HUNTER VALLEY 99ERS USERS GROUP

HOME COMPUTER NEWSLETTER

"Morpeth" from the *Illustrated Sydney News*, 16th October, 1865

REGISTERED BY AUST POS
PUBLICATION No. NBG8023

Hunter Valley 99ERS USER GROUP

OCTOBER 1989

YOUR COMMITTEE

PRESIDENT
Peter Smith
8 Glebe St.,
EAST MAITLAND 2322
Phone 336164 V/tl 493361640

VICE PRESIDENT
John Paton
1 Parien Close,
RUTHERFORD 2320
Phone 326014 V/tl 493260140

SECRETARY
Brian Woods
9 Thirlmere Pde.,
TARRO 2322
Phone 662307 V/tl 496623070

TREASURER
Noel Cavanagh
378 Morpeth Rd.,
MORPETH 2321
Phone 333764

Software Librarian
Stewart Bradley
14 Hughes St.,
BIRMINGHAM GARDENS 2287
Phone 513246

EDITOR
Allen (Joe) Wright
77 Andrew Rd.,
VALENTINE 2280
Phone 468120

PURCHASING/HARDWARE CO-ORDINATOR
Alan Franks
822 Pacific Highway,
MARKS POINT 2280
Phone 459170

SOCIAL SECRETARY
Robert (Bob) MacClure
75 Deborah St.,
KOTARA SOUTH
Phone 437431

PUBLICATIONS LIBRARIAN
Ken Lynch
9 Hall St.,
EDGEWORTH 2285
Phone 585983

COMMITTEE MEMBERS
Don Dorrington
36 NELSON St.,
BARNSLEY 2301
Phone 531228

Tim Watkins
36 The Ridgeway,
BOLTON POINT 2283
Phone 572836

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 submited 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 HV99 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.

FROM PRESIDENT PETER

My, it's almost that festive time of the year again. Doesn't it come around quickly?

Some interesting things are going on in the club at the moment.

Congratulations to those prize-winners from our last meeting. A number of modules and a mini-mem and a car chamois went to a number of our members... No need to buy tickets, just being a member was enough as Max McVee, Graham Kittle, John Paton, Ken Lynch and David Spicer found out.

This meeting sees the long-awaited demo of Joe Wright's latest version of Genealogy running under Forth while next meeting sees Forth's Richard Terry presenting a number of his contributions to the TI community.

At that November meeting, we will conduct a new event for our club- an auction/swap session. The club hopes that members will take the opportunity to sell un-needed equipment and also pick up a good bargain ready for someone to 'give' to them for Christmas.

The auctioneer will be Jim Bradley and the club will profit by a small fee deducted from the sale price. Members wishing to sell should remember that gear should be accurately marked with name, reserve and any faults which exist. Anything defective and not so marked will have to be refunded by the vendor.

The Christmas party for the kids looks as if it is shaping-up well with lots of ideas for the kids, wives and all who attend. A competition will be completed on that day with tickets going on sale at the October meeting. Prizes are being organised and at the moment look like being very worthwhile. Please support it to the full.

Regards...
Peter.

EDITOR'S LAMENT.

Very important!!. A questionnaire has been included in this months newsletter It is related to the newsletter, what you think of it, what if any changes you might like etc, we would also like your general comments to be written on the bottom of the sheet. Then return the sheet to me the EDITOR, I will then pass them along to the person who proposed it for analysis and appraisal. While on this subject I want to thank that person for his suggestion to run the questionnaire.

You will also notice that the glossary for this month (C) is not complete, this is because it would have given me a blank page to complete it. The remainder of C will be included with D next month.

Also no Extended basic tutorial this month, they will continue next month.

Christmas is getting closer, don't forget your input to the Christmas Bumper issue of the newsletter.

Content of the newsletter. It is a policy of our group that the Editor has total control of what goes into our newsletter. It is at his discretion to accept and reject articles for printing. Currently I have not found any article which could not be printed. Generally I will print anything that I think will be of general interest to our members and others who read the newsletter. articles not directly related to the TI will be published so long as they have a broad general interest to people interested in computers. Examples are programming methods not strictly related to the TI but applicable to computers in general. Small articles not related to computers but also of general interest will also be printed so long as most of the newsletter is devoted to computing. We have people with many talents not strictly associated with computers. I see no reason why their knowledge should not be shared within reasonable limits. What do **YOU** think?

Joe Wright.

Brian Woods REPORTS From the Secretary's ** Desk **

MONTHLY MEETINGS

The November monthly meeting will be the final one for the year. At that meeting we will be having a demonstration of some of Richard Terry's handiwork in Forth as well as an auction sale for those members who wish to buy or sell some TI related hardware. At the conclusion of the auction we will have a Christmas/breakup party, with the club supplying something to eat. Members who will be in attendance are asked to BYOG if you so require - tea & coffee will be available for the drivers! Make the effort & come to this meeting.

SOFTWARE LIBRARY DATABASE

Thanks to the efforts of many in the group, and especially Peter & John, the first instalment of the Software Library Database has been passed on to members. I hope you all look through it carefully and contact Stuart, the Software Librarian, and put your order in for some of those 'you beaut' programmes that are available for your computer.

BOB CARMANY UPDATE

Joe Wright has received word from Bob that he will be in Newcastle in late February next year and is looking forward to meeting as many members as possible at the February meeting. Those who correspond with Bob I know will be anxious to meet him, and those who don't will get the chance to meet the man responsible for much of the input to our newsletter.

ZENO BOARD

As reported some months ago, Eric Zeno was working on a new piece of hardware for the TI. The August issue of MICROpendium had the following report...

"Eric Zeno says the 'Internal Board', also known as the 'Zenoboard', is now being manufactured in quantities of 100."

"According to Zeno, the board allows the user to add 32K memory, a clock circuit, Extended BASIC and the Speech Synthesizer to the interior of the TI console."

"The boards are $US17.50 each. Documentation, approximately eight pages of schematics, builders notes, parts lists, software for the clock and the parts placement overlay, is available for $US1 per copy."

"...shipping outside the US is $US8 for the first board & $US6.75 for each additional board."

"Zeno says the board includes three additional, switched GROM sockets. Any circuit configuration can be used. The board requires no additional power and is compatible with all other known hardware and software, he says."

For information, or to order, contact Zeno at 414 Highland Road, Pittsburgh, PA 15235 USA.

THE BLOB

From the September issue of TI-Focus, the newsletter of the Channel 99ers Users Group in Hamilton, Ontario, comes this small program written by Jim Peterson.

```
100 CALL CLEAR :: CALL MAGNIFY(4) ::
CALL SCREEN(2)! the blob by Jim
Peterson
110                                CALL
CHAR(96,RPT$("3C7EFFFFFFFF7E3C",4))
:: J = -1
120 FOR L=1 TO 28 :: CALL SPRITE(#L,
96,16,L*4+20,10,0,L+8) :: NEXT L
130    FOR   L=1   TO   28   ::   CALL
MOTION(#L,0,L*J) :: NEXT L
140 J=J*-1 :: GOTO 130
```

ASSEMBLY LANGUAGE INSTRUCTION SET

This article came from Bits, Bytes & Pixels, the newsletter of the Lima 99/4A Users Group. It was written by Andy Frueh of that Group.

This is for Assembly programmers. I have never seen a chart of the TMS9900 instruction set without a lot of `Greek' in between. This chart simply gives the code, and what it stands for in what may be a simpler to understand `dictionary' form.

A	add a word
AB	add a byte
ABS	absolute value
AI	add immediate
ANDI	and immediate
B	branch
BL	branch & link
BLWP	branch load workspace pointer
C	compare word
CB	compare byte
CI	compare immediate
CKOF	clock off
CKON	clock on
CLR	clear operand
COC	compare ones corresponding
CZC	compare zeroes corresponding
DEC	decrement by one
DECT	decrement by two
DIV	divide
IDLE	idle
INC	increment by one
INT	increment by two
INV	invert, ones complement
JEQ	jump if equal
JGT	jump if greater than
JH	jump if high
JHE	jump high or equal
JL	jump if low
JLE	jump if low or equal
JLT	jump if less than
JMP	jump unconditional
JNC	jump if carry clear
JNO	jump if overflow
JOC	jump if carry set
JOP	jump if odd parity
LDCR	load communications register
LI	load immediate
LIMI	load interrupt mask immediate
LREX	load ROM & execute
LWPI	load workspace pointer immediate
MOV	move word
MOVB	move byte
MPY	multiply
NEG	negate, two complement
ORI	or immediate
RSET	reset
RTWP	return workspace pointer
S	subtract word
SB	subtract byte
SBO	set CRU bit to one
SBZ	set CRU bit to zero
SETO	set ones
SLA	shift left, zero fill
SOC	set ones corresponding (word)
SOCB	set ones corresponding (byte)
SRA	shift right, circular
SRL	shift right, logarithmic
STCR	store from CRU
STST	store status register
STWP	store workspace pointer
SWPB	swap bytes
SZC	set zeroes corresponding (word)
SZCB	set zeroes corresponding (byte)
TB	test CRU bit
X	execute
XOP	execute operation
XOR	exclusive OR

I hope that this helps decipher some Assembly programs, & maybe get a few XB programmers started on Assembly. The idea of having all of the commands on one page, and in an easy to follow dictionary form seems like a great idea. I hope it helps someone!

FOR SALE

* 99/4A console with power supply & TV Encoder
* Expansion system:
  - single disk drive DSDD
  - 32K memory expansion
  - disk controller
  - flexible interface
* Speech Synthesizer
* 'Computemate' CP 80 printer with RS232 loom
* All instruction books
* Programs:
  - TI Extended BASIC
  - Beginners BASIC
  - Touch Typing Tutor
  - Terminal Emulator
  - Disk Manager 2
  - Personal Record Keeper

Contact John Fulham 049 682028

AL'S MARKET PLACE UPDATE WITH *Alan Franks* ©

Well nearly another year gone and our orphen is still going strong, due I think to sheer determination on the part of the remaining TI user groups. TI itself must be wondering when its going to finaly lie down. Anyway enough rambling on this month there is a couple of full systems going for anyone who wants to expand.

The first system consists of an American Console, PE box, 32k card, RS232 card, Disk Controller card, One Single Sided Drive, TI printer, Monitor, Multiplan, Editor Ass, and an assortment of other modules and disks.

The second system consists of a Console, PE box, 32k card, Axium Interface with PIO or RS232 switch, Second Drive, Joysticks, Speech, Ex Basic, Multiplan, TE2, Editor Assembler and a few games modules plus books and magazines.

So if anyone is interested in buying either one of these systems or part there off, please let me know as soon as possible as I am looking at buying these systems and splitting them up if enough people are interested.

Also note the system for sale in the secretary's column.

All men need something to poetize and idealize their life a little-something which they value for more than its use, and which is a symbol of their emancipation from the mere materialism and drudgery of daily life.

Theodore Parker.

PRINTER PROBLEMS

by RON KENNEDY HV/99

My problem consists of a Corcomp RS232 card and a Star NX 10 printer. These two work in perfect harmony for writing text and other normal uses. But! when I want to use Graphx or other similar programs I enter an era of non co-operation. Each unit will work perfectly with other like equipment but refuse to operate together. I have run the printer with ALAN FRANK'S RS232 card and have run ALS printer with the Corcomp card. The ribbon cable for the printer is made to the specs in the instruction manual. ANY suggestion will be considered favourably. PLEASE contact 049 43-8926.

NOTHING
* * * * * * *

Well this is reader punishment time. With the newsletter complete I had this space left to be filled, by WHOM?? I asked the 4A. "You MATE! that is who".

So if I have to accept pain then why not the reader? Good idea, no black holes, no gooeys, no outback races, no funnelweb spiders. No! just plain old nothing, I happen to be big on nothing, its one of my strong points, Bull! it is my strongest point. Any time you want nothing done just ring me. I might not even answer to phone just to prove I can do nothing.

Effiency I hear you say, THATS ME. If nothing gets done then the energy consumed must be zero. Any equation divided by my energy usage which is zero is gonna be a good effiency figure by any standard. Maybe if I started doing less than nothing I could even produce energy. Then again that would be doing something wouldn't it. Completely mess up my efforts.

NOW! if you don't want to be like me then do something, WRITE FOR THE CHRISTMAS BUMPER ISSUE. Anything will do so long as you remember that I have also ready decided to do nothing so YOU can't do that Ha!.

JOE

A COUPLE OF CALL KEY CONCEPTS TO CONTEMPLATE

by 64296572747356671561445944515 5

I recently unearthed some notes on my early encounters with TI Basic according to URG. Among them were a couple of applications of CALL KEY that I haven't seen in print. As I am narrowly read on things TI, they are probably old hat. Nevertheless, even if this article is no more than a refresher, here they do be.

My first problem with CALL KEY was in trying to comprehend URG's explanation of the various status values, with the references to new keys and same keys as well as performances. There must be some difference between a previous performance and a last performance?

After running a few checks it all became clear. After a CALL KEY statement -

S = Ø if no key is tapped (I don't press keys on my 4A)

S = 1 if a key is tapped. And if you keep on tapping a key (it makes no difference if it's the 'same' key or a 'new' key), S will keep on recording 1 as its value. If you hold down a key, S will record 1 on the first pass and then

S =-1 as long as the key is held down.

Following on from which, the usual format for CALL KEY is

```
CALL KEY(key-unit,R,S)
IF S=Ø THEN etc
```

Of course there are variations such as IF S<>1 or IF S<1 and no reference to S at all. However, this is not a CALL KEY tutorial, so there won't be any in-depth exposé of S.

One point URG doesn't mention is what happens with R. Of course, when a key is tapped, R records its ASCII value. But if no key is tapped, R maintains a value of -1. So why not

```
CALL KEY(Ø,R,S)
IF R=-1  THEN  etc
```

It works. Any enlightenment from knowledgable readers as to why R=-1 is not, or should not be used??

It also seemed obvious that, when 4A encounters the CALL KEY/IF S=Ø bit, it scurries (or whatever computers do that counts as scurrying) back and forth between the two lines expectantly awaiting a key tap.

Knowing that computers can scurry very quickly indeed when they have a mind to, I decided to check out 4A's CALL KEY scurry rate with a counter. Surprisingly, it's a mere saunter - about 1000 times a minute.

Following right along, here was an obvious means to prompt tardy key-tappers.

```
10CALL CLEAR
20PRINT"TAP A KEY":::
30CALL KEY(Ø,R,S)
40C=C+1
50IF C=49 THEN 70
60IF R=-1 THEN 30 ELSE 110
70PRINT"HURRY UP!":::
80CALL KEY(Ø,R,S)
90IF R=-1 THEN 80
100PRINT"ABOUT TIME!":::
110PRINT"** DONE **"
```

There's about a 3 second delay before C totals 49 and triggers the prompt. My guess is that 1 scurry is what URG refers to as a performance.

For my own programming use (in Basic) I've scratched together a minimum-frills CALL KEY subprogram. It has no validation or similar extras as I reckon I know what I'm doing - on odd occasions anyway. It uses the same counter idea to provide for multi-key inputs.

Whilst CALL KEY is regularly used for single-key inputs, INPUT seems to be the usual preference for longer data items. I have seen a multi-key CALL KEY subprogram using GOTOs, but not the counter type (being narrowly read). Anyhow, here's how my version hangs together.

```
100REM PART OF MAIN PROGRAM
110REM REQUIRING FOR INPUT
120REM YEAR OF BIRTH
130CALL CLEAR
140H$="IN WHAT YEAR WERE YOU BORN?"
150FOR L=1 TO LEN(H$)
160CALL HCHAR(8,3+L,ASC(SEG$(H$,L,1)))
170NEXT L
180KU=Ø
190CKV=4
200CKP=1
210GOSUB 12010
220REM INPUT DATA IS NOW
230REM AVAILABLE FOR USE
240PRINT VAL(CK$)
250STOP

12000CALL CLEAR
12010CALL KEY(KU,R,S)
```

```
12020IF R=-1 THEN 12010
12030IF CKP<>1 THEN 12050
12040PRINT CHR$(R);
12050CK$=CK$&CHR$(R)
12060V=V+1
12070IF V<>CKV THEN 12010
12080IF CKP<>1 THEN 12100
12090CALL CLEAR
12100RETURN
```

All of which is fairly obvious. The variables are set in the main program as required. KU is for the key-unit (already spotted that hadn't you?). CKV is the counter variable - set for 4 this time (4 digit year input). If a display of the input is required as keys are tapped, CKP is set to 1. For long input items, a screen display is necessary to obviate mistakes - and a validation sequence.

How about the case of a day-date input which may require either 1 or 2 digits do I sense some perspicacious reader wondering? As mentioned, it's for personal use, so, for a single digit day I'd just whack in ∅8 for example, which VALs out as 8.

As I was knocking up this article - hence the ramshackle presentation - it occurred to me that I could possibly amend my subprogram to incorporate a 50-scurry counter to cater for any size input. Then, when no key tap occurred for about 3 seconds, 4A would figure 'that's the lot' and get on with the rest of the program. It might be a smidgin slower than INPUT (even with its need to ENTER at the tail end), but what's a couple of seconds in a lifetime of computering?

Not this time though - I'm just about knocked up already. And so, that's it. Oh - if you want to decipher the 642965 --- bit, it can be used as data. But it's definitely not worth the effort. Just thought I'd throw it in at the finale as extra padding.

```
10CALL CLEAR
20DATA
30READ E$
40CALL COLOR(2,1,4)
50CALL VCHAR(5,28,42,236)
60CALL HCHAR(20,5,42,316)
70CALL COLOR(2,2,4)
80FOR L=1 TO LEN(E$) STEP 2
90CALL COLOR(2,L/2,(L+3)/2)
100CALL HCHAR(12,9+(L-2)/2,
  L+VAL(SEG$(E$,L,2)))
110NEXT L
120PRINT"** ALL "
```

Did you try it? Feel fortunate, the prototype had CALL SOUND as well. Happy computering!

E

Beating Around the BUSH

With Ron Klienschaffer

There is that old saying that things come in threes, well let me tell you mate, that is not quite correct, listen while I relate some of the happenings here in the sticks.
For starters the old truck (the one that went missing for a while) decided that it would blow the hoist seals and wouldnt dump a load, then the excavator had a fit and it also blew the main lift ram seals, after this my house generator (TI power supply?) broke its cemented in base bolts and was trying to walk off down the paddock everytime I started it, was that all?, no way mate!, the old utility that carries the mining generator had a spasm and tore out the diff centre.

If that wasnt enough along came Atrax's and Squishers and other nasties like Bufadr's, I tell you its enough to make a cowboy shoot his horse.

The first lot of problems were bad enough, after all to replace a half ton hydraulic rams seals only requires a lot of getting very dirty, busting the proverbial poofoo and using some assorted and carefully selected words at times, but the problem set by T.McG. in the August newsletter caused a lot of heavy thought (very painfull) and a bit of mild hair tearing.
After the initial study of the article and gaining some comprehension about exactly what the routine does it was onto the keyboard and clatter up some source code to do the job, piece of cake, lets see?, sheeez 56 words of code,

not real good, like the man said, back to the drawing board!. After a bit of fiddling with the original source a few ideas started to creep into the old gray matter, s l o w l y the code started to reduce in length untill the final breakthrough, BINGO 37 words and bug free. So what next?. after a few phone calls and a bit of pleading I managed to get Tony's new Editor files for assembly for beta testing, funny how a sticky beak cant help himself, I just had to load the files and poke around in them a bit.

It is curious how tasks to do the same job can be accomplished in different ways and this routine is no different, my code is very different to that of Tony's and I wonder just how many ways it could be written, that is the beauty of assembly, something new can be found every time you write in the language, and the learning process never stops.

PS: If you hear a loud bang and read reports of a fireball in the north of the state you will know I blew up all my machinery!.

F

TI-Writer U CODES

Reading some back issues of Bits, Bytes abd Pixels and found an article by Jim Peterson on the use of CTRL U codes in TI-Writer. I have listed below some of the codes given.

These codes are for a Gemini 10x printer, no harm to try them on your printer is it!.

CTRL U, FCTN R, CTRL U SHIFT G -- Print in double-strike mode.

CTRL U, SHIFT O, CTRL U -- Set print pitch to condensed print.

CTRL U, SHIFT R, CTRL U -- Cancel condensed print.

CTRL U, SHIFT N, CTRL U -- Print in double-width for one line only.

CTRL U, SHIFT T, CTRL U -- Cancel double-width print.

CTRL U, FCTN R, CTRL U, SHIFT H -- Cancel Double-strike.

CTRL U, FCTN R, CTRL U, SHIFT E -- Emphasized mode.

CTRL U, FCTN R, CTRL U, SHIFT F -- Cancel emphasized mode.

CTRL U, FCTN R, CTRL U, SHIFT -, CTRL U, SHIFT A, CTRL U, -- Underline characters.

CTRL U, FCTN R, CTRL U, SHIFT -, CTRL U, SHIFT A, CTRL U, -- Cancel underline.

CTRL U, FCTN R, CTRL U, SHIFT S, CTRL U, SHIFT , CTRL U, -- Print superscript mode, which is always double-struck and unidirectional.

CTRL U, FCTN R, CTRL U, SHIFT T -- Cancel superscript and unidirectional.

Any combination of codes can be used, including those which move the print head or move the paper (line feeds).

CTRL U, SHIFT H, CTRL U -- moves the print head back one space and can be used to overprint characters.

CTRL U, SHIFT M, CTRL U -- placed before the end of the line will send the print head back to the beginning of the line and reprint it with the rest of the line.

Remember that these control characters are deleted by the printer and the line of print is shifted left by the number of spaces that they occupied. If you are inserting codes into text that has been prepared in column format, or right justified through the formatter, the best way is to use CTRL 0 to get the open-square cursor; position it over the character to the right of the point of insertion; FCTN 2 to insert characters; tap the space bar asmany times as there are characters to be inserted (be careful not to shove the end of the line into oblivion!); FCTN S to backspace and then fill the blanks with control characters.

Good luck
Joe Wright.

Random Bytes from Bob Carmany

Misunderstood and overlooked are two words that adequately describe the topic of this months' column -- XB extensions. Over the years, there have been several extensions written for the XB environment. Some of them are simple and some are very complex but they all have one thing in common ---they take advantage of the A/L LOADing and LINKing capabilities inherent in XB.

For our purposes, we are going to be discussing only "fairware" and out-of-production commercial programs. With that in mind, let's start at the beginning. One things that all of these software packages have in common is that they are composed of a series of A/L routines that are LOADed into memory with the XB CALL LOAD command. Then, when you want to access the routine from an XB line of code, you use CALL LINK along with the required parameters --- a small price to pay for what you get!

With most of these routines, there is a considerable overlap in the routines that are used. They each have a 40-column mode and have a variant of the XB DISPLAY AT and ACCEPT AT routines. There are enough variations between them, however, to justify having a copy of each. There is one notable exception!!

Several years ago, AMERISOFT came out with a "Graphing Package" that was truly remarkable. Basically, it was the renowned APESOFT graphics commands reduced to LINKable XB form. There are a series of routines to allow the user to draw virtually any shape and dump that creation to the appropriate printer. It even has the capability to create true windows in XB. The 30 new commands that the AMERISOFT package contains make it a very distinctive addition to the XB environment. The fact that this package is unique is enough to recommend it as an addition to your library. Unfortunately, it is out of production but you might be able to find a copy "floating around" somewhere.

The next two packages that we are going to take a look at are very similar. XXB (eXtended eXtended Basic) is a series of routines created by Barry Traver et. al. that is now in Vn 1.5. There are the obligatory 40-column routines and several others that are of notable interest as well. There are disk read and write (RAW) routines, a series of disk management routines, and some valuable PEEK and POKE routines for moving data to various places in memory. There are even some utilities for changing linenumber references within a program itself.
One of the most intriguing aspects of this package is the inclusion of a MERGE routine to eliminate the necessity for CALL LINK entirely. For a slight sacrifice in speed, you can access the routines directly as if they were resident commands in XB. For example, CALL LINK("DISPLA"ROW,COLUMN,MESSAGE$) becomes simply CALL DISPLA(ROW,COLUMN,MESSAGE$). All in all, a very interesting package.

XDP (eXtended Display Package) is another similar extension of the XB environment. Many of the routines are the same functionally as the XXB package but there is enough of a variation to warrant to look. Both 28-column and 40-column modes are available as well as some other utilities as well.

The latter two packages are now available in the UG software library having made the trip with this column. Remember that they are "fairware" and the appropriate donations to the authors are encouraged.

From here, the list goes on and on! There is the Display Enhancement Package, STAR and several other entries of both "fairware" and commercial quality. You can use them in your XB programs to create some truly dazzling effects ---like recalling previously saved graphics screens and otherwise stretching the XB environment to its full potential.

Well, I'm rapidly getting another 'Buffer Full' message so 'til next time . . .

TI-Writer Once Again
Tony McGovern

Several years ago I wrote a HV99 Newsletter article entitled "TI-Writer under the Hood". As far as I know, this general overview of its Editor's internal operation, as distinct from all the how to use it articles, still seems to be the only published discussion seen here of the Editor's internal workings. This is not a very good reflection on the state of information sharing in the TI-99 community, as quite a few people must have studied its workings over the years, even if only a few have ever seriously attacked it.

Recently I resumed work on a substantial rewrite of the Editor in preparation for updating Funnelweb to Vn 4.20, and while the details are fresh in mind I will write about the operation of the buffer manager in the Editor. For a gentle introduction dig out your old HV99 and read the original article again. This time we will go into a lot more detail, at the level of talking about the routines used, even if we still stay clear of detailed coding.

To remind you of how it goes, the hi-mem segment of CPU RAM contains some buffers, the buffer manager code itself, and starting at >A410 the pile of squished lines growing upwards and the line table growing downwards from >FFC8. When these meet up you have the dreaded "Text Buffer Full" condition. Each entry in the line table is a pointer to the squished text record in the Edit buffer pile corresponding to that line number used as an index into the line table. Records are only ever added to the top of the Edit buffer, and if a line is deleted it is removed and the pile collapsed down so there are no gaps left.

The buffer manager code as bequeathed to us by TI contains 4 major functions called as BLWP procedures and a small cast of supporting routines. The main procedures are, to give them TI's labelling, GETLN, INSTLN, UPDTLN, and DELTLN. As you can see from the names these are line oriented. The minor routines are

BINDEC - converts a line number in binary form to the 4 digit decimal you see down the side of your screen.

CKLIN - given the line number it returns the line table and edit buffer pointers and the length of the record.

CKROOM - this is the one that returns the 'Text Buffer Full' if it isn't satisfied.

MOVMEM - removes a record from the edit buffer pile by dropping down all the records above it in a mass move of the Edit buffer contents.

SQUISH - this is the one we talked about in last month's Assembly programming contest, that does run length encoding of text lines before they are placed on the top of the Edit buffer pile. SQUISH, and the unsquisher in GETLN, have had a functional update in that the encoding has been improved slightly to give greater apparent buffer capacity. The improvement varies depending on the document file from none at all, through about 3% for a file like this one , to the 5-10% range for assembly source files. Buffer capacity is never decreased. As before buffer encoding is purely internal to the Editor.

These services are called on by the main routines as needed I will start from the simplest function (not necessarily the simplest code) in looking at how these work. All the buffer management functions in the standard TI-Writer are performed by these routines, and as we shall see, in very inefficient fashion because the basic services are all line oriented.

GETLN - this procedure fetches a specified line from the Edit buffer for purposes such as refreshing the screen buffer, for Find String etc, and for Saving to disk. It uses CKLIN to find the record in the Edit buffer and then unsquishes the record as part of the service. At the level of detail coding this was the sloppiest effort by TI on any of the routines. I have left the function alone but have refined the code extensively. At the intermediate level the original code is inefficient in that it copies the line from the Edit buffer into the squish buffer before unsquishing to the destination buffer, where it could have been done . directly from buffer to destination and now is of course.

INSTLN -- whenever a new line has to be added to the buffer, the INSertLiNe procedure is called. This uses SQUISH to encode the line into the squish buffer, and it is then copied at the top of the Edit buffer pile, with CKROOM testing for "Buffer Full" before this is done. Then the new pointer is inserted into the line table and all pointers above it shuffled along to make room. Nothing too much needed updating in this routine.

UPDTLN -- when you edit a line and changes are made, it has to be replaced in the Edit buffer by the UPDateLiNe procedure. This fetches the data for the existing line with CKLIN, SQUISHes the new line, and then checks the length against that of the existing line. If this is unchanged the record is written back over the earlier version without further ado. If the lengths don't match then it has to delete the existing line and insert the new line. A subtle improvement on the original is that it now uses only the increase in length when it calls CKROOM.

DELTLN -- when you use <fctn-3> or <ctrl-N> to delete a single line DELeTeLiNe does the job. For normal documents this is the most expensive operation of all, because when a line is deleted from the Edit buffer the whole buffer above where that line was stored must be moved down. The reference in the line table must also be deleted and the rest of the table moved along, with pointers to any record that has been moved adjusted appropriately. Yes, all that goes on every time you delete or even just edit a line. Since the line table has only a 2-byte word aligned entry for each line (to give the CPU memory address of the actual record) it is usually much shorter than the the Edit buffer. However just how much of the Edit buffer has to be moved depends on past history. A well ordered buffer is one in which the records start from the bottom at >A410 in order of their line numbers. A file freshly loaded from disk is in this state. Editing any line enough to change its length will, as we have seen already, end up with it pulled out of sequence an placed at the top of the pile.

TI-Writer uses these buffer manager routines in the form that we have described them, always in a line at a time fashion. The most annoyingly slow function of all in TI-Writer is Delete Lines and this was the one I decided to attack first as it is also called by other functions. Why is it so slow ? The reason is that each time a line is deleted, it is done as a separate task and a complete mass move of the Edit buffer and update of the line table is performed for deletion of that line. Code polishing improved this noticeably, but a new approach is clearly needed. So a new procedure was added.

DLBLOC -- when given a range of line numbers to delete this keeps checking successive lines to see if they form a contiguous block in the Edit buffer. Each such block is deleted as a single unit from both Edit buffer and line table. So with a well ordered buffer even large blocks are now deleted almost instantaneously. You could imagine a large buffer in the opposite of a well ordered state - then the process would proceed a line at a time, but usually there is some degree of block ordering. What you can see is that TI's original method is guaranteed to give the worst possible result every time. Oh well, they invented GPL too! DLBLOC is now also called from Reformat when it has finished making a reformatted copy of a paragraph to get rid of the original paragraph, and so Reformat is now speedier.

Move Lines and Copy Lines are also very much slower in TI-Writer than one might reasonably expect them to be. Once again this is because line by line services are used to construct block related functions. Apart from speed a bad failing of Move Lines is that it can give rise to a "Text Buffer Full" condition. If you think about it this should not occur as no text is being added. It happens because TI-Writer implements Move Lines by first doing a Copy Lines - which is where the full buffer disaster may strike, and then deletes the original lines, adding insult to injury by doing it one at a time always. At this stage DLBLOC had already speeded things up very noticeably, but clearly something much better is needed. In principle all that is necessary is to shuffle around blocks of the line table, and the Edit buffer need not be altered at all. So a new general purpose support routine, MVLINS, was written that could be called as whole or part of buffer functions.

MVLINS -- this is a routine that does the line table block shuffle. It is written as a generic routine, and the block limits are pre-ordered by the calling procedures, William came off his Amiga for long enough to make an initial suggestion on how to do it in linear time in block size, and I think the resulting code is the most elegant piece of 9900 code I have yet written.

TI-Writer did Copy Lines by using GETLN to fetch each line and INSTLN to add a copy to the Edit buffer and line table. Once again this is about as inefficient a method as could be dreamed up, and made worse by all the unsquishing and resquishing involved. A further bad feature is that if the buffer fills during the process you are left with a partially copied mess.

CPBLOC -- the obvious solution here is to do direct copying of the lines from their existing position in the Edit buffer to append them to the end of the buffer, and to add new pointers to the end of the line table. In this process a new entry point CKCOPY to CKROOM is invoked which checks for room in the buffer before copying each line. At this stage the copy process can still be abandoned leaving everything as it was since nothing has yet been done inside the previous Edit buffer and line table limits - only additions to the ends. If all is satisfactory then the end pointers for the buffer and line table are updated. The Edit buffer is now in fine shape but we have a whole block of line pointers out of position, unless they were originally meant to be added at the end. But all it takes now is to call MVLINS with the right pointers and the job is all done.

An aspect of Copy Lines that bares a little thinking about is that the new approach would make it possible to do a Copy Lines with the "After" line number in between the "From" and "To" line numbers. This could be done because the copy of the block is made before any reinsertions are made, but it is not implemented because the Copy Lines and Move Lines call a common routine to take down the line numbers and do range checking. The conditions for Move Lines are more restrictive of course.

A further modification to the buffer manager code which has not yet been incorporated, is to abandon the fixed squish buffer and float it at the top of the Edit buffer, with appropriate modifications to the code. There is no net gain in buffer capacity, but since lines are always added to the top of the Edit buffer (except for the direct replacement in UPDTLN) it will mean that the squished line is already in place without having to be copied into position again. Strictly speaking it is only the direct replacement path in UPDTLN that forces provision of an explicit squish buffer, since you cannot tell that a replacement line is the same length until it has been squished. The other major area besides the buffer manager that will bear looking at is the Load/Save module. The change just discussed will help this along by reducing overhead in the file reading process. Another way to help it along was discussed as an aside in the Living with Spiders series on interfacing to Funnelweb published in the HV99 newsletter many months ago. This is the technique used in the E/A Utilities object loader, and extensively exploited in the LINEHUNTER utility program that has been part of the Funnelweb package for what seems like very long time now. The idea is to use the full DSRLNK only for the initial access to a device, usually a disk DSR, saving the CRU base address and the entry address, and using these directly for future accesses to the same file. Linehunter gets its speed by keeping track of these separately for both MAIN and COPY files.

The revised buffer manager has been incorporated in the 40 column version also, and the speed-up is such that it now seems impossible to lose a keystroke on end of line automatic wordwrap. It is still just possible to lose a repeated character in the AVPC version, but remember that this has to write up twice as much on the screen in between each keystroke, and do it via the 8-bit bus to the PE-Box as well. I must compare this against the Mechatronics 80-col unit sometime. Another upgrade to the 40 column version is that <ctrl-Y> now functions as a full margin release as has been the case in the AVPC version.

Another new feature which is proving its worth right now in writing this document is the provision of dual tab settings. The command line entry "DF" is now obsolete as SD does the job better, and has been replaced by "ST" for Swap Tabs, which swaps the current tab set with an alternate set. So when I want to write inset paragraphs all I have to do is go to command mode, ST, and Enter. Both tab sets are now saved with the document and if an old document with only one tab set is read in this is made the initial alternate set also. In the AVPC version the current tabs show on the line 26 ruler when called for editing by T or ST. Pressing <ctrl-=> cleans the ruler line to numbers only. I have also added a NTSC/PAL toggle from <ctrl-N> in the command mode. The control key may change but it seems a worthwhile inclusion (assuming your monitor will handle it) as I find it easier on the eyes to read the less elongated screen characters with less "lininess" to the display. Interlace mode becomes pretty horrendous though at 50 Hz frame rate. I still have to sort out why the output from the 9938 seems so much worse than standard TV.

Users of the Programmer's Editor will find a whole new mode of operation. In the Word Processor pressing <ctrl-O> toggles the cursor between the solid word-wrap shape and the hollow cursor to indicate fixed mode. Previously the Pgm/Ed allowed only a modified form of fixed mode to give E/A compatibility of saved files and to make sure no reformat could ever be done on program source files. Now <ctrl-O> toggles in a diamond cursor to indicate that you are in assembly source mode (ASMode from now on). It has always been a nuisance switching between upper case for the assembly code itself and to lower case for the comments. In ASMode you may leave the alpha lock off, and type both code and comments on the same line in lower case, with <shift> needed only for capitals as in normal typing. It doesn't matter that the assembly instructions are in lower case because ASMode converts them to upper case before updating the line in the Edit buffer. I recall from the brief look we had at My-Word for the Geneve a year or more ago that a "C" could be added to the tab line to control conversion to upper case. Atrax Robustus would never be satisfied with anything so inflexible. What ASMode does is to parse partially each line as an assembler source line. Comment lines are ignored, and the line parsed for the first three fields (two if no label starts in the first column) separated by blanks. Some error checking is done to catch a couple of common typing errors that can be difficult to find by eye. It indicates these by giving a little audible error bloop as the line is converted, and ceases case conversion at that point to make it more obvious. In the Opcode field only alpha characters are allowed as all legal opcodes and directives contain only these - it dislikes things like L1 or SOCB. In the Operand field ASMode leaves any entry
between single quotes unaltered - it is not allowed to touch yo
text data strings. It also checks for an uneven number of sing
quotes. Further it notices any "." not part of a text string
lets you know - as in LI R8.6 for instance. There are a coupl
minor bits of anomalous behavior, but they are only incidenta
very occasional, and a fix is not a high priority in assignin
code space. It bloops on the filenames in COPY directives, w
in standard TI Assembler syntax are indicated by normal doub
quote marks and usually contain one or more "."s not isolate
within single quotes. Some opcodes do not have an operand f
and if these also carry comments the first word of the comm
any, will be treated as if it were the operand field and co
to upper case. Most commonly these are the RTWP opcode anc
pseudo-instruction. Usually the function of these is so l
and obvious that they don't deserve a specific comment, but n
harm is done anyway. In practice neither problem is a real
bother, but if they do grate on any sensitivity just turn ASMode
off, or never turn it on in the first place.

This new ASMode code is written over the Reformat code on the way in so that it will no longer be reasonable to have immediate re-entry to the Editor interchangable between Word Processor and Program Editor. In fact instant reentry is a very marginal feature at best, and may end up being omitted altogether if it gets in the road of more desirable features.

From our anonymous Foster Parent

(An Unpaid Unsolicited Advertisement)

"You nearly finished with your 99/4A for tonight Bruce? I'm ready for bed."

"Yeah, Charlene, I was ready to give it away anyhow."

The Place: Somewhere in Suburbia where a fortunate 4A has found a foster home.

"How are getting along with Basic?"

"I think I've got the hang of it. I found that using it and writing about it seems to make it easier. I'm even writing an Article For The Newsletter."

"You Bruce - for the HV99 Newsletter?"

"Yeah, Charlene, all contributions are greatfully accepted. And when I came across that book you bought a couple of years ago - Real Men Don't Eat Kwitchy."

"That's not how it's pronounced Bruce!"

"Whatever - anyway it gave me the idea for a title for my Article - Real Men Program in Basic."

"But you're a timber-cutter Bruce - or rather were. Do you think you know enough about computering yet to write an Article For The Newsletter?"

"I'm not sure Charlene, but I reckon it's a bonzer title!"

"If you say so Bruce. Pity you can't write an article about timber-cutting. You could call it Real Men Don't Use Chainsaws!"

"Fun-nee! But maybe that's not such a bad idea Charlene. I could write one about timber - or trees rather - you know, what kinds to select for different soils and stuff like that. But none of the fancy varieties - it's best to stay with the natives."

"What, when you go camping in the bush Bruce?"

"Native trees Charlene, native trees!"

"Sorry about that Bruce. Anyhow that idea's no good for a computer newsletter."

"Oh, I don't know Charlene. There was an article about gooeys a couple of issues back."

"About goannas - in the newsletter?"

"Yeah, even about catching one by chasing it towards a bloke standing in the middle of a clearing. Trouble was the warning wasn't included."

"Warning Bruce?"

"You know - about tucking trousers into socks."

"Yeah, of course Bruce, gooeys and snakes don't always see eye to eye."

"Charlene!"

"Sorry Bruce. Anyhow think about your article tomorrow, it's getting late."

"Good idea Charlene. I want to be rested ready to start that Article For The Newsletter in the morning."

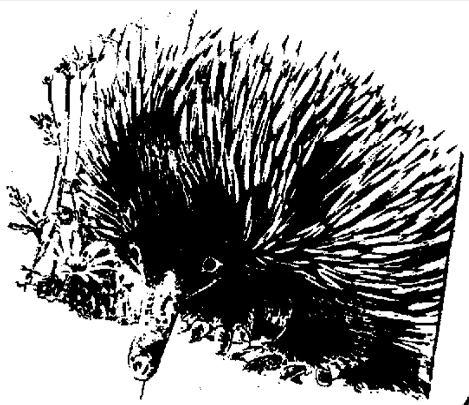
"I've got news for you Bruce!"

"Hmmmmm -- "

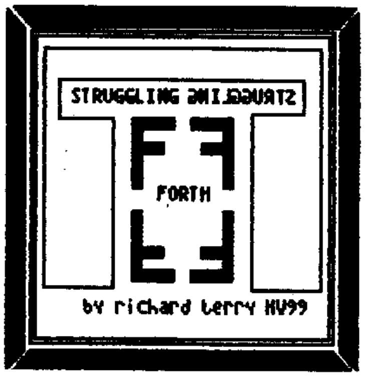
"Bruce, I've been thinking - you certainly don't have to eat quiche to be a real man."

"Keesh? What's keesh?"

If you don't write for the
CHRISTMAS BUMPER
ISSUE
I'll come up and eat all
your ANTS !



```
SCR #114
 ----------------------------------------------------------------
 -                                                              -
 -                                                              -
 -                                                              -
 -   STRUGGLING FORTH                                           -
 -                                                              -
 -   HV99ERS  OCTOBER ARTICLE 89                                -
 -                                                              -
 -   GENERAL PROGRAM CODE                                       -
 -                                                              -
 -   using flow charts to create                                -
 -   bar menu as example                                        -
 -                                                              -
 -                                                              -
 -                                                              -
 ----------------------------------------------------------------
```



WRITING GENERAL PROGRAM ROUTINES.

Next in our series on program writing we come to the battery of non-specific or relatively non-specific support routines. These are the fabric of your program. They range from the simple IF's and DO....LOOPS which are already supplied in the Forth kernel, to new routines you write yourself to do such things as detect certain keypress's, input data from the keyboard, validate the input, routines such as checking if only digits are inputted, string routines and many more. In fact far to many to be covered by the scope of these articles, so what I will do at a later date is to print a copy of the more useful general routines I use, along with an explanation of how and when to use them.

When designing a program I tend to think or forth WORDS in a stratified level - from the more primitive lower level ones such as IF.., to the more complex yet still general routines such as our example today, words to control bar-menu's which give us visually pleasing programs.

If you have a simple routine in mind such as L>U (Convert lower case to upper case ascii-see screen 116), then nothing more is needed than a quick jot down on a piece of paper to work out the components needed and the stack order.

However, once we start to think in more complex terms, things can quickly get more difficult, and rapidly get out of hand. A complex routine such as BAR-MENU as listed on screen 121, is, in many ways identical to an entire forth program itself. In fact looked at simplistically most forth words, even the most simplest are like a mini-program to themselves.

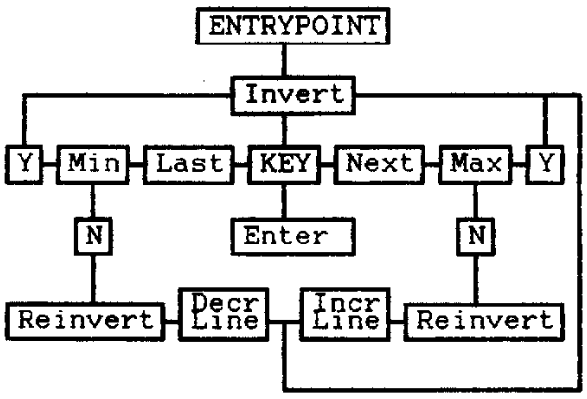
Its easy to get lost whilst producing these more complex routines, and let our logic fail us. Those readers who have perused my earlier examples of barmenu codings of many moons ago will notice a vast difference compared to today's version. In fact its worth digging out your old magazines (ref:) and comparing the two. They both work, the first adequately, the second brilliantly (tomorrow who knows what!!!). From there they part company. The first was written in an ad-hoc manner and the second was written using a flow diagram.

FLOW DIAGRAMS AND FORTH CODE.

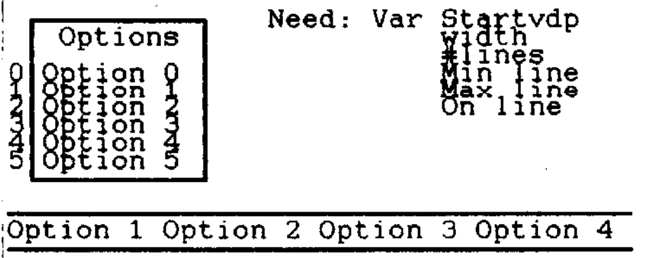
I had always been unhappy with the flickering of the ist codings. I spent a little time recently pottering over the codings. found them difficult to understand, was tempted into patching them, failed. became frustrated. and all in all wasted several hours of time. Off goes the TI at the power point and off I stormed!

Driving home gave me time for reflection. Why not use the good old flow diagrams?. Well. for one. no-one had ever taught me what a flow diagram was let alone how to use them properly, or rather I hadn't been prepared to listen with the impatience of youth.

Anyway, I sat down at the kitchen table and drew the following diagram:



Blow me down if it didn't seem to be right first time! Next I sketched a typical vertical and horizontal menu and annotated the possible variables needed to write the routines.



So ecstatic was I, I preceeded to design the code as the pundits say we should ie from the top down. Following the logic stream produced the following:

```
: BM   INVERT-BAR
       GET-KEY (allow E,X,Ent)
       CASE 69 OF NEXT 0 ENDOF
            83 OF LAST 0 ENDOF
            13 OF      1 ENDOF
       ENDCASE ;
```

Note at this point we have not got a "working" definition, because none of the lower routines ie GET-KEY, NEXT,LAST etc exist yet.

Skipping the GET-KEY detect because it was easy I moved on:

```
: NEXT ?AT-END-MENU 0=
       (1=not at end)
       IF RE-INVERTBAR
          INCREMENT-LINE#
          INVERT-NEXT-BAR
       THEN ;
```

And so on.

ANOTHER ASIDE ON LEDGIBLE CODE

I know I keep harping on this point but.....

It is important to note that alot of the coding is actually written in easy to read english statements not gibberish. It is testimony to the brilliance of Charles Moore, the originator of Forth, that this ability to write code and test it before its support exists, and the ability to do it so logically in english, is possible. He showed amazing genius and foresight. It is absolutely essential to keep your code readable for both yourself and your readers. I often compromise and change the names several times during writing to find a name that is compact, yet english enough to be descriptive of what your forth word actually does. From the byte scrooges I hear a deafening cry: " but INCREMENT-LINE# is 15 bytes long just for the name, let alone the code - what a waste of space". Back to your forth manual boys, ever heard of a little something called WIDTH!!!!! (see another article in this magazine called What use is.....).

Anyway, a little experimentation with VXOR and a little logic showed how to invert the characters on each line. Try sitting at the terminal and type:

```
0 0 AT ." TESTING" 128 0 VXOR
```

What happens???

Now examine the final coding to see the end product. Its been neatened up, well commented and generalised to allow its use with horizontal and vertical menus. for the lazy typists amongst us I include a single screen version, with a sole difference of including a possible action upon the user hitting the F9 key as well as up/down/enter.

Just another brief note on comments. They are essential!!!! Their importance cannot be overstated. I can speak with some authority both personal and medical on the biological effects of Ageing on memory, something our younger members will scoff at, an attitude with which we can identify and smile at knowingly. Unfortunately its a fact that the older one gets, the quicker one forgets. Our normal daily neuronal dropout is often accelerated with help from the wonderful beveridges available from our local vineyards. It is somewhat depressing to return to an application you wrote several months before and not to be able to remember why you did what on the surface of it seems awefully bizarre!

You will note that preceding the routines screens there is a screen with comments describing what it is intended to do. Each of the other screens has a similar heading:

\ Bar menu —Scrn Contents Date

This comment line, which can be accessed by you using INDEX helps rapid scanning of a disks contents and helps keep an application logically together, so it can also be archived on another subroutine disk as well.

With the actual coding, as far as possible I stick to the following routine:

```
0-->14   15->34       35-64
NAME   FORTH WORDS \ COMM ENTS
```

I've experimented and for me this gives easy to understand and aesthetic looking code. You can choose your own format but keep it ledgible. Obviously I don't stick to this as a hard and fast rule if other factors override.

Finally on screen 123 I've included a non-commented example to show how the codings work.

Next article I'll continue with program general codings and include some support words you may find useful in your future applications, and which obviously will be included in our DESIGN EDITOR.

ADDRESS FOR CORRESPONDENCE

RICHARD TERRY
141 DUDLEY RD
WHITEBRIDGE
NSW 22990

(049) 436861

```
Program:BARMENU                                          Date:5Oct89

Screen # 115

\ Bar menu's                  RHT20Jne89

The following screens contain all the codings to produce bar
type menu's. The code is applicable to both horizontal and
vertical types of menu's

It is a totally general definition.

Variables exist to to change the speed of movement of the bar up
and down the menu, its start vdp position, width, number of line
s etc
```

```
Screen # 116

\ My core routines          21Sept88

\ Convert lower to upper case
: L>U          DUP 96 > IF 32 - THEN ;
\ delay loop
: DELAY        0 DO NOP LOOP ;
\ stack Eg n3,n2,n1,3(of them) n4 flag true if n4 = n1,n2,n3
: VALIDATE  0 SWAP ROT 0 DO ROT OVER = ROT MAX SWAP LOOP SWAP
\ Eg 10 13 14 (funct keys) 3 (of them) 10 (funct key) VALIDATE
\ place copy of nth number on top of stack
: PICK 2 * SP@ + @ ; : P PICK ;               \  expects n
\ duplicate top n numbers of the stack ,exp count to duplicate
: NDUP     DUP 0 DO DUP >R PICK R> LOOP DROP ;
\ A B C = 65 66 67 3 ?KEYS-ALLOW.exec stopped till valid input
: ?KEY-ALLOW  BEGIN DUP 1+ NDUP ?KEY L>U VALIDATE IF 1
              ELSE DROP 0 THEN UNTIL >R  0 DO DROP LOOP R> ;
```

```
Screen # 117

\ Bar menu - variables        RHT20Jne89

1500 VAR SPEED                \ rate of bar movement
0 VAR ST-VDP                  \ vdp position of line 0
0 VAR LINE-WIDTH              \ width of each menu option
0 VAR MIN-LINE#               \ mininum line number  0
0 VAR MAX-LINE#               \ maximum line number  0 +n
0 VAR ON-LINE#                \ line currently on
0 VAR VDP-INCR                \ amount to inc each line by
0 VAR VERT/HORIZ              \ 1=vertical menu 0=horizont

\ increment line value by 1
: INC-LINE                    \ expects nil
          1 ON-LINE# +! ;     \ leaves nil
\ Decrement line value by 1
: DEC-LINE                    \ expects nil
          -1 ON-LINE# +! ;    \ leaves nil
```

```
Screen # 118

\ Bar menu -display bar       RHT20Jne89

\ pre-routine for displaying inverted or normal line
: (BAR)                             \ expects 128 OR -128
          LINE-WIDTH @              \ parameters give number of
          0                         \ bytes across the menu line
          DO
            DUP                     \ dup byte to VXOR
            ST-VDP @                \ ist byte of menu
            ON-LINE# @              \ current line are on
            VDP-INCR @ * +          \ inc this to match line
            I +                     \ inc vdp  to write to
            VXOR                    \ write 128/-128 to vdppos
          LOOP
            DROP                    \ drop unused copy of 128/-1
          ;                         \ leaves nothing on stack
```

Screen # 119

```
\ Bar menu routines          RHT20Jne89

\ Invert screen position of nominated line
: BAR-ON      128 (BAR) ;             \ expects nil,leaves nil
\ revert screen position of nominated line
: BAR-OFF    -128 (BAR) ;             \ expects nil,leaves nil
\ checks to see if at ist line of menu
: ?MENU-START                         \ expects nothing,leave flag
              ON-LINE#  @             \ current line bar is on
              MIN-LINE# @             \ lowest line able to be used
              =  ;                    \ 1 = are at start of menu
\ checks to see if at last line of menu
: ?MENU-END                           \ expects nothing leaves flag
              ON-LINE#  @             \ current line bar is on
              MAX-LINE# @             \ highest line able to be used
              =  ;                    \ 1 = are at end of menu
```

Screen # 120

```
\ Bar menu -control routines  RHT20Jne89

\ advance bar if indicated
: NEXT                            \ expects nil
              ?MENU-END   0=      \ 1=not at end of menu
              IF   BAR-OFF        \ if not turn bar off
                   INC-LINE       \ increment line number
                   BAR-ON         \ and turn bar on again
              THEN ;              \ leaves nothing on stack
\ retreat bar if indicated
: LAST                            \ expects nil
              ?MENU-START 0=      \ 1=not at start of menu
              IF   BAR-OFF        \ turn bar off
                   DEC-LINE       \ decrement line
                   BAR-ON         \ turn bar on again
              THEN ;              \ leaves nothing on stack
```

Screen # 121

```
\ Bar menu routines          RHT20Jne89

\ does the whole thing
: BAR-MENU                            \ expects nil
              BAR-ON                  \ turn line 0 on in rev-video
    BEGIN     VERT/HORIZ @            \ 1=menu is vertical
              SPEED @ DELAY           \ delay movement of bar
               IF   69 88 13          \ 1=vert:allows E,e,X,x,Enter
               ELSE 68 83 13          \ 0=horz:allows S,s,D,d,Enter
               THEN 3 ?KEY-ALLOW      \ allow these,leaves ascii
       CASE 69 OF LAST 0 ENDOF        \ E
            83 OF LAST 0 ENDOF        \ S
            88 OF NEXT 0 ENDOF        \ X
            68 OF NEXT 0 ENDOF        \ D
            13 OF      1 ENDOF        \ enter
       ENDCASE                        \
    UNTIL     ;                       \ leaves nothing on stack
```

Screen # 122

```
\ Bar menu routines          RHT20Jne89

\ stack expects:vert/hor flg,max line,width,vdp incrmt,startvdp
: SET-MENU     ST-VDP       !
               VDP-INCR     !
               LINE-WIDTH   !
               MAX-LINE#    !
               VERT/HORIZ   !
               0 MIN-LINE#  !
               0 ON-LINE#   !
               ;
;S

 NOTE: the speed of movement of bar up and down the menu has
 been pre-set at 1500 NOP'S delay. This must be adjusted within
 th program before running if you want to change it.
```

Screen # 123

```
\ Bar menu routines          RHT20Jne89

\ Example of use of the menu
: LINE        40 0 DO 95 EMIT LOOP ;

: MENU1       PAGE    ." **MENU**"
              12 2
              DO 0  I AT
                 ." Option " I 2 - .
               LOOP
              1  9 8 40 80
              SET-MENU BAR-MENU ;

: MENU2       PAGE 0 20 AT LINE
              0 21 AT 4 0 DO ."  Option " I . LOOP LINE
              0 3 10 10 840 SET-MENU  BAR-MENU ;
;S to load application load screens 116 -> 123
```

Screen # 124

```
\ Ti-forth       Bar menu wrds RHT9Aug89

1000 VAR SPEED  0 VAR ST-VDP 0 VAR LINE-WIDTH 0 VAR MIN-LINE#
0 VAR MAX-LINE# 0 VAR ON-LINE# 0 VAR VDP-INCR 0 VAR VERT/HORIZ
: INC-LINE 1 ON-LINE# +! ; : DEC-LINE -1 ON-LINE# +! ;
: (BAR) LINE-WIDTH @ 0 DO DUP ST-VDP @ ON-LINE# @ VDP-INCR @
  * +  I + VXOR LOOP DROP ; : BAR-ON 128 (BAR) ; : BAR-OFF -128
 (BAR) ; : ?MENU-START ON-LINE# @ MIN-LINE# @ = ;
: ?MENU-END ON-LINE# @ MAX-LINE# @ = ; : DELAY 0 DO NOP LOOP ;
: NEXT ?MENU-END   0= IF BAR-OFF INC-LINE BAR-ON THEN ;
: LAST ?MENU-START 0= IF BAR-ON DEC-LINE BAR-ON THEN ;
: SET-MENU SPEED ! ST-VDP ! VDP-INCR ! LINE-WIDTH ! MAX-LINE# !
           VERT/HORIZ ! 0 MIN-LINE# ! 0 ON-LINE# ! ; : BAR-MENU
BAR-ON BEGIN VERT/HORIZ @ SPEED @ DELAY IF 69 88 13 15 ELSE 68 8
3 13 15 THEN 4 ?KEY-ALLOW CASE 69 OF LAST 0 ENDOF 83 OF LAST 0
ENDOF 88 OF NEXT 0 ENDOF 68 OF NEXT 0 ENDOF 13 OF 1 ENDOF 15 OF
BACK ENDOF ENDCASE UNTIL ; \ Bar menu words in a nutshell!
```

The highest of character, in my estimation, is his, who is as ready to pardon the moral errors of mankind, as if he were every day guilty of some himself; and at the same time as cautious of committing a fault as if he never forgave one.

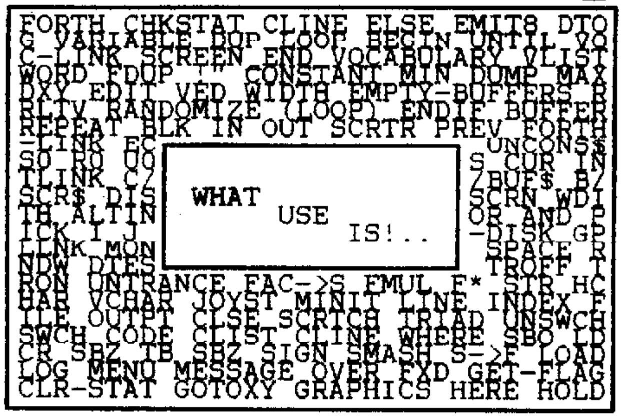
PLINY THE YOUNGER -- Epistles

I believe that every right implies a responsibility; every opportunity, an obligation; every possesion, a duty.

John D. Rockerfeller, Jr -- Speech 1941

Life is not so short but that there is is always time enough for courtesy.

Emerson -- Social Aims.


FORTH CHKSTAT CLINE ELSE EMIT8 DTO
G VARIABLE DUP LOOP BEGIN UNTIL VQ
C-LINK SCREEN END VOCABULARY VLIST
WORD FDUP !" CONSTANT MIN DUMP MAX
DXY EDIT VED WIDTH EMPTY-BUFFERS B
BLIV RANDOMIZE (LOOP) ENDIF BUFFER
REPEAT BLK IN OUT SCRTR PREV FORTH
WHAT USE IS!..
HON UNTRANCE FAC->S FMUL F* STR HC
HAR VCHAR JOYST MINIT LINE INDEX F
ILE OUTPT CLSE SCRTCH TRIAD UNSWCH
SWCH CODE CLIST CLINE WHERE SBO LD
CR SBZ TB SBZ SIGN SMASH S->F LOAD
LOG MENU MESSAGE OVER FXD GET-FLAG
CLR-STAT GOTOXY GRAPHICS HERE HOLD

When I first started programming in Forth, having no computing background I used to get bamboozled by what use some of the standard forth words were, so I thought I'd write a little beginners column indicating what use some of the terms are, trying to include words I've used in the examples in the magazine article of the month

!" (pronounced string-store)

We are all probably familar with the fact that when most strings are stored they have the count preceding them eg:

```
bytes 0 1 2 3

      3 67 65 84

         C  A  T
```

The complexity of string handling is one of the paradoxes of forth. Getting the count byte there automatically is not quite as easy as it would seem and one has to develop quite complicated routines to emulate the extended basic type accept string, add string, segstring etc.

!", which must be terminated by a ", will deposit whatever you enclose in its brackets to whatever address you nominate, when you have a condition **not needing** a preceeding count. For example in many programs you may want to have a disk catalog, which when working flashes up the type of program or file (1-5). You can create the data it will pull out at an address using !":

```
0 VAR TYPES 33 ALLOT

TYPES

!"DIS/FIXDIS/VARINT/FIXINT/VARPROG
RAM"
```

Then during your program when you have opened the catalog file and worked out the type of file 1-5, you can write a little routine to jump into this data array and retrieve the appropriate name:

```
\ pronounced printfiletype

: .FILETYPE \ expects n (1-5)
            1- 7 *   \ offset
            TYPES +  \ add to adr
            7 TYPE   \ print out
            ;        \ leaves nil
```

By the way. You'll notice that in the glossary of Ti-forth this word must be loaded by loading -COPY, which when you do so will also load the disk copy words and copy words your application may not want. To avoid this simply copy out the few lines you need and place them on your own forth screen within your application which needs !".

NOP

What use is an instruction doing nothing!!!! The manual states it is useful for patching as in assembly code. Means little to my poor brain. I've found it useful in two areas. Firstly in writing a routine to delay the flow of the program:

```
: DELAY       \ expects n eg 5000
             0 DO NOP LOOP ;
```

or to put into temporary definitions you are testing which will later be doing something.

+! (pronounced Plus-store)

Expects on the stack a number, and an address (which typically contains another number) ie:

```
n adr --leaves nil
```

It is typically used to change the value at an address which is being used as a counter. In this months Struggling Forth article there is the example of making a bar-menu. Within the coding there are words to increment and decrement the line number the inverted bar is on:

```
0 VAR ON-LINE#
```

This variable, initialised as 0 from within a later definition is our counter. Remember we usually count from zero (0).

```
: INC-LINE  1 ON-LINE# +! ;
: DEC-LINE -1 ON-LINE# +! ;
```

There are innumerable situations where this technique is invaluable. Note you can "add" negative numbers, ie subtract as well.

WIDTH.

I've taken my usual swipe at the byte scrooges who write illegible code in my article. Width is a variable which contains the number of letters which will be saved in the compilation of a definitions name. For example if your definition is:

```
: INCREMENT-LINENUMBER .....  ;
```

and the value in width is its default value ie 31. the whole of this name will be saved - at the penalty of using 20 bytes of dictionary space. If you typed:

```
2 WIDTH !
```

then during compilation it would only save the first two letters, namely IN, which would save alot of space. However there is a catch. later definitions which refer to INCREMENT-LINENUMBER may not work properly if there is another definition starting with IN... on a screen after the original definition. When the dictionary is searched the first IN found will be interpreted as the correct one - when in fact its not the one we want. To avoid this conflict if you adjust the value in width you must ensure the first few letters of all you names are different. In practice, because forth is so compact, it is not necessary to adjust the value in WIDTH, but there are times during compilation of a large application you may need to do so.

From
Kevin Cox
a programme to calculate interest.

```
5 CALL CLEAR
9 DISPLAY AT(11,2):"!----- -
------- --------!"
10 DISPLAY AT(12,2):"!BASIC
COMPOUND INTEREST!"
11 DISPLAY AT(13,2):"!_____
________ ________!"
20 DIM PR(30),MP1(30),MP2(30
),MP3(30),MP4(30),MP5(30),MP
6(30),MP7(30),MP8(30),MP9(30
),MP10(30),MP11(30),MP12(30)
25 IMAGE " $########.##"
30 INPUT "     PRINCIPAL AMO
UNT                (NO COMMAS)
                     ":X
50 INPUT "      NUMBER OF YE
ARS                   ":Y
70 INPUT "     ANNUAL PERCEN
TAGE            (WHOLE NUMBE
RS)                 ":Z
90 PR(1)=X
100 AI=Z*.01/12
110 FOR I=1 TO Y
120 MP1(I)=PR(I)*AI+PR(I)
130 MP2(I)=MP1(I)*AI+MP1(I)
140 MP3(I)=MP2(I)*AI+MP2(I)
150 MP4(I)=MP3(I)*AI+MP3(I)
160 MP5(I)=MP4(I)*AI+MP4(I)
170 MP6(I)=MP5(I)*AI+MP5(I)
180 MP7(I)=MP6(I)*AI+MP6(I)
190 MP8(I)=MP7(I)*AI+MP7(I)
200 MP9(I)=MP8(I)*AI+MP8(I)
210 MP10(I)=MP9(I)*AI+MP9(I)
220 MP11(I)=MP10(I)*AI+MP10(
I)
230 MP12(I)=MP11(I)*AI+MP11(
I)
240 PR(I+1)=MP12(I)
250 NEXT I
260 CALL CLEAR
270 PRINT "    MONTHLY COMPO
UNDING"
275 PRINT
280 PRINT "$";X;"FOR";Y;"YEA
RS AT";Z;"%";
290 PRINT "YEAR   BEGINING
  ENDING "
300 PRINT "====   ========
  ======"
305 FOR I=1 TO Y
310 PRINT I;:: PRINT USING 2
5:PR(I);:: PRINT USING 25:(P
R(I+1));
320 NEXT I
333 PRINT
334 PRINT
335 PRINT
340 DISPLAY AT(23,4):"ANOTHE
R COMPUTATION? (Y/N)" :: CAL
L KEY(0,K,S):: IF S<1 THEN 3
40 ELSE IF K=89 THEN 5 ELSE
350
350 STOP
```

A.N.OTHER to T.I. ADAPTOR (m)

from Al Lawrence

For those with A.N.Other joystick and who want to use them with your TI at little or no expense.

If you have access to a set of old " USELESS TI controllers " cut both cables at about 9" (230 mm) from the console socket end. Open up the old Joysticks and there you will see 5 Diodes(1n4001) on a board in each Joystick. You can unsolder these for reuse with the new adaptor. Diodes prevent feedback signals from the other controller.

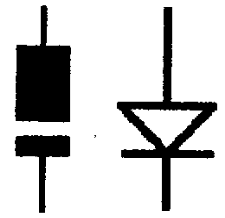
Follow the 3 B's. Beg, Borrow or " BUY ? " 2 -D9 plug's plus shells.

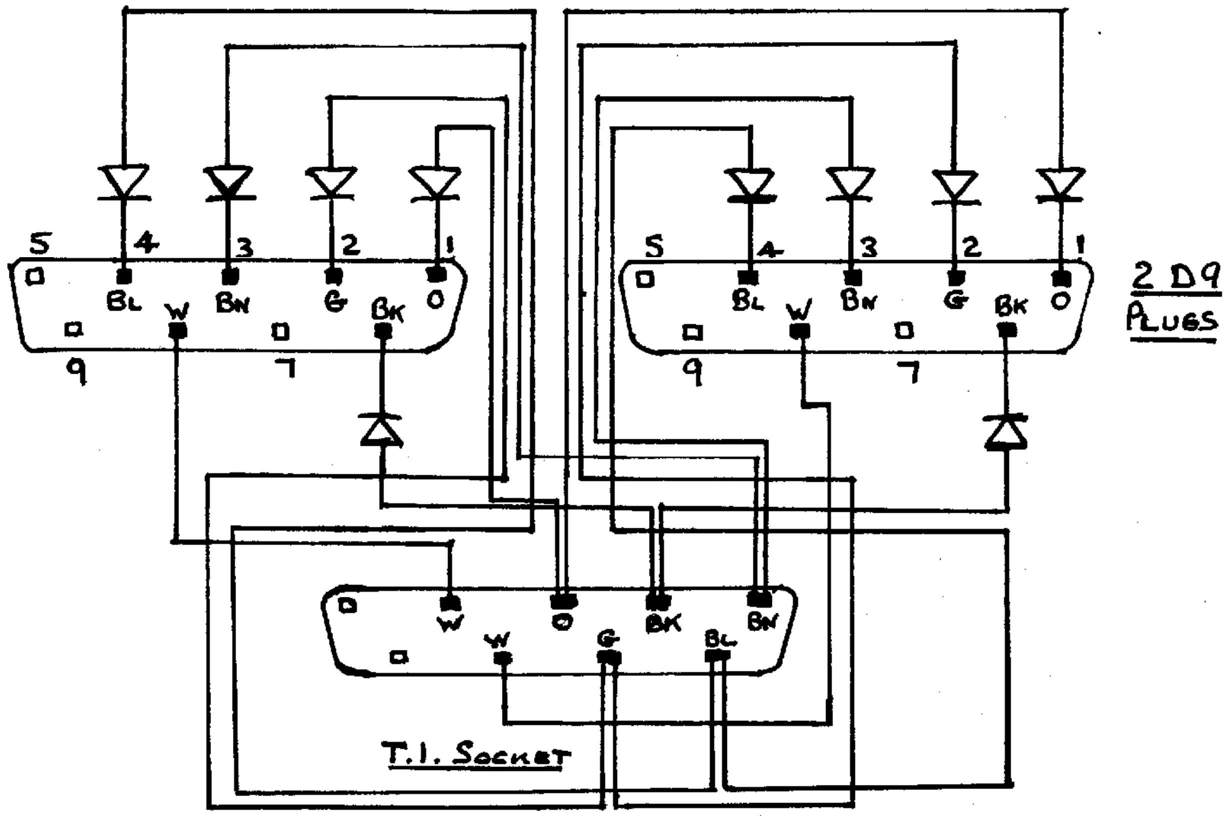
Connections are shown looking at where the wires are to be attached to plugs.

Note Diode direction before removal but in case of Alzheimer's the bar on the case is towards joysticks, eg.
Work neatly and you will have no problems fitting everything into the shell's. Use solderless type if unsure of your soldering ability.

Below are the colours and function as found. Check that all the connections correct and not touching anywhere.
Sleve with plastic tubing if necessary.

W = White ____> Common
O = Orange ____> Up
G = Green ____> Down
Bk = Black ____> Fire
Bn = Brown ____> Left
Bl = Blue ____> Right





THE INFORMATION PAGE

Ⓝ

FORTH.
******
THE SIG is on again at Richard's surgery on 7th November. Starts at 7:00 pm sharp. This means that we all sit around talking generally about Forth for about 30 minutes then get down to tin tacks. Hopefully next months newsletter might show some of the programming efforts of those who attend the SIG.

COMMITTEE MEETING.
****
Second tuesday of the month at Boolaroo ambulance station again. Big disappointment last meeting; some body had removed the table tennis from the hall. All those who got there early for a game missed out, life is tough isn't it. Main discussion was the Christmas do for the kids, raffle results and the next raffle. More disc drives for the club, the $50.00 new slimlines. Also discussed was the visit next year of Bob Carmany in march or february.

EXTENDED BASIC.
****
Third tuesday 20th November at Bob McClure's house again. Congratulations to the Basic group for their programming effort to produce the Raffle drawing programme now used at our General meetings. The group members put this together without the assistance of their leader Gary Jones. Although I am informed that Gary has plans to add bells and whistles. If you can make it these nights are really a good evening.

GENERAL MEETING.
****
Fourth Tuesday at the Jesmond Community centre. This meeting Richard Terry assisted by Joe Wright will be giving a demo of Forth and showing some of the programmes that they have in various stages of development.

MEMBERSHIP
****

Membership fees for the HUNTER VALLEY 99'ers.
Australian Membership........A$25.00
Overseas Membership..........A$40.00

Address membership request and enquiries to Brian Woods, his address is inside the front cover.

CALENDAR 1989
****

Social secretary, Bob McClure has collected the deposits for the Kings Theatre night out. Merv Walker who is involved with the Kings theatre tells us that we can expect a really enjoyable night out.

Christmas party for the kids is mentioned in the president's column.

GREAT MAIL OUT.
****

This is it, the mailing list has been completed, envelopes will be handed out the October general meeting. Hopefully they will be in your homes well before christmas.

[illegible]

How did you find this issue of the newsletter? (not - it was JUST there in the mailbox). We (editors are allowed the privilege also) were wondering whether you found anything of interest in the various articles.

To generate some feedback to see how we (the authors/contibutors are also included this time) are performing, readers are requested to complete the rating form below. Additional comment would also be appreciated.

Then, if you decide to send the form back to us, results of the ratings will be assessed. This should enable us to select the type of articles you want. As well, contibutors will be advised of the assessments so that they can smarten up their acts if necessary. However, as the computer program used for the assessment is designed to always give encouraging results (known as positive feedback), you'll probably end up getting more of the same.

Don't be deterred. Put your name and address on the back of the form and you may be the winner of the lucky draw for the major prize - refund of your postage!

Question 1. If you were writing the article, what changes would you make?
- A. None
- B. Some
- C. Have it put down
- D. Iove sprained my wrist

Question 2. How did you judge the article?
- A. Too long/too short
- B. Too boring
- C. Just about right
- D. Didn't read it

Question 3. Did you undstand the article?
- A. Yep!
- B. Sort of
- C. I drowned
- D. What was the question again?

Question 4. What have you gained from the article?
- A. Nothing
- B. Something
- C. How about lose?
- D. Didn't read it

Question 5. What do you look for in computer articles?
- A. Information
- B. Knowledge
- C. Humour - with a touch of pathos
- D. Some sign that they'll soon become extinct

Question 6. With the newsletter in general - what changes would you like to see?

A. More articles (if you can get them)

B. Well - I do have some input to make, but with my busy schedule I can't really find the time to write - least of all answer dumb questions

C. None - it's fine as it is (I guess)

D. I've sprained my wrist again

Question	1	2	3	4	5	6
Article						
A						
B						
C						
D						
E						
F						
G						
H						
J						
K						
L						
M						
N:-						
O						
P						

NAME:- ______________________ (OPTIONAL) ons

GLOSSARY OF TERMS.
****

CARRY
*****
One or more digits,produced in connection with an arithmetic operation on one digit place of two or more numerals in positional notation,that are forwarded to another digit place for processing there.

CCD
***
Charge-coupled device.A means for very dense serial-access storage of bits as tiny packets of electric charge moving along the surface of a semiconductor chip.

CENTRAL PROCESSOR UNIT(CPU)
****
Part of a computer system which contains the main storage, arithmetic unit,and special register groups.It performs arithmetic operations,controls instruction processing,and provides timing signals and other housekeeping operations.

CHARACTER
****
A letter,digit,or other symbol that is used as part of the organization,control,or representation of data. A character is often in the form of a spatial arrangement of adjacent or connected strokes.

CHARACTER CHECK
****
A check that verifies the observance of rules for the formation of characters.

CHECK BIT
****
A binary check digit,e.g.,a parity bit.

CHECK DIGIT
****
A digit used for purpose of performing a check.

CHECKPOINT
****
A place in a routine where a check or a recording of data for restart purposes,is performed.

CHIP-ENABLE INPUT
****
A control input that when active permits operation of the intergrated circuit for input,internal transfer,manipulation,refreshing,and/or output of data and when inactive causes the intergrated circuit to be in a reduced-power standby mode.

CIRCULATING REGISTER
****
A shift register in which data moved out of one end of the register are re-entered into the other end as in a closed loop.

CLOCK
*****
1.A device that generates periodic signals used for synchronization.
2.A register whose content changes at regular intervals in such a way as to measure time.

COBAL
*****
(Common Business Oriented Language)A business data processing language.

CODE
****
1.A set of unambiguous rules specifying the way in which data may be represented,e.g.,the set of correspondences in the standard code for information interchange.Synonymous with coding scheme.
2.In telecommunications, a system of rules and conventions according to which the signals representing data can be formed,transmitted,received,and processed.

3.In data processing,to represent data or a computer program in a symbolic form that can be accepted by a data processor.

COMMUNICATION LINK
****
The physical means of connecting one location to another for the purpose of transmitting and receiving data.

COMPILE
****
To prepare a machine language program from a computer program written in another programming language by making use of the overall logic structure of the program,or generating more than one machine instruction for each symbolic statement,or both,as well as performing the function of an assembler.

COMPILER
****
A program that compiles.

MR R CARMANY
1504 LARSON STREET
GREENSBORO
NC 27407
USA