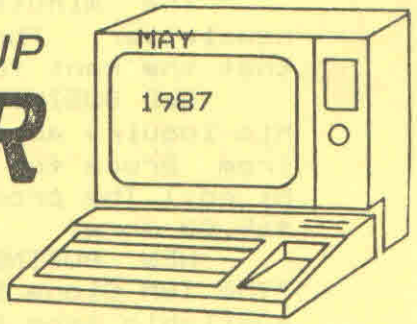
CEDAR VALLEY 99'ER USER GROUP NEWSLETTER

CEDAR RAPIDS/MARION, IOWA



****

OFFICERS

PRESIDENT:
Jerry Canady
6616 Kent Dr. NE
Cedar Rapids Iowa 52402
(319) 377-9382 (Home) or
(319) 395-2494 (Office)

VICE PRESIDENT:
Bruce Winter
702 Fernwood Dr. NE
Cedar Rapids, Iowa 52402
(319) 393-0610

SECRETARY:
Bill Paeth
923 Owen St. NW
Cedar Rapids, Iowa 52405
(319) 396- 6470

TREASURER:
Jim Harrington
4420 Tama St. SE #15
Marion, Iowa 52302
(319) 377-1865

COMMITTEES

PROGRAM:
Ed Edwards
102 N. Davis St.
Anamosa, Iowa 52205

PUBLICITY:
Paul Mortensen
3179 Country Park Dr.
Toddville, Iowa 52341
393-6022

EDUCATION:
John Johnson
398 Forest Dr. SE
Cedar Rapids, Iowa 52403
366-4541

EDITOR:
Jim Green
288 Windsor Dr. NE
Cedar Rapids, Iowa 52402
377-4073 (Home) or
395-1898 (Office)

****NEWSLETTER TOPICS****

1. Future Meeting Dates
2. Next Meeting Notes
3. Minutes From May Meeting
4. Reprint on XBasic Programming
5. Assembly Language Class
6. Library Corner
7. From the Mailbox
8. One Liners

****FUTURE MEETING DATES****

Please mark the following dates on your calendar for future meetings: JUNE 8, JULY 13, AUGUST 10.

********NEXT MEETING********

Monday, June 8, 7:00 PM at the JA building, 330 Collins Rd. NE. Ed's program will be a surprise. Library will be available for your use; if you bring your system, you will be assured of time to copy a program or show off your latest!

****MINUTES FROM THE MAY MEETING****

The Cedar Valley 99er User Group meeting was called to order at 7:07 PM on May 11, 1987 by President Jerry Canady. Eleven members were present.

The minutes of the April meeting were approved as printed in the newsletter. The treasurer's report was approved as read. It was noted that the rent for our meeting room has been paid for the next three months.

OLD BUSINESS--Bruce Winter announced that he had received an answer to his inquiry about quad density for the TI-99/4A. The info is available from Bruce for those who want to check it out. (Also see comments on page 8; ed.) The prom set would be about $45.00, and the drive would be about $69.00 more.

NEW BUSINESS--We received a new book for the library. The title is "The IBM Clone Buyer's Guide". It and all the lending library will be available from Bruce.

COMMITTEE REPORT--Education Committee chairman John Johnson announced the availability of an assembly language class or help. John has been using Assembly for some time now and is willing to help others. He recommends learning assembly as it helps you understand your computer and how it works. John recommends using the book "Introduction to Assembly Language for the TI Home Computer" by Ralph Molesworth. Several copies are known to be in the hands of members. Additional copies would be ordered if a source can be found. If you are interested in either the manual or the course or both, contact John Johnson or one of the club officers. (See John's article later in this issue.)

It was also announced that Jim Trainor is willing to teach 'c'. Any takers? Give your name to John Johnson.

QUESTIONS FROM THE MEMBERSHIP--Q. Is any help or book available to teach Logo II to kids? A. Logo II is written on an adult level. Logo I is written on a younger level. Check out a Logo I manual for help or ideas.

Q. Is it possible to change basic programs to the multiple statement method used in XBasic? Is there a program available to do this automatically? A. Yes! No! Anyone should be able to rewrite using the multiple statement form but we have no such program in our program library. A program is available to do the opposite, but the program would still be in XBasic. This is primarily a debugging tool.

A general discussion on various subjects followed including: memory expansion, the Myarc 9640, the Clyde College Cassette Loader, clearing the 32K memory, ram disks, etc. Jerry Canady announced he was going to do some work on his P-box; he will be glad to bring it for examination without covers at the June or possibly the July meeting.

Program for the evening--Bruce Winter made available the club's program library for browsing and copying of programs.

Submitted by Bill Paeth, Secretary

TUTORIAL REPRINT

In order to bring some new ideas to those members who still write their own programs, I am reprinting the following four pages from the Atlanta newsletter, with thanks to them and to Funnelweb Farm. This tutorial may help us remember how to use Extended Basic, and it surely gives some useful tips of the trade. I hope Funnelweb Farm follows through with the second part of this well-written article. jcg

FUNNELWEB FARM EXTENDED BASIC TUTORIAL
PART 1

Extended Basic Tutorials from FUNNELWEB FARM

---

1. 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 June 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(Q,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 of 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/994A 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 TI99/4A 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. Lack of proper "procedures" has always been the major limitation 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 civilized. For example the magazine Australian Personal Computer in an older issue (Mar 84) carried a review of the IBM PCjr computer just released in the US. The Cartridge Basic for this machine apparently does not support procedures.

Perhaps IBM doesn'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 localized 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 for 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 behavior of TI Extended Basic comes from programming experience with Version 110 of XB on a TI99/4A 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 condole 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 of 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 doesn't 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.

THE END FOR NOW.

ASSEMBLY LANGUAGE CLASS

At the next regular club meeting on June 8th, there will be a sign-up sheet for an assembly language class. The class will be held either before or during the regular meetings and start on the July 13th meeting. We will use the book "Introduction to Assembly Language" by Molesworth and we hope to cover one of its chapters each time we meet.

If you are interested please be at the next meeting or send word with someone so we can order a book for you. The only cost I know of at this time is for the book which is listed in the Tenex catalogue at $8.49.

The book covers using both the Editor-Assembler and Mini-Memory cartridges. The minimum requirements for the Editor Assembler version include the cartridge, one disk drive, and 32K memory expansion. The Mini-Memory needs only MM cartridge and cassette recorder. Hope to see you there.

John Johnson

THE LIBRARY CORNER

Well, here goes! I was asked to write something about some of the new software that the club has received. (Ed. note: Hope to make this a regular feature!)

We now have the Monopoly game for our computer. It is an Ex. Basic program on disk. It could possibly be converted for a cassette system. Would anyone like to try? It is for 2 to 8 players. The computer is the banker and takes care of all transactions. Each player's money and property is kept track of and can be reviewed at any time. It is a little slow, but is very well done. The only problem is that there is no way to save a game.

We also have a program for a Gemini 10X printer. It sets the printer up to print Old English characters. It is an Ex. Basic program and is written for a parallel printer (PIO). I ran the program, and then I loaded Funlwriter and printed a letter. It was printed using Old English characters. This is a sample of the alphabet called out by this program.

𝔗𝔥𝔦𝔰 𝔦𝔰 𝔞 𝔱𝔢𝔰𝔱 𝔬𝔣 𝔬𝔩𝔡 𝔈𝔫𝔤𝔩𝔦𝔰𝔥 𝔠𝔥𝔞𝔯𝔞𝔠𝔱𝔢𝔯𝔰.

𝔄 𝔅 ℭ 𝔇 𝔈 𝔉 𝔊 ℌ ℑ 𝔍 𝔎 𝔏 𝔐 𝔑 𝔒 𝔓 𝔔 𝔘 ℜ 𝔖 𝔗 𝔘 𝔙 𝔚 𝔛 𝔜 ℨ

𝔞 𝔟 𝔠 𝔡 𝔢 𝔣 𝔤 𝔥 𝔦 𝔧 𝔨 𝔩 𝔪 𝔫 𝔬 𝔭 𝔮 𝔲 𝔯 𝔰 𝔱 𝔲 𝔳 𝔴 𝔵 𝔶 𝔷

1 2 3 4 5 6 7 8 9 0

If you would like these programs or any others, please let me know. With any luck, a new list of all the club's software will be available by the July meeting. All the programs have now been reviewed. I just have to get the reviews together in one list.

Bruce Winter

FROM THE MAILBOX

The following articles of interest appeared in other user group newsletters received this past month. If you see a topic that you would like to know more about, please let one of the officers know soon. He will see that you have access to the newsletter you are interested in.

A new prom set is reported available that will allow you to upgrade your TI disk controller card to handle double sided, double or quad density disks. Price is $45 from Ryte Data, Millenium Computers, 210 Mountain St., Haliburton, Ontario K0M 1S0, Canada. (from Shoals TIdings, March 1987) (Ed. note: Further information has revealed that this chip will not give DSDD capability to standard disk drives. The chip is designed for quad density drives, which are labled 96 tracks per inch. Thus, on two sides of one of these newer drives, one can store the equivalent amount of data (360K bytes) as on the usual DSDD disks many of us now use.)

An article on LOGO procedures (Shoals TIdings, March 1987).

Also from the Shoals 99ers comes word of an interface that allows connection of an IBM style keyboard to your TI. Price is $80, not including the keyboard, which may be purchased in the discount markets. Contact MLsystems, 17 Mathewson St., Johnston, RI 02919, (401) 232-5134. (Shoals, Apr 1987)

A how-to explanation of how to clean the print head on an Epson printer, AT YOUR OWN RISK. The procedure looks like it may apply to other printer types, as well; An explanation of how to customize your menus on Funnlwriter; Producting art with the word processor, an article by Ann Dhein, from Waterloo; a schematic diagram for a single chip 32K memory expansion. (Cleveland Area 99-4A Users, May 1987)

Instructions for correcting printer default in PRBASE, using a sector editor program; a beginner's encouragement to learn assembly language. (Southern Nevada Users' Group, May 1987)

A review of data base programs; a counter-review to a bad review of PRBASE; parts list for the TI Diagnostic Module, ordered from TI for about $8.50. (Topics, L.A. 99ers, May 1987)

A new load program for your XB program disks; a catalog program for cassette libraries; part six, Getting On Line; an explanation that even I can understand about using a multi-column printing program, "Multi-Print". (PUG Peripheral, May-June 1987)

The Smart Programmer has several articles for Gram Kracker (TM) owners, some assembly language utilities, and a reprint of Tom Freeman's article on using check sums to validate the proper keying in of a published program. The article comes complete with assembly source code and XB Call Loads. (The Smart Programmer, December 1986)

A review of the Mechatronics Extended Basic II +, a review of Schedule Manager from Asgard Software, miscellaneous programming tips. (Rocky Mountain Tictalk, April 1987)

ONE LINERS

by Tony Falco

One liners are an exercise which develop programming skill. By their obvious limitation in size they must be compact and efficient. The three one liners which follow all have to do with the number pi. That elusive, non-ending, non-repeating decimal which approximates the ratio of the circumference to the diameter of any circle.

The first one does low resolution graphics designs. The designs are in reality plots of graphs of polar functions. Complete execution of the program takes about 22 minutes.

The second one approximates pi by randomly tossing imaginary "darts" at a square which has a circle inscribed in it. The darts always land in the square, and by counting how many end up inside the circle we can calculate pi because the probability of a dart hitting inside the circle is pi divided by four. After each 20 tosses the program reports the number of tosses and the approximate value of pi. The longer it runs the better the approximation. The program is stopped by using FCTN-4.

The last program starts by considering a regular hexagon (six equal sides and six equal angles). We use its perimeter as an approximation for the circle's circumference. We then repeat this process (that is we calculate the perimeter) for 12 sides, 24 sides, 48 sides and so forth doubling the number of sides each time and stopping when the number of sides reaches 6,144. (That's pretty close to a circle.) Try this one on another brand of computer and compare the value obtained with the actual value of pi. You will see how accurate your T.I. is compared to other machines.

(Courtesy of
M U N C H
Newsletter)

```
1. CALL SCREEN(2)!! FOR P=3 TO 30 !! CALL
 CLEAR !! CALL COLOR(2,P/2+1,1)!! FOR A=
0 TO 5*PI STEP PI/36 !! R=11*SIN(P*A/2)!
! CALL HCHAR(12-R*SIN(A),17+R*COS(A),42)
!! NEXT A !! NEXT P
```

```
1 N=N+1 !! RANDOMIZE !! X=RND !! Y=RND !
! H=H-(X*X+Y*Y<=1)!! IF N/20=INT(N/20)TH
EN PRINT "TOSSES=";N;"PI=";4*H/N;;;!! G
OTO 1 ELSE 1
```

```
1 CALL CLEAR !! PRINT "# OF SIDES","PI"!
;!;!! N=6 !! S=1 !! FOR X=1 TO 11 !! PRI
NT N,S*N/2 !! N=2*N !! S=SQR(2-SQR(4-S*S
))!! NEXT X
```

NEXT MEETING

JUNE 8

7:00 PM --- JA BUILDING

HARDWARE AND SOFTWARE REVIEWS

DON'T MISS THE FUN!!!!!!!

Cedar Valley 99'er Users Group
288 Windsor Dr. NE
Cedar Rapids, Iowa 52402

Send To:

GARY BISHOP
124-222
860 WESTVIEW DR
MARION IA 52302