VOLUME 2 ISSUE 1 $1.75 JUNE 1986

"Serving 99'ers Since 1984"

THE SMART PROGRAMMER

As most of you are now aware, this issue is the first produced by Bytemaster and represents the merger of *Super 99 Monthly* and *The Smart Programmer* (formerly produced by Millers Graphics). So, it is quite appropriate that we begin with a message from Craig Miller.

It has been awhile since the previous issue and once again I want to thank everyone for their patience and understanding. In the Sept. '84 issue we said that we were going to produce one more 64 page issue to complete everyone's subscription. Well, I'm glad to say that that didn't happen because that would have been the end of *The Smart Programmer*. From your calls and letters I know you didn't want to see it end and we didn't want that to happen either. We really liked doing the newsletter, but it was impossible to do it and come out with new products at the same time. Now, with Richard Mitchell as publisher and Editor, it will continue on, on a regular basis. This not only allows us to continue to write articles for the newsletter, but we can also continue to produce new products for the 4A. So, this merger of newsletters will help keep our orphans going strong!

Some of you have called or written to ask why we are no longer on Compuserve™. That is one of those questions that is best answered with the old "No comment". However, I'm pleased to say that Millers Graphics has signed up on GEnie™. This service is considerably less expensive than Compuserve™ - $5.ØØ/hr for 3ØØ or 12ØØ bps. They have also set up a message area for Gram Kracker™ owners called the Kracker Box in the TI Round Table area on Page 575 where we will be adding new information. The TI Round Table section is new, but it is growing fast thanks to all of the great people that are joining in. Hope to see you there. Oh yes, our Electronic Mail box there is MG. See Richard's article on GEnie™ for sign-up information.

Many of you have asked us about our plans for the future and for the 4A. Let me say that the 4A has been our sole supporter for the last 4+ years. It is what we do 1ØØ% of the time. We do not make any products for Atari, Commodore or any other brand of computer and we aren't planning on it. as long as you are out there supporting us we will be supporting the 4A. We really don't want to leave the 4A, it is our first and only home computer! As a matter of fact we have quite a number of new products in the works. As they near completion we will let Richard know so he can pass the information on to you. Doug Warren and I also have a number of articles planned for *The Smart Programmer* as well as the continuation of memory maps. So, there is still a lot to be done!

We were wondering if you would like to see us release some tutorial video tapes for the 4A. We thought we could do some 1 hour videos showing the use of, say, Gram Kracker™, Explorer, DISkASSEMBLER™, Advanced Diagnostics and any new products, as well as other tutorials. Drop us a note and let us know what you think. Speaking of DISkASSEMBLER™, we are now shipping this new piece of software that was written by Tom Freeman. Tom has done a great job and it really shows - this is a very powerful and smart disassembler.

Well, as usual, I've gone on long enough. Let me close this column out with a sincere THANK YOU for your support, and a great thanks to Richard Mitchell for his support of the 4A, which will allow *The Smart Programmer* and the 4A to continue on for a long time to come!

Craig Miller
Millers Graphics
1475 W. Cypress Ave.
San Dimas, CA 91773

Many of you have inquired as to how the new subscription system will work.

For those of you who were subscribers to *The Smart Programmer*, you are due 4 issues, so the last issue on your previous subscription will be dated September 1986.

Those of you who were subscribers to *Super 99 Monthly* should note that next month the expiration date on your label will be extended 4 months, as there were no issues of *Super 99 Monthly* during our transition period. You will now receive 16 pages instead of 12, at no extra charge.

Subscribers who sent in payments after March 19 at our old rates will have your payment pro-rated to our new rates and any partial amount will be shown as a credit on your label.

Now, for the case of those of you who subscribed to both *The Smart Programmer* and *Super 99 Monthly*. Ordinarily, publication mergers result in subscribers receiving the longest running subscription period of the two publications. Instead, what we have decided to do is grant a discount based on the number of pages you were entitled to and the rate at which you subscribed versus the number of pages you will receive. You'll receive the longest running subscription and your discount will be noted on your next label. The discount can be applied toward any Bytemaster product, including a renewal.

In all cases, the expiration date on your current label is not accurate.

Q&A

Does Multiplan™ allow alphameric fields?

Multiplan™ offers many of the features of a BASIC language. By making use of Multiplan™'s string functions, you can use a portion of a string in another cell and you can even convert a string into a number. The following demonstrates converting a teacher's gradebook, which includes both a letter and number grade as a string in cell R1C1 into a letter only in R1C2 and a number only in R1C3.

```
          1                    2
1 "B090"                MID(RC[-1],1,1)

          3
1 VALUE(MID(RC[-2
  ],2,3))
```

The above is only one way of accomplishing the task. For the specific example, a teacher's gradebook, a more flexible approach would likely be to use a LOOKUP table. We'll examine uses for LOOKUP in a future issue.

Is there a communications network that provides affordable service for 99/4A users?

GEnie™ provides 300 or 1200 bps

service at $5 per hour (night hours, day is $35 per hour), which is the best deal we've found. There is a TI Round Table for 99/4A users that includes a message base, download section (Xmodem and ASCII) and conferencing area. The Sysop is Mark Sumner, whose firm, CSI Design Group, has produced several fine 99/4A programs (99ØØBASIC, Windows and Spy's Demise™, to name a few). There's a sign-up fee of $18 that provides you with 3 free hours and a manual. Most areas of the U.S. can access GEnie™ through local numbers. Call 1-8ØØ-638-8369 for on-line tours and sign-up (it's half-duplex). Key "HHH" to begin, then at the "u#" prompt, key "5JM11999,GENIE". GEnie™ will guide you from there. Once you sign up, keying TI will get you to the TI Roundtable and if you want to read the message base (BBS) without pausing, key BRO ALLS NOR (Browse All Scroll No Reply). There is a conference on GEnie™ each Wednesday and Sunday at 9 PM Eastern time. We'll try to have a representative in conference on Sunday evenings. Our Mail address is Bytemaster.

What significant hardware products have been released recently?

One of the major factors that sets the 99/4A apart from other computers is its availability on power-up (Have you noticed that 99'ers aren't enthralled with loading from disk so we can load from disk through some <cough> "advanced" DOS system?). Two battery-backed products released in recent months lend outstanding new capabilities to the 99/4A on power-up.

Millers Graphics' Gram Kracker™ is a device that plugs into the module port and yields full access to address space previously available only as pre-programmed GROM. Depending on the module environment, RAM can also be modified and/or previously unavailable RAM can be accessed. We'll be covering GK regularly, so it is sufficient to mention now that the product functions reliably.

The Horizon RAMdisk is a PEB card that provides access to up to 18ØK of files (including programs) on power-up, functioning (from the user's perspective) exactly like a disk drive except much faster. The Horizon RAMdisk provides user-selectable CRU addressing from >1ØØØ to >17ØØ, so that the card cannot be rendered incompatible with other cards. The card uses low-power CMOS static RAM chips and Ni-Cad batteries. Users of the Horizon RAMdisk are consistenly reporting being very satisfied with the product. Bare printed circuits with instructions, software and documentation are $5Ø each ($42 in quantities of four or more through user groups) and parts are approximately $72 for a SS/SD card or $1Ø5 for a DS/SD card. Ready-to-run cards are also available. For more information, contact Horizon Computer, Limited, P.O. Box 554, Walbridge, OH 43465, USA.

---

16 ITEMS ON THE MAIN MENU

In the Gram Kracker™ manual it states that you can only have 9 items on the TI Menu at one time. After 9, funny looking sprites start to appear on the screen. Here is a solution for this problem if you are using MSAVE and you want more than 9 items on the menu. This patch takes out the double spacing between items listed on the menu. This allows more items to appear before the Power Up routine starts to place them in the Sprite Attribute Table.

1. With an operating system installed in Gram Ø, select Edit Memory from the Gram Kracker™ menu. Next press FCTN = for Hex, FCTN 1 for Gram Memory and FCTN 5 for Search.

2. Use ØØØØ for the Start address and 1ØØØ for the Finish address. Next press FCTN 9 and type in A3 52 ØØ 3A for the Hex string to search for, press the left arrow once and then press ENTER to start the search.

3. When the Hex string is found (our's was at gØ2E1), press FCTN 5 to leave search and then press FCTN 9 to put the cursor in the memory

window. Turn on Bank 1 to disable write protection and then change the hex string to:

A3 52 ØØ 1A

4. Next, press FCTN 5 to activate search again, and then press FCTN 9 to put the cursor in the Search String Input area. The start and finish addresses are OK where they left off. Type in ØØ Ø2 28 6Ø for the Hex string to search for, press the left arrow once and then press ENTER to start the search.

5. When the Hex string is found, our's was at gØ38Ø, press FCTN 5 to leave search. The cursor should be in the memory window. Turn on Bank 1 to disable write protection and then change the 3rd, 7th and 12th bytes on the first row from:

ØØ Ø2 28 6Ø ØØ D6 28 AA 43 95 D2 29

to:

ØØ Ø2 4Ø 6Ø ØØ D6 4Ø AA 43 95 D2 41

This corrects a problem with the foreign language auto start check routine. It was using >8328 and >8329 for temporary storage. We have moved the temporary storage place to >834Ø and >8341. This keeps it from wiping out part of the stack of start address for the program selections. With 16 items on the menu the program start addresses are stacked at >83ØØ through >833F.

6. After you have made this modification, press CTRL = to leave the Memory Editor. Turn on Write Protection. Check your modification by pressing Reset and then go to the TI Menu. If everything is OK you might want to save this modified Operating System by using the Load Save Console selection on the Gram Kracker™ Menu.

If you make this modification, you will need to install another character set (CHARA1) into the TITLE SCREEN Characters with the NEWCHARS Utility on the Gram Kracker™ disk (see page 27 in the Gram Kracker™ manual and the addendum that came with it). The standard Title Screen characters are 8 pixels high so there isn't any blank space between rows when they are not double spaced and it makes it hard to read.

Now, when you have more than 9 items on the menu the selections will become the :;<=>?@ characters instead of numbers since these are the characters that follow in ASCII order. NOTE: You can actually place 17 items on the menu BUT the first selection (i.e. TI BASIC) will NO LONGER WORK because its start address in scratch RAM gets destroyed!! Be forewarned, placing 17 items on the menu will put you into a precarious situation!

LIST Width Modifications

With Extended Basic loaded into the Gram Kracker™ you can change the LIST "device" width for your output device. This allows you to easily list your programs to printer in 28 columns, 132 columns or any width you choose. This same change will also change the DIS/VAR file width if you LIST to disk.

To make this change load Extended Basic into the Gram Kracker™ and then use the Gram Kracker™'s Edit Memory selection. Next press FCTN = for Hex, FCTN 1 for Gram Memory and FCTN 5 to activate the Search function. The Start address is 9ØØØ and the Finish address is 98ØØ. The Hex string to search for is:

ØØ 12 ØØ ØØ ØØ

When this is found press FCTN 5 to leave Search and FCTN 9 to put the cursor in the Memory Window. Turn on Bank 1 to disable Write Protection and move the cursor to the third ØØ after 12 and change it to the width you would like (in Hex). In our XB this was found at g917Ø and the byte to change was at g9174. Examples:

```
00 12 00 00 00 = default 80 column
00 12 00 00 1C = 28  column listings
00 12 00 00 84 = 132 column listings
00 12 00 00 FE = 254 column listings
```

The area you are changing is part of the default PAB for an Extended Basic LIST to a device. Since most of it is zeroed out it allows the card's DSR (i.e. RS232 or DSK) to set its own default for width. When you place a value here the card will use it instead of the default of 8Ø (>5Ø).

If you want to LIST a 28 column program to disk and then load it into TI-Writer or the E/A Editor you will need to convert the file back into DIS/VAR 8Ø format. To do this simply run it through the following XB program, where TEST is a DIS/VAR 28 file and TESTA will be the DIS/VAR 8Ø file to be loaded into an editor.

```
> 100 OPEN #1:"DSK1.TEST",VARI
  ABLE 28
> 110 OPEN #2:"DSK1.TESTA"
> 120 LINPUT #1:A$
> 130 PRINT #2:A$ :: PRINT A$
> 140 IF EOF(1)THEN CALL CLSAL
  L ELSE 120
```

If the file is large, you can easily convert it from DIS/VAR 28 to DIS/VAR 8Ø with a sector editor such as Advanced Diagnostics. To do this, find the File's Header (File Descriptor Record) by doing a Find File. The "Sector" pointer at the top of AD's screen points to the File's Header Sector. Edit this sector and change the 17th byte, in hex, from 1C to 5Ø and then rewrite the sector. NOTE: This will only work if you are converting files to a longer logical record length, i.e. DIS/VAR 28 or DIS/VAR 4Ø into DIS/VAR 8Ø.

To change the LIST width in console BASIC, copy BASIC from GROMs 1 and 2 to GRAMs 1 and 2, then Search GRAM 2 (g4ØØØ to g5FFF) in a manner similar to that described above for Extended BASIC. We found our bytes to be changed at g4822. The changes would be the same values as for Extended BASIC.

Notes On STRASG

Gram Kracker™ users should proceed with caution when making use of GRAM 2 for anything other than TI BASIC. The STRASG utility routine in Editor/Assembler and Mini Memory does a GPLLNK to >ØØ38 and branches to >4D12, TI BASIC, for engaging the Get String Space routine. So, obviously, if the program you are running makes use of STRASG while you have TI BASIC disabled, the program will be unable to complete the routine and will crash. One of the most commonly used programs that accesses STRASG is the BSCSUP program on the E/A disk. We're looking into the possibility of a fix that would bypass the BASIC access, but for now don't use STRASG while GRAM 2 is loaded with anything other than TI BASIC.

A TI-Writer and Fast-Term Menu

Many readers have asked how to have a terminal emulator with a large text capture buffer available from the TI-Writer module. Paul Charlton has recently released a version of Fast-Term for Super Cart (a Super Cart is an Editor/Assembler module with RAM) users. Fast-Term (Cart) is an update to Fast-Term, which is a Fairware program (suggested payment for all of Fast-Term is $15, available from Paul Charlton, 1110 Pinehurst Ct., Charlottesville, VA 22901. It runs from all assembly support modules, such as E/A, and requires disk and 32K. As with all Fairware, send disks, mailers, postage, etc. for the program. Disks should be at least 720 sectors if you want the source code.). Fast-Term (Cart) is not a full implementation of Fast-Term, but rather is primarily useful for taking advantage of its large text capture buffer. To install Fast-Term (Cart) into a GK TI-Writer module, follow these instructions (Be sure to save anything you have in GRAM 2 or at RAM c6ØØØ to c7FFF before proceeding!):

1) Load Editor/Assembler and turn Bank 1 on, emulating a "Super Cart".

2) From Option 3 of E/A, Load and Run, run the program. Fast-Term (Cart) will now appear on your console Menu.

3) Go to the GK Editor. Select address c6ØØØ. Select a START address of 6ØØØ, FINISH 7FFF and DESTINATION g4ØØØ.

4) Enable GRAM 1-2. Press <FCTN> <2>, Move. Fast-Term (Cart) will now be in GRAM 2.

5) Leave the GK Editor and load TI-Writer.

6) Return to the GK Editor and reverse your Move process by selecting address g4ØØØ, START 4ØØØ, FINISH 5FFF, DESTINATION c6ØØØ.

7) Be sure you have GRAM 1-2 enabled and Bank 1 on. Press <FCTN> <2>, Move. Your TI-Writer module now includes a console Menu option for Fast-Term (Cart).

8) Save your new TI-Writer and restore the GRAM 1-2 switch to TI BASIC. To use Fast-Term (Cart), be sure your Bank 1 is on.

Next month, we'll have some more ideas on how you can use GK to make use of Fast-Term even easier than it already is (it's a very good program).

Editor Assembler LOW MEMORY EXPANSION after CALL INIT

>2000	>A55A	Constant that indicates CALL INIT has been executed.
>2002	>2128	Start address of NAME LINK routine (i.e. Start Name)
>2004	>2398	Start address of Tagged Object Code Loader from GPL
>2006	>225A	Start address of CIF (Convert Integer to Floating Point)
>2008		Start of Variable Storage area
>2022	UTLTAB (Utility Table Area)	
>2022	>0000	Default START address for program just loaded
>2024	>A000	First Free address in High Memory
>2026	>FFD7	Last Free address in High Memory
>2028	>2676	First Free address in Low Memory
>202A	>3F38	Last Free address in Low Memory
		also pointer to default REFs and DEFs thru >3FFF
>202C	>0000	Saved Checksum
>202E	>0000	Saved Pointer to FLAG byte in PAB (in VDP)
>2030	>0000	Saved GPL return address
>2032	>0000	Saved CRU base of Peripheral (>1100 for Disk Controller)
>2034	>0000	Saved Entry address of DSR
>2036	>0000	Saved Device Name Length
>2038	>0000	Saved Pointer to Device Name in PAB (in VDP)
>203A	>0000	Saved Version Number of DSR
>203C		Start of 80 BYTE RECORD Buffer
>2094		Start of UTILITY Workspace Registers
>209A		Start of DSRLNK Workspace Registers
>20BA		Start of USER Workspace Registers
>20D9		Start of LOADER Workspace Registers
>20FA	>0064	Data 100
>20FC	>2000	Data >2000 (H20 and H2000)
>20FE	>2E	Byte Decimal Point '.'
>20FF	>AA	Byte >AA for Validation

Editor Assembler LOW MEMORY EXPANSION Continued

>2100	UTILITY VECTOR TABLE (ie: BLWP @KSCAN)	
>2100	>2094	GPLLNK Utility workspace pointer
>2102	>21C4	Start address for BLWP @GPLLNK
>2104	>2094	XMLLNK Utility workspace pointer
>2106	>2196	Start address for BLWP @XMLLNK
>2108	>2094	KSCAN Utility workspace pointer
>210A	>21DE	Start address for BLWP @KSCAN
>210C	>2094	VSBW Utility workspace pointer
>210E	>21FA	Start address for BLWP @VSBW
>2110	>2094	VMBW Utility workspace pointer
>2112	>2200	Start address for BLWP @VMBW
>2114	>2094	VSBR Utility workspace pointer
>2116	>220E	Start address for BLWP @VSBR
>2118	>2094	VMBR Utility workspace pointer
>211A	>221A	Start address for BLWP @VMBR
>211C	>2094	VWTR Utility workspace pointer
>211E	>2228	Start address for BLWP @VWTR
>2120	>209A	DSRLNK Utility workspace pointer
>2122	>22B2	Start address for BLWP @DSRLNK
>2124	>20DA	LOADER Utility workspace pointer
>2126	>23BA	Start address for BLWP @LOADER
>2128	Start of Name Link routine. (Finds Start name in REF/DEF Table)	
>218A	Routine to Return to Assembly Language from GPLLNK	
>2196	Start of XMLLNK Routine. (Link to system Utilities)	
>21C4	Start of GPLLNK Routine. (Link to GPL Routines)	
>21DE	Start of KSCAN Routine. (Keyboard Scan)	
>21F4	Start of VSBW Routine. (VDP single byte write)	
>2200	Start of VMBW Routine. (VDP multiple byte write)	
>220E	Start of VSBR Routine. (VDP single byte read)	
>221A	Start of VMBR Routine. (VDP multiple byte read)	
>2228	Start of VWTR Routine. (Write to VDP register)	
>225A	Start of CIF Routine. (Convert Integer to Floating Point)	
>22B2	Start of DSRLNK Routine. (Link to DSR routines)	
>2398	Start of Tagged Object Code Loader when coming from GPL	
>23BA	Start of LOADER Routine. (Loads Tagged Object Code - DIS/FIX 80)	
>2676	First Free address in Low Memory (Pointed to by >2028)	
>3F38	Last Free address in Low Memory (Pointed to by >202A)	
	Start of Default E/A REF Table.	
>3F38	UTLTAB >2022	Pointer to Utility Table in Low Mem
>3F40	PAD >8300	Start address of Scratch Pad Ram
>3F48	GPLWS >83E0	GPL Workspace pointer
>3F50	SOUND >8400	Location of the Sound Chip
>3F58	VDPRD >8800	Address for VDP Read Byte port
>3F60	VDPSTA >8802	Address for VDP Read Status port
>3F68	VDPWD >8C00	Address for VDP Write Byte port
>3F70	VDPWA >8C02	Address for VDP Write (set) Address port
>3F78	SPCHRD >9000	Address for Speech Read port
>3F80	SPCHWT >9400	Address for Speech Write port
>3F88	GRMRD >9800	Address for Grom/Gram Read Byte port
>3F90	GRMRA >9802	Address for Grom/Gram Read Address port
>3F98	GRMWD >9C00	Address for Grom/Gram Write Byte port
>3FA0	GRMWA >9C02	Address for Grom/Gram Write (set) Address port
>3FA8	SCAN >000E	BL address for key scan routine in Console ROM
>3FB0	XMLLNK >2104	BLWP address for XMLLNK Routine
>3FB8	KSCAN >2108	BLWP address for keyboard scan

Editor Assembler LOW MEMORY EXPANSION Continued

>3FC0	VSBW	>210C	BLWP address for VDP Single Byte Write
>3FC8	VMBW	>2110	BLWP address for VDP Multiple Byte Write
>3FD0	VSBR	>2114	BLWP address for VDP Single Byte Read
>3FD8	VMBR	>2118	BLWP address for VDP Multiple Byte Read
>3FE0	VWTR	>211C	BLWP address for VDP Write To VDP Registers
>3FE8	DSRLNK	>2120	BLWP address for DSRLNK Routine
>3FF0	LOADER	>2124	BLWP address for Tagged Object Code Loader
>3FF8	GPLLNK	>2100	BLWP address for GPLLNK Routine

Editor Assembler HIGH MEMORY EXPANSION

>A000	START OF HIGH MEM-EXPANSION
>A000	First Free address in High Mem (Pointed to by >2024)
>FFD7	Last Free address in High Mem (Pointed to by >2026)
>FFD8	XOP 1 Workspace
>FFF8	XOP 1 First Instruction (Usually a JMP, B, BL or BLWP)
>FFFC	Workspace Pointer for LOAD Interrupt (non-maskable interrupt,
>FFFE	Start Address (PC) for LOAD Interrupt not DSK1.LOAD)
>FFFF	END OF HIGH MEMORY EXPANSION

NOTE: When RORG (Relocatable Origin) DIS/FIX 80 files are loaded by the E/A Tagged Object Code Loader they are loaded at >A000 by default. The loader looks at the value in >2024, which contains >A000 at first, and loads the next word(s) according to this value.

If the file contains any AORG directives the loader will load the code where the programmer specified (i.e. AORG >C000) instead of where the pointer says to. Also, with AORG code this pointer at >2024 is not updated as the file is loaded.

Since the E/A Tagged Object Coode Loader resides in Low Memory Expansion, >23BA - >2675, you can not use this loader to load this area of memory since the loader will get wiped out. Also, since this loader uses other routines in Low Mem and other addresses for storage, it should not be used to load >2000 - >23B9 either. However, you can use the Mini Mem's loader to load this area of memory since its loader resides in the cartridge ROM and leaves ALL of Low and High Mem free for your programs.

If the file is a PROGRAM IMAGE type file the Program Loader built into the E/A module (5 RUN PROGRAM FILE) loads the file back into the memory it was SAVED from with the SAVE Utility. This loader is written in GPL code and resides entirely in the E/A module so it also leaves ALL of Low and High Mem free for your programs.

5TH 1- =FORTH

by Mariusz Stanczak

First, let's cover our mail. Some of it has gotten pretty old, though I hope most of the trivial problems we all have in the beginning have long been solved.

There has been an aura of misconception around FORTH. If your letters are any indication, it seems that its presence could be attributed in large part to a misunderstanding of what FORTH is and/or what it is for. It is not a conventional language by any means. Its strength and beauty comes, indeed, from a very unconventional idea of leaving all the doors to the system opened. Traditionally, one would try to match a language to the problem to be solved. There are some serious drawbacks (availability of languages on one's machine, one's ability to make the proper choice, etc.) to this approach. FORTH's greatest power comes from the fact that it is, first of all, a programming environment for creating such a language that best describes your solution to the problem. Very important words that describe this environment are: interpretative, standarized and extensible. And, atop of all that, is offering a choice of either a total machine independency, which makes the 99/4A "look like" any other computer running FIG-FORTH, or total machine control. The FORTH system, as it comes to your computer, is not designed to be a language for writing of complex applications. Instead, it supplies you with tools to build what Leo Brodie, in his book THINKING FORTH (Spectrum Books), calls lexicons. Brodie states:

> *What you should do is write your own languages in FORTH (lexicons) to model your understanding of the problem, in which you can elegantly describe its solution.*

Brodie steps through the entire premise of using lexicons to extend FORTH into application-oriented languages, so be sure to read the book!

Once we understand the benefits of the Brodie methodology, it becomes quite obvious why FORTH's absolute speed is a secondary consideration. FORTH is primarily a development environment, and it offers the tools for achieving ease and clarity needed at that stage. When our application is written and thoroughly tested we can shift our energy towards the goals of speed and compactness of the final product (This is a subject in itself so we will explore some means of achieving those goals some time in the future.). Still the speed of TI-FORTH programs on the 99/4A is the source of disappointment for many of you. It's not even remotely close to what TI wanted (?) it to be, i.e. to be only about two times slower than equivalent assembly program. WYCOVE-FORTH users are much better off. Their systems outrun TI-FORTH in all but disk access operations and are 2-1Ø times faster.

There are three major variations of threaded interpretative languages (see the book Threaded Interpretative Languages by R. G. Loeliger). Both FORTHs for the 99/4A are of the indirect threaded variety, which is right in the middle (between direct and token threaded) as far as the ratio of overhead during execution versus compactness of code is concerned. Being a copy of FIG-FORTH, WYCOVE-FORTH's colon definitions exhibit very little optimization, so there is some room for improvement, but it's not really worth the effort. TI-FORTH, on the other hand, seems to be an unfinished product. Moreover, it is by design fatter than Wycove's by about 5K and, on average, it goes (also by design) through 2 to 3 more links. Not much, short of redesigning the system, can be done to improve TI-FORTH's speed, although, I am told, some people are working on it (good luck!).

> Just a thought: since most of that "extra" 5K in lower memory is composed of assembly routines (marked on the memory map as "99/4 support for FORTH" and "Assembler support") that are currently accessed through one extra level of indirection (LINK register), the communication with those

routines could be (?) changed to a direct threaded mode using JMPs for going in and a normal return through NEXT.

The promised dictionary update for WYCOVE-FORTH is a set of very fast shift instructions. The shift words below are named after their assembler counterparts as their behavior mimics them exactly.

```
SCREEN # 21
 0 ( CREATEd Shift Instructions) BASE @ HEX
 1 CREATE SRA  ( n ct -- n_shifted ) SMUDGE
 2   C036 ,  \ MOV *SP+,R0  get count
 3   C056 ,  \ MOV *SP,R1   get value
 4   0801 ,  \ SRA R1,0 <-ct defaults to R0
 5   C581 ,  \ MOV R1,*SP   >PS
 6   0459 ,  \ B *NEXT      return
 7 CREATE SRL  SMUDGE
 8   C036 , C056 , 0901 , C581 , 0459 ,
 9 CREATE SLA  SMUDGE
10   C036 , C056 , 0A01 , C581 , 0459 ,
11 CREATE SRC  SMUDGE
12   C036 , C056 , 0B01 , C581 , 0449 ,
13 BASE ! ;S
14
15 Routines above are both more compact
16 and faster than the ones proposed in
17 the manual.  The shift count is a value
18 between 0 and 15. If the value equals 0
19 then the effective count eguals 16.
20 -NOTE- The above words will work with
21  systems from version 2.0 up; v1.0 uses
22  different registers. Check your manual.
23 -END OF NOTE-
24
25
```

As you have probably already noticed, I do most of my playing around in WYCOVE-FORTH; so, inevitably, this will sometimes get me in trouble with all of you that use TI-FORTH. Last time, contrary to my assurance that the SSMOVE will work with either system, the code published applied to the former environment. The word R/W-CLOSE is specific to that system and is needed to force FORTH to read the directory of a floppy newly inserted into a drive on the next I/O on the disk subsystem. I hope you caught that.

One more thing before we close for this issue. When you send your work to us, which we do appreciate, please(!!!), try your best to make it readable. Use meaningful names, do not cram your screens, do not optimize the code unless you retain its readability. For a whole collection of other do's and don'ts you can choose from freely, let me refer you again to THINKING FORTH. Basically, help me to understand your code and its possible applications. Most of all, comment whenever necessary and describe not only the net effect a word has on the PS but also all items needed and/or changed by it.

In the spirit of the above, all of the code you will see in this column will be mainly of instructional value. Unless otherwise noted, it will not be optimized and it might take more memory and/or run slower than possible implementations. Believe me, there is code that you would not want to decipher from its optimized form.

Beginning with this issue, *The Smart Programmer* will start arriving at your mailboxes monthly; it will contain more of everything that makes it the best magazine around for the 99/4A. So, this column on FORTH will also expand, I hope, proportionally to the rest of the magazine and to the time I can spend writing it (the time it takes me to write the column has exceeded my worst expectations; but on the brigther side, the results have also taken care of my English class assignments). Your contributions are very much welcomed. If you stumble upon some interesting code, data structure implementation in Forth, article that sparks questions or ideas, write to us and make 5TH 1- =FORTH your forum on FORTH for the benefit of all.

Until the next time. MS

P.S.: *Dr. Dobb's Journal* devoted its #10, instead of #9, issue to FORTH in 1985. It sports a complete description and listing of a FORTH target metacompiler for FIG-FORTH which, unfortunately, will not run (as far as I could tell on first examination), on the 99/4A under WYCOVE-FORTH without extensive modifications to the source code (which I will not get at its asking price), but it might run under

TI-FORTH. If anybody wants to give it a shot, let me know how successful your efforts are. The published compiler, in many design details, is quite similar to what I have been working on for some time now, although our approaches to getting over the quirks of the FIG model are diametrically different.

Lastly, a disclaimer for all the malcontents spreading rumors around; Wycove Systems Ltd. hasn't pulled out of the market and it is still distributing and supporting its product, WYCOVE-FORTH, for the 99/4A, which, by the way, is Wycove's side, side business.

Screen 100

```
 0 Since the next few screens have gotten pretty crowded
 1 and I did not have time to edit them, here follow a few
 2 words on their content.  The six screens cover a high
 3 level FORTH decompiler that in its original version was
 4 published in "FORTH Dimensions".  Adapted for the /4A,
 5 it still has a minor bug I agreed to live with.  The top
 6 word, RDCP, expects to be followed by the definition of
 7 interest.  Pressing <ENTER> key will drop the process to
 8 next lower level.  Pressing <q> key will exit the
 9 program and any other key will continue run on the same
10 level.  A non-zero value in KEYSW will permit a
11 continuous operation as if the <ENTER> key was depressed
12 throughout the run.
13
14
15
```

Screen 101

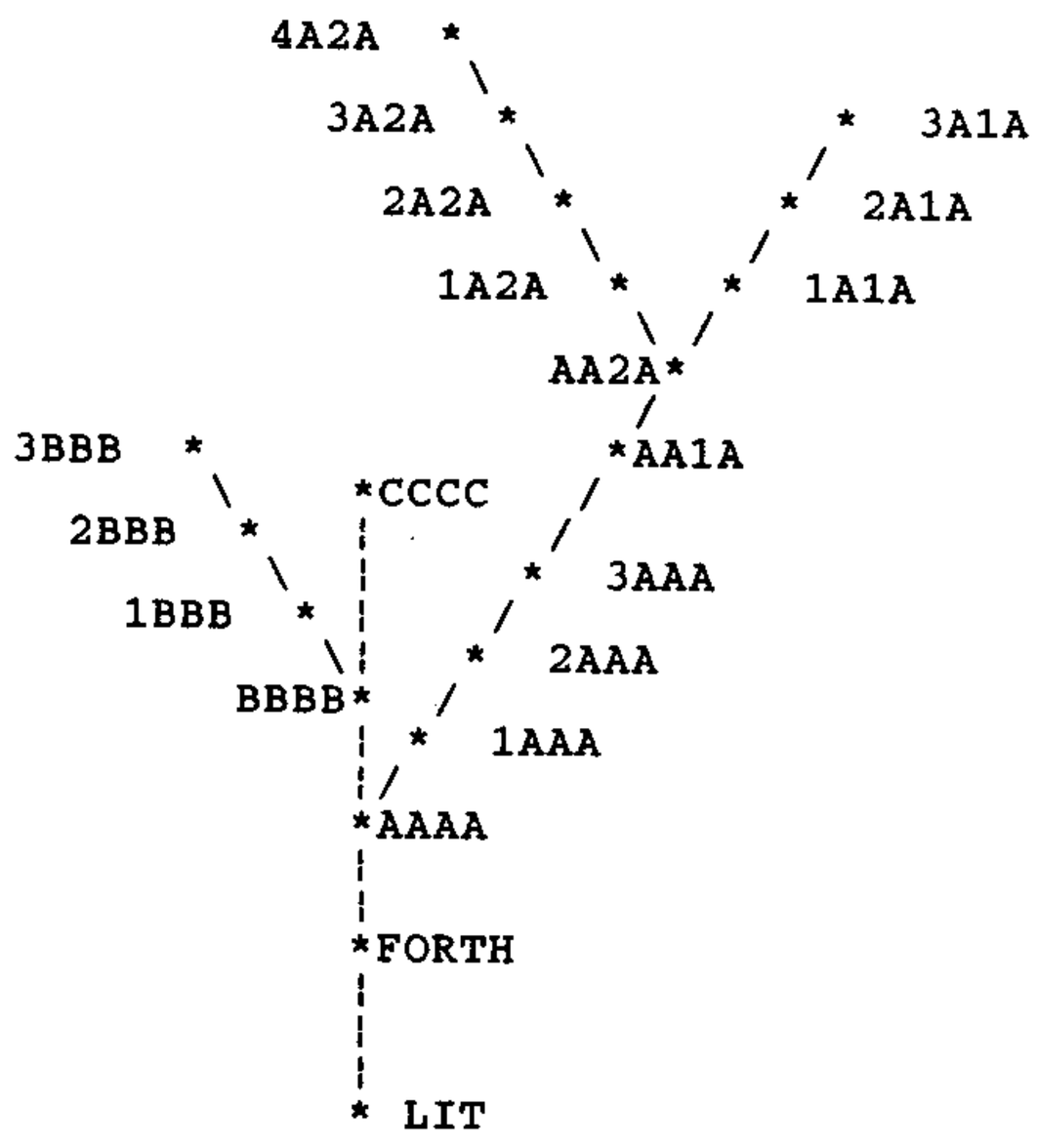
```
 0 ( Recursive Decompiler - 1 ) BASE->R HEX
 1
 2  0 VARIABLE GIN
 3  0 VARIABLE BASESAV
 4  0 VARIABLE KEYSW
 5  0 VARIABLE RDFEN
 6  ' BL   2- @ CONSTANT CONST.ADR
 7  ' BASE 2- @ CONSTANT USERV.ADR
 8  ' GIN  2- @ CONSTANT VAR.ADR
 9  : U. 0 D. ;
10  : GOV OVER @ 2+ ;
11  : 2+D 2+ DUP ;
12  : .G@ 0 4 D.R GIN @ ;
13  : DIN CR DUP .G@ SPACES ;
14  : GIN+ CR OVER .G@ 2+D GIN ! SPACES ;
15                                                    -->
```

```
Screen 102

 0 ( Recursive Decompiler - 2 )
 1  : GCHKTYP
 2       CASE ' LIT      OF 1 ENDOF
 3            ' BRANCH   OF 1 ENDOF
 4            ' ØBRANCH OF 1 ENDOF
 5            ' (OF)     OF 1 ENDOF
 6            ' (LOOP)   OF 1 ENDOF
 7            ' (+LOOP)  OF 1 ENDOF
 8            Ø SWAP
 9       ENDCASE ;
10
11  : 1KEY ( pfa -- pfa c ) KEYSW @
12       IF ?TERMINAL
13         IF 51 ELSE DUP RDFEN @ U<
14           IF BL ELSE ØD THEN THEN
15       ELSE KEY THEN ; -->
```

```
Screen 103

 0 ( Recursive Decompiler - 3 )
 1
 2  : GCHK ( pfa -- pfa [altered] ) 1
 3       CASE GOV ' COMPILE =
 4         OF 2+D @ 2+ NFA ID. ENDOF
 5            GOV GCHKTYP
 6         OF 2+D @ SPACE U. ENDOF
 7            GOV ' (.") =
 8         OF 2+D COUNT TYPE DUP C@ 1- + ENDOF
 9       ENDCASE 2+ -2 GIN +! ;
10
11
12  : CK: ( pfa -- pfa f )
13    DUP CFA @ ' : CFA @ = ;     -->
14
15
```

```
Screen 104

 0 ( Recursive Decompiler - 4 )
 1
 2  : DISTYPE ( nfa -- )
 3    DUP C@ 4Ø AND
 4      IF ." [COMPILE] " ID.
 5      ELSE DUP ID. PFA CFA DUP @
 6        CASE CONST.ADR OF ." const " EXECUTE U. ENDOF
 7             VAR.ADR   OF ." var " EXECUTE @ U. ENDOF
 8             USERV.ADR OF ." user " EXECUTE @ U. ENDOF
 9             DROP
10        ENDCASE
11      THEN ; -->
12
13
14
15
```

```
Screen 105

 0 ( Recursive Decompiler - 5 )
 1  : (GOESINTO) ( pfa -- ) CK:
 2       IF 2- DIN ."    :" 2+D NFA C@ 40 AND
 3         IF ." IMMEDIATE" THEN
 4         BEGIN DUP @ DUP ' ;S CFA =
 5           OVER ' (;CODE) CFA = OR 0=
 6         WHILE
 7           2+ GIN+ DUP NFA DISTYPE
 8           1KEY CASE
 9             51 OF SP! BASESAV @ BASE ! QUIT ENDOF
10              D OF MYSELF ENDOF
11             55 OF DROP DROP R> DROP -2 GIN +! ENDOF
12             DROP ENDCASE GCHK
13         REPEAT CR
14           GIN @ 6 + SPACES 2+ NFA ID.
15         THEN DROP ;  -->

Screen 106

 0 ( Recursive Decompiler - 6 )
 1
 2
 3  : RDCP BASE @ BASESAV ! HEX
 4         -FIND
 5       IF DROP 0 GIN ! CK:
 6         IF (GOESINTO)
 7         ELSE NFA DISTYPE
 8         THEN
 9       ELSE ." Not in Dictionary"
10       THEN
11       CR.BASESAV @ BASE ! ;
12
13
14       HERE RDFEN ! R->BASE ;S
15
```



RESTORE

While RESTORE has several forms and many uses, one aspect of its use is not fully documented in the Extended BASIC or BASIC manuals.

When using a complex PRINT to multiple devices, it would be useful to loop through the devices. However, when attempting to access a particular record in a RELATIVE file and trying to use the same statement to PRINT to a printer, an error message is evoked due to printers not being multi-record devices. The solution is to use RESTORE prior to the PRINT statement. Here is an example:

```
> 100 ON WARNING NEXT
> 110 OPEN #1:"PIO"
> 120 OPEN #2:"DSK2.REPORT",DI
  SPLAY ,FIXED 80,UPDATE,RELAT
  IVE 7
> 130 DISPLAY AT(5,1)ERASE ALL
  :"SELECT DAY (MON=1 - SUN=7)
  ":"1":"SALES":"0":"SALES TAX
  ":"0"
> 140 ACCEPT AT(6,1)VALIDATE("
  1234567")BEEP SIZE(-1):D
> 150 ACCEPT AT(8,1)VALIDATE(D
  IGIT,".")BEEP SIZE(-7):A
> 160 ACCEPT AT(10,1)VALIDATE(
  DIGIT,".")BEEP SIZE(-7):B
> 170 FOR I=1 TO 2
> 180 RESTORE #2,REC D-1
> 190 PRINT #I,USING "DAY #
  SALES $####.##   SALES TAX $
  ####.##   TOTAL $####.##":D,
  A,B,A+B
> 200 NEXT I
> 210 CLOSE #1 :: CLOSE #2 ::
  END
```

If more than one disk file were to be accessed, line 18Ø could be changed to an IF-THEN statement, with "2" being changed to "I" and the FOR-NEXT loop being incremented through the proper number of files.

Many of you have indicated that you needed an example of complex use of PRINT USING, so, hopefully, the above example will help you to understand that statement also.

REPLACE STRING

The TI-Writer command Replace String (RS) is very powerful. Pages 86 and 87 of the manual offer a number of examples of the usage of Replace String.

While the manual makes it clear how to replace a string, it makes no mention of inserting a string.

To insert a string, the first requirement is, obviously, that the same number of spaces be blank at the end of the line as the number of characters you wish to insert. Next, you must enter Fixed Mode, by pressing <CTRL Ø> (Word Wrap Mode may give you different, possibly undesirable, results). Position the cursor at the point in the text immediately before where insertion should begin (for the first character in a file, insert a line before your text and later delete it). Then press <FCTN 9> or <CTRL C> to enter Command Mode. Key <RS> <ENTER>. Then, key the column number twice, the string to be replaced as null (no character, not even a blank) and the string to insert, as follows:

```
Ø Ø //R/
```

"R" is the character to be inserted. The column for insertion is the first column (Ø). Upon pressing <ENTER>, the "Replace String (Yes,No,All,Stop)?" prompt will appear, allowing insertion in all or selected lines.

So, is there anything special we can do with insert? Sure! One idea is to add a printer tab command (character 9) to the beginning of a line(s) or imbedded within a line(s). To use character 9, you'll need to go into Special Character Mode, by pressing <CTRL U>, then <SHIFT I>, followed by another <CTRL U> to return to Fixed Mode. You'll see a small "9" with a small dot above it on your screen. Of course, before you use the printer tab, you'll need to send the commands to the printer for horizontal tab positions, using (for most printers) ESC D nn...n nul. For a guide to using Special Character Mode, which would be required to set the

horizontal tab positions, refer to page 146 of the TI-Writer manual. The key sequence to establish one tab at printer column 41 would be <CTRL U>, <FCTN> <R>, <CTRL U>, <D>, <)>, <CTRL U>, <SHIFT 2>, <CTRL U>.

The reverse process, to delete characters, will also work. The following is a valid command:

```
Ø Ø /R//
```

Again, the parameter that may appear blank on the printed page is actually a null, no character.

Have fun with Replace String!

In Memorium

On April 5, 1986, Thomas P. Weithofer passed away at the age of 21. Among Tom's accomplishments was authoring PILOT 99, a Fairware Computer-Assisted Instruction program written in FORTH.

The Weithofer family suggests that those who wish may make memorial contributions to:

The National Cystic Fibrosis Research Foundation
3379 Peach Tree Rd. NE
Atlanta, GA 3Ø326

Details as to the continued availability of PILOT 99 will be announced in the near future.

Bytemaster Products

Many of you have inquired about products available from Bytemaster (for prices, see page 16).

All 18 issues of *Super 99 Monthly* (Volume I, Issues 1-12; Volume II, Issues 1-6) are currently available as back issues (but are going very fast). There were many topics covered, including creating Multiplan™ SYLK files from Extended BASIC, dumping an Extended BASIC screen for printing through the Formatter of TI-Writer, building a numeric keypad for the 99/4A, working with the CorComp Disk Controller's Toolshed statements, using a 4Ø column screen from Extended BASIC and many more topics.

The issues of *The Smart Programmer* released by Millers Graphics have been sold out for quite some time. Due to the demand for those issues, we are considering re-prints and hope to announce the date of availability soon.

The non-FORTH programs from *Super 99 Monthly* are available as a two-disk set.

Super 99 Handicapper is a thoroughbred horse racing handicapping aid. It converts key numbers into more understandable figures and offers a morning line figure. Version 2.Ø of the program will soon be available.

We have several more software products in the works. Let us know what you'd like to see us produce. Our projects will likely be oriented toward applications software for small businesses and families.

We're working on release of a catalog and will announce its availability soon. In the meantime, thanks for your patience during our transition period. Your response to our producing *The Smart Programmer* has been overwhelming. Now that everything is set up, we'll be back to prompt shipments again.

Diskazine

For those of you looking for another reliable source of information for the 99/4A, try *Genial TRAVelER*, a diskazine (magazine on disk). Barry A. Traver is the Editor. A subscription is $3Ø (U.S. and Canada), $5Ø (overseas) for 6 issues and is available from Genial Computerware, 835 Green Valley Drive, Philadelphia, PA 19128.

BYTEMASTER ORDER FORM

The Smart Programmer

- [ ] $18.00 U.S. AND CANADA FIRST CLASS
- [ ] $15.00 U.S. THIRD CLASS
- [ ] $20.00 FOREIGN SURFACE
- [ ] $32.00 FOREIGN AIRMAIL

Super 99 Monthly

- [ ] $18.00 Complete set of back issues
- [ ] $ 1.00 Back issues - ea. (U.S. Third Class)
- [ ] $ 1.50 Back issues - ea. (Canada and U.S. First Class)
- [ ] $ 2.50 Back issues - ea. (Foreign Air Mail)
- [ ] $12.00 Programs on disk (non-FORTH)
- [ ] $15.00 Super 99 Handicapper (req. XB, 32K, Disk, Printer)

Name_____

Address_____

City_____

State_____

Zip Code_____

Country_____

Payments accepted by check or money order in U.S. Funds, coded for processing through the U.S. Federal Reserve Banking System. No billings or credit sales. Dealer inquiries invited.

The Smart Programmer is published monthly by Bytemaster Computer Services, 171 Mustang Street, Sulphur, LA 70663. All correspondence received will be considered unconditionally assigned for publication and copyright and subject to editing and comments by the Editor of *The Smart Programmer*.

Editor Richard M. Mitchell

Staff Craig Miller
Charles M. Robertson
Mariusz Stanczak
Steven J. Szymkiewicz, MD
Barry A. Traver
D.C. Warren

Bytmaster Computer Services
171 Mustang Street
Sulphur, LA 70663

Bulk Rate
U.S. Postage
PAID
Sulphur, LA 70663
Permit No. 141