my letters and in this column. "How can you make *that* with your printer?" they usually ask.

The answer is simple and not so simple. The different fonts built into my Gemini 10X are fairly easy to get to. I made a template of my Transliteration Key and access it through my *T.I.WRITER* (in this case, the *FUNLPLUS!* version, though any form of TIW will work). This lets me do the following:

Underline when and where I like

Be in condensed

Or enlarged

Or italics

Or letter quality. This is what you're reading. This is normal printer type. Obviously this is preferable to most people.

You may even combine:

Underlined italics, for example.

All this is simply done automatically through IFfing the template which has the TL codes on them.

If persons are interested I could devote a column or two to this kind of thing, complete with all the codes and how to get to them automatically. Let me know.

But the fancy stuff like the header of this column is what most people would like to be able to do.

These kinds of things (pictures, fonts, designs, labels, letterheads, signs, banners) are all done with some very special programs. Compared to the costs of similar programs for other computers they are very inexpensive, but they are not free.

I'll be devoting the next few columns to explaining how the different packages work and showing examples. These packages are the ones I use: GRAPHX, TI ARTIST, FUNLPLUS!, BETTER BANNERS, FONTWRITER, and CSGD. I use the last (CHARACTER SETS AND GRAPHIC DESIGNS III) to make the IMPACT-99! logo.

Of the six graphics/text programs listed above which is the best? None is *better* than the other. Each does something different than the others, but now - thanks to FONTWRITER - all the works done by any of them can be tied together. More on that when I take up FONTWRITER.

But first let's discuss Broderbund's PRINT SHOP. This program, which is out for most computers, is not available for the TI.

Everyone loves PRINT SHOP. It is so user-friendly a four-year-old could operate it beautifully the first time around. The program makes greeting cards, banners, signs, etc. with incredible error-free ease. And there are now hundreds (including a lot of PD files) of graphic designs and fonts for the program. The commercial disks for the whole works for the IBM or Apple is well over $200, the basic program under $50. People who HATE and FEAR computers LOVE to print with PRINT SHOP.

But you pay a price for convenience. I don't mean just money, though that is *definitely* a factor. The price you pay is sameness. A PRINT SHOP anything looks, well, printshoppy. There is not much flexibility. Granted, you can choose your font; you can choose your graphics; you can even choose size, form of font, layout - all to a limited degree. Limited.

With the five commercial programs above (and my FUNLPLUS!) I am unlimited. I can create virtually any type of font or graphic that can fit on a printed page and lots that extend beyond. I can couple any of the above with my

word-processor to create dazzling text/graphic printouts, a lot like the MacIntosh stuff. (CSGD III even lets you use TIWRITER versions to type in six new, large, unusual fonts in full or split columns WITH the graphics!) But there is a price. Ease. All this stuff requires reading, learning, experimenting, time. For people willing to put the time in and invest a few dollars (GRAPHX, FONTWRITER about $25 each; TI-ARTIST, BETTER BANNERS about $20 each; FUNLPLUS! $8; CSGD about $18; and various companion disks from $7.95 [from Asgard] to $18). Probably a basic structure for someone with an Epson-compatable printer who is just starting out with this exciting design world would be FONTWRITER, FUNLPLUS!, TI-ARTIST, and CSGD III. That $68 investment would give you far greater flexibility than PRINT SHOP. It would also be a fairly reasonable task to set for yourself to learn how to use these pieces of software. Each has excellent documentation providing you take the trouble to do all the things along with the manual while you are learning. (Though, to be honest, I am always too anxious to get into my new "toy" when I get it to look at the manual. I should follow my own advice because I inivitably have to redo everything because I wouldn't take the trouble to read the documentation BEFORE using the programs.)

If you've never designed your own letterheads or labels or drawn and printed any graphics with the incredible GRAPHX and/or TI-ARTIST then you are in for a real treat. Your computer is far more powerful than you have yet imagined. (Asgard even has an automatic slide-show for the GRAPHX pictures to amuse your friends and amaze your enemies.)

In short, the world is your oyster if your TI is working toward its potential.

We'll discover a lot about that potential in the next few columns.

3,2,1 IMPACT! until next time.

(Jack Sughrue, Box 459, E.Douglas, MA 01516)

COOKIES

THE MINI-SERIES THEY SAID COULD NOT BE DONE!
Combining Chemistry, Math, Art, and Good Taste
The Math Class Proves They Have Good Taste!

FUNLPLUS

JACK SUGHRUE
BOX 459
EAST DOUGLAS MA 01516

MIKADO

THE ORIENTAL GAME OF PICKUP STICKS

MUNCH OFFICERS AND NUMBERS (all in 617 area)

President	Wm. Corson Wyman	839-4134
Vice President	Hector Beaudreau	
Secretary	Al Cecchini	
Treasurer	Jim Cox	869-2704
Editor	Jack Sughrue	476-7630
Hardware Chair		
Programs Chair		
Adv Prog. Chair	Dan Rogers	248-5502
Club Reviewer	Jack Sughrue	--------
Library	Al & Lisa Cecchini	--------
Software Library	Don Mason	754-6630
	Hector Beaudreau	
Mail & Messages	Wm. Corson Wyman	--------

****

LIBRARY NOTICE

Several items in our library have been missing for some time. Everyone, PLEASE check around your home for materials that might belong to the library. If you find any, please bring them to the next meeting, or call an officer to have someone pick up the item(s). Some of these items were only loaned to our library and should be returned to their rightful owners.

****

ADVERTISING RATES:

Double Page	(10.5" by 8")	$25.00	per insertion
Full Page	(5" by 8")	$13.00	per insertion
Half Page	(5" by 4")	$ 7.00	per insertion
Quarter Page	(5" by 2" or (2.5" by 4")	$ 5.00	per insertion

Classified (non-commercial) ADs are FREE for MUNCH members.

****

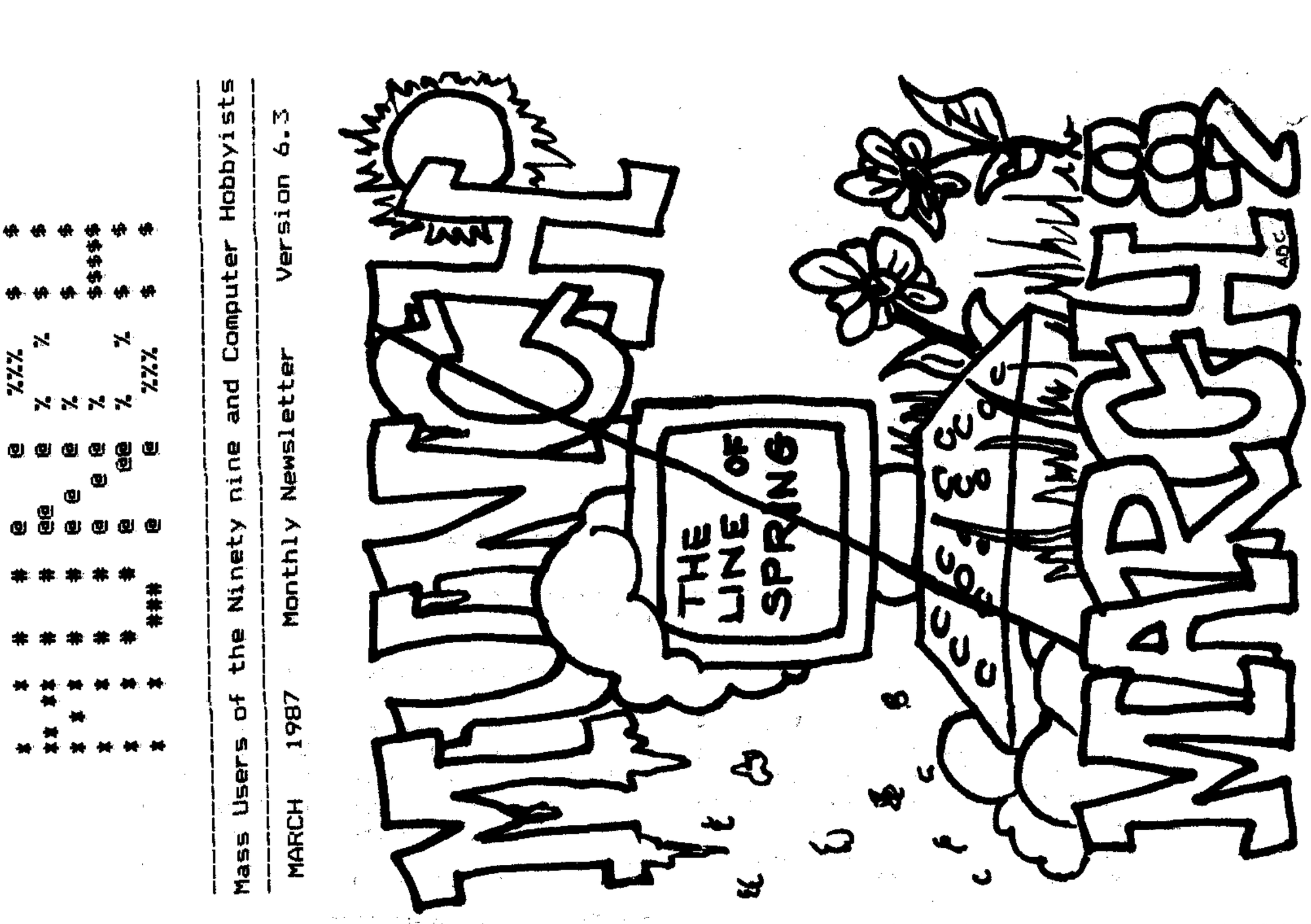
...RAFFLE...RAFFLE...

This month we will have a Very Special Raffle!

The raffle is open to all who attend. The drawing will be held just prior to the business meeting. Remember:

***** YOU MUST BE PRESENT TO WIN *****

****



M.U.N.C.H.
P.O. Box 7193
560 LINCOLN STREET
WORCESTER, MA. 01605

FIRST CLASS

****
IMPORTANT NOTICE
****

MARCH meeting will be on March 17th, 1987 (*Begorra!*)
at University of Massachusetts Medical Center
(Come to the VISITORS entrance and follow the signs for MUNCH...)