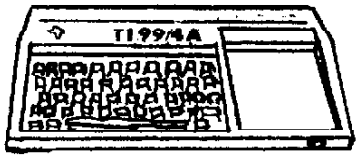
KC 99'er



****
* Volume 6 KC 99'er BBS 436-9074 Issue 4 *
****
* Kansas City TI99/4A Users' Group, P. O. Box 12591, N.K.C., MO 64116 *
****

APRIL EDITION

++
++ ++
++ ++
++ KC 99'er SWAP-n-SHOP ++
++ ++
++ APRIL 12 1987 ++
++ ++
++ ++
++ OPENS 2:00 PM at ARTHUR MAG CENTER ++
++ ++
++ cost $1.00 per person ++
++ ++
++ DOOR PRIZE DRAWING every 20 mins. ++
++ ++
++ ++
++

EXTENDED BASIC TUTORIALS

(The HV 99ers of Australia
from MID SOUTH U. G.
(edited by Steven DeGeare)

INTRODUCTION:

These tutorials were originally written several years ago to try to improve the abysmal standard of programming apparent in magazine and User Group Newsletters. The standard of published programs written in XB has on the whole improved since then. There does seem to still be demand for tutorial material on XB, so I hope this series will still be of value to begining programmers.

The aim of this series on TI EXTENDED BASIC was always to concentrate on those features which had not received due attention in User-group Newsletters or commercial magazines. The user defined sub-program, or some other feature of XB. A much neglected source of help is TI XB tutorial tape or disk. The programs in this collection are unprotected and so open for inspection and it's worth looking at their listing 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? Subjects to be covered are:

1 USER-DEFINED SUB-PROGRAMS (now known as UDSP)
2 PRE-SCAN SWITCH COMMANDS
3 CODING FOR FASTER RUNNING
4 BUGS IN XB
5 CRUNCHING PROGRAM LENGTH

Initially the discussion will be restricted to things which can be done with the console and XB only.

Every dialect of BASIC, TI XB 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. See example.

```
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. Very useful, but there are problems. If the line number of the subroutine is changed, other than by RESequencing of the whole program then the GOSUB will go astray. 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 info passes to and from the sub-programs through a well defined list of parameter and return variables. The price paid for the power and expressiveness of TI basic and XB is slowness of the GROM/GPL implementation.

DEF function is a primitive form of user defined sub-programs found in almost all BASICs. Often its use is restricted to a special set of variable names, RNA,FNB ... But TI basic allows complete freedom in naming DEFed functions as long as they don't clash with variable names.

CALL LINK to machine code routines is only available in XB when the memory expansion is attached. The TI console has only 256 bytes of CPU RAM for the TMS9900 lurking in there.

Get your TI XB manual handy and look through the section on SUB-PROGRAMS. Unfortunately most TI 99/4A manuals avoid explanations in depth ... So we are now going to try to do what TI chickened out of.
The UDSP 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 the built-in ones are. Unlike those 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'.

Now let's look at how XB handles sub-programs. The RUNning of any XB program goes in two steps. The first is PRESCAN, that interval of time after you type RUN and press ENTER before anything happens. During this time the XB interpreter scans through the program checking a few things for correctness such as there being a NEXT for each FOR. Next 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 the program.

Well then, what does XB do with the user sub-programs? First of all XB locates the sub-program names that are not built into the language. It can do this by finding each name after a CALL or SUB statment, and then looking it up in the GROM library index of built-in sub-programs names. See example:

```
100 CALL NOTHING
```

TI basic will go out of its tiny 16K brains and halt executiion 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-Programs code comes at the end of the program, with nothing but sub-programming after the first SUB statement. XB scans and establishes new variable storage areas, starting with the variable names in the SUB xxx(parameter list), for each sub-program, 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. See example:

```
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 ... 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 Basic assigns nominal values to all variables mentioned in the program as part of the PRESCAN, zero for numeric and null for strings. 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 :: 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. XB does not 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 UDSP 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.

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

NEW FROM MONTY SCHMIDT: **GPL LINKER V1.1 Run Time Version**

GPL Linker is an ingenious program that places the power of Graphics Language Programming (GPL) at your command. No extra hardware is required beyond standard 32k and disk system. In short, Linker creates runnable program files from compressed (or uncompressed) GPL Assembler object files. You can then run these programs with "Option 5 Run Program Files" of the Editor Assembler Module.

Up to 24k GPL programs can be developed and run on standard 32k systems. Included in the run time version are two demonstration programs and "CONVERT," a public domain conversion program that converts MS BASIC statements to TI BASIC statements.

Price: $21.00 CDN funds $15.00 US funds.

now $49.95
w/Linker $59.95
plus Intern $69.95
add $3 shipping

GPL price Reduction

ENHANCED GPL Assembler V2.1

NØW with high memory loader package

UNLOCK ALL THE SECRETS! New GPL Assembler Version 2.1 available exclusively through Ryte Data.

This program provides the power to write, edit and assemble true GPL programs for the TI 99/4A. Create code that accesses console operating system routines directly. Develop programs that use the GPL Interpreter and all the features of the TI 99/4A.

This package includes the GPL Assembler disk, printed documentation, GPL tips and hints, update support service and commented GROM/ROM listings (with the book "INTERN"). An example for a command module type GPL program is included with source, object and list files on disk.

Requires: 32k memory, disk drive(s), TI Editor Assembler package. Printer/RS-232 recommended.

R/D COMPUTING

Technical Newsletter

with Bill Gronos on assembly!

We have a vision. Our vision is one of continued TI 99/4A support. We're dedicated to the power of the machine. From the novice to the experienced computer user; for management, home, education, entertainment or advanced applications our publication "R/D COMPUTING" is for you. TI never revealed all the important inner workings of the 99/4A. We bring you this vital information every month.

A major feature of R/D COMPUTING is the regular "upgrade projects." These electronic construction projects are designed to give the 99/4A owners more features and improvements. For example, it is possible to increase the speed of your computer with a very simple part and switch. Each month we present new circuits, diagrams and projects for your computer.

From the moment your new subscription arrives at your home, you will have access to critical technical information that makes your computer more valuable, powerful and versatile.

We believe that the TI 99/4A deserves new products, innovative hardware, software, information and a dedicated technical publication. This is what makes a computer "viable" in the fast paced microcomputer industry. Now that the 99/4A has been 'opened up,' all the secret information is available. You can have all these benefits and more each month. SUBSCRIBE NOW!

19

$14/year - back issues 3 - 15 available

THANK YOU! Our business has grown 300% this year. To show our appreciation we are **giving away** hundreds of dollars in TI products to 99/4A owners or users groups. To enter drawing, (no purchase necessary) send your name and address on a postcard to Ryte Data. For **subscribers** to R/D Computing we are giving away XBII plus, 32k memories, GPL Assembler package, etc. **Enter your subscription today!**

Prices listed in U.S. funds.

New catalogue available.

Super Clock Support

Designed for the CorComp Clock Peripheral—Triple Tech Card or Stand-alone models. This utility package provides more functions for use in your Extended Basic programs. Direct access to the clock ROM at assembly speed gives you these features: three independent timers to set and read; alarm function; two interrupt routines to display time and date on screen with CTRL T—continuously or on your command; all time and date displays are in 12 or 24 hour format using **TEXT.** This program also allows the week, date and time to be set independently rather than all together.

Program disk is not copy protected to allow you full use in your Extended Basic programs. Package includes disk and instructions. Only $17.95 plus $2

XBII plus

As reviewed in Micropendium October 1985. This command module gives you all the features of Extended Basic PLUS 40 new commands. Totally compatible with TI's XB, this enhanced version gives your programs more power to access your 99/4A. Commands such as MLOAD, MSAVE, VPEEK, VPOKE, GPEEK are superior to most other Basic environments. various demo programs and new applications using high resolution graphics make this module a "must" for Extended Basic users. Comes complete with a 95 page manual. Requires console and 32k. $75.00 (US) plus $2 shipping.

BASIC Compiler V1.1

New Basic Compiler that is finally easy to use! Supports virtually all Basic and Extended Basic commands in existing programs. Simply load and compile programs from a menu driven directory on your screen. No extensive re-writing, variable declarations or conversions are required. Compiler produces code-list in one pass containing all variable addresses and jump list. Package includes Extended Basic Loader, Floating Point Loader, Integer Loader, Disk Menu program and DSR program for the Compiler support. This Compiler cannot unravel DEF statements and stops on the END statement—no SUB's allowed. TRACE, BREAK, ON ERROR, CALL LOAD and CALL LINK may produce execution errors.

Requires 32k, disk. Price: $20.00 plus $2 shipping (US funds).

Ryte Data (705) 457-2774

MasterCard

MILLENNIUM COMPUTERS
210 MOUNTAIN STREET,
HALIBURTON, ONTARIO K0M 1S0
TELEX 06-986766 TOR. ATTN: RYTE DA

BASIC PROGRAMMING: CASSETTE DATA FILES-by Bob Pass
Edmonton (Alberta) 99'er Computer User's Society
reprinted from DVUG newsletter May 1986

Some of you may not be aware that you can use your cassette recorder to do more than just load or save programs. Your cassette can also save data files which can be read into the console by a running program, modified by the user, and saved for later reference. By learning to use the Basic commands, OPEN#, INPUT#, PRINT#, AND CLOSE#, you can open up new horizons with your TI-99/4A by being able to save and recall data from cassette.

One important point to get CLEAR is the concept of 'BUFFERS'. The word buffer is used to describe an area of computer memory(or hardware) that is used to temporarily store data that is written into and out of the computer.

Buffers are required whenever the computer must talk or listen to another device which does not operate at the same speed or in the same manner as the computer does. For ex. since you cannot type at computer speed, the keyboard on your machine uses a buffer to pass information to the processor. Similarly, a cassette recorder simply cannot handle data at computer speeds; consequently the computer must use a buffer to transfer information to the device.

Briefly, a buffer is a block of memory of fixed size which is compatible with the output device. When the buffer is emptied, more data is written into it until the data transfer is completed. An IMPORTANT point to realize is that the transfer of data from buffer to the external device is done automatically only if the buffer is full. If the buffer is only partially loaded when the application program ends, this data could be lost unless you instruct the system to close all open files (buffers). This will cause the system to finish dumping the buffer to cassette. The last data item is always an END OF FILE (EOF) marker.

When data is read back into the computer, the process is reversed, with the computer looking for the END OF FILE marker so that it knows when to stop reading the buffer and shut down the external device.

The buffers have a numerical tag. In TI BASIC or XBASIC, you can specify a tag for 1 to 255 with each buffer being distinguished from others by the tag number. Buffer number 0 is reserved for system use and is, in fact, the keyboard (and screen) buffer.

You can use more than one buffer at time for different purposes, however the number of buffers that are open at one time is limited to a default of 3. If you need more than 3, use the CALL FILES(n) command, where n is any number from 1 to 9. It must be used in the following manner:

NEW <ENTER> CALL FILES(n) <ENTER> NEW <ENTER >

Now load your application program in the usual way, and you will have the required number of files or buffers available. CALL FILES may NOT be used within a program. Consequently, any program requiring more than 3 buffers must have the appropriate CALL FILES executed first. Each buffer that has been reserved occupies 518 bytes of RAM (except the first which takes up 1052 bytes of RAM), so it is wise to keep the required number of buffers as low as possible to conserve memory space.

DO NOT USE A PROGRAM TAPE TO STORE DATA

OPEN #n -- This command prepares the sytem to transfer data to an accessory device. The buffer (FILES) number (n) is specified by you as well as the device name such as CS1 to which the data is to be written to or read from. Additionally, you must specify the structure of the data file to be written on the cassette. Until you have become familiar with the use of cassette files use "CS1",INTERNAL,SEQUENTIAL,FIXED for your file structure. Furthermore, you must tell the sytem the size the data string to be written will be (so it will know how to read the data back later) by placing 64,128, or 192 after the FIXED notation. You must plan the maximum length of each

data item to be stored. If you choose FIXED 64 in the OPEN # statement and then write a data statement 70 characters long, the last 6 character would either be lost or would overflow into the next character string, producing an unwanted concatenation or a 'trashed file'. On the other hand, if your string was only 60 characters long, the system would automatically pad the string to 64 characters with blank characters, which are removed when the data is recalled(read).

PRINT #n -- This command causes the system to transfer (print) data from the computer to the devide identified by and in the format specified by the OPEN# statement whose buffer number corresponds.

CLOSE #n -- This command will cause the computer to empty the specified buffer number of pending data by completing the transfer sequence. TREAT OPEN # and CLOSE # commands like matched bookends. DO NOT place statements in program which might transfer control out of the program block defined by the two commands. If you experience a program error msg. during a file transfer sequence, DO NOT USE <FCTN QUIT>!!!!!! This will cause all data in the buffers to be lost. INSTEAD, type BYE,RUN,OLD,SAVE,LIST or else EDIT a line number. Any of these actions will cause the buffer to close properly.

CHARACTER SETS MISSING.
By Tony Parla

Recently our U.G. acquired a whole lot of programs from Amnion and I decided to go through most of them to see what the programs were as the filenames were only numbers and the only way to find out was to load and run them. All 301 of them.

You can imagine what a process that was, especially when about 10% of the programs would crash and give you an error at one of the "call color" lines. I was using extended basic.

While I was going through some of the Tips from Tigercub by Jim Peterson, and wondering how one guy could have so much talent, I found the answer in #40 Tips.

I quote. "When TI developed Extended Basic, they took away the ability of Basic to redefine or color the characters in sets 15 and 16, ASCII 144 to 159, in order to make room in memory for sprites. That is why Basic programs which use sets 15 and 16 will crash if you try to run them in Ex. Basic." Unquote. Thanks Jim.

MORE ON DISK DRIVES
(By way of MICROpendium by Jeff Shaw, Troy, New York)
Condensed and edited by
Tony Parla

Both drive motors come on when either drive is accessed. However, the drive that was not chosen turns very slowly and does not draw full power. Our information indicates that the power supply is adequate for this project.

The one possible area of concern is the PE box's voltage regulator. The regulator is supposed to maintain 12 volts DC. If too much current is drawn, it is possible that the voltage will oscillate around the 12 volts.

We have recently learned that an upgrade for this part is available. The PE box's 1 ampere regulator may be replaced with a 5 amp. (Sylvania ECG-933). This is readily available in most electronics stores.

Even with the original PE box's regulator I have been able to power both drives as well as five peripheral cards (not including PE box of

which (Foundation CPM) has the power requirements of two cards. (Unquote).

Speaing for myself, I tried to run two drives with the PE box supplying the power, but after about an hour of running the drives started acting erratic so I metered the voltage and it was dropping down as far as 6 volts. The regulator was dropping out.

I made me a power supply using a switching type console power supply and it has worked out very well. No more problems. I made one up for one of our club members and his is working OK. I just changed the 12 volt regulator to a 1 ampere from Radio Shack. A 7812 type. Left the +5 volt alone and didn't use the -5 volts.

FROM THE AMNION COLLECTION:

I decided this should be be modified for our members who get their newsletter our BBS (which is of course)

THE KC 99er 436-9074

```
100 REM NEWSLETTER READER
110 !FROM PRGRAM BY PETER HODDIE
120 REM MODIFIED FOR NEWSLETTER BY
STEVEN DEGEARE 3/11/87
130 @=1 :: ON ERROR 220 :: C$="RS23
2/2\LF"
140 DISPLAY AT(2,3)ERASE ALL:"** NE
WSLETTER PRINTER **"
150 DISPLAY AT(5,2):"OUTPUT TO ";C$
 :: ACCEPT AT(6,2)SIZE(-27):C$
160 A=65 :: D=0 :: OPEN #2:C$
    ,VARIABLE 254
170 DISPLAY AT(5,()):"    ENTER FILE
NAME:":"    DSK1\";SEG$(M$,6,15)::
ACCEPT AT(6,5)SIZE(-15)BEEP:M$
180 OPEN #@:M$,INPUT :: PRINT #2:
190 DISPLAY AT(10,5)ERASE ALL:"PRIN
TING ";M$
200 LINPUT #@:L$ :: GOSUB 250 :: IF
EOF(@)THEN CLOSE #@ :: GOTO 230 ELSE
 200
210 D=0 :: PRINT #2: :: GOTO 200
220 DISPLAY AT(10,()BEEP ERASE ALL:
"ERROR -- PROGRAM IS ":"RESTARTING
\\\\\" :: FOR DE=1 TO 300 :: NEXT D
E :: RUN
230 PRINT " ANOTHER FILE Y/N" :: IN
PUT AN$ :: IF AN$<>"N" THEN 170
240 PRINT " DONE" :: STOP
250 D=D+@ :: IF D<>B THEN PRINT #2:
L$ :: RETURN
260 PRINT #2:" " :: D=0 :: RETURN
```

HERE ARE A FEW OF THE BETTER TI RELATED PUBLICATIONS
(your editor)

MICROpendium
P.O. Box 1343
Round Rock, TX 78680
(515) 255-1512 ($20.50/yr)

The largest TI only publication. Write for a free sample issue.

The Computer Shopper
407 South Washington Avenue
Titusville, FL 32781
(305) 269-3211 ($21.00/yr)

Carrys the informative section on the TI by Ron Albright.

Genial TRAVeLER Diskazine
835 Green Valley Drive
Philadelphia, PA 19128
($30.00/yr published bimonthly)

This is a very good magaxine (on disk) with great programs by Barry Traver. (from MID-SOUTH 99/4A editor).

The Smart Programmer
(see Walter Blood for details)

HERE IS ONE FROM CANADA:

CLUBLINE-99
P.O. BOX 1005 STATION A
HAMILTON, ONTARIO L8N 3R1
($25)