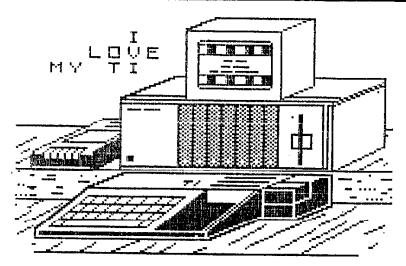
HUNTER VALLEY 99ERS USERS GROUP

MOME GOMPOTER MEWSLETTER









1988









REGISTERED BY RUSTRALIA POST PUBLICATION NUMBER NBG8023
THE SECREATARY HYSSERS, B RRCOT CLOSE TARRO NSW 8388

YOUR COMMITTEE

all telephone numbers are STD area code 049

PRESIDENT

Alan Lawrence 35 Bayview St., WARNERS BAY 2282 Ph. 484509

VICE PRESIDENT Allan (Joe) Wright 77 Andrews Rd., VALENTINE 2280 Ph. 468120

SECRETARY

Albert Anderson 6 Arcot Close, TARRO 2322 Ph. 662602 Viatel 496626020

TREASURER

Peter Smith 8 Glebe St., EAST MAITLAND 2322 Ph. 336164 Viatel 493361640

SOFTWARE LIBRARIAN

Stewart Bradley 14 Hughes St., BIRMINGHAM GARDENS 2287 Ph. 513246

EDITOR

Brian Woods
9 Thirlmere Pde.,
TARRO 2322
Ph. 462307 Viatel 496623070

PURCHASING CO-ORDINATOR

Alan Franks 822 Pacific Highway, MARKS POINT 2280 Ph. 459170

COMMITTEE MEMBERS

Noel Cavanagh 378 Morpeth Rd., MORPETH 2321 Ph. 333764

Rodney Gainsford 56 Sedgewick Ave., EDGEWORTH 2285 Ph. 583515

John Paton 1 Parlen Close, RUTHERFORD 2320 Ph. 326014 Viatel 493260140

CONTRIBUTIONS

Members and non members are invited to contribute articles for publication in HV99 NEWS.

Any copy intended for publication may be typed, hand written, or submitted on tape/disc media as files suitable for use with TI Writer (ie. DIS/FIX 80 or DIS/VAR 80). A suitable Public Domain word processor program will be supplied if required by the club librarian.

Please include along with your article sufficient information to enable the file to be read by the Editor eg. File Name etc. The preferred format is 35 columns and page length 66 lines, right justified.

All articles printed in HV99 NEWS (unless notified otherwise) are considered to be Public Domain. Other user groups wishing to reproduce material from HV99 NEWS may feel free to do so as long as the source and author are recognised.

Articles for publication can be submitted to the Editor, ALL other club related correspondence should be addressed to The Secretary.

DISCLAIMER

The HU99 NEWS is the official newsletter of the HUNTER VALLEY NINETY NINE USER GROUP.

Whilst every effort is made to ensure the correctness and accuracy of the information contained therin, be it of general, technical, or programming nature, no responsibility can be accepted by HV99 NEWS as a result of applying such information.

The views expressed in the articles in this publication are the views of the author/s and are not necessarily the views of the Committee, Editor or members.

TEXAS INSTRUMENTS trademarks, names and logos are all copyright to TEXAS INSTRUMENTS.

HV99 is a non profit group of TI99/4A computer users, not affiliated in any way with TEXAS INSTRUMENTS.

THA As me, the

as gro ta: the Ui; rei

to:

Ri

de: ef coi

70) Ne Rai

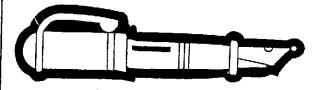
th 51 Ro

de GE A Su

be

FA We

PRESIDENT'S



with

Al Lawrence

THANK YOU

Α

As this is the final Prez's pen from me, I take this opportunity to thank the outgoing committe for their support during the past year.

It is hard to single out any person as everyone all put effort into the group by various means. Realising that without Brian Woods' massive task each month for our output to all TI'ers via the Newsletter and the Secretary Albert Anderson's time consuming input and output to World User Groups (as well as renovations at home) it would be remiss if I did not put them at the top of the thank you list.

As well I must mention Tony and Will McGovern, Bob Carmany, Ron Kleinzchafer, "Joe" Wright and Richard Terry for articles and other great contributions.

Our technical whiz, Alan Franks also deserves all our thanks for the efforts he has put in repairing consoles, as well as being the purchasing co-ordinator for the used eqipment department. Thanks for all your valuable assistance, Al.

Neil Quigg for the Quest RD200 Ramdisk a big thank you. While on this subject, if you have the full 512k it is now possible thanks to Ron to divide it into Disk A and Disk B (this is still being developed).

GET WELL SOON, JACK

A speedy recovery to USA member Jack Sughrue after his auto accident and best hishes from all the HV 97'ers.

FAIRWARE

We at the HV99'ers STILL collect group donations to sent to various authors. As we know how costly it is to buy and send individually. We will continue this policy to help support them in the future by collecting donations and forwarding them on in bulk.

MR Ue only sell disks for cost of MEDIA plus nominal COPY FEE and the ONUS is on you the USER to forward any FAIRWARE, FREEWARE, SHAREWARE or by WHATEVER name it is distributed under, to the AUTHOR by any means. That is why we do not charge \$ 5.88 for a disk, as then the recipients do not think we pass the EXCESS to them by some mysterious channel.

So why not all AUSTRALIAN users or if you are a small group send your donation to us to pass on ?? This will be acknowledged and you will help us encourage more great PROGRAMER's to support the TI for a long time, so if you use any type of FAIRWARE and have forgotten, you will never be too late as we will pass monies on so long as you make clear which Fairware you sent it for. Better still send one LARGE cheque for us to distribute to several if you make use of more than ONE!!! Just let us know the Authors and also how to divide it up, as well as any comments on the programs use to you.

Help support those who support us.

I wish the new committee a very successfull future and as a member I will continue to support the group.



random bytes

WITH BOB CARMANY

What to write about this month? That is always an interesting questions! It wasn't bad when I first started this column because almost everything I wrote was 'new'. Now, however, I find myself pushing just a bit to come up with a viable topic for this monthly endeavor. Let's see what we can find to cover this month!

Over the years, XB has become the "neglected" programming language. Assembly language is faster but you pay for the speed by writing 20 or so lines of low-level code to accomplish what I single XB statement does. Besides, I KNOW what DISPLAY AT means but I'm not so sure about those registers!

I belong to the "hack and slash" school of programming. I use a method similar to Attila the Hun --- plug a segment of code in and cut it to pieces if it doesn't work. Trial and error are great teachers!! When I write a program from scratch, I rarely do it in the proper order. I'm just as likely to start at the end or the middle of the program as at the very beginning.

Watching me write a program is sometime akin to a Marx brothers movie --mayhem and madness. But despite all of the unorthadox means and disarray, there are certain things that I do consistantly throughout --- things that I have found make for the best and fastest programs.

FILL UP PROGRAM LINES. You should realize that each line number takes up memory --two bytes to be exact. In applications where memory space is critical, the savings can be substantial. On top of that, it takes time for the computer to process each line in succession --more time, in fact, than to process a full, multi-statement line ٥f code. Type until the cursor stops you can't enter anymore characters and press (ENTER). Then, press <FCTN-8> and you will be able to enter more code.

2) USE PRE-SCAN. If you have written a rather lengthy program, make use of the fact that you can turn the prescan off and on. The long interval between the time the program is booted into memory and the time that it actually executes is due to the fact that the program scans every line for variables and constants and CALL statements before it starts. There are two ways that you can do this. If your program is shorter than 45 disk sectors, you can use PRE-SCANIT! to do the "dirty work" for you. On longer programs, use NEATLIST or one of the other cross reference programs to list the variables and constants and put them at the beginning of the program. When you have them in two or three lines of code (along with the CALL statements), turn the pre-scan off and then back on at the appropriate place at the end of the program. The program will start almost immediately!

3) WRITE PROGRAM MODULES. If you create self-contained program segments, you can beta-test them independently of the rest of the program. It is much easier to de-bug 25 lines of code than to try to follow the program flow de-bug 150 or 200 lines of code. Remember, you can always RESequence after you program have it assembled. That is one a£ reasons that I use the scatter-shot method of programming --- writing the end of the program or middle long before the beginning written.

NEVER WEAR OUT YOUR FINGERS TYPING SOMETHING THAT THE COMPUTER CAN DO FOR YOU. Why should I step through a program line-by-line looking for typos (I have never written a program without one) when I can let the computer do the work for me. I use PROGRAM WRITER to write what lines of code that it can write and usually end up saving the entire lot as a D/V 80 file before I run it. Just to make sure, I run the D/V 80 file through a special dictionary of CALLs, etc. to check for any mis-spellings that I have missed. I convert the D/V 8Ø file back into a program with XLATE and I'm ready to go! Then, I exterminate the logic "bugs". By the same token, take advantage

common sections of code by making them SUBROUTINES and use whatever segments of code that you can from other programs you have created.

5) ALWAYS HAVE A CURRENT BACK-UP COPY OF YOUR WORK!!! This is the most important of the rules of programming. I have had the rather uncomfortable feeling of seeing 4% sectors or so of work mysteriously vanish by pressing the wrong key (or as an offering to the "disk god"). It is always easier to reload that re-create code. I usually backup my work every 25 - 3% lines of code to alternating files on a backup disk. That way, I won't lose more than that should the system crash or the program "eat" what is in memory.

6) DEBUG FROM A PRINTED COPY. It is always much easier to look at a written copy of the program and find logic errors than it is to look at the screen and try to find them.

BETA-TEST!! BETA-TEST!! BETA-TEST!! One of the most difficult things for a programmer to do is to de-bug his own work. The reason is that you know what it is supposed to do and what keys are supposed to be pressed. someone who has never seen your program and let him try it out---if there are any errors, they will usually pop out then.

When I follow these simple XB programming rules, everything seems to work out just fine --despite the fact that I haven't started in the proper place in the program!!



THE MYSTERIOUS K

This article came to us from the February 1989 issue of CIN-DAY Newsletter, and came from an article written by Bob Albrecht & George Firedrake.

Computers use a very simple code, called binary, to represent information. Binary is very simple; it uses only two symbols, 0 and 1. The symbols 0 & 1 are called binary digits, or bits.

In a typical personal computer information is stored in the memory of the computer. The memory consists of many thousands of bits organized in bunches of bits in memory locations.

One memory location can hold eight bits of information. A bunch of eight hits is called a byte. So one memory location can hold eight bits or one byte. The memory of a typical personal computer has many thousands of memory locations.

* One memory location can store 8 bits.

* A group of 8 bits is called a byte.

* So a memory location can store one byte.

* A computer memory has many thousands of locations, so the memory can store many thousands of bytes.

Perhaps you have heard about the mysterious K. Feople say a computer has 128K or 256K or 512K - or more - bytes of memory.

* 1K bytes equals 2° bytes equals 1024 bytes.

Use the computer to change 1K bytes or 256K bytes to ordinary numbers:

type PRINT 2010

it prints 1024

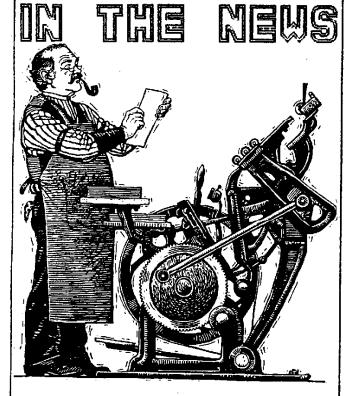
type PRINT 256 * 2^10

it prints 262144

it prints 524288

REMEMBER

1K bytes = 240 bytes = 1024 bytes



compiled by BRIAN WOODS

HALL OF FAME

In the April issue of ROM, the newsletter of the Users Group of Orange County in the US announces that four people active in the TI world have been inducted into their Hall of Fame. They are Barry Boone, Jim Peterson & our very own Tony & Will McGovern.

With regard to Tony & Will they said:

"These two residents of Australia are best known for Funnelweb. This program, which has been proclaimed by some as 'the most significant program ever written for the TI', takes TI Writer to its maximum abilities and then adds features never consolidated into one program before Funnelweb. It is, simply put, an entire operating environment that will support just about any disk based TI application."

"They have tweaked TI Writer and added new control keys, a better character font on screen, faster word wrap and much, much more."

"The McGoverns have also contributed to the TI knowledge base. Their newsletter articles, which are often copied, cover many

areas of interest for Extended Basic and Assembly programmers."

MELBOURNE TI FAIRE

The date for this years Faire has been set. It is on:

Saturday October 7th

The Faire will be held at Deepdene Park, Whitehorse Road, Deepdene, commencing at 2.00pm.

According to the Melbourne group's newsletter, the Brisbane group will be running a Multiplan raffle to be drawn at the Faire. Tickets should be on sale soon for that. New products from Australia, the US and Canada are expected to be on show & the Melbourne group's software competition winner will be announced. The will also be (hopefully) an auction of TI parts and other odds and ends will also be organized.

Make sure you keep the date in mind. As we get a little closer to the date we will see those who want to go and arrange transport. accommodation etc.

PAGE MAKING SOFTWARE

The May issue of MICROpendium carries an advertisement for a recently released piece of software. The program is called Page Pro 99 and is being sold by Asgard.

Some of the features are:* It lets you compose a 66 line page
full of text, graphics and lines
quickly and easily...a what you see
is what you get screen, and the
ability to paste in pictures and
type text.

* Allows up to 28 pictures of any size or shape anywhere on the screen, and allows you to type text in a complete large font & a small font of your choice.

* With this utility you can create forms, advertisements, flyers, maps, graphs, charts, labels, certificates, signs, reports and newsletters.

The price is \$US24.95 plus \$US4.75 for airmail delivery, and is available from: Asgard Software PO Box 10306 Ruckville, MD 20850 USA

<u> PR-BASE HINT</u>

From the May issue of the LA 99ers newsletter comes this tip that originally came from the PT9 Newsletter... "Sorting and so on are based on ASCII strings and everything works according to the ASCII value of the characters. Thus, while 4 comes after 2, 22 will come before 4! Use leading zeros on a number you intend to sort...then you will correctly sort 02, 04, 22."

"Selective Indexing search works on your input up to the first space, so that 'good day' will sort on only 'good'. To sort by the whole string you must insert a question mark thus 'good?day'."

TERRIES CORNER

In her column in the May issue of the LA 99ers Newsletter, Terry had this to say...

"... well lots of people like lots of things happening, as long as someone else does them. How about burnout, you all? Haven't enough workers dropped by the wayside because of lack of actual support? If the system starts to wear down, it is NOT the fault of the current doers. If you are one of the myriad of takers, any chance of a switch of roles? Try it, you might like it!"

AND IT'S GOODNIGHT FROM HIM

Well folks, this issue wraps up my spell as editor. I have been putting the newsletter together for a little over 3 years now, and I feel that it is time for a fresh approach to the editing of the newsletter. As well, I recently sold my TI gear and purchased a new computer. 50 it would become difficult to do all the things necessary in printing our magazine.

I have enjoyed my time editor immensely and have made new friends around the world as a result. I would in closing thank those who have contributed to all our magazine over the years including Al Lawrence, Joe Wright, Tony McGovern, Jack Sughrue, Ron Kleinschafer, and of course one of our most prolific contributors, Bob Carmany. Without your contributions gentlemen the HV 99ers Newsletter would have failed long ago...long may you all write!!!

I do not of course intend to desert the Group, after all, your not a bad bunch of blokes. I will he doing my bit to see that the Group continues to prosper into the future. Why don't YOU do your bit for the Group as well, and not leave it to the same people all the time? Have another read of the article above — 'Terries Corner'!

WHAT MAKES ED TICK

يونا

YOUR ROVING REPORTER

As our club seems to be facing a watershed in its history and many of the key people are finding the drain on their time too great, it seemed a good idea to try and find out a little bit about the various jobs that help to keep our club operating in order to see whether a solution can be found to the problems we are facing.

With that in mind, I made our Editor, Brian, the first guinea pig, and asked him these questions. I hope you find the answers enlightening and challenging and that you may find that there is something constructive YOU can do to help.

Rep.: "Just how long have you been Editor of the Club Mag., Brian?" Ed.: "It's been three years now." "That's a fair while considering the work load. Just how much time do you spend each moth getting the Mag out? Ed.: "Well, actually it's got a bit better as I've got better organised. I've had sheets printed up for the regular articles with the headings already done so that it's just a matter of 'Spell Checking' them, making sure that it sounds right, that the article actually says what the person means, cutting it out and pasting it in. It's actually got better. It probably takes up to about 20 hours a month, depending on the size." Rep.: "Do you have to organize the printing as well?" Ed.: "Yes, when I'm done, it's taken to the printers at the B.H.P.

they print it for us. I pick it up

and put all the address labels on

the back ready for delivery."
Rep.: "So what would happen if we have to get it printed commercially?"

Ed.: "Weil, we couldn't afford to have it done commercially. They do it for us and make a small charge. They're very good about it, and we appreciate it."

Rep.: "Is there a chance that we'll lose this facility if you don't continue as Editor?"

Ed.: "No."

Rep.: "How about pressure. Are ther any pressures in this job?"

Ed.: "Well, I like to have the magazine done and ready for people at the meeting. The pressure is getting articles from people in time to have it all done and pasted up and at the printers on Monday morning then that gives them two days to have it done. Sometime the machines break down and it can't be done. That happens occasionally." Rep.: "Is it hard to organize people to get articles in on time?" Ed:

to get articles in on time?" Ed.:
"Most of the people who contribute
are regulars and they know the
score, so that's not usually a
problem."

Rep.: "What kinds of skills have you found helpful in this job?"

Ed.: "Being able to spell and having a reasonable grasp of grammer. That's about all."

Rep.: "Typing?"

Ed.: "Ah...it helps, but it's not vital because I don't have to do a lot of typing. I can type reasonably quickly anyway...two fingers, so typing is not a real problem."

Rep.: "How does being Editor affect your family and leisure time. Does it cut into it too much?"

Ed.: "No. My wifes used to it."

Rep.: "You must have a good wife."

Ed.: It does cut in a little, but it's my responsibility to have it done and I take that responsibility seriously."

Rep.: "When are you able to slot this in to your time? Do you do it at night or at some other time?

Ed.: "Well, mostly, I've been on afternoon shift the week it's been required so I get most of the work done during the day. But it can be done quite easily in 3 or 4 nights at a few hours a night.

Rep.: "O.K. what aspects of the job are the most satisfying? Or do you find it a drudge?"

Ed.: "I find it extremely satisfying. I wouldn't have done it for this long if I'd found it a drudge. I'm getting a little stale now, not because it's a drudge, but because I think someone else who's more enthusiastic could do a better job."

Rep.: "What part is the most satisfying?"

Ed.: "The end product. I'm quite proud of it and I think it rates as one of the best."

Rep.: "Knowing the work load involved, do you think there is anyone who would willingly subject themselves to these pressures?"

Ed.: "No comment."

Rep.: "A hard one. If the job of Editor ceased to be done, what do you think the survival rate of the club would be?"

Ed.: "Very slim. Honestly. Tonight, we've got about ten people here. We've got minety in the club, the others must be in the group for some reason. If they're not here. it must be for the magazine. Conceited as that may sound. they can't come to the meetings, they must be in the club something. The only other thing is the magazine. No magazine, club. *

Rep.: "Thanks Brian, that gives us a a lot to think about. Well, folks, this is your Roving Reporter signing off.



FUNNELWEB FRRM PRESENTS

EXTENDED BASIC TUTORIALS REVISITED

TOHY HcCOVERH

Pari 1

I. INTRODUCTION

These Tutorials priginally were unitten several years ago to try to improve the abysmal standard programming apparent in magazine and User Group Newsletters, and commercial game programs at time. They appeared first Sydney TISHUG NewsDigest and later ones in the HV99 Newsletter the formation of that group as a separate entity. The standard of published programs written in XB has on the whole improved since but not as much as it should have, and as advanced users move on, beginners take their places on the basic console/XB computers and need to learn how to use them. XB is no longer the prime language on the expanded TI-99/4a, but it is still a

very expressive and powerful language, with features that have only been caught up with in recent times on newer and more pretentious computers. There does seem to be some continuing demand for tutorial material on XB, so I hope this series will still be of value to new and old programmers.

The series is intended neither as an elementary course for absolutely raw beginners nor as Ιt i⊊ meant interested user who is willing some effort into understanding how Extended Basic works in order to make best use of it, and wishes to develop a feel for how the machine actually goes about its business. Plus assorted ravings and ramblings This issue of the Tutorials represents a mild going over of original files, mainly to outdated to empa the material.

aim αf this Extended Basic was concentrate on those features which had not received due User-group newsletters or commercial magazines. In fact most of the programs published in these sources up to the original time of writing little use of that most powerful feature of XB. defined sub-program, or other features of XB. Part of reason for this was the practice of

authors. 3 practice still sometimes in evidence in magazines, of rehashing the same material several different machines and publications, forcing the use of the lowest common denominator of Basics. Worse still were the many programs were object lessons in how to write tangled and obscure code. much neglected source of help 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 easily understood overall structure to a program.

Well, what are we going to talk about then ? Subjects covered are

- (1) User-defined sub-programs
- (2) Prescan switch commands
- (3) Bugs in Extended Basic
- (4) Crunching program length

Initially the discussion will restricted to things which can be done with the console and XB only. Original intentions were to cover LINKing of assembly routines but the series petered out before that. Ross Mudie has written extensively on this topic in a series of TND articles in recent years so the gap has been filled. Actually, for most game programming in pure XB with no assembler help, the presence of the memory expansion doesn't speed up XB all that much as speed still seems be limited by the built-in sub-programs (CALL COINC, etc) which are executed from GROM through interpreter. The real virtue of the expansion system for 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 a game. Even so careful attention to MB programming can often provide the necessary speed. As an example, the speed of the puck in TXB, our first major exercise in XB coding, is a factor of 10 faster in the finally released version than it was in the first pass at coding the game. Curiously it has turned out that the coding of this game is so finely fitted to the way TI MB handles things that the puck speed to be reduced for Myarc XBII to

handle it without losing track, even though it is supposed to be much faster.

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

ь

D

n

ŧ

290 300 GOSUB 2000 310 2000 CALL KEY(Q,X,Y);; IF Y= 1 THEN RETURN ELSE 2000

This simple example waits for returns the ASCII code for a fresh keystroke, and might be called a number of places in the program. Very useful, but there are problems. If the line number of the subroutine changed, other than RESequencing Of the whole program (and many dialects of Basic for microcomputers weren't even that helpful at the time) 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 or tail remarks. 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, "Y" is used as a variable anywhere else in the program value will рe 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 these, οf built-in The first of these, built-in sub-programs, are already well known from console Basic. The important thing is that they have recognizable in CALL statements, and that information passes to and from the sub-programs through a well defined list of parameters and 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 in which the powerful TMS-9900 CPU is hobbled by being forced to interpret another language (GPL) for an imaginary 8-bit processor.

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 TMS-9900 lurking in there. All we note now is that the TI does it in a very civilised fashion, LINKing by name to relocatable assembly routines. Ask your PC or Apple or C64 owning their Basic supports friends that. The Funnelweb system supports all necessary assembly development functions with the XB module in an expanded system. TI XB contains a standard set of interface routines that support the details of linking. Unfortunately these have a great weakness in that transfers of strings between XB's VDP storage and the memory expansion is done byte by byte with a complete reset of the VDP read and write addresses for each byte instead of transferring a block at a time. The result is that assembly string handling doesn't always speed things up all that much and this has been a reason for lack of success of some XB support packages.

You should have your TI Extended Basic Manual handy and look through the section on SUB-programs. 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 get into the right frame of mind for your expectations. This process helped by figuring out, in general terms at least, just how the computer does what i t 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 built-in ones Unlike are. 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 information in either direction. These sub-programs provide programming concept as "procedures" in other computer languages, for instance Logo, Fortran. The lack of proper "procedures" has always been the major limitation of many BASIC dialects as a computer language. TI XB is one of the BASICs that does provide this facility. Not all BASICs, such 25 the GW-Basic supplied with IBM style ATs that we bought recently for the lab, are

civilised. Perhaps the suppliers of these machines don't really want or expect anyone to program their machines seriously in Basic. You will 4 ind that 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. 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 to be mentioned later. At the same time XB extracts the all variables, sets aside names of space for them, and sets up the by which it associates procedure 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, and appears 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 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 pre-scan, and Basic then has its at 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 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 ٥f all XВ locates the sub-program names that aren't built into the language. Ιt can do this by finding each name after a CALL or SUB statement, then looking it up in the GROM library index o f built-in sub-program names. You can run a quick check on this process 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 sub-program code comes at the end of the program, with nothing but sub-programs after the first SUB statement (apart from REMarks unich are ignored anyway). Now this is a good way to do things as the main program is right up front there for you to inspect. One of the great about Pascal is that annoyances despite its pretensions to structured style it forces you to hide away the main program right at the end after all the procedures. MB them 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. Ιt 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 :: RE AD X :: PRINT X 120 SUB NOTHING 13Ø DATA 2 14Ø 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, 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 12Ø SUBEND

subject of misleading or incorrect camment magazines and newsletters. A little reflection on how XB handles the details will usually clear up difficulties.

BASICs assign nominal values to All of this discussion all variables mentioned in the behaviour of TI Extended Basic comes program as part of the prescan, zero from programming experience for numeric and null for strings, unlike some languages (some Basics 1981 title screen. Earlier Versions even) which will issue an error and message if an unassigned variable is Australia, but TI generally seems to presumed upon. This means that XB have taken a lot of trouble to keep can't work like TI LOGO which has a new versions of programs compatible rule that if it finds an undefined with the old. On the other hand TI variable within checks the chain αf procedures until it finds a value Editor/Assembler manual has under that name. which erases all information left within a procedure console Basic. when it is finished with it, XB retains from CALL to CALL the values Another of variables entirely contained in experiment will demonstrate what we the sub-program. Some recent Basics an other machines now

subprogram variables to be specified either as static (can be very handy) or transient (which saves storage space in between CALLs), but XB is at the fully static end of the spectrum. 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 SBP R(Ø):: NEXT I 11Ø SUB SBPR(A):: A=A+1 :: B =B+1 :: PRINT A:B 120 SUBEND

The first variable printed is reset to Ø 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 array occurs. XB doesn't tolerate attempts to re-dimension arrays, 50 information on arrays can only be Dassed dawn the chain ۵f 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 This point is not explicitly made in many times removed, will result in the XB manual and has been the an error. Recursive procedures, an essential part of TI LOGO, are NOT with XB sub-programs possible since CALLing a sub-program does not set up a new private library of values.

> with Version 110 of XB on a TI-99/4a with consoles are not common in a procedure it has also been very reticent about CALLing the details of how XB works. However, unlike little to say about it either, less the by far even than it tells about

> > simple programming lasics mean by saying that XB sets up a allow separate Basic program for each

sub-program. RUN the following

100 X=1 :: CALL SBPR :: BREA

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 time will be in the main program, where you can 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 variables and DEFed functions 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 to be enclosed in quotation marks. statements, CHARacters, SPRITES, COLORs, and File specifications.
- from the CALLing main or subprogram by the parameter lists in CALL and SUB statements. XB does 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 sub-program are static, and preserved for the next time the sub-program CALLed. i S Some languages such as Pascal delete all
- (e) XB sub-programs may not CALL themselves directly or indirectly in The first example is an example of a a closed chain. 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 CHR\$(199) flag and a length byte other programs.

PROGRAMS THAT WRITE PROGRAMS

вов спвилия

Part 2

In the last article we briefly on the three flags that are used by programs that write other programs. CHR\$(199) is the one that is used to denote a quoted string. That means that the string is to be physically enclosed in quotation marks. The only stipulation is that you must include a length byte to tell the computer exactly how many characters are in the string to be enclosed between the quotes.

CHR\$(200) denotes an unquoted string --either a numeric constant, or other word that is not

CHR\$(201) tells the computer that a line number reference in coming and (c) All other information is passed it is the exception as far as line numbers are concerned. Three bytes instead of the customary two bytes in length.

Now that we have had the refresher course, let's look at some examples of what a line of code would look like. Each line is going to begin with PRINT #2:CHR\$(INT(LN/256))& CHR \$(LN-256*INT(LN/256)). That is the line number algorithm that we covered in the last column. For the purposes of this exercise, we are traces of a procedure after it has going to replace that statement with been used. present the actual code itself.

> Subject to this simple PRINT statement with a quoted string:

> > ***&CHR\$(156)&CHR\$(199)&CHR\$(5) &"HELLO"&CHR\$(Ø)

CHR中(156) is the token Value for PRINT and it is followed by the with the CHR\$ value of the number of letters in HELLO. The line of code is ended with the null character $CHR\$(\mathcal{G})$ to tell the computer that the end of the line has been reached.

Assuming that the value of LN (the line number) is 500, the resulting line of code that is produced is: 500 PRINT "HELLO"

Here is an example of a line of code using CHR\$(200) for an unquoted string. We are going to once again use 500 as our line number value.

****&CHR\$(157)&CHR\$(200)&CHR\$(4) &"INIT"&CHR\$(0)

We have the token value for CALL which is CHR\$(157) followed by the flag for an unquoted string which is CHR\$(200) and a length byte. INIT is then printed followed once again by the null CHR\$(0) to end the line. Here is the resulting line of code as it actually appears:

500 CALL INIT

рd

re

Now we come to the last of the flags --- CHR\$(201). It is a bit more complicated than the other two. We are going to begin with the premise that we have used LNUM as an input variable to put our line number reference into.

****&CHR\$(155)&CHR\$(165)&CHR\$(2Ø1) &CHR\$(INT(LNUM/256))& CHR\$(LNUM-256* INT(LNUM/256))&CHR\$(Ø)

Notice anything familiar? We used the token for ON followed by the token for ERROR and then the line number reference flag and our line number algorithm and the line ending null.

Before we end this segment, let's look at an example of how more than one of the flags can be used in a line of code.

The resultant line of code would look like this:

500 DISPLAY AT(ROW, COL): "HELLO"

I'm sure that this all seems a lot of work for a couple of simple lines of code. The fact is that we can combine common code into a string variable and then use simple ACCEPT AT statements to write the code.

For example:

DIS\$=CHR\$(162)&CHR\$(240)&CHR\$(183) &CHR\$(200)&CHR\$(LEN(STR\$(ROW) &STR\$ (ROW)&CHR\$(179)&CHR\$(200) &CHR\$(LNN(COL))&STR\$(COL) &CHR\$(182)

This would reduce DISPLAY AT(ROW, CO L) to the string variable DIS\$. Simply by entering the values for ROW and COL from the keyboard, a line of code could be written to disk. That is the key to the situation --- common portions of code could be stored as string variables and then with a single keypress or short series of input values, a line of code could be The advantage is that it created. is free of typos and the lines of code are produced much faster than you could type them in by hand.

The possibilities for using taken values to write D/V 163 files are almost endless. The MAKELOADER MAKELOADER program mentioned in the first article of this series usod modified disk directory utility to read the filenames from the disk and created a MERGE file with tokenized versions of "RUN DSK1.filename" to actually load and run the program selections. There are other varied uses to experiment with in the use of takenized reserved words. advantage of this little-used facet of the TI and make programming easier!

This is the end of this two-part tutorial. If you are interested in creating programs that write other programs, please consult documentation to PROGRAM WRITER which is now being distributed by the UG. There is a copy of it in the library as well. As the docs state, it isn't meant to be the "end" but rather the "beginning" for those adventurous souls who would like to write programs the easy way --have the computer do it for you!!!

TI-WRITER PROBLEMS

This article appeared in the May 1989 issue of CIN-DAY Newsletter, and first appeared in the SFV 99er TImes.

PUNCTUATION

Typists will always use two spaces after a punctuation mark ending a sentence. TI-Writer, for some strange reason does things a little different. For example:

THE PERIOD (.) - TI-Writer will always put 2 spaces after every period that has been followed by a single space. This is fine if the period is at the end of the sentence. But what if you are using an abbreviation within a sentence? The formatter will put 2 spaces here also, but you properly only want one. What you need to do in this case is use the required space symbol (^) after the period of an abbreviation. This will give you the desired one space when using the formatter. (A period followed by no space will appear as just that.)

THE EXCLAMATION (!) & QUESTION MARKS (?) — in these cases the formatter will not automatically give you 2 spaces as it properly should. To make your document look correct you will need to add one space and the required space symbol (^).

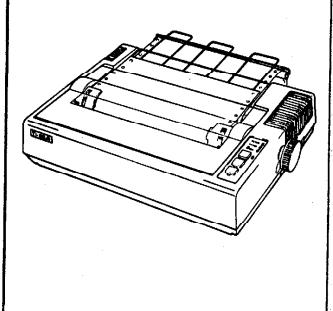
THE PERIOD & DECIMALS — the formatter thinks that any line which begins with a period is a formatter command and will delete the whole line. If by chance your document contains a value such as .10 and the wrap around caused by Fill and Adjust of the formatter puts it at the beginning of the line the whole line will disappear! To correct this you could put a zero in front of your decimals ie 0.10

ASTERISK (*) & NUMBERS - if you are printing out of the formatter and your document contains an asterisk followed by two or more digits, the asterisk and the two digits will disappear. For instance, A*256

becomes A6. What's happening here that the TI-Writer program misinterprets the asterisk and 2 digits as an instruction to input data from a 'value file', as in mailing lists. This is described on page 111 αf the TI-Writer instruction book. To correct this problem you will need to type 2 asterisks followed by 2 dummy numbers, then the actual digits. For example, type A**00256.

REQUIRED SPACE (^) - if you tie two words together for the purpose of underlining (using &) with the required space (^), the Fill and Adjust command of the formatter will leave gaping blanks in your line. If you tie too many words together the line will extend beyond the right margin. It would be better to put a separate % or 0 in front of each word. Be sure to include the spaces between the words. If you want a ^ to appear in your text you will need to transliterate it (see page 107 of TI-Writer instruction book). The & and the D are typed twice in succession to get them to print ie && or aa.

OTHER PROBLEMS - Other problems have been noted in TI-Writer that cause erratic and destructive commands, but they are not fully documented.



THE GREAT COVER UP

ut

2

by ROH PRATT

Not long ago, I visited a factory here in Newcastle where illuminated Perspex signs are made and showed interest in their use of computers in cutting the letters and locating them in position. The console in the workroom was hooked up to a pantograph machine which did the actual cutting out of the letters and I've seldom seen a more hostile environment for a computer. There were aluminium shavings and acrylic dust ALL over it! Then I saw that it was sealed in a plastic bag.

Well, whenever I come out to the den to enjoy a little writing on the TI, our pet cat climbs up alongside me and wants to get into the act too (talk about kitten on the keys), and I don't like the idea of cat hairs in the works. As well, there's the usual dust etc. floating about and the risk of spilling the cuppa (YOU don't drink at the computer DO you!). Another problem which plagues me occasionally is the need to clean the module port connection which gets dirty and causes lockup, usually at very inopportune times.

After seeing the plastic coated computer, light bulbs came on, my wife's plastic bag sealer came out of the cupboard and in ten minutes I had a very handy envelope which slips neatly over my dear old TI leaving enough room to breathe through vents. They the say simplest ideas work best and this does too. I elected to leave the module port covered, as Ex Basic is about the only cartridge I use and I can get plenty of games (for the kids of course) and Assembler on Of course if you want to use the port, its only a slit with the knife to free it up.

Of course, it feels a bit funny to type on, you have to be a bit more precise when hitting the keys, but it keeps the 'Old Girl' a lot cleaner. You might like to try it yourself. If you haven't a bagsealer, try using sticky tape.

Lastly, one from "Silly Question Corner". Why can't the gizzards from an Extended Basic cartridge be mounted inboard on the console thus ending the lock-up blues forever? Oh well, it's probably a silly idea anyway.

FORTH TOOLBOX

BOB CORHOHY

One of the best things about Forth whether is is Wycove (my favorite) or TI-Forth is that you can design and name a particular word to suit a particular function. The language is extensible virtually without limits. The words can be named to describe the function that they are designed to perform.

About a year ago, I was engaged writing a series of character sets for Wycove Forth that would be selectable as a menu option. In the course of writing the option, I had to define an ASCII character as a constant to delimit the character sets that I was storing from character 128 up. I re-defined the characters in the range 128-255 to have several character sets in memory at the same time. Anyway, here is the definition of ASCII which writes a character as constant.

: ASCII (C -- c)
I IMMEDIATE 1
BL WORD HERE 1+ C
[COMPILE] LITERAL;

How about a text mode screen dump? This one appeared in Lutz Winkler's fine series of Forth tutorials (mandatory reading for Forth programmers). With apologies to Lutz, I have extracted this word. It requires -PRINT to be loaded to work.

HEX : SCREEN-DUMP SWCH

Ø3CØ ØØ DO I DUP 28 MOD Ø=

IF CR THEN VSBR EMIT

LOOP CR UNSWCH ;

DECIMAL

Lutz suggests that you use it for recording those "experiments" in the interactive mode before they scroll off the screen. While we are printing, I ran across this one from John Burkett that does a better job of printing a screen index than INDEX supplied by TI-Forth (screen to follow).

One interesting difference that I have found between TI-Forth and Wycove Forth (I'm using version 3.0) is that TI-Forth needs a special loader even from F'WEB. Not so with Wycove!! Wycove Forth loads 25 standard E/A program image files from the F'WEB loader. In fact, is one of the reasons that I like it --- everything is "tuned" toward the standard TI system. The files access as standard and the I/O functions are also in line with TI's.

As with most broad-based languages, c99 like and others, Forth applications can be "ported" from "dialect" to another. In other words a Wycove application could þe re-written in TI-Forth with minimal The fact is that Wycove Forth sound screens have been directly written into TI-Forth to provide sound facility.

This column is a bit short this month and I would welcome any questions or applications that you would like to see of two screens or less to be written in either Wycove or TI-Forth. Until next time --may the Forth be with you!!!

C#97

Mr R Carmany 1504 Larson St GREENSBORO NC 27407 USA

DROP ELSE ů. DISK INDEXER" ." Enter index title:" PAD 80 EXPECT ; -TRAILING TYPE -TRAILING TYPE LOOP CR 1+ SWAP (from_scr to_scr --ILE SWCH CR PAD 80 -Tf SPACES R Z SPACES I BLOCK 64 F PAUSE IF LEAVE ENDIF CLS & 5 GOTOXY 12 Ø 1Ø GOTOXY PAD BØ ELANKS . 2 SPACES BEEP ! (PRINT-INDEX JEB 21APR85 ļ from to NEWINDEX († F.C.,)
DO I DUP ?ASCII
IF 4 .R 2 SPAC PRINT-INDEX SPACES . PIHDR **50 -- 10** €

١

Ì