

HOOSIER USERS GROUP USERS GROUP HOOSIER GROUP HOOSTER USERS HOOSIER USERS GROUP

HOOSIER USERS GROUP HOOSIE

#### THE HUGgers HOOSIER USERS GROUP People Helping People

GROUP HOOSIER USERS HOOSIER USERS GROUP USERS GROUP HOOSIER GROUP HOOSIER USERS HOOSIER USERS GROUP USERS GROUP HOOSIER GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER USERS

May 1991

The HUGgers Newsletter

Volume 10, Number 5

JUST WHAT I NEED

hv

Allan Cox

I imagine that most computer users have, at one time or another, wished that they had a program that would do something different than what they have, or wished they knew where to find the program they need. I they looked inward, they would find it in their own mind. Yes, I am referring to writing their own program.

At the mention of writing a program, the first reaction with some is that it is too complicated. A program is a combination of a lot of VERY SIMPLE statements or commands, that are arranged in such a way as to accomplish a purpose. Creating a program can be a joy and satisfaction that is renewed each time you use it. You can have the satisfaction of knowing that others can also enjoy using it.

When the TI computer became so popular, and user's groups abounded, my wife and I belonged to the TIBUG user's group here in Birmingham. We had several excellant programers, who taught us a lot about programming. My first program was a sprite program of a red airplane flying around clowds that I called RED BARON. My wife wrote a similar one of a RED BALLOON. This broke the ice for us. They were short and simple, and we suprised ourselves with them.

One of our members wrote the first LEAPFROG game, and he also wrote the first BLACKJACK program, and both were first class programs. Another member wrote a COUPON program, for keeping track of product coupons. Most members wrote some kind of program.

Doing genealogical research as a hobby, the first program that I wrote was FERFETCAL, a perpetual calendar program that tells you what day of the week a date was. It was short and simple, worked fine, and filled my need. I have seen other similar programs since then.

I then wrote BALLRANK, a program for calculating the scores of college football games. In four years I have calculated the scores of 1841 games and had 1321 correct, for almost 72% correct. Figuring 35 to 40 games each week gives me something to do with myself, instead of sitting around like a bump on a log. It keeps my mind active, and each week I look forward to how well I have done, so it fills another need of mine.

Two other genealogy programs are STIRPSLINE and GENBUG PRINTOUT. STIRPSLINE is used to keep records of each of my ancestors, and GENBUG PRINTOUT is used to print family sheets from my GENEALOGY WORKSHOP records.

I have written other programs for my different needs by taking the time to figure out how I would do it, and then went ahead and done it, and so can you. There is someone else out there that has the same need as you, and you can help them by sharing. Put it on a disk and if you cannot take it, then send it to your user's group Editor.

The Editor, or someone clse, should write a review of the program so that the members will have a concept of what it does. Your Editor will be thankful to you for submitting it to him. Also, the members will thank you for being thoughtful enough to allow them to have a copy of it. So give it a try, and fill your needs. See how much happier you will be!!!

Editor's Note: This was downloaded from Delphi on March 12, 1991 and concerns a new procedure for "Return Authorization Numbers" for repairs on Myarc products. It has been slightly altered for ease of reading. It was posted as Message #20307 5-JAN 13:02 9640 GENEVE

JAN. 5, 1991 RMA'S and HARDWARE From: MYARC, Inc. To: ALL

As always please ship to the following address:

Myarc Inc. 2624 Ranier Drive, N.E. Birmingham Al. 35215

Office phone is 205-854-5843 with the RMA,s being given in the following manner the phone should be free for other problems.

Under the new procedure your RMA number will be given by you, it is the DATE SHIPPED with the LAST FOUR DIGITS of your PHONE NUMBER. For example: if I shipped today the number for me would be 01/05/90 0173 the last four digits are my home phone number. Please include key by key what the problem is and include your full address and zip code, and a daytime phone number. Also, if you want to know if your package has arrived please ask the carrier to do this so that they will verify the arrival of your package and let you know.

## \*\*\* ATTENTION \*\*\*

There is a new newsletter available now supporting the Mechatronics 80 column card, Dijit AVPC, Asgard EGI, and OPA's TIM. It is Called: Bits & Bytes.

Bits & Bytes is a monthly publication put out by the South Bay TI Users Group of Silicon Valley that specializes in 80 column card support. Every month it is packed full of the latest industry news, programming articles and projects for you to get more out of your 80 column TI or Geneve. If you would like to know more about this newsletter, or would like to buy a subscription, please write to:

Bits & Bytes Newsletter P.O. Box 11037 Campbell Ca, 95011-0037

A one year subscription which included 12 issues of 10-20 pages is \$15. Back issues are \$1.25 and are avaiable back to January 1991. If you would like to begin your 12 month subscription with the January issue, please indicate and we would be glad to.

This publication is the publication of the South Bay TI Users group and contains meeting information and UG business as well as 80 column news.

[ EDITOR'S NOTE: This was downloaded from Delphi on 4/20/91 ]

THIS ARTICLE IS REPRINTED FROM TIGERCUB NEWSLETTER PROGRAMMING MUSIC THE EASY WAY

PART 2

by Jim Peterson

In Part 1 I showed you how to set up a musical scale to create notes, and how to merge in various little routines to create a variety of musical effects, but I didn't tell you how to figure out what numbers to put in between those GOSUBs. So, here is the little program that makes it all easy.

100 CALL CHAR(127,"000F080F0 868F870**000F0**8080868F87000080 8080868F**870**0008080808689870" ):: CALL CHAR(131, "0000000000 0609070") 110 CALL CHAR (132, "000012004 8302040**0000**22100810200000201 0201030**2000**000003CFF"):: CALL CHAR(136,"000000FF3C") 120 CALL CLEAR :: S\$="GFEDCB A" :: CALL CHAR(45, "000000000 FF"):: 7\$=RPT\$(S\$,3):: FOR R =2 TO 22 STEP 2 :: IF R=12 T HEN 130 :: DISPLAY AT(R, 1):R PT\$("-", 28) 130 NEXT R :: CALL CHAR(98," 0020202834242830") 140 FOR R=1 TO 21 :: DISPLAY AT(R, 1):SEG\$(A\$, R, 1);:: NEX T R 150 DATA 127, 127, 128, 128, 129 , 129, 130, 130, 131, 131 160 DATA 1/16, 1/8, 1/4, 1/2, 1/ 1 170 FOR R=1 TO 20 STEP 2 :: READ N :: DISPLAY AT(R, 15) :C HR (N); :: NEXT R :: FOR R=3 TO 19 STEP 4 :: DISPLAY AT(R ,16):".";:: NEXT R 180 C=132 :: FOR R=1 TO 17 S TEP 4 :: DISPLAY AT(R, 17):CH R\$(C);:: C=C+1 :: NEXT R 190 FOR R=1 TO 17 STEP 4 :: READ M\$ :: DISPLAY AT(R, 20): M\$::: NEXT R 200 DATA 35,33,32,30,28,27,2 5, 23, 21, 20, 18, 16, 15, 13, 11, 9, 8, 6, 4, 3, 1 210 FOR R=1 TO 21 :: READ N :: N=N+CHR+(N):: DISPLAY A T(R,6):STR\$(N);:: NEXT R 220 G\$="b" :: Z=-1 >: GOSUB 320 :: IF F=0 THEN 230 ELSE COSUB 330 :: 6010 240 230 G\$="#" :: Z=1 :: GOSUB 3 20 :: ∠TF F()0 THEN GOSUB 330 240 DISPLAY [ (24, 1): "Shorte st note? 1/" ;; ACCEPT AT (24 ,18) VALIDATE ("12468") SIZE (2) BEEP:L ::

T\$="1/"STR\$(L):: RESTORE 160 :: FOR J=1 (U 5 :: READ L\$ :: IF L\$=T\$ THEN 260 250 NEXT J :: GOTO 240 260 DISPLAY AT(24,1):"Is it dotted? Y/N" :: ACCEPT AT (24 , 19) VALIDATE ("YN") SIZE (1):D\$ :: D=1-(D\$="Y") 270 T=-3+J\*4 280 FOR R=T TO 19 STEP 4 :: DISPLAY AT(R, 11):STR\$(D);:: DISPLAY AT(R+2,11):STR\$(D\*1. 5);:: D=D\*2 :: NEXT R 290 GOTO 360 300 FOR R=1 TO 20 STEP 2 :: READ N :: DISPLAY AT(R, 15):C HR\$(N);:: NEXT N 310 GOTO 310 320 DISPLAY AT(24,1): "How ma ny "G\$" on upper scale?" : : ACCEPT AT (24, 28) VAL IDATE (" 01234567")SIZE(1)BEEP:F :: R ETURN 330 Y\$="" :: FOR J=1 TO F :: DISPLAY AT(24,1):"On which letter?" 340 ACCEPT AT(24,18)VALIDATE (S\$)SIZE(1)BEEP:L\$ :: IF POS (Y\$,L\$,1)()0 THEN 340 ELSE Y \$=Y\$<u>L\$</u> 350 S=1 :: FOR K=1 TO 3 :: P =POS(A\$,L\$,S):: DISPLAY AT(P ,2):G\$;:: DISPLAY AT (P, 6):ST R\$(ASC(SEG\$(N\$,P,1))+Z);:: S = P+1 ::NEXT K :: NEXT J :: RETURN 360 OPEN #1:"PIO" :: FOR R=1 TO 22 :: FOR C=3 TO 30 :: C ALL GCHAR(R,C,G):: CALL HCHA R(R,C,30):: R\$=R\$CHR\$(G):: NEXT C :: PRINT #1:R\$ :: R\$= "" :: NEXT R :: STOP Get yourself a piece of sheet music and compare it to the screen display from that program. You will see that music is written on two sets of 5 lines. The upper set is marked at the left end with something like a fancy script capital S; it is used to write the higher notes, including the melody, which a pianist plays with the right hand. The lower set, marked with a sort of a backward C, contains the luw notes played with the left hand. Your sheet music probably has a wide space between the sets, to make room for the lyrics, but there are really only three notes between them. The screen display shows letters at the left, which are not on the sheet music. Those are the names of the notes, which we will have to refer to a couple of times to get started: observe that the notes are named A through G and then repeated. The numbers along the left side are the numbers you will key in to play those notes. However, the screen display is set up in the key of C.

which is played entirely on the piano white keys. The sheet music you want to program from may be in a different key, so -

The computer is asking you how many there are of something that looks like a squashed lower case b — I guess that's why they call it a flat? It means that the note will be played a bit lower, on the black key just left of the white key — and we will program it one number lower. So, look next to that capital S and see how many flats there are. If none, type Ø. Otherwise, the computer will ask which letters they are next to. Type them in, one at a time, and presto — the computer will put them on the staff and adjust the numbers accordingly.

If there were no flats, the computer will want to know if there are any sharps — those are what you get by typing a shift 3 on the keyboard, and they mean that the note is played on the black key above the white key, and is programmed one number higher. Now, the computer needs some information in order to help you set up the length of your notes - how long they are sounded. The various notes are depicted at the right. A 1/16 note is a little black egg with a stem (it may go up or down, makes no difference) and two flags on the stem. A 1/8 has only one flag and a 1/4 note has none. A 1/2 note is a hollow egg with a stem and a whole note has no stem. Those little doodads to the right of the notes are rests, used to indicate a silent pause of the same length as that note - more on that

Look through your sheet music and find the shortest note. Tell the computer. It will want to know if any of those shortest notes are dotted — have a little dot to their right, as the screen display shows. A dotted note is played half again as long as normal. Presto again, the computer will show you the duration number to key in for each note. Then, if you have a printer attached, it will print out an XBasic screen dump of that screen — you will have to squash your own b's and sketch in the notes and rests.

later.

If your software library contains an assembly screen dump, delete that la 👈 program line and put in a CALL INIT, CALL LOAD and CALL LINK to get a better printout - or ask me for it. If you don't have a printer, why not copy those numbers right onto the corresponding lines and spaces on your sheet music, and number some of the Now we're ready to make music! notes. Let's keep it simple at first, just a single note melody - and I hope you picked a simple piece of music. Clear the TI's brain with NEW, then merge in that line 100 scale from part 1 by MERGE DSK1.SCALE . In the same way, merge in one of those line 1000 CALL SOUND routines. Fut in a temporary stopper line 999 STOP, and a line 110 D=200 to set the duration. The melody is almost always on the upper set of 5 lines. If a note has 2 or 3 eggs on its stem, as they usually do, the upper one is the melody note we will get into harmony later. Start with line 110. Check your chart to see what number denotes the length\_ of the first note - maybe 2, if so k in T=2 :: Then check to see what number applies to the position of the upper egg of that note. Maybe 22, so key in A=22 :: GOSUB 1000 Enter RUN, and if you've done everything correctly, you will hear the note. You might decide already that you want to change that 200 in line 110. Now for the second note. If it is of the same length as the first, you don't have to type anything - that's what makes this shorthand method so quick and easy. If the note position is also the same, you don't key that in either - just another GDSUB 1000. If you have EZ-KEYS or another "hot keys" program, you can program a control key to put in the GOSUB 1000 with just one keypress — wish I had thought of that when I was programming music by the diskfull! So keep plugging along, keying in durations and notes. After every half dozen notes or so, type RUN to see if everything sounds OK so far - it's easier to catch errors before they as .. too far back in the music. You can get up to 5 screen lines on

\_stick to 3 lines. You will note that he sets of notes are divided by vertical bars. You might program the notes between bars on a separate line. then add a ! followed by the words of the song that go with those notes — I find that a very good way to track down sour notes. Regarding those bars — it might help you sometime to know this. At the heginning of the music, right after the big script S and the flats and sharps, you will see something like a 3 over a 4, or a 4 over a 4, or whatever – but often a symbol such as a barred C is used instead. A 3 over a 4, for instance, means that the notes

one line number, but you might better

to 3/4 - might be three quarter notes, or two eighth notes and two quarter notes, or whatever, but they will add up to 3/4. Sometimes the very first notes will add up short, but in that case the very last ones will make up the difference.

between two of those bars will add up

The notes between those two bars make p a bar of music, and the emphasis is on the first note — for instance, that 3/4 is the 1-2-3, 1-2-3 beat of waltz time.

While you are keying in that music, you might come to one of those rests. You can just key in its T= value and then A-Ø for a silent note. However, computer notes stop so abruptly that somehow a rest just doesn't sound right, so I often just use the previous note instead.

You may come across one of those flat or sharp symbols next to a note in the music. Give the note a number 1 lower if a flat, one higher if a sharp, and the same for any subsequent occurrences of that note, until you find next to it a symbol that looks like the sharp sign with half its legs knocked off; that means to go back to normal. You might also come across that symbol to tell you to play a normally flat or sharp note as if it was not.

I think that covers all that you bsolutely have to know for now, and I have horrified all serious students of music just about enough. There are all

kinds of other squiggles on the sheet music but usually they are not essential in programming music. There is one other time-saving shortcut that I should tell you about right now. Most music consists at least partly of musical phrases, of a series of notes, which are repeated two or more times within a melody. So, the first thing you should do before you start programming a song is to search through the music for such phrases.

If you find one, of more than a few notes, that is repeated elsewhere — and make sure it is repeated exactly the same — mark it off each place it occurs and label it 500. If you find a second repeating phrase, label it 600, and so on.

Then, when you start programming, start with line 500, key in that series of notes first, and end it with RETURN. If you have another phrase, put it in lines starting with 600, again ending with RETURN.

Now, start programming from the beginning of the song in line 120, but when you come to one of those phrases, just put in GOSUB 500 - the program will jump to that line number, play those notes, and come right back to where it was.

In Part 3, we will get into programming in 3-part harmony, bass notes, auto-chording, and all kinds of things.

MONTHLY MEETING LOCATION
LITTLE HOUSE NEXT TO THE
ST. ANN'S SCHOOL
2839 S. McCLURE
INDIANAPOLIS, IN
MEETING STARTS
AT 2:00 P.M.
NO MAY MEETING

DOWNLOADED FROM: COMPUSERVE

## BACK TO BASICS TI EXTENDED BASIC MICROSOFT GWBASIC.

## A Discussion by Art Byers

Last year I wrote a four part series extolling the virtues of the Basic language and of TI Extended in particular for use on the 99/4A. Those pieces have been reprinted in many TI user group newsletters, so most 99'ers are well aware that Basic is one of my favorite computer languages.

It is only natural, therefore, that I would investigate Microsoft GW Basic which is most often supplied as a "freebee" when an IBM Compatable PC is purchased. In the small space of a computer club newsletter, I cannot go into depth of detail. After all the GW Basic manual I have is well over 400 pages, and TI-XBasic manual with addendums is over 240 pages. My intent is to give you my own views and let you go further from there, if you so desire.

I have heard many conflicting stories about the origination of Basic for the Here is what I believe is most likely true: Regular TI-Basic, Console Basic as it is often called, was written by Microsoft for Texas Instruments. TI Extended Basic, which is obviously based on TI-Basic, was written by a team of TI in-house programmers. It is no surprise. therefore, that GWBasic and TI-XPasic are very much alike. They are so similar that many of the more simple programs can run on both TI and PC systems with only minor syntax modifications.

Here are two examples: (1) XB uses a double colon "::" as a separator for multi statements on the same line. GWB uses a single colon ":". (2) CWB uses a semi colon after text in an INPUT where XB uses a colon. ie: in GWB 100 INPUT "Enter your name":N\$ — in XB 100 INPUT "Enter your name":N\$.

The overwhelming majority statements, commands and functions are either identical or there is a simple substitution of name such as SEG\$ in TIXB and MID\$ in GWB. Both versions can link to assembly routines. Both can avail the programmer of useful peeks and pokes into varied memory locations, Both have the ease of debugging program generic to interpreted languages. All this is welcome news to a Basic devotee interested in transporting programs from one make of computer to another.

In fact, when reading the rest of this article, keep firmly in mind that MOST of XB and GWB statements, commands and functions are the same right down to the last parenthesis and comma!!!

# WHICH IS MORE POWERFUL?

Although There are areas where consider TI-XB to be definitely much superior to GWB, there can be no argument or question that overall GWB has more features. statements. commands and available memory than TI-XB. (As an aside I'll say that the new Advanced Basic for the MYARC 9640 is better than and more powerful than GWB. However, most of us are using the 99/4A and must XBasic.) use Nevertheless, if you suppliment TI XB with some assembly graphics routines Extended Basic not only can hold its own very well, but also in two very important respect outstrips Let's take a quick look.

# Amount of Memory Available.

First of all, sheer memory space available for Basic programming and required array storage is 60k in GWB as compared to about 37k in XB. Of the 48k memory in the 99/4A, Extended Basic uses either 768 bytes or 960 bytes of VDP RAM just for screen display (32 or 40 column mode). VDP RAM is also used for string variable array storage, DSR buffers, line number tables, and color and screen tables, etc. Effectively, if you have opened a line to a printer and opened a couple of disk files (at 512 bytes each) you can safely figure at least

13k available for string storage in VDP RAM. XB uses 24k high memory for programming as well as storage of numerical arrays and constants. lower 8k is there for assembly support linked to basic programs but not for Basic programming. Even if we make the usual assumption that XB has the full 48k of RAM available one way or another for use by XB programming, GWB has 25% more. You may laugh that a 640k RAM PC has so much overhead that only 10 percent is available for Basic programing. That is not really the point. GWB simply has more memory available, period! (Actually the TI has more than 90k total memory with various ROM chips, but let's not get into that right now as no way does it. have 640k of cpu addressable RAM).

The above is much mitigated by the fact that the 99 was designed as a HOME computer. Very few programs written for home use should require 60k of memory. In six years of programing, I have NEVER run out of memory on the 99/4A for an XB program — though at times I had to do memory squeezing tricks and be very efficient with my code. If the amount of digging you have to do will be handled by a garden spade, is not a deisel powered back hoe overkill?

### WHERE GWB SHINES?

Many many more features make it an easy and convenient programming language Let me give you just a few examples:

- (1) GWB simply has more goodies for the programmer such as statements and functions to use the PC's built in clock including retrieval of time and date, elapsed time, as well as the ability to set the clock. Another useful statement is SWAP, which facilitates sorting data. There are may more examples I could list here.
- (2) a GWB program can use Shell commands to temporarily leave the Basic program and use a batch file, or use the many MSDOS commands such as reading directories, rename files or disks, etc and return to the execution

- of the Basic program when the outside task has been completed.
- (3) The 99/4A has excellent file handling. So does GWB but XB lacks GWB's ability to LOCK and UNLOCK files. Even woefully deficient Apple II Basic has that feature. Why Microsoft left it out of TI Basic, I do not know. Another omission from XB is the RESET statment to close all open files and update the disk directory. That is an important adjunct to any error trap routine or any program that requires swapping of disks.
- (4) A whole host of graphic commands such as CIRCLE, DRAW, PAINT etc are part of GWB. To use these in TI XB we must link to assembly routines such as Quality Soft's Draw and Plot Mechtronics Expanded Graphics which contains many links to Basic. latter uses so much overhead, however, that the memory remaining for XB programming is severely restricted. An fine alternative is the Extended Basic Module automatically dumps graphic routines into lo-memory and has other enhancements.
- (5) Another real plus for GWB is the ability to use a few simple Basic statements and commands to save a whole screen onto disk, to recall the screen. text or graphics make no difference.
- (6) There are many more features made possible because GWB is a larger version of Basic. These include additional looping commands such as WHILE/WEND. All I can advise is that you take a look at a manual in a bookstore if you want to investigate further.
- (7) What may well be GWB's strongest point is that it is tied to a very powerful MS-DOS. It is this DOS connection that may be, for a majority of Basic programmers, the deciding factor in favor of GWB.
- (8) GWB is substantially faster in execution than TI XB. I have written

articles before on "Apparent Speed". I won't repeat them here more than to say I consider TI XB sufficiently fast for almost every home use except Telecommunications. A Telecom program can be written in GWB that will handle up to 2400 baud, (though probably best only up to 1200 baud as you must program in stop/start (Ctrl S and Ctrl Q) transmissions to the remote to allow GWB to catch up with itself at 2400 baud). Above that a compiled laguage program is required. XB could handle 110 baud or 300 baud. A LIMITATION OF GWBASIC Much to my surprise, GWB limits you to a maximim of 255 dimensions in an array, (DIM A\$(255)), This applies to both string and numeric variables. To those of us in XBasic used to making DIMension statements much larger than this, either single (DIM A\$(400)) or multi dimensional (DIM A\$(100,4)) this is a potentially disabling choke collar on a very fast greyhound. I am inclined to believe this is deliberate. If the TI with its limited RAM can dimension huge arrays, certainly Microsoft could have done the same with anywhere from 256k to 640k in the standard XT clone. Why?? Perhaps because they contemptuously don't think of the home user as writing sophisticated programs that would require large arrays. Perhaps they want to be sure that for sophisticated programs, users must go out and BUY software written in other languages, for it is a truism that the average user will program in Basic, IF they program at all. There are ways of programming around this restriction but they sure make things unnecessarily complicated and difficult.

#### WHERE XB SHINES

TI-XB has several areas where even after all these years, it still outstrips GWB.: (1) The 99/4A, although a design of the 70's, was far ahead of its time in graphics and graphic definition. Even without the high number of pixels that the latest PC's have on their monitor screens, the TI can hold its own. What is more, the 99/4A's use of and ability to control Sprites is the equal or superior, even today, of many new computers. The normal IBM clone does not yet have sprites!!! What is

more, all this graphic and sprite capability (such as the MAGNIFY command) is part of XB. GWB does not have sprites. (2) XB statements and commands for screen display and and input are superior on the TI. I consider TI's DISPLAY AT(row, column) SIZE() very much better than GWB's LOCATE(row, column, cursor, raster, raster). I think the ACCEPT at (row, column) SIZE ()VALIDATE() to be more logical and easier to use than GWB's methods. I even like the TI's CALL KEY much better than GWB's INKEY\$. It gives more information and is more powerful. (3) Here is the number one unequalled way that TI XB has it all over GWB: The ability to write truly structured programming by writing your own subprograms that you can CALL by the name you give them. For Example: CALL INTERESTCALC (PRINAMOUNT, RATE, PERIOD) is tons more meaningful than GOSUB 1300. This alone weighs so very heavily in favor of XB. That for many 99/4A (and 9640) users it may well be the deciding factor in making their judgement of which version of Basic is the better. So I suggest, again, that some time when you are near one of the very large book stores that have an extensive computer section. (B Dalton on Central Avenue Yonkers is the best in our area) you peruse any GWBasic manual. I suspect you may just come away feeling much better about TI's Extended Basic. \*

IF anyone is interested in the above equipment for sale, call me at:

# HUG OFFICERS

President Gary McQuade 888-5654 V.President Bryant Pedigo 255-7381 Secretary Jeff Overton 299-2333 Treasurer Walter Farmer 539-2679 Librarian Bryent Pedigo 255-7381

\***\*** # # PASTE UP PARTY # HELP GET THE NEWSLETTER # DUT # # FIRST FRIDAY OF JUNE # AT BOB STAHLHUT'S HOUSE # CALL 856-4962 # ###########################

#### DEC

Hoosier Users Group Baud rate: 300/1200/2400 On Line 24 Hours Daily

317-782-994A

Now with a Hard Drive 40 MEG ON LINE

#### JUNE MEETING

There will be "NO" June meeting. A Picnic is planned for members and their families, -- Poolside -- at Garvs home.

Date and time to be announced in the June Newsletter.

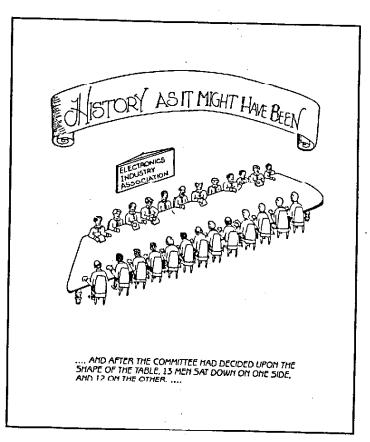
#### DISCLAIMER

This newsletter is brought to you through the efforts of the officers and members of the HOOSIER USFRS GROUP. Every member is encouraged to submit articles.

If you have an article you would like to share with the other members mail it to:

Bryant Pedigo 6461 N. Sherman Drive Indianapolis, IN 46220

Opinions expressed are those of the author and not necessarily those of the HOOSIER USERS GROUP.



Dan H. Eicher P.O.Box 605 Mooresville, IN 46158

Z6/KEW

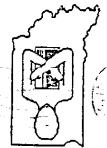
DATAO BMIT ONLYSSH VEH ON JAIRBITAN



Forwarding and Address Correction Requested

P.O.: Box 2222 / 46206-2222 <sup>M</sup>

HOOSIES OZEBS GBOOK



# APPLICATION FOR MEMBERSHIP

Below you will find an application for membership to the Hoosier Users Group, Active membership entitles you to the Newsletter, up and download on the HUGbbs, attendance and voting rights at regular club meetings, access to the HUGger Library of Programs, special club activities and special guest speakers for one year.

Make check or money order payable to Hoogier Users Group. Send completed application to:

HOOSIER USERS GROUP P.O. Box 2222 Indianapolis, IN 46206-2222

| please p  | rint                    |               |       | ,           |
|---|-------------------------|---------------|-------|-------------|
| Check One:  |                         | cut on line   |       | _           |
| Active Member   New: \$20   | NAME                    | YAQOT<br>TATE | _     | <del></del> |
| Renewal: \$18   Dues will be due in   May of each year. New   members subtract \$1.50 |                         |               | APT # | _           |
| <b>.</b>  | CITY                    | STATE         | ZIP   | `           |
| Amount   Enclosed   | PHONE ( )<br>INTERESTS/ | · .           |       |             |
| #DI<br>SI   | COMMENTS                | ·             |       | -           |