SHIFT838 Newsletter

MAY 15, 2015 VOLUME 1, NO 5

This newsletter is dedicated to the ongoing support for the Texas Instruments 99/4A and Myarc Geneve 9640 user community and is published by SHIFT838.

First I want to thank Chris Schneider (founder and editor of SHIFT838) for allowing me to help with this month's issue. His newsletter has been a welcome shot in the arm for all enthusiasts of the /4A and I jumped at the opportunity to offer my services as guest-editor for the month of May. For this month's issue of SHIFT838, I have chosen to focus some energy and attention toward Forth on the TI-99/4A. I believe it is one of the most powerful, yet underutilized languages available to game and application developers for the system, and hopefully hearing about it from the experts and seeing some Forth code will encourage some of you to give Forth a try.

--Owen Brand, "Opry99er"--

Thanks to all that have subscribed!

Interviews

What is Forth?

Forth is a stack-based, LIFO (Last-In-First-Out) Reverse-Polish-Notation programming language that emphasizes an extensible dictionary over syntax.

What does that all mean?

I thought I'd ask two of the gurus of our TI community (Lee Stewart and Mark Wills) a few questions about their journey with Forth, how they came to be authors of unique cartridge-based Forth languages, and why (in their own words) Forth should be considered by software programmers for new TI games and applications going forward. Our conversation follows...

Q: Thanks for agreeing to talk with me here, guys! I'd like to start out by asking each of you to briefly explain your respective Forth releases and your individual goals at the outset of your projects.

Lee: fbForth evolved from a project I started not long after I got my hands on TI Forth in late 1983 or early 1984, which was to convert the TI Forth Manual from the TI Writer version I acquired from the DC Users Group into WordPerfect. I wanted to reformat and edit it to make it easier to use. That project languished until about four years ago when I found all the TI-99/4A activity on the AtariAge forum, which I immediately joined. I converted the manual to OpenOffice Writer format and produced an enhanced edition in PDF about nine months later.

While completing a second edition a year later, I began champing at the bit to convert TI Forth from sector-based I/O of Forth blocks to a file basis. File I/O of blocks makes a lot more sense because it is easier to use disks of different sizes and it is a lot safer than sector I/O, which can destroy a disk if one is not careful. Mark Wills' TurboForth was my principal inspiration for making this conversion. I finished this file-based TI Forth, which I renamed fbForth for "file-based Forth" and released as fbForth 1.0 in November, 2013. It had a few enhancements, among which were an 80-column text mode, an enhanced editor and a joystick routine that can use the CRU like TurboForth or KSCAN like TI Forth.

Almost immediately after the release of fbForth 1.0, I started work on a cartridge version, which culminated in its release at the Chicago Faire in November, 2014. It is much enhanced over the previous version. The 40/80-column editor was rewritten in assembly language (ALC) as were all of the graphics primitives, which dramatically increased their speed. The editor rewrite was heavily influenced by the TurboForth editor. I also included a new font editor. My implementation of the Geneve MDOS floating point library was also included.

Mark: TurboForth was the result of sitting down to design and build a single-board computer. I decided to use a TMS9995 (same CPU as the Geneve) with a very simple memory map, and a serial port. Having a serial port would mean that I wouldn't have to implement the hardware for a keyboard or a display.

I bought the components in a fit of enthusiasm, and started thinking about what kind of operating system I would put on this marvelous little computer. After looking at many different options, I finally stumbled, or should I say, re-stumbled across Forth.

I'd been introduced to Forth in 1989 by Stephen Shaw (a UK TI99 legend). He demonstrated the language to me on a TI (TI Forth). I thought he was out of his mind. Reverse Polish Notation? Who the hell wants to write stuff backwards? Stacks? Pah! I dismissed his advice on the spot (in 1989 I was 18) and never gave Forth a second thought, until I saw TI Forth listings published in TI*MES (a UK 4A magazine) whereupon I'd laugh indignantly at how stupid it was. Oh what it is to be young and very stupid!

Fast forward to 2008/2009 (if I remember correctly). I stumbled into some articles on the internet on how simple it was to implement a Forth system on minimal hardware, and not only that, but the programming language was the operating system, and the operating system was the programming language. No borders. No barriers. Even the majority of Forth itself could be written in Forth. That was enough to drag me in, hook, line, and sinker!

I soon dropped the idea of building a little computer. The chips are still sitting in a drawer somewhere. I instead resolved to write a Forth system for the 4A. That lead me on to a journey of discovery that culminated in TurboForth. TurboForth itself is quite a large project. Thousands of lines of code. Many parts of it were re-written as I discovered more about Forth (more a symptom of my failure to do adequate research than anything else!), it was developed in fits and starts, as most hobby projects are. It's a truly international project; large parts were written in Aberdeen, Scotland. I remember writing the editor in about 5 hours on a sunny day in my office (I had some down-time!). The file I/O was written in the International Hotel in Lang Fang, China. The 32 column words (DCHAR, HCHAR, COLOR, SPRITE and others) were written in Tashkent in Uzbekistan. Other parts were written in Bhukara in Uzbekistan.

My goals for TurboForth were simple: I wanted it to be as fast as I could make it. Large parts of it are written in assembly language, and a lot of it is highly optimised - having been written, and re-written a number of times to save an instruction here and an instruction there. I even reversed the direction of the stacks long after I had (supposedly) "finished" (when is a software project finished?) because it made the code shorter and faster. Faster is good, and shorter meant I could squeeze more code into the 16K ROM!

Q: Good stuff there fellas.

It seems that you both had differing reasons for starting your projects, but somehow ended up in a similar place... How did the two of you come to know one another and what (if any) influence did you have on each other's projects?

Mark: From memory, my first conversations with Lee were to do with a program to find a unique Social Security Number that Lee had written in TI Forth, then he converted it over to TurboForth. IIRC, there were a few issues that came up during the porting of the code. Along the way, Lee discovered that V1.0 of TurboForth didn't implement floored integer division, or, it did, but incorrectly, leading to problems. That led to a gargantuan discussion (all via AtariAge) of how floored integer division worked per the Forth-83 Standard I was following. I was writing code and sending builds of TurboForth to Lee to try out! It took a few attempts, but I got there in the end!

There have been quite a few collaborations along the way. For example, when I developed the assembler for TurboForth (a port of the TI Forth assembler) Lee assisted with proof reading the documentation. Lee also helped with the documentation of the 32-bit library. In addition, Lee also contributed an entire floating point library (complete with extensive documentation) for TurboForth. I helped Lee with reviews of the fbForth manuals and the odd bug report. It's been really nice watching fbForth come to fruition. He works much faster than I do!

Lee: If I ever work faster than you, Mark (which I doubt), it's because I have a lot more discretionary time. I *have* been retired for 6 years.

Q: Lee, I know that maintaining perfect compatibility with the standards of the original TI FORTH was important to you. Was that constricting at all? Surely at some point you had a few decisions to make on whether "to stray or not to stray"?

Lee: Maintaining perfect compatibility with the standards of the original TI Forth was certainly important as a goal; but, from the outset, that was impossible because of the change from direct-sector access for I/O of blocks (screens, in figForth terminology) to file access.

Regarding whether or not "to stray", I tried not to do that unless I had no choice. Not having a choice usually involved the new, file-based I/O for blocks. Appendix E in each of the manuals for fbForth 1.0 (disk version) and fbForth 2.0 (cartridge version, including significant enhancements) details words removed, added and changed. If I strayed anywhere, it was probably in adding words to the resident dictionary to enhance fbForth. In fbForth 2.0, I hoisted most of the optional words into the resident dictionary in ROM space because I could. This dramatically sped up the graphics words, some of which were so slow as to be practically unusable, especially **LINE** in bitmap mode. The only drawback to this is that a larger dictionary increases compile time for the interpreter for words toward the top (earliest entries) of the dictionary—the dictionary is searched as a linked list from the most recently defined word. This, of course, negatively affects the load time of disk blocks.

Q: There have been some unbelievable developments in the TI community of late... Namely cartridges with an insane amount of ROM AND GROM. Any thoughts about expanding your languages to incorporate more words or functions in the cart space?

Lee: The cartridge version of fbForth (v2.0) is much expanded over v1.0. I figured out a way to split out the dictionary headers from the main resident dictionary bank; but, there is very little room in either of those banks for more expansion. More banks will allow for more ALC in ROM; but, I would have to think of a clever way to allow the dictionary bodies and headers to exist in more than one bank each before the added space will matter for adding more words.

One thing to keep in mind when adding words to the dictionary is that increasing the size of the dictionary increases the time it takes the interpreter to load source code from blocks. It also increases time from the terminal; but, that is not nearly so noticeable.

Mark: Yes. TurboForth lives in a 16K EPROM. I'm quite impressed that I managed to squeeze as much in as I did. I did some fairly insane amount of optimising - revisiting code over and over - moving things around. Bob Carmany, who did most of the testing of TurboForth (starting way back in 2009!) was constantly floored by my emails saying "Hey Bob, you know that feature we didn't have room for? We do now!" - but it's reached the absolute limits now. It's packed out.

Q: One last question for the both of you.

In our community (which seems to be growing fairly steadily these days) there are many options for the would-be developer...

In your viewpoint, why should a programmer look at Forth for game and application development over, say, Extended BASIC or assembly?

Mark: Bang for buck and memory efficiency. Forth, both fbForth and TurboForth are many 10s, and in some cases, hundreds of times faster than TI BASIC or Extended Basic. Further, they open up the entire machine to you. Nothing is hidden from you, as it is in TI BASIC or Extended Basic.

In addition, there's no edit/compile/load/run/repeat type sequences like there is when you write in Assembler or C. Just type your code in and try it, just like BASIC. There definitely is a learning curve when learning Forth. It's not an easy language to pick up. It's made even harder if you are used to BASIC programming, because Forth is so different. It's like having a baby! It turns your world upside down for a while and you can't figure out just what the hell it's doing, or wants for most of the time. But eventually you get to know each other, and the rewards are huge, and it's with you for life. Just like a baby!

Lee: What he said!

To quote from Leo Brodie's *Thinking Forth*, "Forth is a language, an operating system, a set of tools, and a philosophy. It is an ideal means for thinking because it corresponds to the way our minds work."

I think the biggest hurdle to learning Forth is its stack-oriented nature. Once that idea sets in, there is no going back.

TurboForth & fbForth Resources

AtariAge TI-99/4A forum:
http://ti99.atariage.com

Main fbForth thread with release stuff always up to date in the first post:
http://atariage.com/forums/topic/210660-fbforthti-forth-with-file-based-block-io-post-1-updated-12052014/

Main TurboForth resource:
http://www.turboforth.net

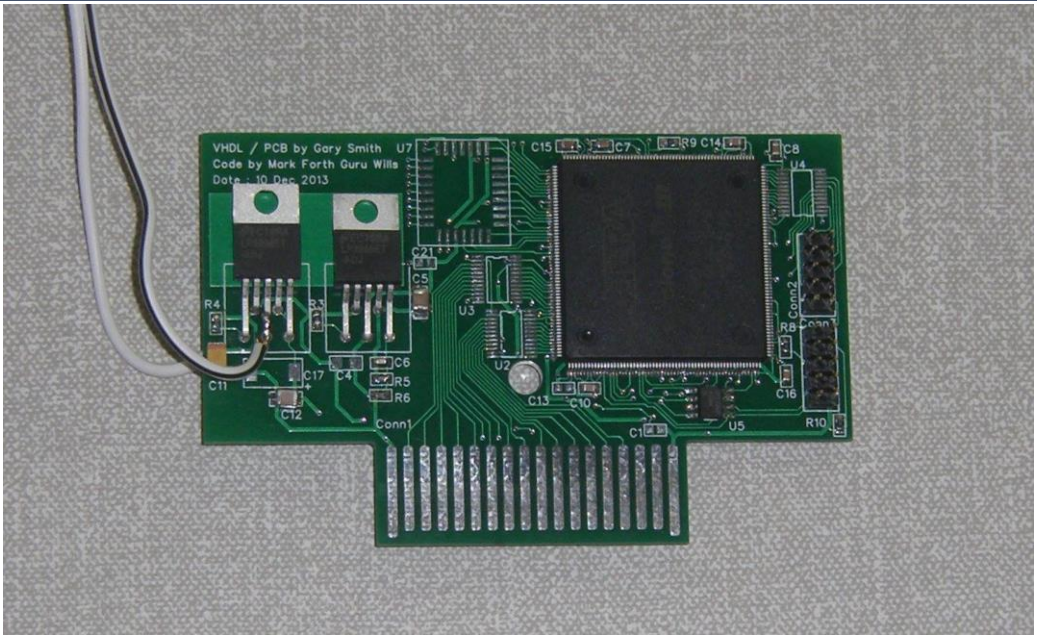
Both have resources listed here:

http://atariage.com/forums/topic/153704-ti-994a-development-resources/

Hardware

In the world of Forth on the TI-99/4a, nothing in the future looks more intriguing than a project by Mark Wills and Gary Smith, sometimes referred to as the "Fourth" cartridge (not FORTH). It is an FPGA-based cartridge running 4 (four!) separate Forth processors, each capable of executing 3 instructions per clock cycle. These processors use Forth as their machine language (which I find fascinating). It will be paired with a unique Forth interface on the TI side, one which will resemble TurboForth, but will not be derived directly from the existing TurboForth architecture. The prototype PCB board is complete and in the hands of the developers.

There are talks of adding an SD card reader/writer allowing for the TI to run a DOS-like operating environment. The possibilities with this project are virtually endless, and I, for one, cannot wait to see this thing plugged into my own /4A system.



Coding

In keeping with the Forth theme, Lee Stewart and I worked on a small conversion project together. I wanted to put together an XB/Forth comparison, so I hastily wrote some XB code, and Lee helped me with the Forth version. (Please note that fbForth had to be slowed down MANY times to maintain the speed (or lack thereof) of the XB program. In short, it is freakin' fast!!! You can see that early on in the program listing with the "DELAY" definition.

```
 XB DEMO (TI Extended BASIC)

100 C=1
110 CALL CLEAR
120 CALL SCREEN(9)
130 CALL CHAR(42,"00E0FFD5FFFFE74200007F55FF7F7722007052727FFF7F32")
140 DISPLAY AT(6,C+1):"*+++,"
150 DISPLAY AT(6,C):" ";
160 IF C=12 THEN GOTO 180
170 C=C+1 :: GOTO 140
180 CALL SOUND(75,1397,1):: CALL SOUND(75,220,1):: CALL SOUND(75,1397,1)
190 DISPLAY AT(12,4):"RIDE THE FORTH TRAIN!!"
200 GOTO 200

FORTH DEMO (fbForth 2.0 (by: Lee Stewart)):

: DELAY 1280 0 DO LOOP ;

: TADA BEEP DELAY HONK DELAY BEEP
   ;

HEX
: GRAPHSET
     GRAPHICS 9 SCREEN
     COLTAB 20 19 VFILL
     00E0 FFD5 FFFF E742 2A CHAR
```

```
    0000 7F55 FF7F 7722 2B CHAR
    0070 5272 7FFF 7F32 2C CHAR
  ;

DECIMAL

: MOTION
    12 0 DO
      I 6 GOTOXY ."  *+++," DELAY
    LOOP
  ;

: BANNER
    4 12 GOTOXY ." RIDE THE FORTH TRAIN!! " CR
  ;

: WAIT
  BEGIN ?TERMINAL UNTIL
  TEXT ;

: FTRAIN GRAPHSET MOTION BANNER TADA WAIT ;
```

Program Explanation:

Both of these listings are "type-in" ready. For the fbForth program, simply open a fresh block in fbForth 2.0 and type in the code and flush to disk. Then load the block.

This is a very simple program… It turns the screen red, displays a little train, sends it to the middle of the screen, one column at a time, then displays a banner with a little sound effect. Very simple.

(Pictured above is a screenshot of BLOCK #1 of 2 of the 'FTRAIN' program in the fbForth 2.0 block editor. The second picture is the program running in fbForth)

These programs run at virtually the same speed, thanks to DELAY, defined by us in fbForth 2.0 as a blank loop which executes 1280 times before terminating.

In Forth, small words are defined (like DELAY above) and then combined in a larger definition to execute a series of words (like the FTRAIN definition)… Once all this is done, you need only type the highest level word in the block in order to run the program. In this case, we defined 6 unique words and one word that combines them all. Let's look at these word definitions one at a time:

DELAY: Creates a delay loop (needed to slow fbForth execution down to match XB)

TADA: Plays the sound effect (uses the user-defined DELAY word in its definition)

GRAPHSET: Sets the graphics for the train (colors and char definitions)

MOTION: Motion is actually a loop which moves the train across the screen (uses DELAY in its definition also)

BANNER: Displays the message beneath the train

WAIT: Similar to the XB command in the XB code (>200 GOTO 200)

FTRAIN: Highest level word… combines all the words together in something of a "sentence."

One of my favorite things about FORTH is the ability to read high definitions like sentences. You can read it left to right and "see" in your head what it should do. In this case:

: FTRAIN GRAPHSET MOTION BANNER TADA WAIT ;

Notice the "colon" at the beginning of every definition and the semicolon following. These are words, just like DELAY or MOTION. The colon is a word that tells the compiler "what follows me is a definition" and the semicolon is a word that tells the compiler "what precedes me (back to the colon) is a definition". Each word must be separated by a space in order for the compiler to understand what your definition is doing. This can be a stumbling block for new Forth programmers.

FORTH truly offers the user the freedom to "talk" to the computer, and both fbForth

and TurboForth are fully featured with hyper-fast libraries and tons of built-in words. I suggest to everyone reading this—Give FORTH a try, either fbForth or TurboForth. You might just fall in love.

Software

Since we have discussed fbForth and TurboForth in some detail, let's take a look at a program written by Mark Wills in TurboForth called "Dark Star."

This game is extremely fun, challenging, and has a real replay factor! I like particularly that the fully-commented TurboForth code is on the turboforth.net website for viewing.

For those of you who are considering "Thinking Forth," this source code could easily be the "Tombstone City" of TurboForth. The game is free for download at turboforth.net, and there is a video on YouTube of the game in motion. Truly one of my favorite time-killers on the TI-99 written in ANY language.

From the Author (Mark Wills):

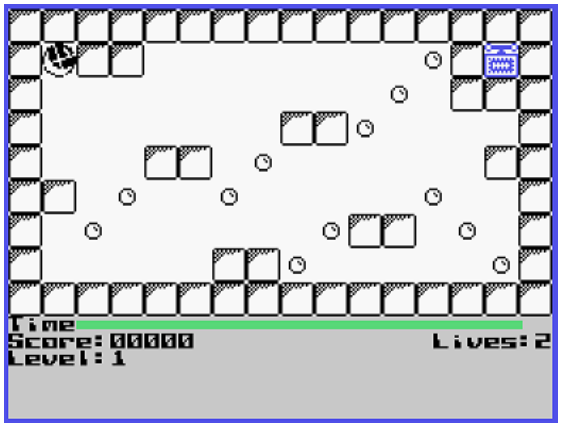
Dark Star is a puzzle game. You need to collect all the 'power crystals' on each of the 25 levels, against the clock. You control a robot that looks a little like a spinning beach ball (!). The robot can move up, down, left and right. When moving, it continues moving until it hits a wall object.
Not all of the power crystals are reachable; you will have to use a second robot as a temporary 'blocker' to give the first robot something to butt up against.
The game gives a reasonable demonstration of some of the capabilities of TurboForth, featuring:

- Character animation
- Speech
- Sound
- Keyboard or Joystick operation
- Screen scrolling

A number of Forth programming techniques are used:
- Vectored execution
- CREATEd words
- VALUEs

This game is based on the version for the Jupiter Ace, a true Forth machine. I didn't have access to the source code of the Ace version, but I copied the look and feel of the graphics, adding a timer element and 3 lives. The level data was taken from the Z88 compiler website, where Dark Star is included as a demo project for the compiler.

ScreenShot of Dark Star



Resources

Contact information
To contact me please feel free to visit my website and click on the 'Contact' tab.

http://shift838.wix.com/shift838

Newsletter Topics
If you would like to participate in the writing of this newsletter or provide any topics for this newsletter please contact me via my web site.

Sites

There are a few of sites that I think should get their own list below. These are for the TI Hall of Fame and TI-99ers Unsung website. Please visit these below sites as both have great information.

http://www.ti99hof.org/index.html

http://www.ti99ers.org/unsung/

Also the below site has a list of all the TI-99ers that have passed.

http://ti99ers.org/modules/Inspire/remember.htm

Below resources are just a handful of sites that support the TI-99/4A and/or Geneve 9640 computers. It is in no way a full list. This section will be included in all future newsletters. If there is a site that you think should be mentioned then please contact me.

Web sites / FTP Sites

http://www.99er.net

http://www.ninerpedia.org/

ftp://ftp.whtech.com

http://shift838.wix.com/shift838

http://www.ti99-geek.nl/

http://www.mainbyte.com

http://www.atariage.com

http://www.harmlesslion.com

http://www.ti99iuc.it

http://www.turboforth.net

http://www.ninerpedia.org/

Yahoo List Groups:

https://groups.yahoo.com/neo/groups/TI99-4A/info

https://groups.yahoo.com/neo/groups/TI994A/info

https://groups.yahoo.com/neo/groups/Geneve9640/info

https://groups.yahoo.com/neo/groups/turboforth/info

Active BBS'

HeatWave BBS

Access: Dial-Up and Telnet
System: Geneve 9640
Software: S&T BBS Software
Location: Arizona
Content: TI and Geneve file libraries, message bases, door games and e-mail.
Telnet to: www.heatwavebbs.com port 9640 Dialup : 602-955-4491 @ 8-N-1

The Hidden Reef

Access: Dial-Up
System: TI-99/4a Modified
Software: S&T BBS Software
Location: New York
Content: TI and Geneve file libraries, message bases, door games and e-mail.
Dialup : 718-448-9402 @ 8-N-1

The Keep

Access: HTTP and Telnet
System: Pentium 4 running Windows 2000
Software: Worldgroup BBS Software (up to 256 user connections)

Location: Tigard, Oregon
Content: TI and Geneve file libraries, message bases, door games, multi-user and multiplayer games and e-mail.
Telnet : www.thekeep.net port 23 Web browser to http://www.thekeep.net

The Keep has TI File libraries, Message bases, e-mail, door games, multi-user and multiplayer games. The keep also has a modem line connected for anyone that would like to contact The Hidden Reef BBS from the internet through The Keep.

Simply telnet to www.thekeep.net on port 23, login to The KEEP and then type /**GO DIALOUT** at the main menu, then D1 to dial out to The Hidden Reef. It's that simple.

Vendors

SHIFT838 – Provides used TI equipment as acquired. Check with me often. A lot of the items need rehoming from other TI Users.

Arcade Shopper – Provides old and new TI equipment, upgrades and new runs of PCBs at www.arcadeshopper.com

Repair Centers

Richard Bell

Repairs available on limited basis, please contact Richard at swim4home@verizon.net for wait-time before sending any repairs

Tim

Myarc-related hardware repairs on a limited, as-available basis. Contact Tim at insane_m@hotmail.com for wait times or to request service.