ISSN 0265-5195

# Forthwrite

July
2001

Issue 112

news events people reviews projects programming

# Forth for NEAR Spacecraft

FIGUK magazine:

“Quikwriter” Proposal

Forth Gesellschaft - Tagung 2001

The FIG UK Awards of 2000

Arithmetized Logic in Forth

“Extreme Mindstorms”

JenX - A very simple XML parser

Three Free Forths and an OS too!

July
2001

Issue 112

## events

euroFORTH 2001 ……..…….. 15
Forth Gesellschaft - Tagung 2001 …... 16

## projects

“Quikwriter” Proposal ……..12

## news

Forth News ……..…….. 2

## reviews

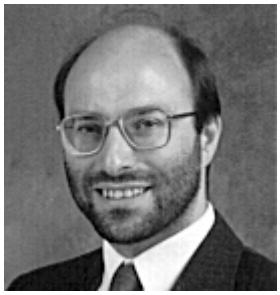
Forth for NEAR Spacecraft ……..…. 5
“Extreme Mindstorms” ……..... 25
Three Free Forths and an OS too! ….. 36

## programming

Arithmetized Logic in Forth …….. 20
JenX - A very simple XML parser …… 33

## people

Chairman’s Message ..…….…. 9
The FIG UK Awards of 2000…….. 19
Letters ……40



# Editorial

Lots of good stuff in this extra-large issue – hope it was worth the wait.

As promised, Joe Anderson reports on the remarkable NEAR space probe and Dave Abrahams delivers a detailed review of the new Lego Mindstorms book to whet your appetite for Forth robots.

We have more on XML in this issue and the next. This looks set to be a pioneering effort by FIG UK members working together.

Paul Bennett returns to describe his explorations with Forth on Linux and I report on the recent German FIG conference.

As well as our new Chairman (see his message inside), we now have a new Librarian. Graeme Dunbar has taken this over from Sylvia and will be publishing regular items on Library topics.

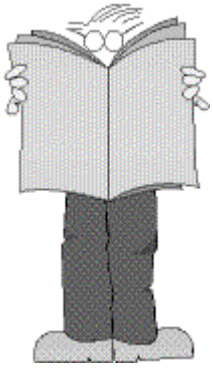
The University of Teeside is now a subscriber to Forthwrite, part of our out-reach to higher education.

Finally, congratulations to our Award Winners for Year 2000 – see inside for details.

PS. Don't forget the monthly IRC session. Our next one is Saturday 4th August on IRCNet channel #FIGUK from 9:00pm.

Until next time, keep on Forthing,

Chris Jakeman



Dave Abrahams
0161 477 2315
d.j.abrahams@cwcom.net

# Forth News

## 64-BIT FORTHS

In a discussion on comp.lang.forth Michael Coughlin predicts 64-bit Forths will be needed "sooner than we expect." Elizabeth Rather points out that they're already available.

"A 64-bit Forth will hardly ever need double precision for anything! Actually, there are 64-bit Forths on SPARCs."

## Ficl V2.06

John Sadler has announced the release of V2.06 of Ficl, an open source ANS Forth designed to be incorporated into other programs, including (especially) firmware-based systems. Ficl 2.06 is licensed under the terms of the BSD license and is available from:

http://ficl.sourceforge.net/

## eForth

Bill Muench has moved his web site for eForth to: http://homepage.mac.com/forth/

## BETA Testers Needed

Gary Chanson is looking for experienced Forth programmers to beta test Quest32. It is "an elaborate 32 bit development system for Windows 2000, Windows NT, and Win9x and is written in a dialect of Forth which is derived from Forth-83 and FIG-Forth."

Email applications please to gchanson@shore.net

## IRE-2001 Workshop on Java Virtual Machine

A "Workshop on Intermediate Representation Engineering for the Java Virtual Machine" will be held in Orlando, Florida, USA on July 22-25, 2001 at the 5th World Multi-Conference on Systemics, Cybernetics and Informatics.

Authors interested in submitting papers should see the workshop web page:

http://www.cs.may.ie/~jpower/ire2001/

## ANS Forth INTERNATIONALISATION

Following the papers published at euroFORTH 2000, a proposal and an implementation is now available from MPE.

http://www.mpeltd.demon.co.uk

## Machine Forth and Color Forth

Jeff Fox announced on comp.lang.forth, two new list servers for Machine Forth and Color Forth. To subscribe to theselist servers send an email to

MDaemon@chaossolutions.org

with "subscribe MachineForth" or "subscribe ColorForth" on the first line of the body then reply to the confirmation mail without changes.

Jeff writes:

"OKAD II is now written in Color Forth and is better than ever. Chuck now has descriptions of basic components in Forth source code and compiles the chip object when moving between the Forth editor and the chip simulator in OKAD, it appears to be instantaneous or at least faster than you can perceive."

http://www.UltraTechnology.com

## ProForth VXF Forth version 3.4

MPE's VFX Forth v3.4 for Windows is now available, it's even faster and with


internationalisation support. The price of the standard edition has also been reduced.

http://www.mpeltd.demon.co.uk/

## ProForth VXF Forth - free version

The evaluation version of VFX is available as a free download from the MPE website.

"The full system with a short nag screen, no timeout, no kernel sources and no turnkey generation."

## RTX2000

After an enquiry on clf about the Forth chip manufactured by the now defunct Novix company, Elizabeth Rather points out that it is still available from Harris as the RTX2000 in RAD-hard versions.

"We have recently done an application for NASA involving its use for an extremely accurate position encoder for a satellite."

## Shboom Chip

The ShBoom chip is sold by Patriot (http://www.ptsc.com) as the "Ignite 1".

Forth Inc. offer the SwiftX cross-compiler product for it; look for it under its previous name PSC1000 on:

http://www.forth.com

## TPFORTH version 3.2

A new version of TpForth has been released which adds the multi-language support (MLS) and the native code generation (NCG) systems. Download it free from:

http://www.technopoint.net/tpforth

## Forth and Super-scalar processors

In a discussion on clf, Stephen Pelc reported that there is some research going on at York University, UK, by Chris Bailey and others on the use of Forth with Super-scalar processors.

## Chuck Moore's Web Site

Chuck Moore, inventor of Forth, has now published his own web site:

http://www.mindspring.com/~chipchuck

This features "25x", an array of 25 "X18" microprocessors on a single chip, each running at 2,500 MIPS. The system is designed using Chuck's proprietary tools written in ColorForth.

Joe Anderson
0131 662 4007
jia@jia.abel.co.uk

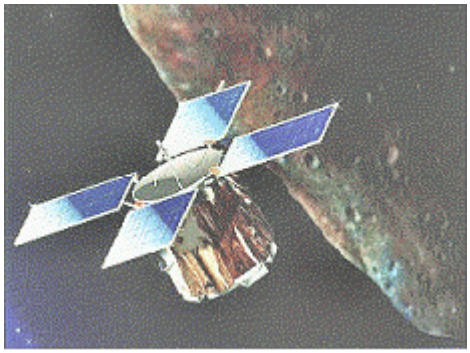
# Forth for NEAR Spacecraft

## Joe Anderson

In a recent headline-grabbing event (Feb 12th), the NEAR space probe was diverted, after successfully completing its mission, to land on the Eros asteroid – the first such landing ever attempted.

All the instruments and the command and data handling system were programmed in Forth.

You may have read "Did You Know?" reporting in the April issue about an article on the use of Forth in yet another space application and will not have been surprised, for some of you may remember previous articles making a link between Space and the Forth language. This link is the reason that the NASA web-site[1] is one I frequently visit, and I have been there especially often recently as there has been so much interest in a unique event, the first landing of a spacecraft on an asteroid. The spacecraft in question has been given the name NEAR Shoemaker, the NEAR standing for "**N**ear-**E**arth **A**steroid **R**endezvous".



Artist's impression of the NEAR Shoemaker spacecraft approaching the asteroid Eros.

The background is that nearly all the asteroids we know about are whizzing around in the asteroid belt, but a small number are much closer to Earth, and follow different orbits. These are known as the "near-earth asteroids" but we know very little about what they really are, what they are made of, where they come from, and so on. Astronomers can be expected to take every opportunity to investigate one of them. It was calculated that such an opportunity would arise when one of these, with the juicy name of Eros, would come very close to Earth early last year – to an astronomer 196,000,000 miles is "near Earth".

It was decided to launch a "low-cost" spacecraft in 1996 to find out more about this lump of orbiting rock, as it would take the full four years to get to the meeting-point. Incidentally, the timing of such a mission is quite critical, as an asteroid like Eros has a highly elliptical orbit, usually completely out of sync with

[1] See list of useful sites

the Earth's, and the occasions when the two are relatively close are thus comparatively few - the next "close encounter" will be in 2012.

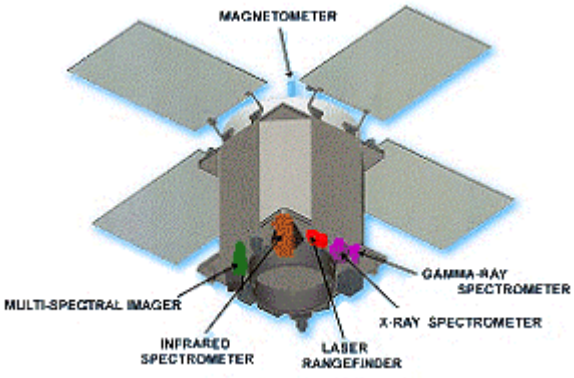
For those of you interested in figures, the orbit of Eros takes it round the Sun once every 1.76 Earth years. It is about 21 miles long by 8 by 8, and takes only 5.27 hours to rotate round its axis, a fast rotation that could have led to problems. It has no atmosphere that we know of, and because of its tiny size, has very weak gravity, a feature that came in very handy later on, as we shall see.

Management of the project was entrusted to Johns Hopkins University Applied Physics Laboratory, where a considerable expertise in this field has been built up. Most of the information in this article comes from JHU, and in particular from John R. Hayes, who has responded most helpfully by e-mail[2].

To get the sort of information the astronomers were after, the spacecraft had to carry certain specific instruments for measuring various aspects:

- an X-ray/gamma ray spectrometer, XGRS
- a near-infrared imaging spectrograph and magnetometer, NIS/MAG
- a multispectral camera fitted with a CCD imaging detector, MSI
- a laser rangefinder, NLR.

This artist's impression shows the general layout of the spacecraft, the disposition of the solar panels, and where these instrumentation packages are located on it.




All the instruments were programmed in Forth. John reports that other vital aspects of the project were also programmed in Forth, in particular the dual-redundant Command & Data Handling (C&DH) processors – "These relay telemetry to the ground, forward commands to the instruments, manage the recorder, handle spacecraft autonomy, etc." explains John.

Originally the spacecraft was meant only to orbit Eros and take measurements of its various properties. It duly spent a year orbiting the asteroid, collecting and transmitting data to Earth. It is reckoned that this mission has provided the scientists with an enormous amount of material, far more that had been originally anticipated, to assist them in their investigations into the origins and composition of such asteroids. One source has estimated that about ten times more data has been gathered than had originally been expected. Such plethora of data has had a double effect. In the first place it has enabled the astronomers and physicists to determine many facts about the asteroid and

[2] All the good facts in this write-up are John's and any blunders are mine.

answer many questions, but it has also raised a whole series of new puzzles, which you can follow on the web-sites listed.

At the end of the period, when the spacecraft had fulfilled its original mission, the question arose of what to do with it next. Some bold individual (for a proposal like this is too bold to come from a committee) put forward the suggestion that it be brought down on to the surface of the asteroid. After all, if it then was damaged or became non-operational, it had really done all that had been set out for it to do, and in any case, the asteroid would soon be entering a part of its orbit where the solar panels would be out of the sun, and hence all power to the instruments would be lost. The decision was then taken to attempt to land the spacecraft on the asteroid - the first time any such event had been tried in the history of space or astronomy - a genuinely unique event.

**Suggested web-sites**

http://www.jhuapl.edu/

Johns Hopkins University Applied Physics Laboratory site.

http://near.jhuapl.edu

Devoted to the NEAR project with some stunning material.

http://forth.gsfc.nasa.gov

See this site for Jim Rash's "Space Related Applications of Forth". Mentioned by John as being of particular interest to our members.

http://www.nasa.gov

The main NASA site with links to other interesting locations, e.g. the Goddard Space Flight Center, the Marshall Space Flight Center, etc. Navigate to the nssdc.gscf.nasa.gov site for a full history of the NEAR flight and summary of what it brought in. (nsscd = National Space Science Data Center, and gsfc = Goddard Space Flight Center). .

http://www.astronomy.com/

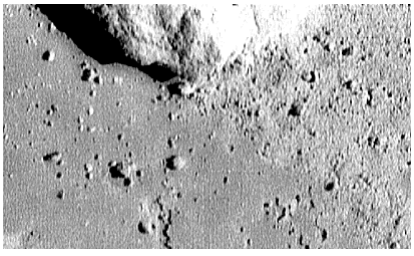
A magazine which gives a good coverage of events such as this.

http://www.intersil.com/

The Intersil web-site with full details of the RTX21010 chip.

The relatively fast rotation of this asteroid obviously made a safe landing on a suitable surface a matter for close control and precise timing; in the event (dare I say, "to everyone's surprise"?) it worked out perfectly. The controllers fired the thrusters several times during the four and a half hours descent, to slow the rate of descent to 7 mph from 20 mph. In the final 45 minutes, i.e. with the spacecraft about four miles from the landing site, pictures were shot at the rate of about one every 30 seconds "- some showing surface details smaller than 4 inches across", according to one source. Now the benefit of the weak gravity can be appreciated; even a heavyweight setting foot on Eros could do so with the delicacy and lightness of a butterfly, and for Shoemaker, the landing was even gentler.

The spacecraft touched down safely on the asteroid on Monday, 12 February 2001, having taken a series of 69 pictures during this final phase. Please go to the John Hopkins web-site (http://www.jhuapl.edu) and have a look at this sequence of pictures showing the surface of the asteroid getting nearer and nearer, until the final picture, shown here, where the spacecraft has come to rest on the asteroid.



The final picture from the spacecraft, showing the surface of the asteroid as seen when the spacecraft came to rest on the surface – a historic moment.

Although this was originally intended as the end of the project, the landing was so gentle that the instrumentation was still working well, and it was decided to extend the mission for another ten days, to gather data from this unprecedently close position. In particular data from the X-ray spectrometer was considered to be invaluable, for at this distance it could analyse material for several inches below the surface of the asteroid. This meant some rapid re-programming, which was done in record time. And then another extension of a further four days was granted, to squeeze all the data possible out of this most successful mission.

The part played by Forth in this exciting and historic event has been mentioned above, but as this is our common interest and your reason for reading this journal, perhaps a little more information can be given. John made it clear that Forth was the obvious choice, not only because of its economy and speed, but also because there was a suitable chip available. This chip is the RTX2010RH (the RH standing for "radiation hardened", a *must* for space applications).

| **Software Subsystem** | **Command and Data Handling** | **Guidance & Control** | | **Instruments** | | | | **Systems Engrg.** |
|---|---|---|---|---|---|---|---|---|
| | Command & Telemetry Processor | Attitude Interface Unit | Flight Computer | MSI Inst. DPU | NIS/MAG Inst. DPU | XGRS Inst. DPU | NLR Inst. DPU | |
| **Lines of Source Code** | 13,500 | 13,000 | 18,000 (guess) | 7,600 | 5,600 | 11,400 | 6,400 | |
| **Application Size/Available (KB)** | 56/64 | 90/128 | 200/512 | 63.3/64 | 50.4/64 | 55.9/64 | 49/64 | |
| **CPU & Language** | RTX2010 Forth | RTX2010 C & Forth | 1750A Ada & Assembly | RTX2010 Forth | RTX2010 Forth | RTX2010 Forth | RTX2010 Forth | |
| **CPU Rating** | 3 MIP | 6 MIP | 1.7 MIP | 6 MIP | 6 MIP | 6 MIP | 1 MIP | |
| **CPU Utilisation** | 40% average 58% peak | 35% | 70% average 100% peak | | | 20% average | | |
| **Development (man months)** | 56.8 | 54 | 124 | 15 | 17.2 | 13.8 | 9.7 | 25.4 |
| **Lines/man-hour** | 1.37 | 1.39 | 0.84 | 1.88 | 1.88 | 3.64 | 2.20 | |
| **Developers** | D.Artis B.Heggestad L.Linstrom | P.Haring, W.Frank C.Ray D.J.Waddell | R.Pham C.Ray T.Strickwerda L.Fisher D.Haley G.Heyler DJWaddell S.Hutton | B. Ballard | J. Hayes (also did boot code & common code for all DPUs) | S.Schneider | A.El-dinary | S. Lee (also system stuff & autonomy rules) |

NEAR Flight Software Statistics (rough approximations)

D.A. Artis 10/31/96

Like many of us, I have tried to use Forth at work, but found I had to do most of my professional material in C or C++, simply because so few others knew Forth or could be persuaded to take an interest in it, so I asked John how he had got his programmers - were they physicists who had an interest in Forth, Forthers who were interested in physics or astronomy, or what. John's answer was revealing and rather modest. He confessed that he took on staff and then himself taught them Forth for the purpose.

He also added, something that will strike a chord in many of us, that some people took to Forth quite well, but others just could not get the hang of it at all and programmed in C instead. My theory is that those with an interest in either the technicalities of logic, or the technicalities of electronics, i.e. how the chip actually works, find the rationale behind Forth fits in well with their thinking, but those interested more in a narrative style of thinking or describing go for one of the other languages.

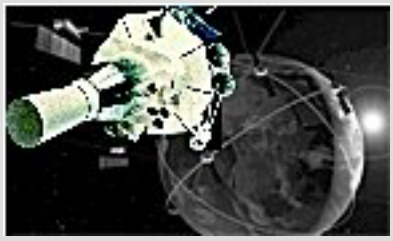
In conclusion I would like to thank John for his very generous help and also Dr. Dave Williams of Goddard Space Flight Center, who gave permission for the web-site material to be published here.

---

*Joe Anderson started out in Forth thanks to Byte magazine, a 6800 kit and a teletype. Now retired from working in airborne radar, he is likely to be found helping out at the National Gallery on The Mound, Edinburgh.*

The work done for this long-running project was completed some years ago, but Forth continues to reach out into space.

Forth Inc. has posted details of a project completed in the past 6 months for a space-based infra-red instrument to track missiles.



# *Chairman's Message*

## *Jeremy Fowell*

Since I have taken over from Chris Hainsworth I would first like to wish Chris and Sylvia success and happiness with their new life in Spain. Chris had been Chairman for quite a few years and contributed a lot to FIG UK during that time. I hope I can follow his example.

My contact with Forth began with an advert for polyFORTH in an American magazine in the early nineteen eighties. Although the ad. didn't say very much, I still clearly remember a strong feeling that there was something very significant going on here. At the time I was struggling to learn about the 8080 processor using an early home-built computer. It seemed a black art, the machine didn't appear to have an assembler and I couldn't afford to buy a proper one.

It wasn't until 1987 that I discovered FIG UK and joined up. By this time we were writing 8080 assembler on a BBC Micro with a Z80 second processor. We bought two floppy drives and it was the first real computer I ever got to use. Soon we had a Forth compiler from MPE to experiment with. The 8080 application had now grown to many pages and kept us very busy coding and debugging, along with hardware design and test. Sadly we lacked the experience and time to rewrite the whole lot in Forth. However the idea of Forth had taken hold.

Over the years I have been able to use Forth more and more, but it was not until PygmyHC11 was completed that I felt that the Forth I really wanted had finally arrived.

There is much still to do - the Incremental Compile project is nearing completion and then we need building blocks such as keyboards and small displays with hardware and software to act as examples. Maybe then one or two complete (but simple) projects[3].

Although only a small part of the Forth picture, I hope this is one way we can attract newcomers to the subject.

As Charles Moore once said "We need to beat the drum".

---

[3] See the Keyboard Project elsewhere in this issue.

# F11-UK

provides everything needed in a professional-quality low-cost Forth controller board.

Use it in industrial or hobby projects to control a wide range of devices using the well-known multi-tasking Pygmy Forth.

Designed for hosting from a DOS or Windows PC, you can test your application as it runs on the F11-UK board itself. The board was developed by FIG UK members to provide an easy way to explore the world of controlled devices – a niche where Forth excels.

The kit includes both hardware and software and is supported and sold to members at a nominal profit through a private company.

**Software**

**PC-based PygmyHC11 Forth compiler** running under DOS produces code for Motorola HC11 micro-controller.

**Code is downloaded** via standard serial link from the PC to the FLASH memory (or RAM) on the F11-UK single board computer (SBC).

**No dongle** or programming adaptor of any kind is required.

**Forth running on the SBC is interactive** which makes debugging and testing much easier.

**Multitasking and Assembly included.**

**The serial link can be disconnected** to enable the SBC to function as a stand alone unit.

**All source code provided** - 78 pages or so (unlike many commercial systems).

**Around 30 pages** of additional documentation is supplied including a full glossary of the 300 or so Forth words in the system.

**Email mailing list** for discussion and limited support.

**Hardware:**

**Processor:** Motorola HC11 version E1 – 8 MHz (2 MHz E-Clock).

**Memory:** 32k x 8 FLASH
32k x 8 battery backed SRAM
512 x 8 EEPROM onboard HC11.

**I/O:** 20 lines plus 2 interrupts (IRQ and XIRQ).

**Analogue in:** up to 8 lines using onboard 8-bit A/D.

**Serial:** 1) RS232, UART onboard HC11
2) Motorola SPI bus onboard HC11.

**Expansion:** Via HC11 SPI serial bus using 2 or more of 20 available lines.

**Timer system:**
Inputs: 3 x 16-bit capture channels
Outputs: 4 x 16-bit compare channels.

**PCB size:** 103 x 100 mm.

**Price to FIG UK members:** £47.0 plus postage and packing (£2 UK, £4 overseas) plus $25.0 (US Dollars) for registration of 80x86 Pygmy Forth with the author Frank Sergeant.

**Delivery:** ex-stock.

**More information:** jeremy.fowell@btinternet.com and 0121 440 1809

# "Quikwriter" Proposal

Here are some edited messages which I hope will explain themselves. If you have anything to contribute, I urge you to do so. Not only does this project promise real benefits but there is plenty of scope for some interesting programming – Ed.

From Chris Jakeman to all FIG UK members on e-mail – 7th May:

Hi everyone,

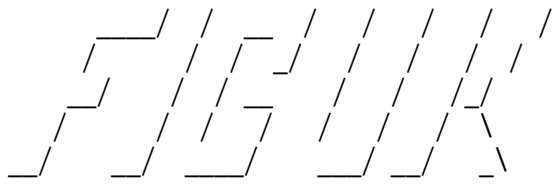
Jenny Brien has a requirement for a device which might be purely hardware or might be a mix of hardware and software. He has asked me to send this request for advice out by e-mail as he doesn't want to wait until the June issue of Forthwrite.

If you have suggestions to make or if this project might be of interest to you, please e-mail to Jenny at jennybrien@bmallard.swinternet.co.uk.

PS. I'm especially interested in this project as it might:
- help the community
- involve the F11-UK kit
- the amount of programming involved could range from simply translating signals up to a flexible, adaptive, configurable user interface.

Bye for now

Chris Jakeman



Forth Interest Group United Kingdom
Voice +44 (0)1733 753489 chapter at http://www.fig-uk.org

FROM JENNY BRIEN:
I've been doing voluntary work lately with an association for the disabled, who have just got a computer suite installed. Though there are plenty of chord keyboards and other one-handed text input devices, I can't find anything really suitable for the folk who have little or no finger mobility.

I have in mind something like a short joystick that can be operated with the palm of the hand, with a sprung central position and eight directions of movement, to produce "Quikwriting" (see: http://www.mrl.nyu.edu/~perlin/demos/quikwriting.html) and interface to a normal PC keyboard socket.

Would any of the membership be able to produce such a device, or know of suitable hardware?

[cont.]

A diagram from the Quikwriting article indicates that out-and-back movements give the central character, but a shift to the neighbouring zone gives a different character (compare “e” and “h”). Interestingly, the research that led to Quikwriting was designed for faster text input, not to overcome disabilities.

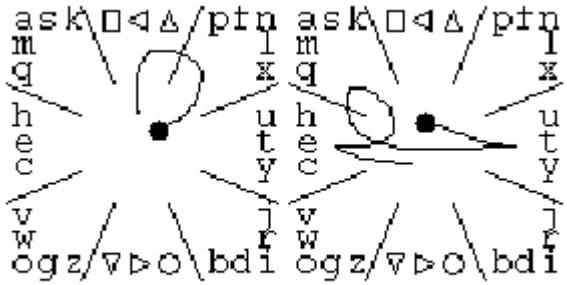


Figure 2: Writing the letter 'f', and the word "the"

COMMENT FROM CHRIS (based on further e-mail & May's IRC session)
Jenny began by thinking of a purely hardware device which was a plug replacement for the PC keyboard. If anyone knows of such a thing, please tell.

I was thinking that a games joystick could be the ideal mechanism, if we teamed it up with a F11-UK kit. All shapes and sizes of joysticks are available in the computer shops, mostly with a spring return. If we added some connectors to the F11-UK kit, it could pretend to the joystick that it was a PC and to the PC that it was both a keyboard and a mouse.

I have a suitable joystick I can donate to the project and also an F11-UK (sadly still unassembled).

PS. I would love to see an interface which monitored the relative frequency and adjacency of each keystroke actually used and then offered to switch to a more convenient layout. That would be really neat.

We had more input from Jeremy Fowell, Paul Bennett and Jan Coombs. First Jenny provides a bit more background.

Hmm... I'm thinking of something which could be used with any PC, so my first thought as a direct plugin replacement for the keyboard. No software. The nearest equivalent is the Cykey chord keyboard (developed from the old Microwriter Agenda) but that doesn't emulate all keystrokes. Perhaps not important, when you can mostly control with a mouse or other pointing device. Besides, with only 32 variations on a single stroke, you can't get too clever. What do you need - alphanum, delete, tab - what else?

Quikwriting was invented for the Palm Pilot, and the Java demo uses a mouse. It's easy to program but a bit awkward to use, and I think the self-centring mechanism would improve it no end. But then, we need a replacement for the mouse too, so perhaps they could be combined and use the mouse port, treating the writer as a five-button mouse. I don't know if Windows can do that.

Jeremy wrote in to say that the cheapest solution would be to add software to the PC to use the input from a special keyboard, if one could be found, or from a joystick. However, an easier (and portable) solution is to put some intelligence between the joystick and the PC, so that the joystick appeared to be a standard keyboard and mouse. He confirmed that the F11-UK kit would be appropriate for prototyping and a cut-down version using the HC908 would be cheaper for large numbers.

Paul Bennett had the same idea and suggested a possible joystick.

> My immediate thoughts turned to an adaptation of the Pizza-key. In case you haven't come across any of these yet, they are 4, 6 or 8 way switching devices operated from a single button. The Pizza-key rocks in all eight directions to close the contacts. They are used on some of the more rcent equipment (Game-boy [4-way], Sony Playstation some CD Walkman's, VCR's and DVD players).
>
> Adding such a key to the F11-UK project board would enable suitable key activation decoding to occur and produce the required strings as output to the RS232 port. I suppose Jenny's people will want a slightly larger one and this is easy enough. I might have enough switches to make an eight-way prototype around here.

Jan Coombs suggested butchering a cheap (£8) PC keyboard for a low-cost solution that requires programming the PC.

> A good hardware starting point may be a standard £8 PC keyboard. The PCB is often very small and fixes compatibility issues quickly.
>
> The PC keyboard likely has unused scan codes, so it may be possible to simplify the debugging process by attaching the new switches as an extension to a working standard keyboard.
>
> You might choose to use 8 infrequently used keys for your new device e.g. <F5> to <F12>. I would want to use the keyboard for programming while testing the new input device. If you need to retain independent use of all 102 keys and have the new device connected, then a two-key sequence coiled be used to indicate the new device codes.
>
> The PC keyboard sends separate key-down and key-up codes to the PC, and this could assist decoding the movement of the new device. There are BIOS (and DOS?) functions to retrieve these codes, but likely not Windows support.

# *euroFORTH 2001*

The 17th annual euroFORTH conference on the Forth programming environment and Forth processors is being held on November 23 – 26 at Schloss Dagstuhl, near Saarbrücken, Germany.

This annual conference is held in the UK every third year and, after the 1999 venue in St.Petersburg, it returns again to Schloss Dagstuhl. (See Paul Bennett's detailed report in issue 99). The conference language will be English.

For conference details, see http://dec.bournemouth.ac.uk/forth/euro/ef01.html. FIG UK member Bill Stoddart is the Program Chair and invites papers (both academic and business) by 26th August, please, to the Proceedings Editor Peter Knaggs (pknaggs@bournemouth.ac.uk). Topics of especial interest include:

- Forth applications and language extensions.
- Open protocols and standards, including TCP/IP, HTTP, XML etc.
- Virtual machine application and design.
- Stack-based architectures.
- System configuration and Open Boot.
- Other topics likely to be of interest to the extensible language community.

**Charles Moore** is the Guest of Honour, so this is a rare chance to meet the inventor of Forth on this side of the Atlantic.

Attendance with a single room in the castle for 2 nights and full board costs €350 (£220)[4]. Discounts are available for students sharing rooms.

[4] Bill tells me that prices have been kept to a minimum to encourage the widest possible attendance. Let's take advantage of that – Ed.

Chris Jakeman
01733 753489
cjakeman@bigfoot.com

# Forth Gesellschaft - Tagung 2001

## Chris Jakeman

The German FIG combines their AGM each year with a residential conference. This year, I was invited as Guest Speaker – my first visit.

Germany being a large country, this event takes place at a different location each year. This year's organiser was Martin Bitter, the Editor of the magazine and known on our IRC sessions for his interest in teaching using robots programmed in Forth. Martin had chosen a residential college in Hamminkeln, near the Dutch border.

Not only was the college very pleasant, but this arrangement meant that we could stroll to the restaurant and retire to the bar with ease.

The presented papers were spread over Friday evening and Saturday, with the promised Forth Robot challenge that evening and Sunday morning set aside for the AGM.

About 18 attended and, although the papers were of a high standard, this was clearly a meeting of good friends with a relaxed atmosphere and lots of laughter. A number of wives had made the journey, providing a civilising influence and escaping the stream of Forth talk for shopping and sight-seeing trips together. Several teenagers made an appearance to take part in the robot challenge. At DM300 (under £100) for everything except the bar, this seemed good value.

I am delighted to report that the group gave me a very warm welcome, (also a Gesellschaft sweat-shirt and mug) and helped me join in from the outset.

After giving my own paper on WebForth and talking about FIG UK, I did my best to follow the other papers, some more easily than others:

**Fred Behringer** spoke on Arithmetic Logic which arises out his robotics work and uses a simple mathematical transformation to provide a general solution to a difficult problem in machine control – see elsewhere in this issue.

**Wolfgang Allinger** showed the neatest portable terminal I've ever seen. He has taken a standard Palm Pilot, added Quartus Forth and programmed it to talk to his instrument pack via a serial link. Apparently, the infra-red PC link can also be used for communicating to other devices too, so his portable terminal could also be wireless.

**Heinz Schnitter** presented an Open Network Forth which he has used with Egmont Woitzel for the control system for the Munich Accelerator Facility[5]. All the devices on the network run the same object-oriented Forth and one Forth can execute a method on an object on another Forth. Communications use plain text just as though one Forth was a user typing at the terminal of another. The acknowledgement to a successful command is "ok" which, when received, rather neatly executes a deferred word on the sending system. The acknowledgement to an unsucessful command is "ko" which similarly executes a deferred word.

I note that communications between processors commonly use binary messages, such as RPC or CORBA. However text is much easier to debug, log and monitor and the computer world seems to realise this at last with the recent rise of XML for messaging.

**Klaus Zobara** showed his work on management of a surgical instrument that cuts using a very high frequency current "like a hot knife through butter". He

*"like a hot knife through butter"*

presented a very neat Finite State Machine and also a Decision Table. The paper contained restricted material, but I am hopeful that these particular aspects might be published separately.

**Hans Eckes** showed his high-performance controller board based on the Patriot PSC1000 chip. This is very compact (similar size to F11-UK) with the processor on a daughter board to provide flexibility. Running at 80MHz, the chip performs a fast Fourier transform (FFT) of 1024 points in just 0.14 seconds. This board is for sale and was built primarily for the Reilhofer KG company to provide extra performance (see below).

### Between The Papers

Because of the language barrier, the chat between the sessions was somewhat easier to follow than the papers! Several members run small businesses that have thrived on the speed advantages from using Forth. Unsurprisingly, they are innovators in other ways too as follows.

**Adolf Krüger** explained his Forth company's unusual approach to project financing. Instead of the traditional approach of predicting the cost of a project and asking for 30% in advance, he completes the project and then sells the results to his client at a price based on the actual cost of the work. Occasionally

[5] An English language version of this paper is available.

the client refuses and the work is wasted, but he finds this approach has many benefits.

**Johannes Reilhofer** runs Reilhofer KG (http://www.rhf.de), a company specialising in automated instruments to test gearboxes and transmissions. This requires high performance and unattended operation as his customers include both the largest car producers and some Formula I racing teams. Jaguar cars use his equipment to test all their gearboxes. Each gearbox is automatically loaded into a special booth and taken under power through all the forward and reverse gears in just 20 seconds. The vibrations are analysed during that 20 seconds to find faults in components or assembly.

To achieve accurate results in such a short time, Reilhofer has developed some special techniques and has commissioned and adopted high-performance Forth processor boards. They have also introduced simple but innovative statistical techniques which allow the software to divide the transmission systems into good and bad without requiring any test limits from an engineer. This radical step simplifies the test configuration dramatically and, two years on, this new approach has been fully accepted by the users.

**Swap Dragon** – this is awarded only once in a lifetime and goes this year to Martin Bitter for this year's successful conference, his recent work as Editor of Vierte Dimension and his continuing teaching using robots programmed in Forth.

**Ulrich Hoffman** has been appointed the new Director of Forth Gesellschaft and will be managing the web-site too. [Late News: Fred Behringer will assist in editing the Web site and he intends making the site available in English too. Volunteers from FIG UK who can help Fred polish his translations will be warmly welcomed.]

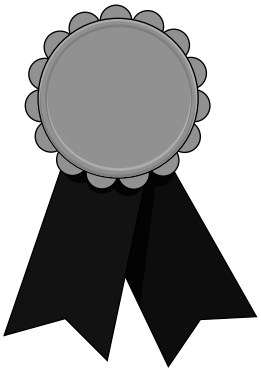
### Robot Challenge

Forthwrite has reported before how Martin Bitter uses robots built from Lego Mindstorms controllers and programmed in pbForth in his teaching. This year, for the first time, the conference included a challenge which Martin had organised around 4 identical sets of robots. The 4 teams raced to program the devices to plot the words "`Forth 2001`" and (silly) prizes were awarded before we retired once again to the bar.

### A Conference for FIG UK

Why don't we have a similar conference for FIG UK? After all, FIG UK has a similar membership and travel distances are rather shorter. Given a willing and energetic volunteer, it would be worth serious consideration. For next year, though, I propose an alternative.

Forth Gesellschaft have many years' experience in running successful conferences and getting away from Britain for a weekend is an attractive proposition after the winter. With this in mind, how about some of us getting together to form a British contingent to the next German conference? (Munich 19th-21st April 2002). I can guarantee a warm reception.



# *The FIG UK Awards of 2000*

The FIG UK Awards of 1999 were won by Jeremy Fowell and Alan Wenham. These awards are given to encourage effort and recognise achievement.

Free membership

To everyone who sent in their nominations - "thank you". Looking back, a lot of good work was done during 2000, but our judges, the officers of FIG UK, have now chosen their winners for 2000. They each receive:

- a place in our web site's Hall of Fame
- this mention in Forthwrite
- ***a year's free membership***.

Achievement

**Keith Matthews**: for 13 years of work behind the scenes as Treasurer getting FIG UK to run smoothly and keeping it that way.

Forthwrite

**John Tasgal**: for his articles (available on-line as a special issue) explaining Chuck Moore's Machine Forth and Color Forth.

We congratulate Keith and John on winning - enjoy your year of free membership!

Fred Behringer
behringe@mathematik.tu-muenchen.de

# Arithmetized Logic in Forth - An Introduction

## Fred Behringer and Chris Jakeman

This is a simplified version of an article which Fred Behringer has prepared for the German FIG “Vierte Dimension” magazine. The present article, whose presentation is down to Chris Jakeman, is intended to be an introduction for those less mathematically-inclined. A more rigorous treatment with the pros and cons will appear in a forthcoming issue of Forthwrite. The subject itself being Mathematical Logic, its language is Mathematics, and as everyone knows, there is no easy route to Mathematics.

How can we find out whether a bit pattern matches a logical rule? For example, we might have 3 independent input bits (possibly from Lego robot touch sensors) and a rule that is true if at least 2 of the 3 inputs are true. (Imagine a jetplane with 3 turbojet engines from the view of Reliability Theory: it crashes if less than 2 engines remain working.)

The simplest way is to build a truth table and Forth makes this very simple:

```
BINARY
HERE
FALSE , \ 000
FALSE , \ 001
FALSE , \ 010
TRUE  , \ 011
FALSE , \ 100
TRUE  , \ 101
TRUE  , \ 110
TRUE  , \ 111

CONSTANT TruthTable

: 2OutOf3? ( BitPattern - Flag )
      CELLS TruthTable + @
;
```

In 2OutOf3?, we interpret the truth table's bit patterns as binary numbers which we use as an offset into the table. When the number of bits grows, this procedure becomes unwieldy. With 16 bits of input, the table grows to 65536 entries, each to be entered manually. The thought of entering larger sizes still becomes difficult to contemplate.

Here, Mathematics comes to the rescue of the automation engineer with a space-saving transformation called the Multi-Linear Form (MLF). Any logical function can be converted to a table of coefficients which in many cases has far fewer entries than the corresponding truth table.

In the 2OutOf3? example, we build a table containing 4 pairs of keys and coefficients:

```
Key (bit pattern)  Coefficient
011                 1
101                 1
110                 1
111                -2
```

The work is done by the MLF defining word (below) that creates 2OutOf3?, pointing it at the table and then provides the analysis when 2OutOf3? is executed.

Where do these mysterious coefficients come from? Mathematics states that logical (ie Boolean) expressions can be converted to arithmetic expressions as follows. If FALSE is represented by 0 and TRUE by 1 (as in many textbooks on Mathematical Logics and any digital data sheet), then the following is true for the single bits x and y.

```
x∧y = x*y          \ x and y = x times y
x∨y = x+y-x*y      \ x or y = x plus y minus (x times y)
¬x  = 1-x          \ not x = 1 minus x
```

Our 8-entry truth table can be written as the logical expression:

$$\neg x_2 \wedge x_1 \wedge x_0 \ \vee \ x_2 \wedge \neg x_1 \wedge x_0 \vee x_2 \wedge x_1 \wedge \neg x_0 \vee x_2 \wedge x_1 \wedge x_0$$

i.e. (not X2 and X1 and X0) or (X2 and not X1 and X0) or (X2 and X1 and not X0) or (X2 and X1 and X0).

How so? Examine each key from the truth table. Ignore the ones that have a FALSE value and consider only the TRUE ones. Replace each bit of value 0 in the key by the corresponding single bit variable in its negated form (eg not X2), and each bit of value 1 by the corresponding single bit variable in its affirmative form (eg X1), connecting the single bit variables by means of ∧, and finally glueing the whole thing together by ∨. This is known as the Disjunctive Normal Form (DNF). (For an explanation, see below.) Using the 3 rules above, we can show that the ∨'s in any DNF can be immediately replaced by +'s, however complex the rule for **x∨y** might appear at first sight, thus saving much computational effort. What we finally get as the equivalent arithmetic expression is then

$$(1-x_2)*x_1*x_0 + x_2*(1-x_1)*x_0 + x_2*x_1*(1-x_0) + x_2*x_1*x_0$$

which can be multiplied out and simplified to the Multi-Linear Form:

$$1*x_1*x_0 + 1*x_2*x_0 + 1*x_2*x_1 - 2*x_2*x_1*x_0$$

These factors (1,1,1,-2) become the coefficients in the table and the bit patterns 011, 101, 110, and 111 become the keys.

Having computed the entries in the table and used MLF to build them into the 2out0f3? table, how does 2out0f3? extract the result? It matches the input bit pattern against each key in turn and uses BNS to decide whether to skip the entry. If not skipped, 2out0f3? just adds the coefficient to the result so far.

BNS compares 2 bit patterns and returns true if all the bits in the key are also present in the input. For example, a bit pattern like 111 will be matched using BNS against all the keys in turn and, since 111 includes all the bits present in 011, 101, 011, and 111, the 111 bit pattern will accumulate by addition the coefficients for all 4 entries in the table: 1 + 1 + 1 - 2 = +1 or true.

```
: 2, ( a b -- )
        , ,
;
: BNS ( BitPatternA BitPatternB -- Flag ) \ True if A includes B
        TUCK AND =
;
: MLF ( Compile: &Base DataPairs*i -- )
      ( Execute: BitPattern -- Flag )
        HERE            \ &Beyond ( addr beyond array of data
                        \ pairs)
        CREATE SWAP 2,  \ Store array &Base and &Beyond
DOES>                   ( -- BitPattern &Base )
        0 SWAP          \ Bury an initial result = FALSE.
        2@ DO           \ Get limits of array
                        ( -- BitPattern Result )
          OVER I CELL+ @ BNS IF
            I @ +       \ Accumulate result
          THEN
        2 CELLS +LOOP
        NIP             \ Drop the BitPattern
        NEGATE          \ Convert +1 to –1 to provide Forth TRUE
;
```

The 2out0f3? word to be created by MLF is to act in a control structure like 2out0f3? IF .. ELSE .. THEN . Forth uses -1 to represent "true". The 2out0f3? word can be built quite simply:

```
BINARY
```

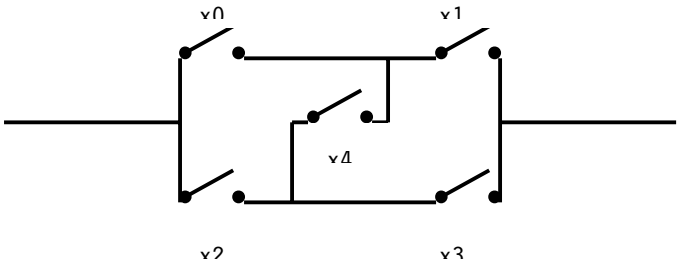
```
        HERE            \ &Base of array of data pairs
        011   1 2,      \ Define the table
        101   1 2,
        110   1 2,
        111 -10 2,
MLF 2Out0f3?            \ Create the analysis word
```

and then tested using input patterns from the command line as follows.

```
000 2Out0f3? .      0 OK
001 2Out0f3? .      0 OK
010 2Out0f3? .      0 OK
011 2Out0f3? .     -1 OK
100 2Out0f3? .      0 OK
101 2Out0f3? .     -1 OK
110 2Out0f3? .     -1 OK
111 2Out0f3? .     -1 OK
```

Note that if we see results that are neither 0 nor –1, then the coefficients must be wrong.

Now that we've seen how it works, let's look at something which would be hard to do without MLF. Here is a circuit of switches, similar to the arrangement of resistances in the Wheatstone Bridge. We've chosen it as the simplest arrangement which is not composed of parallel and/or series parts only, but where the 5 inputs lead to a more cumbersome 32-input truth table.




Note that a closed switch is represented by a "1" in the table below, and an open switch, by a "0". To list all the combinations of open and closed switch position requires a truth table of 32 bit patterns Instead, we present a shorter table of 10 keys and their coefficients for building the MLF word Bridge? instead.

```
BINARY
        HERE            \ &Base of array of data pairs
        00011  1 2,     \ Define the table for inputs x4 ... x0
        01100  1 2,
        11001  1 2,
        10110  1 2,
```

```
        01111 -1 2,
        10111 -1 2,
        11011 -1 2,
        11101 -1 2,
        11110 -1 2,
        11111 10 2,
MLF Bridge?             \ Create the analysis word
```

Where do these ten coefficients come from? Our 32-entry truth table (whose construction we've left to the reader as an exercise) can be written more compactly as the logical expression:

$$(x_0 \wedge (x_1 \vee (x_3 \wedge x_4))) \vee (x_2 \wedge (x_3 \vee (x_1 \wedge x_4)))$$

Using the 3 rules for converting from logic to arithmetic (as above), the equivalent arithmetic expression is then

$$x_1x_0 + x_3x_2 + x_4x_3x_0 + x_4x_2x_1 -$$
$$- x_3x_2x_1x_0 - x_4x_2x_1x_0 - x_4x_3x_1x_0 - x_4x_3x_2x_0 - x_4x_3x_2x_1 + 2*x_4x_3x_2x_1x_0$$

This immediately gives us the table of coefficients and keys needed for creating `Bridge?` by means of the defining `MLF` word as above. That's all there is to it.

There is an alternative to words created by `MLF` which can be used in the same way as `XXX IF ... ELSE ... THEN`. It is known as the Disjunctive Normal Form (DNF). Sometimes it is better to use the MLF method, sometimes DNF is better. More importantly, the DNF can be obtained directly from the truth table (once and forever while writing the program), whereas getting the coefficients for the MLF method is generally more cumbersome (only try to multiply the bridge expression out using pencil and paper!). Fortunately, there exists a method of transforming a DNF (equivalent to the truth table) into the corresponding MLF systematically and possibly automatically too. A thorough discussion of all this will appear in the forthcoming full version of this paper.

---

**From Forth-Gesellschaft:**
In Forthwrite's issue 111 we learned that Jeremy Fowell was elected as Chairman of FIG UK. Good luck to you, Jeremy, and best wishes from the German FIG Forth-Gesellschaft.

The Directors: Thomas Beierlein, Ulrich Hoffmann, Fred Behringer

David Abrahams
d.j.abrahams@cwcom.net

# Book Review
## "Extreme Mindstorms
## An Advanced Guide to LEGO Mindstorms"

## David Abrahams

This new book is the latest addition to our comprehensive lending library. Thanks to Lego Mindstorms, there has been a surge of interest in robotics – an area where Forth has many advantages. David has a personal interest in the subject and, after building some of the examples, provides us with a detailed review showing why he recommends the book so highly. Uniquely, he also provides feedback from one of the authors.

### *Introduction*

LEGO Mindstorms is described as a robotics invention system by the manufacturers. It was first introduced in 1998 and consists of the familiar LEGO construction bricks, beams, gears etc. but with one new item, a large "brick" housing a single board computer. This is the "RCX", with Hitachi H8 CPU, LCD screen and electrical connectors to provide power for motors etc and to read inputs from a variety of sensors.

The system comes with a graphical programming language that runs on a PC with the compiled programme downloaded to the RCX computer via an infra-red interface. The product was initially aimed at children and the standard RCX programming language is suitably simple.

However, the system very quickly attracted the attention of older customers who saw the potential of a ready made powerful micro-computer and easy to assemble mechanical components. Some of these people were knowledgeable engineers and programmers who, frustrated with the limitations of the RCX code, hacked the computer firmware and hardware to see what could be done with the system.

This book is written by four of these people and provides a guide to the current ways in which the Mindstorms system can be extended and the full power of the RCX computer can be explored.

Extreme Mindstorms - An Advanced Guide to LEGO Mindstorms
by
Dave Baum
Michael Gasperi
Ralph Hempel
Luis Villa

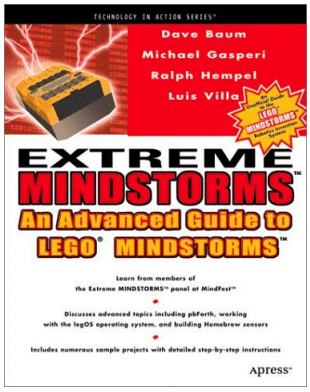
ISBN 1-893115-84-4

Paperback of 347 pages published by Apress at $29.95 (Amazon price $23.96)

### *The System*

The Mindstorms system costs about £150 from Maplins or Toys r Us and provides a way for anyone to experiment with control systems and robotics. Users have created a wide variety of items including XY plotters, maze solving robots and even a working model of  a numerically controlled milling machine.

### *The Book*




The book runs to 347 pages printed on high quality paper with black and white illustrations. There are comprehensive contents and index sections. I noticed one or two minor typos in the text and one illustration was missing completely but none of these errors were serious.

The book could have been improved by the use of colour as this would help to identify different Lego parts and electronic components. However, this would no doubt have increased the price significantly.

### *Part 1 - Mindstorms by Dave Baum*

This part covers chapters 1 to 4 and is written by Dave Baum, a Motorola engineer and creator of the NQC (“not quite C”) programming environment for the RCX. He is also the author of “The Definitive Guide to Lego Mindstorms.

Chapter 1 gives a brief history of the Mindstorms product and how and why the hackers stepped in. There are many references to web sites containing a wealth of information on the internal workings of the hardware and firmware.

Chapter 2 describes the capabilities of the RCX computer brick in terms of its inputs and outputs and the standard firmware. The book does not go into great technical depth here but does give a good basic understanding of the capabilities of the system.

The chapter describes how the RCX can be set to expect different kinds of sensor inputs and NQC is introduced with a short programme to play a tone when the RCX detects a change on the input of a light sensor.

## *“I had no problem following the instructions”*

Chapter 3 starts with instructions for the construction of “Seeker”, a light-seeking robot.  The instructions mainly consist of a series of illustrations of the robot at various stages of construction with some text to clarify certain points. I had no problem following the instructions to build the robot which is then used throughout the book to illustrate the different programming environments.

Chapter 3 then goes on to present a simple programme to drive the robot. The programme is built in small understandable blocks and is given in two forms, the standard RCX code and then the NQC equivalent. The chapter ends with a more sophisticated version of the NQC code using multi-tasking and makes the point that this approach is not possible with the standard code.

Chapter 4 is entitled “RCX 2.0 Firmware” and describes some of the new features introduced with this version which can apparently be applied to older RCX bricks. Included in this chapter are details of how to obtain and download NQC to the RCX. There are many examples of NQC code in this chapter which will be invaluable to anyone learning the system whatever version of firmware they are using.

The chapter concludes with a programme in NQC to drive “Seeker” based on event monitoring which illustrates some of the new capabilities in RCX 2.0.

### *Part 2 pbForth by Ralph Hempel*

Part 2 of the book comprises chapters 5 and 6 and is, of course, why this review appears in Forthwrite. Ralph Hempel is a professional engineer specialising in embedded systems design and the creator of pbForth[6]. In the forward to the book, Ralph describes how he started with the 8086 assembler source code for hForth and ported it to the Hitachi H8 of the RCX in “…about 40 hours over the Christmas holidays…”. This was then posted on the Internet and an enthusiastic response encouraged Ralph to develop the pbForth system further.

Chapter 5 is entitled “Introduction to pbForth”. The chapter is a mixture of information about pbForth and its application to the RCX and quite a bit of text about the Forth language in general and why it is suited to interactive development of embedded systems. In common with the rest of the book, no prior knowledge is assumed and the reader is encouraged to put some effort into learning Forth with for example a section headed “Why Learn pbForth?”. An extract from this section reads:

> *One look at the source code might be enough to send you screaming, but if you understand the philosophy of Forth, it will be easier to embrace. By learning another computer language, especially one that asks you to think differently about problems, you will deepen your bag of programming tricks. Even if you don’t write software for a living, learning a language like Forth can be a fun mental exercise*

## *“Ralph’s experience shows through”*

The chapter includes a brief summary of the structure of Forth, the stack and the dictionary and is then followed by some simple examples of using pbForth interactively. This is possible because pbForth running on the RCX can communicate continuously with the PC accepting key presses and generating screen output.

As the chapter progresses, standard and pbForth-specific words are introduced with the usual stack notation which is fully explained. There is quite a lengthy section on numbers and expressions showing amongst other things how integers can be scaled in Forth to give more than adequate precision without resorting to floating point.

---

[6] http://www.hempeldesigngroup.com/lego/pbFORTH/

Short pbForth scripts are used to illustrate how to access the hardware of the RCX, the LCD screen, sound system, motors, sensors etc.. Ralph's experience with controlling embedded systems shows through in much of these two chapters.

Chapter 6 is used to develop a pbForth programme to control the "Seeker" robot. Simple words are developed to perform specific tasks such as `BOT_FORWARD`, `BOT_REVERSE`, `BOT_LEFT`, `BOT_RIGHT` and `BOT_STOP` to control the robots motion. The Forth tutorial theme continues here with straightforward implementations of the words to start with and then the use of factoring to put the common elements of the words into `MOTOR_SET` which is passed parameters on the stack from simplified versions of the motion words.

Code is developed to display the status of the robot on its own LCD display and this section introduces vectored execution in Forth.

The chapter is interspersed with advice on programming in general and Forth in particular. Techniques which may not be obvious are explained in some detail such as the use of "weighted average" to sample the input from the light sensor.

These concepts are all combined to make a complete control program for the robot. In his summary to the chapter, Ralph writes:

*We have only scratched the surface of what pbForth can do. If you choose to explore Forth in more detail, you will find that you can easily create large arrays for data logging or mapping, use cooperative multi-tasking, and even do complex signal processing….."*

### *Part 3 - LegOS by Luis Villa*

Luis Villa studies computer science at Duke University and maintains the LegOS HOWTO on the Internet. Part 3 consists of chapter 7 "Introduction to LegOS" and chapter 8 "Advanced LegOS".

LegOS is described as a development environment designed around the standard C language. The author claims:

*"..the combination of C and LegOS offers you a great deal of flexibility, power and efficiency that can't be matched by the other MINDSTORMS languages."*

Chapter 7 describes how to obtain and load LegOS via a Windows or Linux based computer. Note: Appendix E of the book is devoted to "Installing LegOS" which requires Linux or a "Unix-like environment" on Windows.

The chapter describes how to use some of the functions available to the programmer via LegOS which provide an interface to the RCX hardware and also support multi-threading. The chapter concludes with a C program listing to control the robot. The program uses memory arrays to "remember" the location of the best light signal and a random response to collisions to make it less predictable.

Chapter 8 "Advanced Legos" begins with example code to read rotation sensors and a section on the sound capabilities of LegOS. There is a listing to produce the tune "Devil with a Blue Dress". There is then a brief mention of libraries available to provide floating point maths and LNP - (LegOS Network Protocol) and then some advice on LegOS debugging. This section starts with:

*Unfortunately, debugging LegOS programs is often the low point of the process of coding with LegOS.*

Most of Chapter 8 is devoted the construction and coding of a new robot. The author names this robot "Trailerbot" although "Pusherbot" might have been more apt. This sounds like a fascinating exercise. The robot is a simple vehicle which pushes a long "trailer". The programme uses artificial intelligence techniques to learn how to push the trailer without jacknifing. It would be very interesting to see this code translated into Forth.

The chapter concludes with some advice on how to go about learning how to program in C and LegOS.

### *Part Four - Homebrew Sensors by Michael Gasperi*

Michael Gasperi works as a principal engineer with the Advanced Technology division of Rockwell Automation. Although this section is principally about constructing hardware, another programming environment is introduced, Visual Basic using SPIRIT.OCX which is an RCX interface provided by Lego in their Software Development Kit.

Chapter 9 deals with passive sensors and chapter 10 with powered sensors. The construction techniques required to build sensors are covered in great detail with for example instructions on how to strip insulation from a wire and how to use a soldering iron. The author provides step by step instructions and many photographs. Even someone who has never attempted to do work of this kind before should have no problems following the instructions. For any non-Lego parts required the author gives part numbers for Radio Shack and Mouser Electronics, both American of course, but it should be easy enough to source bits from Tandy or Maplin.

The example passive sensors described are:

- Coin detector
- Non-directional touch sensor
- Temperature sensor
- Relative humidity sensor
- Angle sensor
- CdS photocell sensor
- Galvanic skin response
- Voltage input
- Battery level sensor
- Tachometer

Example code is provided to read the sensors and the chapter ends with some suggestions for other passive sensors which really amounts to "anything you can think of".

Chapter 10 deals with powered or active sensors and concentrates on developing an "opto-interrupter" sensor and a sophisticated sound sensor using two operational amplifiers.

The circuits are developed step by step with a circuit diagram and a photo of the components in a plug-in breadboard for each step. The terminology, electronic techniques and operation of the circuit are clearly explained at each stage of construction. Several pages are devoted to soldering techniques with clear illustrations showing where to place the iron etc. and what a bad joint looks like.

Voltage v. time traces are illustrated for various test points in the sound sensor to show the purpose and effect of the various components.

The chapter ends with advice and suggestions for packaging homebrew sensors.

### *Appendices*

Appendix A - Internet Resources
Appendix B - NQC API Reference
Appendix C - Frequently Used Forth Words
Appendix D - LegOS API Reference
Appendix E - Installing LegOs

### *Impressions and conclusions*

I liked this book very much. I have worked in industrial automation for several years as a software engineer and also have some knowledge of electronics. Nevertheless, I learned several things from this book which will be useful in other areas and not just for building Lego robots.

I felt that all the authors did well in explaining terms and techniques which might be new to the reader and still covering enough ground to produce something useful. The following items are examples of topics introduced in the book and given clear explanations by the authors.

| | |
|---|---|
| Dave Baum | - pulse with modulation |
| Ralph Hempel | - numbers and expressions in computing |
| | - monitoring transient switch inputs |
| | - weighted average |
| Michael Gasperi | - signal conditioning using op-amps. |

Anyone with little or no programming experience should be able to use the book to make a Lego robot perform tasks by simply following the example code. Understanding the code is a different matter. It is interesting to note that neither Dave Baum (NQC) or Luis Villa (LegOs) attempt to teach the reader how to programme in C. Ralph Hempel, however, does adopt a more tutorial approach to pbForth, see the end of this review for Ralph's views on this.

## *"I highly recommend this book"*

It would be unreasonable to expect to learn one programming language solely from a book like this let alone 3 or 4 languages plus control system programming plus the electronics skills to design sensors. However the book does give a good start with plenty of pointers to more information and help for those who wish to expand their knowledge.

In summary, I highly recommend this book to anyone interested in the Lego Mindstorms system. There is a temptation to say that all the information is on the web anyway but there is no substitute for a well indexed and illustrated book to refer to.

### *Author's Views*

Ralph Hempel kindly responded to some questions I emailed as follows:

Q. Roughly how many hours did it take to write your section of the book?
*A. It took about 4 hours a week between January and July, or about 100 hours overall.*

Q.. What feedback have you had from readers of the book?
*A. Feedback so far is positive, but still scarce. Either I did a very good job of explaining pbForth or nobody really cares....*

Q. Have you had any feedback from the Lego company?
*A. Not directly. I have contacts inside Lego that are very pleased that a simple scripting language like pbForth is available. Of course, they have their own interpreted language, but I dare say it's not nearly as fast as Forth!*

Q. How are sales of the book so far?
*A. Very good, as far as I can tell. The Christmas season saw about 7,500 books sold.*

Q. You have included much more tutorial on Forth than the other authors have on C. Did you feel readers might need more help with Forth or more persuasion to learn Forth?
*A. It's a mind-set thing. The concept of stacks and memory maps and access are very familiar to embedded systems programmers, but not the general C programming public. I found that once the idea of Forth is clearly explained, it is easy to go further. I felt a clear summary was needed, with direct application to the RCX. Most programmers don't care to learn a whole new paradigm like Forth, but those that do are better because they have a deeper bag of tricks.*

*I have a new version coming out based on Chris Jakeman's MAF. I tried to make a code generator for MAF, and almost got it working, but now I have hand-compiled (more or less) the MAF source into an intermediate script language (Tcl) that will let me generate MAF for just about any processor quickly.*

---

**Help:** Source for R65F12 processors urgently needed – see Letters.

# *Deutsche Forth-Gesellschaft*

Would you like to brush up on your German and at the same time get first-hand information about the activities of fellow Forth-ers in Germany?

Become a member of the German Forth Society for 80 DM (£28) per year (32 DM (£11) for students and retirees). Read about programs, projects, vendors and our annual conventions in the quarterly issues of *Vierte Dimension*.

For more information, please contact the German Forth Society at the e-mail address SECRETARY@ADMIN.FORTH-EV.DE

or visit http://www.forth-ev.de/
or write to

Forth-Gesellschaft e.V.
Postfach 161204
18025 Rostock
Germany

Tel.: 0381-4007872

# *JenX - A very simple XML parser*


## *Jenny Brien*


Les Kendall's article in the January issue introducing XML has inspired FIG UK members Jenny Brien and Leo Wong to explore key aspects of XML. Leo has been working on holding XML data in memory and we hope to publish details of this shortly. Jenny has worked in parallel on parsing XML files, making use of the parsing words in ANS Forth as far as possible.

Jenny presents here a sample of the work with more detail expected in the next issue. I have been able to follow these XML projects as they developed and have been impressed to see how rapidly good ideas can develop when members collaborate.

XML, like Forth, is an extensible language; new situations are dealt with by defining new tags to reflect the structure of the data in the file. Leo Wong and I have been working on a way to interpret XML tags as if they were Forth words.

It works because in any XML, the first line of the file is: `<?XML version= .... >`. We can write a word `<?XML` to parse the rest of the file and then `INCLUDE` the file. `<?XML` is designed to keep on parsing until the end of file.

`<?XML` needs a parameter to tell it where to look up and execute the tags it parses out. This is an execution variable `DO-TAG`, and the calling program provides a suitable function which takes a tag name in the form *`ca u`* and executes a suitable action. It may take the form of a string `CASE` statement, or search a wordlist as in:

```
wordlist search-wordlist IF EXECUTE THEN.
```

The parser the user sees is

```
' calling_program JenX file
```

where `JenX` is simply defined as

```
: JenX ( xt -- )  do-tag !  INCLUDE ;
```

Here is an example after Leo Wong (18 May 2001 +)

```
\ Display chapter and verse containing a New Testament word or phrase
\ New Testament from: http://www.ibiblio.org/xml/examples/religion/nt/
\
\  <chapter>
\    <chtitle>Chapter 5</chtitle>
\    <v>And seeing the multitudes, he went up into a mountain: and when he
\       was set, his disciples came unto him:</v>
\    <v>And he opened his mouth, and taught them, saying,</v>
\    <v>Blessed are the poor in spirit: for theirs is the kingdom of
\       heaven.</v>
\    <!—more verses snipped  -->
\  </chapter>

CREATE short-title 50 CHARS ALLOT
VARIABLE nChapter VARIABLE nVerse

: SEEK      \ ca1 u1 ca2 u2 -- c1 u1
            \ ca1 u1  is the word or phrase
            \ ca2 u2  is current tag name
      CASE
      S" /bktshort" $OF        \ tag </bktshort> marks end of book title
            Content short-title PLACE 0 nChapter ! ENDOF
      S" chapter" $OF          \ tag <chapter> sets nVerses to 0
            1 nChapter +! 0 nVerse ! ENDOF
      S" v" $OF                \ tag <v> marks start of verse
            1 nVerse +! ENDOF
      S" /v" $OF               \ tag </v> marks end of verse
            Content 2OVER SEARCH NIP NIP
            IF CR short-title COUNT TYPE SPACE nChapter ? nVerse ?
            [CHAR] : EMIT Content TYPE CR
            THEN ENDOF
      2DROP       \ unknown tag - drop it
      ENDCASE ;

S" forth" ' seek JenX nt.xml
```

The main design aim is that the calling program should not need to define actions for any more tags than it has to. There is no error raised if a tag is not recognised - it is simply ignored. Likewise, `<?XML` does not need to know anything about XML syntax except that tags are anything within `< >`, and that `CONTENT` is anything outside. `CONTENT` returns the `ca u` of the most recent content, held in a temporary buffer `Cbuff`. The internal word `Cbuff+` is responsible for filling it, decoding any entities as it goes. At present, it only deals with the hard-wired entities - *`<`* , *`>`* , *`&quot;`* and *`&apos;`*. A similar temporary buffer holds the whole of the current tag, including any attributes. `Get-tag` extracts the tag name.

```
: TILL \ c -- flag ca u ; parse to char c flag true if char not found
      SOURCE NIP >IN @ - >R PARSE DUP R> = ROT ROT ;
```

```
macro NEXTLINE " WHILE REFILL 0= IF EXIT THEN REPEAT "
```
\ nextline works with `TILL` to process characters until the end character is reached

```
: <?XML ( -- )
BEGIN
0 Tbuff ! BEGIN [CHAR] > till Tbuff+ nextline \ tag may be more than one line
Get-tag do-tag @ EXECUTE
0 Cbuff ! BEGIN [CHAR] < till Cbuff+ nextline \ as may content
AGAIN ;
```

### *Pitfalls*

For elements with 'mixed content' (tags mingled with text) there is a problem, as the first internal tag will empty the content buffer. It would be possible to make the tags smarter, so that instead of emptying each time, the level of `Cbuff` would only be decreased by a closing tag, which would restore it to the level it was when the opening tag was found. An easier implementation is have flags `mixed` and `/mixed` which can be added to tag actions to turn off the buffer-emptying. Other applications might call for different enhancements to `Cbuff+` , but neither Leo nor I want to add more smartness to this version which works fine for most simple files.

### *Future Directions*

In theory, `JenX` could be extended to do anything you need with XML - all you have to do is write the tag handlers! Anything, but not *everything*. Full DTD handling and validation, for example, might be very complex, but it should not be too difficult to read a DTD and pick up on any declared entities. Leo has done some work on using `JenX` to read a file into a tree structure in memory, which is a better option when you have to do a number of operations on the same file. And of course such a structure can be saved to disk too. I can see that a Forth-based XML editor would be a very nice thing to have ...

More progress, hopefully, in the next issue.

Paul Bennett
+44 (0) 1235-814586
PEB@amleth.demon.co.uk

# Three Free Forths and an OS too!

## Paul Bennett

Since the beginning of Christmas 2000 I have been working towards the goal of diminishing my reliance on Microsoft products for my work and leisure computing. After many explorations into the literature and newsgroups regarding several operating systems, I have settled on the FreeBSD package. Coming up to Easter, I have now got a feeling of impending mastery at a very rudimentary level but I realise I am still very much a learner with this Unix-like software. However, the following may help others decide for themselves.

It is not my intention to sell the idea to everyone because it will not suit everyone to change operating systems. In my work I now have a need for the benefits, such as improved security and resilience, on offer under FreeBSD. However, I am far from leaving Forth behind. In fact, I am pleasently surprised to find a number of Forth's at my finger tips for free.

### *The OS*

The Operating System, FreeBSD, is a fully POSIX-compliant operating system derived from the 4.4BSD software sources at Berkeley. The OS was made for running as 32-bit on Intel i386 or better hardware and includes versions for PC98 hardware and Alpha boxes. In fact, on Alpha boxes, it is a 64-bit OS.

Reasons for selecting FreeBSD as an operating system are usually for such aspects as security, resilience and dependability. FreeBSD is made to network and comes with the full complement of TCP/IP protocols up to and including IPv6. FreeBSD is the choice of many ISP's to run Internet services for their clients.

While FreeBSD is mainly CLI-based, the package usually provides for running X-servers, Web-servers and ftp-servers and you can select from a range of GUIs. The code is not overly bloated and seems to run at very respectable levels of performance on most hardware. It can give a new lease of life to an old 386 box and, with some work, you could easily run your own Intranet. The other nice thing is that, with the BSD licencing scheme, the OS is quite inexpensive.

A word of warning for those who are tempted to dump Microsoft Windows for a change to FreeBSD. The change is no picnic as you will have to settle down and get to know your hardware quite well to be able to properly configure the system to work for you. It will involve a great deal of reading and study but the results are well worth it.

At time of writing, the current release of FreeBSD is 4.2-RELEASE and is available as a 4-CD set or is downloadable from the web-site (see the on-line handbook for details).

### *FreeBSD Resources*

FreeBSD web-site at http://www.freebsd.org/
There are also plenty of mirror sites for this so, if international communications are slow you could try one of the mirror sites closer to your own country.

PDSL (Public Domain Software Library):

PO Box 131
Trowborough
East Sussex
TN6 1WS
Tel: 01892-663298

Book:
"The Complete FreeBSD - The operating system of choice for serious internet users" by Gregg Lehey published by Walnut Creek CDROM, ISBN 1-57176-246-9 - comes with four CD's and gives an overview of the whole OS (see http://www.cdrom.com ).

### *The free Forths*

The first of these is FreeBSD's "bootstrap loader" - the first program to run. It's job is to load either the operating system or hardware diagnostics or some other utility.

```
**************************************************************************
FreeBSD/i386 bootstrap loader revision 0.8
(jkh@bento.FreeBSD.org, Mon Nov 20 11:41:23 GMT 2000)
Loading /boot/defaults/loader.conf
/kernel text=0x26e45f data=0x31b18+0x21440 syms=[0x4+0x36b90+0x4+0x3b7bc]
|
Hit [Enter] to boot immediately, or any other key for command prompt.
Booting [kernel] in 9 seconds....

Type '?' for a list of commands, 'help' for more detailed help.
ok
**************************************************************************
```

FreeBSD uses FICL or "Forth Inspired Command Language" as its bootstrap loader. FICL is actually three programs. The initial boot sector programme itself, a second program which the first reads in and runs, then the full FICL loader suite and any other Forth-based files called by the script file loader.conf. If the FICL environment is not entered (by interrupting during the 10 second time-out) it will automatically load the FreeBSD kernel and run that.

FICL provides an environment for pre-kernel loading operations, selection of different environments and loading the kernel. It is an ANS-compliant Forth similar (but not the same as) Open Firmware.

Written in C it is probably not the best example of Forth style but has merit in that it will run a number of Forth programs easily enough (one of the first things I tried).

Perhaps it might be worthwhile for a small group of Forth people to get involved in documenting this environment properly, produce a raw-Forth version and place that

in the FreeBSD domain. Perhaps we might do a tidyer and more compact job of it and stand a chance of getting it into the BIOS ROM for later PC's. Now there is a prospect to ponder; Forth on every PC and Workstation by default.

### *PFE - The Portable Forth Environment*

```
****************************************************************
\ Portable Forth Environment 0.29.0 (Nov 17 2000 07:02:53+00)
Copyright (C) Dirk Uwe Zoller 1993 - 1995.
Copyright (C) Tektronix, Inc. 1998 - 2000.
****************************************************************
```

The Portable Forth Environment (PFE) is based on the ANSI Standard for Forth. The PFE has been created by Dirk-Uwe Zoller and had been maintained up to the 0.9.x versions (1993-1995). Tektronix has adopted the PFE package in 1998 and made a number of extensions, in particular the PFE is now fully multi-threaded, so that a number of Forth Interpreters can run in the same address space. Due these changes, the PFE is mostly incompatible on the C-source level with Dirk-Uwe Zoller's implementation somewhere from the 0.20.x versions on. The Forth-level is not affected though.

Another addition is the dynamic loading mechanism that allows the inclusion of extended functionality into the running Portable Forth Environment. These extension modules are written in the C language to obtain maximum speed as far as the processor's capabilities can be exploited by modern compiler technology.

The Forth Interpreter of the PFE is itself fully written in C, so it is very easy to port to new platforms, especially embedded processors. Forth-sources can be easily rewritten in C using the implementations of PFE so that time-critical sections can run at full processor speed. Even more, any external C-object functionality can be made available to the outer Forth Interpreter merily by providing a word-set export-table that is loaded by the PFE.

The loading mechanisms does allow more than one word-set table to be compiled into a single module-object which is then loaded at once. The PFE does itself consist of a number of such basic word-sets which often bear names as proposed by the ANSI Forth standard. Some of the PFE word-sets do not necessarily need to be initially compiled into the base PFE object due to configure-time options. In most cases, the raw PFE does now not pre-load all the words as documented here, in particular the floating-point word-set extensions are not used on many embedded platforms

### *Usage Information*

The PFE can be compiled as a binary that accepts a set of options. Starting it from a shell should prompt you with the Forth's outer interpreter that you can talk to. It also documents the set of ambiguous conditions which the ANS standard requires.

In use on my FreeBSD system, PFE works as another shell with the full range of command line recall and editing facilities available. Several word-sets are visible at the top level:

```
W EXTENSIONS              W FORTH                   W LOADED
```

The wordsets are quite extensive and too long to list here. There appears to be a hierarchy under the LOADED list which lists:-

```
p wordset:PFE-SMART       p <<load_signals>>        p wordset:PFE-SIG
p wordset:PFE-TERM        p wordset:PFE-SHELL       p wordset:PFE-SYSTEM
p wordset:PFE-DEBUG       p wordset:PFE-MISC        p wordset:FORTH-83-L&P
p wordset:FORTH-83        p wordset:STRING-EXT      p wordset:TOOLS-EXT
p wordset:MEMORY-ALLOC-EXT                          p wordset:LOCALS-EXT
p wordset:FILE-EXT        p wordset:FACILITY-EXT    p wordset:EXCEPTION-EXT
p wordset:DOUBLE-EXT      p wordset:BLOCK-EXT       p wordset:CORE-EXT
p wordset:SEARCH-ORDER-EXT ok
```

### *PFE License*

PFE is free software. You can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. The HTML-version of the LGPL-License can also be obtained from http://www.OpenSource.org. As with all free software, there are no warranties for use cases where the copyright holders have not given an explicit liability.

A later version (0.30.x) of PFE announced by Guido Draheim is available from the net at http://pfe.sourceforge.net/.

### *GForth*

```
**********************************************************************
GForth 0.5.0, Copyright (C) 1995-2000 Free Software Foundation, Inc.
GForth comes with ABSOLUTELY NO WARRANTY; for details type `license'
Type `bye' to exit
**********************************************************************
```

GForth by Anton Ertl and others is another 32-bit ANS Forth and comes with several files of useful code examples.

Gforth is a fast implementation of the ANS Forth language. It works nicely with the Emacs editor, offers some features such as input completion and history and a powerful locals facility, and (thanks in part to FIG UK member Neal Crook), it even has a manual. GForth employs traditional implementation techniques; its inner interpreter is indirect or direct threaded. (There is a great deal of cleverness involved in making it both portable and fast – Ed.) GForth is provided under the GNU licence terms which may be a bit restrictive for some commercial applications but well suited for educational and home use.

Within the extensive number files that come with the program are many useful utilities that could help to build a very nice integrated development environment. There is even a glossary generator package that would be a boon to those who include "\G" commenting along with their source text. This utility alone is worth considering the download.

You can also find GForth at http://www.gnu.org/software/gforth/gforth.html

# Letters

The Magazine Team are always pleased to get feedback and encouragement. Here we have Paul Bennett replying to a comment from Andrew Holt, a request for help from John Matthews and a project suggestion from Doug Neale.

**Paul Bennett**

From: PEB@amleth.demon.co.uk
Sent: 09 April 2001 15:26

Having just received Forthwrite April 2001 (issue 111) I noted the letter from Andrew Holt. He quoted "the best line of code is the one that somebody else writes for you". This may be true some of the time but is not to be regarded as a general rule.

Other peoples work may prove to produce the best line of code only after you have subjected it to tests against your requirements (including integrity and performance targets). Otherwise, if everyone was using other peoples code, where would the new material come from?

Forth has a definite role to play in many areas of computing systems technology. That it can, with some effort, fulfil any role required of it, is a measure of the soundness of the underlying philosophy. Some of what Forth has achieved is only now being seen in other areas of programming.

Taking a look under the covers of some of the latest OS's you may be surprised to see Forth or Forth-like techniques being used in plenty. Forth will be my environment of choice as I am certain to be able to apply full system certification with fully evaluated risks assessments that take into account both the hardware and software aspects. Forth has a very firm place where integrity of the system matters and you need to prove it.

**********************************************************************
Paul E. Bennett .....<email://peb@amleth.demon.co.uk>
Forth based HIDECS Consultancy .....<http://www.amleth.demon.co.uk/>
Mob: +44 (0)7811-639972 .........NOW AVAILABLE:- HIDECS COURSE......
Tel: +44 (0)1235-814586 .... see http://www.feabhas.com for details.
Going Forth Safely ..... EBA. www.electric-boat-association.org.uk..
**********************************************************************

**John Matthews**

From: jjm@aems.demon.co.uk
Sent: 16 May 2001

15 years ago I designed a controller for friend of mine to operate a travelling microscope. The controller was based on the Rockwell R65F12 processor, which is a 6502 that incorporates a Forth kernel in its internal ROM (the program is stored in external EPROM and extra RAM is used for data). The R65F12 came in a 64-pin DIL package (called a QUIP) and ran at 1MHz using a 2MHz crystal.

This IC is no longer supported by Rockwell and my friend has redeveloped the application around another processor. Meanwhile he has been using up old stock. Unfortunately, the replacement is not quite yet in manufacture (I had nothing to do with it! - I am retired) and he has received a sudden batch of orders that he needs to fulfil. He wants to know if anyone is likely to have or know where to obtain old stock of the R65F12 processor. He has already scoured commercial sources as far away as Australia so it would be from someone who may have replaced it in time but not yet thrown away the old stock.

As an aside, the Director of the Imperial Research Fund, Sir Paul Nurse, said 12 months ago that this microscope has speeded up genetic research (e.g. in the understanding of cancer) by a factor of 3.

Do you know anyone that might be able to help?

Regards

John Matthews

**Doug Neale**

From: dneale@w58wmorden.demon.co.uk
Sent: 28 April 2001 17:44

Here is an outline of a possible Forth project that might interest some of our members. First of all let me state that I would have a vested interest in its outcome: it would solve what is beginning to look like a very difficult problem for me.

As you know I am a freelance C and Foxpro programmer for both Mac and Windows platforms. Currently the Foxpro system gives us a pseudo-SQL access to any X-Base database which uses the 'cdx' index file format. Thus one can put the database files on any server Novell, NT, or UNIX and not actually require any server based software to be running.

All the 'front-end' stuff is done in Foxpro with all the usual 'GUI' screens and support stuff, but the data is extracted from the database by my Quark Xtension which is written in C using a third party package called 'CodeBase'. Microsoft have continued supplying Foxpro within Visual Studio for the windows platform, but the Mac version has been stuck on Visual Foxpro 3 for quite a few years. Moreover, the Codebase library does not support SQL on the Mac, only on the PC. For these and other reasons I would like to move away from Foxpro and similar Microsoft products.

What I would like to know is: is it feasible to write an SQL package in Forth which initially would be running on a Windows NT/2000 server. One would then require a Forth client system running on a workstation for both the front end work and the data extraction. The latter component would need to be a version of the Forth client package written in C and implemented inside the Quark Xtension. For general information: Quark Xtensions are to all intents and purposes specialist DLL's.

As well as solving my problem, the project could yield a general purpose client/server SQL system written in Forth initially for the windows platform, but potentially portable to other environments. Perhaps one could call it 'FQL'.

I do not know if we have enough 'know-how' within our membership or whether members would be sufficiently interested to participate in such a project. Perhaps if you could publish this note in the next issue we could discuss the response (or lack of) at the next AGM and decide how it could be project managed.

Regards,
Douglas Neale

## FIG UK Committee

| | | |
|---|---|---|
| ***Chairman*** | **Jeremy Fowell**, | 11 Hitches Lane, EDGEBASTON B15 2LS<br>0121 440 1809 jeremy.fowell@btinternet.com |
| ***Secretary*** | **Doug Neale,** | 58 Woodland Way, MORDEN SM4 4DS<br>020 8542 2747 dneale@w58wmorden.demon.co.uk |
| ***Editor*** | **Chris Jakeman,** | 50 Grimshaw Road, PETERBOROUGH PE1 4ET<br>01733 753489 cjakeman@bigfoot.com |
| ***Treasurer*** | **Neville Joseph,** | Marlowe House, Hale Road, WENDOVER HP22 6NE<br>01296 62 3167 naj@najoseph.demon.co.uk |
| ***Webmaster*** | **Jenny Brien,** | Windy Hill, Drumkeen, BALLINAMALLARD,<br>Co. Fermanagh BT94 2HJ<br>02866 388 253 jennybrien@bmallard.swinternet.co.uk |
| ***Librarian*** | **Graeme Dunbar** | Electrical Engineering, The Robert Gordon University,<br>Schoolhill, ABERDEEN AB10 1FR<br>01651 882207 g.r.a.dunbar@rgu.ac.uk |

Membership enquiries, renewals and changes of address to Doug.
Technical enquiries and anything for publication to Chris.
Borrowing requests for books, magazines and proceedings to Graeme.

## FIG UK Web Site

For indexes to Forthwrite, the FIG UK Library and much more, see **`http://www.fig-uk.org`**

## FIG UK Membership

Payment entitles you to 6 issues of Forthwrite magazine and our membership services for that period (about a year). Fees are:

| | |
|---|---|
| National and international | £12 |
| International served by airmail | £22 |
| Corporate | £36 (3 copies of each issue) |

## Forthwrite Deliveries

Your membership number appears on your envelope label. Please quote it in correspondence to us. Look out for the message "SUBS NOW DUE" on your sixth and last issue and please complete the renewal form enclosed.
Overseas members can opt to pay the higher price for airmail delivery.

## Copyright

## FIG UK Services to Members

**Magazine** Forthwrite is our regular magazine, which has been in publication for over 100 issues. Most of the contributions come from our own members and Chris Jakeman, the Editor, is always ready to assist new authors wishing to share their experiences of the Forth world.

**Library** Our library provides a service unmatched by any other FIG chapter. Not only are all the major books available, but also conference proceedings, back-issues of Forthwrite and also of the magazine of International FIG, Forth Dimensions. The price of a loan is simply the cost of postage out and back.

**Web Site** Jenny Brien maintains our web site at http://www.fig-uk.org. She publishes details of FIG UK projects, a regularly-updated Forth News report, indexes to the Forthwrite magazine and the library as well as specialist contributions such as "Build Your Own Forth" and links to other sites. Don't forget to check out the "FIG UK Hall of Fame".

**IRC** Software for accessing Internet Relay Chat is free and easy to use. FIG UK members (and a few others too) get together on the #FIG UK channel every month. Check Forthwrite for details.

**Members** The members are our greatest asset. If you have a problem, don't struggle in silence - someone will always be able to help. Do consider joining one of our joint projects. Undertaken by informal groups of members, these are very successful and an excellent way to gain both experience and good friends.

**Beyond the UK** FIG UK has links with International FIG, the German Forth-Gesellschaft and the Dutch Forth Users Group. Some of our members have multiple memberships and we report progress and special events. FIG UK has attracted a core of overseas members; please ask if you want an accelerated postal delivery for your Forthwrite.