An Interview with Tom Zimmer: Forth System Developer

Originally published on: Thu, 06 May 2010 23:36:45 +0000

If you've ever used a Forth compiler, chances are you've heard the name Tom Zimmer. Tom's been a staple in the Forth community for a few decades. Tom developed a number of Forth systems for popular 8-bit microcomputers that dominated the home-computer market in the 80's.

Tom is the creator of the freeware Win32Forth system.

What's your educational background?

I received no formal programming training.

I graduated from high school in 1968, long ago and far away. I was interested in electronics at the time, and I had a friend Dick Cappels who bought me the components for a computer, and told me to go down to Wiley Elmar in Sunnyvale CA., and pick up my new computer. The CPU was an RCA CDP-1802, a static processor. It was something of an oddity at the time, most processors were dynamic, and wouldn't run below about 500 kHz. The 1802, being static, would run all the way down to 0 Hz. I had wired the 1802 with 1k of static memory into a simple computer, and programmed it in machine language. It had three clock rates, single step, 10 Hz, and about 500 kHz.

My first exposure to computers. After high school, I worked for Pacific Telephone as a COEM (central office equipment man). That was in the days when job names could specify a gender. After a stint in the military, as a communications controller, my same friend hired me as an electronic tech for a small company that built the first video disk recorders. They were nothing like you might imagine today, being much larger, with many custom mechanical parts. The video recorders contained a micro controller, that was programmed in Forth by Mike O'malley at Berkeley. He did this work on a consulting basis. He would bring us an eprom, we would plug it in, and it would work. We were always amazed when his code worked, because he didn't have any hardware to develop the code on, he claimed to have some sort of simulator that he used for testing. Later Dick had me design a hardware controller for a video disk recorder that was not processor based, because Mike charged us around one or two dollars a byte for

code, and we thought that was expensive. So I designed the controller. My first big hardware design project. I didn't have any formal hardware education either, unless you count a course in electronics in high school. Anyway, the controller worked, and was even shipped in a product, but it wasn't nearly as trouble free as Mike's Forth coded controller version, so we abandoned the idea of using hardware alone to control the recorder.

Anyway my life in electronics and computers was sealed at that point, and I have never looked back.

How did you first encounter Forth?

I already mentioned my first Forth exposure, but the first time I tried to use it, was later when I worked for Calma. They built CAD workstations, and I was hired to work in the hardware diagnostics area. I obtained a barely readable photocopy listing of Forth for the 8080 processor. I typed it into an Intel MDS (Micro controller Development System), assembled it, and got it to run. I had no idea what Forth was supposed to be, but I had heard that it was good for interactive debugging, and I was interested. It had within it the concept of virtual memory, but that was far beyond me at that point, so I just stubbed that all out. At this time, in the later 1970's, I hadn't even heard of the Forth Interest Group (FIG), so I had no contact with that group, or anyone else in the Forth community. I was just exploring this interesting concept of an interactive computer language. Toward the end of my time at Calma, I got a FIG listing of Forth for the VAX, and got that to run. We used Forth to write hardware diagnostics. VAX Forth was quite a challenge, because I could assemble it, but I couldn't (or didn't know how to) link it into the VAX operating system, so I had to dig into some of the system files, to extract system call locations so I could interface with VMS, the VAX operating system.

According to what I've seen on comp.lang.forth, you had developed (or co-developed) Forth software for a variety of microcomputers in the 80's. What events led to your involvmenet in the development of these products?

I was certainly excited about Forth after my experience at Calma. I bought a Ohio Scientific computer, which was 6502 based. I took the 8085 Forth I had evolved at Calma and hand translated it to the 6502 assembly language, so I could run Forth on my Ohio Scientific. I was very young at the time, and I don't know why my wife even put up with

all the time I spent in my work room, but I was so excited about Forth and computers, she just couldn't squash me I guess.

Around 1979, I heard about FIG, and Robert Reiling passed along a FIG listing for the 6502. It looked interesting, and seemed to be accepted by more people than my own Forth was ever likely to be, so Bob and I worked to get it working on the Ohio Scientific. I think Bob typed it in, then turned me loose to get it running on the hardware.

So, I transitioned from my own Forth to FIG around 1979, and moved forward. As various manufacturers were releasing personal computers in those days, I would buy one, and dig into it and develop a Forth for it. It was a way to have fun, and to make a little money at the same time. The next personal computer Forth I worked on was VIC Forth, for the Commodore Vic-20. I can't remember which was next, 64Forth for the Commodore 64, or Color Forth for the Radio Shack Color Computer. Vic-Forth was an 8k cartridge, Color Forth ws a 12k cartridge and 64Forth was a 16k cartridge. Each successive system had more capability.

Why did you implement each as a cartridge?

These computers didn't have disk drives, so the only real alternative as cassette. I had to use cassette to do the development, but I was interested in creating a Forth that would be easy to learn and use, so didn't want the user to have to deal with cassette, except for data storage. Later, in 64Forth, there were also concerns about security, because there were vendors selling cartridge rippers. 64Forth included limited copy protection, that precluded running it out or RAM. It had to reside in ROM, or it would overwrite itself. Cruel, but that was in the days before I switched to making only public domain systems.

Color Forth was a 6809 processor, and was based on a Forth from the only copyrighted FIG listing. It came from a vendor in Southern california, but I can't remember his name. Anyway, I made a contract with him, to split royalties on Color Forth, and it was released. 64Forth was actually the most profitable, it was distributed by HES (Human Engineered Software) in Burlingame Ca. I personally made about 25,000 dollars in royalties from 64Forth, before HES collapsed financially, still owing me almost 9,000 dollars in back royalties. I didn't really care, I was very pleased that 64Forth had sold so well. I believe that they had a lot of inventory that was passed around for several years after that to various Forth vendors, 'til there wasn't any

more interest. Each of these products had a fairly reasonable manual that I wrote, and HES spent a significant amount of money on the packaging for 64Forth and VicForth, so they were very attractive. I'm sure that contributed significantly to their popularity.

How did you go about publicizing / marketing each Forth product? Did you have contacts in the industry at this time?

I didn't have any contacts, but in those days, there was much less software available, so I would just contact a software publisher, and ask them if they wanted to distribute my software with their line. There was a huge hunger for software. Human Engineered Software (HES) was a real developer, they actually invested money into packaging and advertising. They also had contact with cartridge producers that could do "Chip On Board", which eliminated the need for ROM packaging, keeping the production cost low. They produced a very nice package that was used for both VicForth and 64Forth. I am sure that the package alone was responsible for some of the sales.

Had you mastered the assembly languages for the variety of microprocessors at the time? (6502, 6809, etc.)

Assembly language is assembly language, is assembly language. If you have seen one, you have seen them all, with the possible exception of the 1802, which was very different from all the others. I learned assembly language as I went along. Just buy another book, and translate it's instructions mentally to the ones I already knew.

Later at Maxtor, I was employed as a diagnostics programmer for testing their disk drives.

We used 8086s there, we started with Laxen and Perry Forth, we developed Forths for running diagnostics on the high capacity disk drives that Maxtor produced. Forth based software was used in a custom networked environment, to burn-in disk drives for 48 hours, and print burn-in results. I worked there for about three years, and developed several public domain forth, with names like zforth, tforth, hforth, HF, ZF, and F-PC.

Have you written commercial systems other than Forth compilers?

Good question. For a while there it seemed that all I was good at was making Forth systems, and not writing applications. I guess, to me, Forth was an application. Over the years, I have worked on several applications, but they always seem to be based on having to write a Forth system first. I know that many people disagree with this

philosophy, but at the time, I felt I needed to have control of the development system. Now that Visual C++ is so prevalent, we can trust Microsoft to provide the development system (I'm kidding).

That's an interesting statement, though. Do you think that the younger programmers are missing something in their education by not being exposed to Forth?

Absolutely. Most people that are not very familiar with Forth, think it is just a forgotten language of the past. The same thing could be said about our heritage, no matter which country we were born in.

History is important for several reasons, not the least of which is what it teaches us about how to deal with the future. Forth's most important feature has little to do with the fact that it is a stack language, it has instead to do with the way it interacts as a whole with the user. Forths extensibility, structure, modularity and very simple syntax are key attributes that give the programmer freedom to structure solutions for problems in ways that programmers of other languages cannot understand or attempt.

Having access to the full source for your development system gives you the freedom to enhance, or correct problems that the vendor didn't consider. Freedom is very important to me, as it should be to everyone, you just have to remember, that along with freedom, comes responsibility. Forth gives you the freedom, and the power to mold solutions that match the problem. It also gives you the power to shoot yourself in the foot, or in some other even more sensitive area, so if you can't handle the freedom, and the power, then you better stay away from Forth.

Do you presently develop software for a living? If so, what kind of software?

Yes, I work at ThermoQuest, as a programmer. I was hired by Andrew McKewan to assist in porting a very large DOS based Forth application into the WindowsNT environment. We looked at, and even bought the only commercial Forth for WindowsNT available at the time. Unfortunately it wasn't very mature at the time, and we did not have access to the kernel source code, so when we ran into bugs, and philosophy differences, Andrew implemented his own 32 bit Forth kernel one weekend. We got it running using the commercial vendors assembler, which we had a license to use, but we never used any of their source code. I am sure we are guilty of using several of their

ideas though. Anyway, Andrew brought the Forth kernel into work and turned it over to me for "expansion". The kernel started in the public domain, and I never took it out of the public domain during development. I was always careful to separate the code that was proprietary to my employer, from the public domain general purpose Forth system code. An example of this, is that since Win32Forth was a 32 bit Forth system, we were faced with the question of whether to convert all the application source from 16 bits, to 32 bits. Since the application was several megabytes, and we wanted it to be reliable, we chose to leave it 16 bits, and to write a 16 bit to 32 bit translation layer between the Forth and the application. This kept the problems we had to face, down to compatibility issues, and allowed easy porting. We also added a Windows GUI to the application to make it acceptable to the Windows market. The port was completed from start to actual product release in about 9 months, with an average of four programmers working during that time. Still, a large task, but the application proved to be very reliable in the field.

One interesting note, is that the translation layer had within it a lot of debugging code to do range checking on memory operations. When we shipped the product, we left the debugging code active, because we weren't confident enough that we had gotten out all the bugs. Then a year later, when we release the next version of the application, we removed the range checking, and suddenly the application was amazingly faster, and still as reliable, since we had worked out most of the problem during that year. So marketing used that as a new feature, "much faster".

Andrew McKewan, Robert Smith and I were the primary contributors, followed by Y.T. Lin, and Andy Corsack. Later I talked Jim Schneider into writing a full 486 assembler, which he donated, completing the system. Andrew added object oriented programming fairly early, modeled after the MOPS OOP Forth system for the Macintosh. OOP was very valuable in handling the complexity of the Windows API. Over the years several people have donated bug reports, fixes and enhancements to Win32Forth. It was even sold to a commercial vendor for a year, but it proved to complex for their purposes.

Today I program mostly in Visual C++. Originally I hated 'C', but after five years, it is bearable. When programming in 'C' I miss the power of Forth, to create compile-time solutions for difficult problems.

I think I may be burned out to, Forth system development, but who knows what the future will bring. If another interesting computer and OS come along, perhaps I will jump ship and dive into another Forth system development project.

What about BeOS? I saw a post recently in comp.lang.forth asking about Forth systems for BeOS.

I am a Macintosh advocate, and I was interested in the BeOS when it was going to run on the Mac, but now that won't happen, so I really haven't looked at it much lately.

Have you ever entertained the idea of making a Forth compiler for a console gaming system?

No, but I might be interested in writing a Forth for a PDA style device, though there are already Forths for the Palm. I think that market is just starting, and more interesting devices will come along. Perhaps then.

Have you thought about actually sellig Win32Forth?

I have thought of it, but my experience has been that it is very hard to make money selling development systems. Win32Forth is public domain, so others can benefit from it, but also so I can benefit from other peoples contributions. I prefer public domain, over GPL, because it places less restrictions on use. True, anyone can take Win32Forth and turn it into a commercial system, or write a commercial program without giving me or the other contributors credit, but I am also free to use contributors code in commercial applications I write, so while I always try to give credit where credit is due, being able to solve applications problems is what drives me, not receiving credit for some segment of code I wrote several years ago.

Interestingly Win32Forth was purchased a couple of years ago by a commercial vendor for a token fee. They were to document it, and release it as a commercial product. Problem was, Win32Forth is so big, that it didn't really fit within their philosophy of development tools, so it languishes and was ultimately returned to me.

How many copies had been sold of each of your commercial compilers?

I don't have good access to that information, but my recollection is that about 10,000 copies of 64Forth were produced, and I got royalties on about 7000 or those, before HES went out of business. There were

probably 3,000 or 4,000 copies of VicForth sold, and much smaller numbers of ColorForth and OSI Forth.

What prompted you do develop a DOS Forth with an IDE resembling other compilers of the time rather than a traditional Forth IDE?

I am guessing you are talking about TCOM here, since that is the only Forth system I wrote that has a real IDE. TCOM was developed, to make writing an application for DOS easier. One of the problems with all my Forth systems, was their size. They were always big and fat, with lots of tools and libraries of utilities. All that stuff results in large executables.

TCOM was designed from the start, to only include the parts of the language that were needed to support the application being built. The result was very small executables. Of course you still want to debug your programs, so I needed a debugger. Since TCOM produced .COM executables, that didn't contain any debugging information, and I didn't want to burden the target application with any overhead, I chose to produce additional data files that could tell the debugger where the various source lines connected to the target application. This allowed me to create standard assembly style listing files from TCOM executable, and to debug them symbolically. It worked very well.

TCOM eventually included a bunch of target processors, including at least; 8086, 8096, 8080, 68hc11, 6805 and the Samsung Super8, 56000, and 57000 processors. It included a bunch of examples applications for the 8086 target, more than 70 I think. I even wrote a simple basic compiler for the 8086 target of TCOM. TCOM included all the source for all of the compiler, the examples, the debugger, and all the listing generators for each target. TCOM was built on F-PC.

Did you attend industry trade shows in the 80's?

Oh, yes. But only the Forth related ones.

There was a lot of activity in Forth in the 80's. There were several hardware vendors, and a bunch of software vendors. Things are a little quieter now, but I think Forth has just moved underground. It won't ever be a general replacement for Visual C, but it still has wonderful applicability in limited resource environments.

As we see faster and faster computers, with soon to be gigabytes of RAM, and terabytes of harddisk storage, we might think that limited resource environments will pass away, but in the consumer product

area, and pretty much any high volume product area, Forth is a viable alternative. It provides rapid development and debugging, at low cost. I think it will always be the secret weapon of the small developer breaking into the market of the large developer with hundreds of programmers.

How does Forth fit into your future?

Well, I describe myself as a C programmer, who is really a Forth programmer. C has provided employment, and Forth provides tools for hardware and software debugging.

When I work with other C programmers on large projects, I always build in a Forth interpreter into the application, for debugging purposes. The hardware guys love it, because it gives them so much power to figure out what is going on with the hardware. For software debugging, it is great because it gives you an interactive method of figuring out how to talk to the hardware before going off and writing a driver in C.

I think most C programmers look at Forth, and don't really understand why they should be interested in it, and they never bother to spend the time to find out.

I think of Forth like a fine set of hand tools. Microsoft on the other hand, provides the ultimate power tool, Visual C++ with MFC. It's the computer controlled mill, that you need three PHDs to operate. Then you can get your job done really fast, but you hate doing it, because the tool is such a monster, and so unforgiving of mistakes.

MFC provides wonderful information hiding, to solve common problems, but unfortunately you have to know alot about the information that is being hidden, or it won't work properly in many situations. It is like a house built on sand, rather than a house built on rock.

Forth on the other hand, is more like the foundation of rock that you can build your house on. It is simple to understand, and completely bug free. Of course Win32Forth has fallen into the Microsoft trap, in attempting to deal with all the complexity of Windows, it adds huge complexity to what could otherwise be a relatively simple Forth system. The whole OOP thing was added just to help deal with the complexity, and it does help, but at a price. Sometimes I think the price of increased complexity is just too high.

Well, I guess I better get off my soap box, and get back to programming in Visual C++, MFC and my latest project in Java, a whole new adventure.

Thanks for the interview, Tom!