ISSN 0265-5195

# Forthwrite FIGUK

Issue 108 August 2000

## FIG UK Committee


| | | |
|---|---|---|
| ***Chair*** | **Chris Hainsworth,** | Microplex Ltd., 5a Riverfield Road, STAINES TW18 2EE<br>01784 457565 chris.hainsworth@dial.pipex.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*** | **Keith Matthews,** | 20 Spindlebury, CULLOMPTON EX15 1SY<br>01884 34818 |
| ***Webmaster*** | **Jenny Brien,** | Windy Hill, Drumkeen, BALLINAMALLARD,<br>Co Fermanagh BT94 2HJ<br>02866 388 253 jennybrien@bmallard.swinternet.co.uk |
| ***Librarian*** | **Sylvia Hainsworth,** | Microplex Ltd., 5a Riverfield Road, STAINES<br>01784 457565 sylvia.hainsworth@dial.pipex.com |


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 Sylvia.

## FIG UK Web Site

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

## FIG UK Membership

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

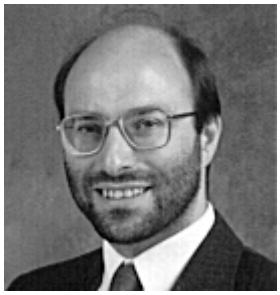
| | |
|---|---|
| National and international | £12 |
| International served by airmail | £22 |
| Corporate | £36 (3 copies of each issue) |

## 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



# Editorial

The big news this time is that Jeremy has launched our F11-UK, the controller board for our FIG UK Hardware Project. Already kits have been delivered, assembled and tested and we hope that this will lead to a number of projects about control which are documented and published on the web site. Jeff Penn intends to report his experiences in building the kit in the next Forthwrite.

Paul Bennett returns with some more Forth as used in industry and Bernd Paysan has graciously allowed us to reprint a paper on using Forth to serve up web pages.

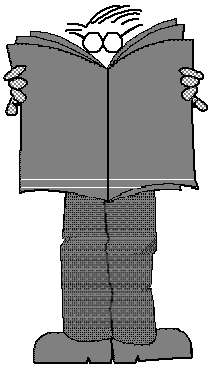
The on-line publication in June of the April Forthwrite has brought us readers and feedback from the USA and Russia and we plan to continue this experiment. An amazing 614 people have visited the download page so far.

Don't forget the monthly IRC session. Our next one is Saturday 2nd September on channel #FIGUK from 9:00pm.

Welcome to some more new members – Graham Bowler and Alex Holden, both interested in embedded systems and Robin Francis whose letter is reprinted in this issue. As ever, I strongly recommend that new members raid our Library to borrow books and back-numbers of Forthwrite and the US Forth Dimensions. It's the biggest and best Forth library anywhere.

Until next time, keep on Forthing,

Chris Jakeman



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

# *Forth News*

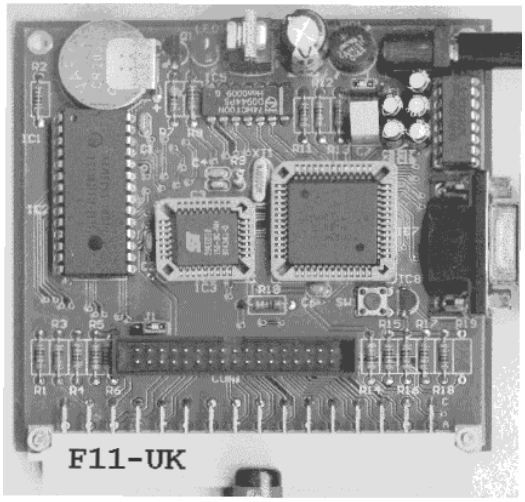
## Feedback

Feedback is always welcome and we have received helpful comments from Marcel Hendrix and Michael Gassanenko - thanks.

## FIG UK News

F11-UK

Top item has to be the release of F11-UK kit – this photo shows the assembled board.




"F11-UK, the long-awaited controller board using an adapted version of Pygmy, has been released by FIG UK member Jeremy Fowell. Special price to members is £47 plus postage but a Pygmy Forth and $20 licence are also required (download from
www.eskimo.com/~pygmy
Already several members have now assembled the board and already have Pygmy running.

F11-UK provides a multi-tasking Forth which can be programmed from your PC, 32K of RAM and 32K of FLASH PROM and a wide range of features such as UART and 8-bit A to D convertor.

For full details sign up to the mailing list by sending the message:
`"subscribe fig-forth-uk"`
to maiser@eee.rgu.ac.uk.

Web Site

Our Webmaster has been busy re-vamping the FIG UK web-site. The aim is to provide a more polished look whilst ensuring continued access for those with older browsers and low bandwidth connections.

The re-vamp coincides with newly downloadable material, including Forthwrite which is in PDF and one issue behind the paper copy posted to members.

The download pages have received over 600 visitors.

AGM

This annual meeting is open to all members on Sat 23rd Sept.. Doug Neale has offered his hospitality once more - see Aug Forthwrite for details.

FIG UK and Forthwrite magazine are now listed in the on-line directory of computer industry resources at

www.iversonsoftware.com/tabularium.htm

under the category Languages and Tools.

## UK News

The date and venue for the euroFORTH 2000 Conference have been changed to avoid the season of higher air-fares.The venue and date are now Winchester, 3-5 Nov., though this has still to be confirmed.

New Products from MPE

See the MPE web site

www.mpeltd.demon.co.uk

for details of the following new products and free ProForth downloads.

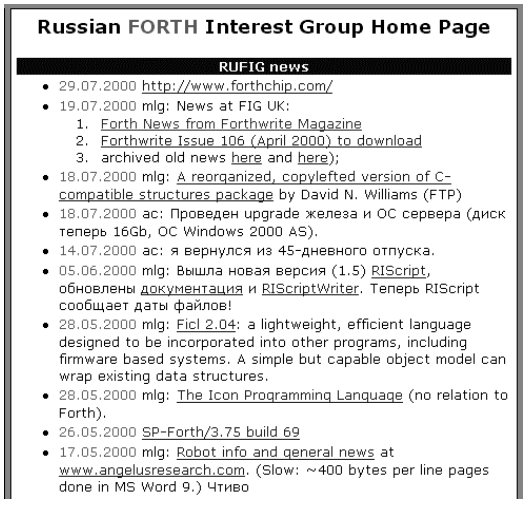
1. Forth Stamp 51 - 89S8252 board plus Forth compiler, assembler, disassembler, ISP programming, interactive compilation and testing. Yours for only US $99, UK pounds 63.75, Euro 105.

2. Windows Forth development system is now at version 3.1. ProForth VFX for Windows features a completely new Forth kernel written to the ANS Forth standard.

3. The Windows-based Forth 6.1 Cross Compilers are now available, providing an interactive development system for embedded targets.

## International News

Forth News from FIG UK is no longer the only archive of UK and international news about Forth.



**Russian FORTH Interest Group Home Page**

**RUFIG news**

- 29.07.2000 http://www.forthchip.com/
- 19.07.2000 mlg: News at FIG UK:
  1. Forth News from Forthwrite Magazine
  2. Forthwrite Issue 106 (April 2000) to download
  3. archived old news here and here);
- 18.07.2000 mlg: A reorganized, copylefted version of C-compatible structures package by David N. Williams (FTP)
- 18.07.2000 ac: Проведен upgrade железа и ОС сервера (диск теперь 16Gb, ОС Windows 2000 AS).
- 14.07.2000 ac: я вернулся из 45-дневного отпуска.
- 05.06.2000 mlg: Вышла новая версия (1.5) RIScript, обновлены документация и RIScriptWriter. Теперь RIScript сообщает даты файлов!
- 28.05.2000 mlg: Ficl 2.04: a lightweight, efficient language designed to be incorporated into other programs, including firmware based systems. A simple but capable object model can wrap existing data structures.
- 28.05.2000 mlg: The Icon Programming Language (no relation to Forth).
- 26.05.2000 SP-Forth/3.75 build 69
- 17.05.2000 mlg: Robot info and general news at www.angelusresearch.com. (Slow: ~400 bytes per line pages done in MS Word 9.) Чтиво

RUFIG, the Russian FIG, has a news archive too with a mix of items in English and Russian at

www.forth.org.ru

"The Unofficial Guide to LEGO MINDSTORMS Robots" by Jonathan Knudsen provides 5 construction projects and 3 "alternative programming environments like NQC, pbFORTH and legOS to develop powerful software for your robots." Ralph Hempel's pbFORTH gets a whole chapter to itself. The book, ISBN 1-56592-692-7, is published by O'Reilly at £16.60.

FIG International

FIG International is being re-organised following the resignation of President Skip Carter and Editor Marlin Ouverson. Please send offers of advice and assistance to board@forth.org.

Board member Elizabeth Rather posted to comp.lang.forth: "If FIG is important to you, now is the time to step forward and see what you can contribute, not only in terms of money, but also logistical support and effort to make things happen."

## Systems

Dave Pochin has revamped the "Getting Started" web site at www.sunterr.demon.co.uk/guide.htm

The guide has now received over 6,000 visitors!

Australian company Colour Vision Systems have developed a Forth OS with TCP/IP capability and are now making this freely available. It makes good use of object-oriented Forth and is used for industrial work. For details, see http://forth.cx

SMAL32 is a DOS based, 32 bit, direct-threaded forth compiler/ interpreter for developing programs in the DOS environment. A collaborative project between English speaker Bruce Hoyt hoyt@voyager.co.nz and Russian speaker Alexandr Larionov laric@forth.ru

"The real power of this system lies in the compiler. SMAL32 can produce very small compiled 32 bit DOS programs containing only the Forth words actually used by the application program plus a 9Kb compressed DPMI interface and loader. The compiler produces code with or without headers."

## Forth on the WWW

The WebForth and eForth for Java pages have been moved to:
webdev.amsystech.com/mlosh/
and
webdev.amsystech.com/mlosh/webforth.htm

At the Silicon Valley Forth Interest Group meeting in February a group led by John Peters japeters@pacbell.net got together for the purpose of developing a web page that will act as a command-line Forth. This would permit anyone to try out Forth online and to collaborate on a project. See:

www.forth.org/svfig/online.html

Editor's Note: The first project is a client-side Forth executing on the client's computer source code from the host's HTML pages. The second is a server-side Forth executing on the host computer source code from the client.

StrongForth is a static type-checked Forth with operator overloading created by Dr. Stephan Becher who hopes to have a web site running soon but meanwhile you can email him to request a copy at:

s.becher@get2net.dk

Derrick Shearer reports on comp.lang.forth that a Forth-based MUD system called MUF is available at

www.furry.com/telzey/fuzzball/muftutor.htm

"MUD" stands for Multi-User Dungeon game (or game-building tool).

Support Material

---

The ForthChip.com website is now open at

www.forthchip.com

The site provides a range of articles concerning M.I.S.C. (Minimal Instruction Set Computer) Architecture, and how it is involved with Forth. Much of the content is provided courtesy of Jeff Fox at UltraTechnology.

For a ANS Forth tutorial on the web try Anton Ertl's work at:

www.complang.tuwien.ac.at/anton/tmp/gforth.html#Tutorial

David Williams of the University of Michigan has made available "a reorganized, copy-lefted version of my C-compatible structures". The package is available at

ftp://feynman.physics.lsa.umich.edu/pub/williams/forth/cstructures/

0121 440 1809
jeremy.fowell@btinternet.com

# Launch of F11-UK FIG Hardware Project

Jeremy officially launched this long-awaited project on the 30th June with an e-mail to the Mailing List. There was a lot of enthusiasm shown at the monthly IRC session and responses have started to arrive.

John Tasgal was one of the first to assemble a board, reporting: "I've just finished putting together Jeremy's excellent F11-UK kit. The PCB is of very high quality and easy to solder."

Paul de Bak, one of our Swedish members, was also quick with the soldering iron, "I've now finished assembling the F11-UK board and downloading Jeremy's code to it. Both the assembly of the board and the downloading of the code went smoothly, apart from one problem I had with downloading (due to Windows: Ed.).

Of course, no product is perfect first time around and the Mailing List now carries some useful feedback messages intended to make every installation go smoothly.

Mike Trueblood has resumed work on the Clock Challenge using his kit and I'm sure we'll hear some more about that soon.

Jeremy has several ideas for applying the F11-UK, saying, "As a thank you for your support I am including a free thermistor with every kit, which can be connected very easily to one of the HC11 A/D inputs to measure temperature. Microchip have an application note on the subject, AN685, which I assume is available on their web site.

Pressure sensors are down to around £10 now and I have also requested a sample of the new Analog Devices accelerometer, ADXL202.

There could be a lot of interesting projects waiting out there . . ."

# 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

01733 753489
c.jakeman@bigfoot.com

# *euroFORTH*

## *Chris Jakeman*

As preparations take place for euroFORTH 2000 in Winchester, we take a look at last year's event.

EuroFORTH '99 took place at St.Petersburg which allowed a strong input from Eastern Europe, where Forth has always appealed more than resource-hungry alternatives. A personal and fascinating report by Reuben Thomas can be found on the euroFORTH site at

http://dec.bournemouth.ac.uk/forth/euro/ef99/report.html

This year, all the papers are published as PDFs at the site above (thanks to Peter Knaggs). For those without Internet access, the papers have been downloaded, printed, bound in a suitable form for photocopying[1] and are available for borrowing from the FIG UK library.

### *Thinking about Forth*

- *Is Forth Code Compact? A Case Study*, by M. Anton Ertl
- *Threaded Code Execution and Return Address Manipulations from the Lambda Calculus Viewpoint*, by M. L. Gassanenko
- *Dynamically Structured Codes*, by M. L. Gassanenko
- *Perspective MetaFORTHness*, by Mikhail Kolodin.

Anton Ertl tackles the issue of comparing Forth with other languages. This is a thorny issue, as a perfect comparison would require matching developments by equally competent teams. Instead, Anton looks at a number of *parser generators* in Forth and 7 other languages and makes some quantitative comparisons.

He finds that the Forth implementations require much less source code and identifies a number of restrictions that Forth doesn't have, which give Forth users a real advantage for parser design.

---

[1] The copyright remains with International FIG who permit the copying of individual articles for personal research.

Michael Gassanenko presented two substantial papers, the first building on work by member Bill Stoddart among others. Previously Michael had developed a scheme for reasoning about the effect of executing a Forth word. The new paper allows a compiler or program verifier to reason about a Forth word including return stack manipulations.

The "dynamically structured codes" in his second paper refer to code that is generated and executed on the fly. This is an unusual technique which offers benefits in reducing the complexity of software. It is closely related to Gordon Charlton's FoSM (see Forthwrite Feb, Aug & Nov 97).

"Perspective MetaFORTHness" by Mikhail Kolodin is a brief paper discussing the ability that modern Forths often lack – the ability to write Forth totally in Forth.

### *Firmware Development*

- *sTTAck: Stack Transport Triggered Architecture*, by Aliaksei V. Chapyzhenka.
- *OpenBoot Dropin Modules*, by Michael Milendorf.
- *Interrupt mechanism for threaded code interpreter*, by Alexey A. Burtsev
- *Assemblers for firmware systems*, by Dr. Sergei A. Sidorov.
- *Machine Forth for the ARM processor*, by Reuben Thomas
- *The TpForth project*, by Reuben Thomas.

Aliakesi Chapyzhenka's paper reports the design of a Forth processor using TTA, an architecture that allows more optimisation opportunities even than RISC and is clearly described by Dick Pountain in Byte Feb. '95. Sadly, Chapyzhenka fails to explain why he has chosen Forth as the language to implement for his TTA design although the stack is clearly central to his work.

Michael Milendorf works for Sun in the USA and his paper discusses the history and development of OpenBoot DropIn technology. Firmware is the ROM-based software that controls a computer before the operating system takes over and OpenBoot is Forth-based. "The implementation of DropIn technology is elegant and simple and shows the advantages Sun Microsystems Inc gains by using OpenBoot for its boot firmware."

Alexey Burtsev describes an interrupt mechanism that has many advantages over the traditional one, because the interrupt routine is not taken until the threaded interpreter reaches NEXT. A cunning arrangement minimises the overhead required to achieve this.

The interrupt routine may be either assembler or high-level source which is a key advantage for Forth. I wonder how this paper compares with "Zero-Overhead Forth Interrupts" by G.Wilson in Forth Dimensions July '94.

Sergei Sidorov describes a portable assembler as a useful but unusual extension to a machine-level debugger. The assembler has been used with 6 different instruction sets.

Reuben Thomas offered two papers, as well as the personal report already mentioned. The first describes a port of Chuck Moore's Machine Forth to the ARM processor. He writes "MF's judicious mixture of novelty and classic simplicity merits careful study, though I for one will not be abandoning the traditional combination of Forth and assembler in its favour."

Reuben also describes the development and current status of the TpForth Project, which delivers a Windows-based IDE for developing Forth to run on a target platform (currently 3 processors are supported). Unusually, the system is "available under a novel 'community licence' which aims to build a community of users while protecting commercial interests." Five Forth companies are already participating and private use is free.

### *Standards*

- *Forth in Russia: present state and standardization efforts*, by Mikhail Kolodin.
- *ANS Forth Internationalisation proposal* (sixth revision), by Stephen Pelc, Willem Botha, Nick Nelson, and Peter Knaggs.
- *ANS Forth and large characters*, by Stephen Pelc, Steve Coul and Peter Knaggs.

Mikhail Kolodin gives an overview of Forth as it is used in Russia, where its frugal use of resources has made it popular. He includes a list of books and home-grown systems (see Smal32Forth in Forth News) and mentions the commercial intranet server E-Serv.

ANS Forth has been influential and a collaborative project is under way to deliver a Russian translation.

Mikhail notes that an official ANS document costs about a year's salary!

Stephen Pelc, Steve Coul and Peter Knaggs presented two papers on using Forth around the world. The first proposes an optional LOCALE word set which covers language, font, date/time formatting etc..

They identify 3 different characters sets – that Development Character Set (DCS) assumed never to change, the Operating Character Set (OCS) of the underlying OS and the Application Character Set (ACS). In many parts of the world, these are unlikely to be the same.

An interesting proposal is `L"`, a word which converts text in DCS into a string identifier suitable for ACS. This enables the source to contain DCS text information (easier to maintain than identifiers) and yet deliver ACS text.

Another aspect of the proposal is the use of macros to embed times, dates etc. formatted according to the current locale within a text string.

The second paper proposes a scheme for Forth to operate in an environment with 16-bit Unicode or even multi-byte characters.

In the Internet environment, these considerations become important (our own WebForth team have tackled 4 locales) and I, for one, am grateful that others are willing to tackle these thorny issues so that I won't need to.

---

# *Diary Date*

## *euroFORTH 2000* 3-5 Nov. in Winchester

To avoid travelling (especially by air) at an expensive time of year, this event has been **postponed** and a **new venue** found. The date has still to be confirmed. Look for details closer to the event on

http://dec.bournemouth.ac.uk/forth/euro/ef00.html

01235-814586
peb@amleth.demon.co.uk
www.amleth.demon.co.uk
www.feabhas.com

# Logging On Statistically Speaking

## Paul Bennett

Another contribution from Paul which reveals the techniques used to apply Forth in an industrial environment.

### Data Logging and Statistics

In working with production machinery it is often required to monitor and record the number of times and duration of particular machine states. The data so obtained may be used to provide information to management information systems. In this article, which is based on work I did some years back, I am only going to deal with logical states.

The collected information can be displayed on local screens or communicated to a central host computer where a number of similar machine states can be monitored. It is useful in the distributed control systems used in factories and process plant.

NOTE:- *I have not necessarily included all the code to provide an immediately running demonstration but what is included, I hope, explains the technique well enough. The code examples included are extracted from the files of a real application and altered sufficiently to hide the client identity. It therefore is probably a suitable follow-on to the "Real World" series of articles I published earlier.*

### Scenario

A PLC[2]-type machine controller has access to machine states via limit switches which are represented by single bits within a block of memory. For this example we shall use a 64-bit array of states which we wish to track. A 16-bit processor is also assumed.

---

[2] Programmable Logic Controller – the PLC was initially developed as an electronic replacement for relays and timers. Typically supporting only low-level programming, its flexibility and reliability make it the most common system for controlling machinery.

## *Logging Facilities*

### Current State Table

This provides the input states to the logging process. If the states of interest are spread about the I/O system, it would be well to gather them into a coherent block of states so that they might be managed more easily. This can often save processing time despite the period spent copying states to the `CURRENT_STATES` table.

| Bit | | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| State Ident | W0 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| | W1 | 32 | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 |
| | W2 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 |
| | W3 | 64 | 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 |

**Figure 1. Map of States (4 cells)**

### Last State History

This is a copy of the state table to keep track of what the state was last time around and to ensure that changes of state can be detected. This table is identified as `PREVIOUS_STATES` and is a full copy of the `CURRENT_STATES` table.

### Count of Events

An event is defined as the transition from false state to true state of the bit of interest. The count of events is the number of false to true transitions that have occurred.

### Accumulated Event Duration

Number of centi-seconds the event state has been at the true value.

## *Memory Space Requirements*

Naturally, the amount of memory space required to provide the logging facilities is greater than the simple collection of the state itself, but the added value is that we now have some statistical information about the states we are monitoring.

The memory requirements listed above add up to

= 2N(1+(1/W)) cells

where N is the number of bit-states to be monitored and

W is the cell width in bits.

As we have seen, in order to keep a historical track (for one cycle anyway) of states, a second similar sized array is required. We then require a variable for each state in the table to hold the count of transitions we notice. Finally, there are the count and timing storage cells where the logged data of interest will be recorded.

The memory map begins to look something like the diagram here.

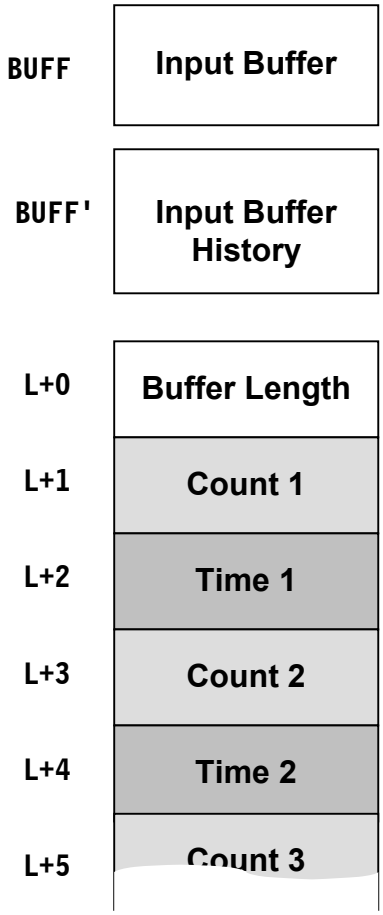



**Figure 2. Memory Block Allocation**

We do not want to have to spend a lot of time creating this sort of array in a painstaking piecemeal fashion and then have to keep in mind its structure when we wish to access it.

The areas BUFF and BUFF' are created normally with CREATE and ALLOT to make the appropriate amount of space available. If you need range-checking on these buffers you can employ it in a similar fashion to that in LOGGED (shown below). This is where a little help from CREATE and DOES> comes in handy to generate an active array that will, given an index, return the appropriate address to access the array directly. We may then manipulate the data at the address as appropriate. The word LOGGED defined below will help create several named arrays of this sort so we may have more than one array that behaves this way.

To ensure we have an address in returned in range, some calculations are performed to range-check the array prior to returning the address of a cell. Each array created with LOGGED returns two valid cell addresses which relate to the count and timer values respectively.

```
\ Logging Task - Active Data Arrays PEB 01/07/93

       : RESERVE 0 DO 0 C, LOOP ; \ Allocate some cleared
                                  \ memory space

       : LOGGED CREATE (S n --- ) \ Compiling
         DUP , 1+ CELLS CELL+ 2* RESERVE

         DOES> (S n --- addr1\addr2 ) \ Interpreting
         DUP CELL+ >R >R
         CELLS R> @ CELLS
         UMIN 2* R> + DUP CELL+ ;
       (G Compiling: Creates an active double stream data array
         for storage of n count and time items.

         Interpreting: Return the addresses of a counter {addr2}
         and timer {addr1} for the data items indexed by n to be
         accessed. )
```

**Listing 1. Active Data Array Creation `LOGGED`**

Editor's Note:- *Paul uses the comment words* `(S` *and* `(G` *which are identical to* `(` *but allow his documentation tools to extract stack comments and glossary entries out of the code (an idea more recently appearing in Sun's JavaDoc tool)*

.

Having created the arrays, we need to be able to access and manipulate the data in the cells. We must first deal with the input data. At the beginning of each cycle, it has to be collected from the state inputs, checked for validity and then placed in the state table at `BUFF` (that bit I do leave to the reader). We then need to do a comparison between this cycle and the previous one. We also need to be careful about initialisation so that we do not count state changes falsely. The initial pass should therefore copy the states to `BUFF'` from `BUFF` so that no inadvertent changes are recorded during the first pass. Thereafter, the comparisons should reflect the differences between scans and be able to correctly account for them.

The initial pass may look a bit like:-

```
: [] (S n1\n2 : <name> -- n3 )
  (G Store the given offset n1 in <name> and calculate the
     new offset n3 from the given size n2 and old offset
     n1.  )
  OVER CELLS CONSTANT + ;
```

```
0   \ Initial offset for offset calculator
1 [] W0
1 [] W1
1 [] W2
1 [] W3
\ Create the main and history buffers
CELL+       \ Add a one cell buffer-zone at end of array.
DUP CREATE BUFF  ALLOT
DUP CREATE BUFF' ALLOT
16 * 1 CELLS / \ Calculate size of logging array array.
LOGGED LOGBUF
```

**Listing 2. Input and History Buffer Initialisation**

### *Extracting The Results*

Having set up the buffers we can set about examining the individual bits to detect change of states and add time increments since last observations. The word `(LOG?)` detects the state change and sets bits accordingly. In the code below, `a` is the main buffer data items from `BUFF` and `b` is the history buffer item from `BUFF'`. If the bit was set and remains set the time increment indicator bit `t` is set. If the bit has changed state from a zero last time to a one this time the log-count bit `c` is set.

```
: (LOG?) (S a\b --- t\c )
   2DUP AND -ROT
   OVER XOR AND ;
(G  For each of the bits in the two cells a and b
    set the corresponding bits in t and c to indicate
    the requirement to increment the associated event
    counter or add the latest time interval. )

: LOGGABLE (S offset --- t\c )
        BUFF OVER + @ SWAP
        BUFF' + @ (LOG?) ;
(G  Resolve, from the buffer bit states, as
    indicated by the offset provided, the need or
    otherwise for incrementing the associated event
    counter or adding the latest time interval. )
```

**Listing 3. Resolving Changed States Data**

The code in Listing 4 below shows how the application might use the code. `LOG-IT` is just a simple example word that works its way through all bits in the state buffers `BUFF` and `BUFF'` and logs the counts and time increments accordingly.

+LOG is performed only when the t and c bits are set for the state of interest. The variable TIME-INTERVAL should be updated just prior to calling LOG-IT during each scan cycle. This may be performed by hardware or software and should be at the time granularity you are using for the system.

```
VARIABLE TIME-INTERVAL
: TIMER-UPDATE (S time-acc-addr\flag --- )
   IF   TIME-INTERVAL @ SWAP +!
   ELSE DROP
   THEN ;

: COUNTER-UPDATE (S counter-addr\flag --- )
   IF   1 SWAP +!
   ELSE DROP
   THEN ;

: +LOG (S counter address\timer address\mask\offset -- )
   LOGGABLE 2DUP OR
   IF   THIRD AND >R
        AND ROT R>
        COUNTER-UPDATE
        TIMER-UPDATE
   ELSE 2DROP 2DROP DROP  \ Nothing to log.
   THEN ;

: (LOG-IT) (S offset\index -- )
   OVER 16 * OVER + LOGBUFF
   ROT 2^N >R ROT R> SWAP
   +LOG ;

: LOG-IT (S -- )
  3 0
  DO   16 0
       DO   J I (LOG-IT)
       LOOP
  LOOP ;
```

**Listing 4. Including Logged Data**

### *Summary*

This article has just touched on a small part of a statistical data-logging function that is part of a SCADA type application. Combined with other data gathering and processing methods, it can form the basis of effective machine monitoring and management. The "real-life" application from which the code extracts have been taken was completed over seven years ago as a replacement for an earlier

(flawed) implementation (the previous implementation obtained inaccurate data which was only established during a thorough audit of the machines and host recorded data).

What you do with the data you have collected depends on the application. Quite often in SCADA type systems, the data is passed via a communications network to a central host where the data is used for long term trending and archiving. Alarms for out of tolerance readings or non-communication can be raised at the host. That way a large number of similar production machines can be monitored effectively with just a few staff members present.

---

Paul Bennett specialises in the control and monitoring of safety-critical systems and offers audits and training courses in addition to software development.

---

A few weeks back we had an enquiry through our web site. Could we put Patrick Hoverstadt of Syncho Ltd. (management consultancy) in touch with Forthought?

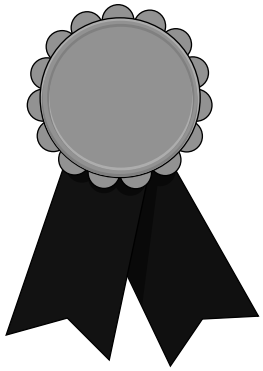
```
Hi Doug,
I'm trying to track down a firm called Forthought who were operating
in the UK 10 years ago. They worked on a program[1] for my company,
which we are interested in re-developing. The only other name I have
is Richard Ocney.
Have you heard of them or him, do you have any information as to what
happened to the company ?
Any information gratefully received.
Regards
Patrick Hoverstadt
```

We sent this out to members on e-mail with a request for assistance, had several useful replies and came up with a corrected name and a current phone number. Thanks to everyone who responded.

```
Hi Chris,
Many thanks for your help in tracking down Richard Olney.
One of the phone numbers supplied was the right one, and we have had
a couple of conversations exploring how we might take things forward.

Best regards,
Patrick Hoverstadt
```

[1]The program was Cyberfilter for real-time process performance measurement.



# *The FIG UK Awards of 1999*

The FIG UK Awards of 1998 were won by Philip Preston and Paul Bennett. These awards are given to encourage effort and recognise achievement.

***Free membership***

To everyone who sent in their nominations - "thank you". Our judges, the officers of FIG UK have now chosen the winners for 1999. They each receive

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

***Achievement***

**Jeremy Fowell**: for his efforts in leading the FIG UK Hardware Project.

***Forthwrite***

**Alan Wenham**: for his work in making German FIG accessible to all.

Our congratulations go to Jeremy and Alan on winning and appreciation for their efforts.

Chris Jakeman
cjakeman@bigfoot.com

# *Did you Know? – Forth v C*

While other parts of Forthwrite bring you all the news and the latest ideas and developments, the **Did You Know?** section highlights achievements in Forth, both recent and historical (taking care always to distinguish hearsay from attested fact).

Attempts are often made to compare Forth and C[3] but reliable comparisons are difficult to make as few projects are done more than once. However Elizabeth Rather is able to report one comparison that seems sound.

"We had an excellent real-world comparison a few years ago, when we and Paul Curtis were both writing OTA[4] virtual machines (VMs), conforming to a very detailed spec. We were writing in Forth, of course, and Paul in C. What makes this interesting is that Paul is an outstanding C programmer, one of the best I know. Our VMs were 32K on 8-bit CPUs, and about 45K on a 68K. Paul's were 64K on average. Performance was equivalent on equivalent platforms. In my opinion, this type of real-world challenge is far more meaningful than 'benchmarks'".
Source - Elizabeth Rather, Forth Inc.

Clearly these days Forth can hold its own regarding speed and size in embedded systems.

---

[3] See *Is Forth Code Compact? A Case Study*, by M. Anton Ertl in the euroFORTH 1999 report earlier in this issue

[4] OTA – Open Terminal Architecture

# FIG UK – AGM

The Annual General Meeting will be held on Saturday 23rd September at Doug Neal's home, 58 Woodland Way, Morden from 2:30pm.

All members who wish to attend are cordially invited to do so. If you cannot attend, but wish to comment on the way FIG is going or the direction you would like it to take, write or e-mail Chris or myself before the meeting.

Anyone who lives in the London area can get to my house easily by Underground as we are just ten minutes walk from the southern terminus of the Northern Line. Please phone for directions (020 8542 2747) if you can't find us in your A-Z.

Some of the topics likely to be discussed are:

- Joint projects such as Clock Challenge and WebForth
- Ideas for the Web Site
- Re-vamp for Forthwrite
- Finances – see Treasurer's report opposite
- How problems with International FIG affect FIG UK
- EuroFORTH 2000

---

**More Editorial**

Have you noticed the new look on our web-site? The idea is to provide a common style for the main pages of the site, while keeping our reputation for rapid downloads and access from old browsers as well as new ones.

## Forth Interest Group UK: Revenue Account for year to 31 March 2000

| | Revenue (see note) | Costs | Surplus/Deficit to 31-Mar-00 | 31-Mar-99 |
|---|---|---|---|---|
| | £ | £ | £ | £ |
| **Membership Subscriptions** | 766 | | 766 | 898 |
| **Forthwrite with advertising** | 24 | 845 | -821 | -888 |
| **Library Purchases etc.** | 0 | | 0 | -162 |
| **Sale of back-issues (one off)** | 0 | 0 | 0 | 45 |
| **Interest on bank deposit** | 3 | | 3 | 23 |
| **Committee travelling expenses** | | 0 | 0 | -85 |
| **Totals for year** | 793 | 845 | -52 | -169 |
| **Previous Accumulated surplus at 31 Mar** | | | 579 | 748 |
| **Accumulated surplus at 31 Mar** | | | 527 | 579 |

| **Balance sheet at** | | 31 Mar 00 | 31 Mar 99 |
|---|---|---|---|
| **Cash at bank and in hand** | | 1221 | 1091 |
| **Less: Creditors** | -40 | | -85 |
| **Revenues received in advance** | -654 | | -427 |
| **Net Liquidity** | | 527 | 579 |
| **Stock: Stationery at cost** | | 0 | 0 |
| **Accumulated surplus at 31 Mar** | | 527 | 579 |

### Treasurer's report to the members of FIG-UK

I certify that the above Revenue Account for the year to 31 March 2000 and Balance Sheet as at that date have been prepared in accordance with my own records of transactions for the year. I have not examined stocks (which are held by other members of the Committee); books for lending, ForthWrite back issues and printed listings have been written off in full.

The subscription rate was increased from £10 to £12 with effect from 1 Jan 2000. Subscriptions received in advance are recognised as revenue in six instalments, when the production and distribution costs of the six relevant ForthWrite issues are incurred. ForthWrite advertising revenue is treated in a similar way.

Keith Matthews, Treasurer 10 August 2000
20 Spindlebury, Cullompton, Devon EX15 1SY

bpaysan@mikron.de
www.jwdt.com/~paysan/

# A Web-Server in Forth

## Bernd Paysan

Bernd Paysan is well-known to many Forth users for his work with Anton Ertl on Gforth, his own bigFORTH and his impressive graphics work (MINOS). I asked his permission to re-print this edited paper from the Hamburg conference to emphasise the quality of work being done in Germany and to show also that Forth has a place in the interconnected world of the Internet.

### *Abstract*

An HTTP-Server in Gforth is presented as an opportunity to show that you can do string-oriented things with Forth as well. The development time (a few hours) shows that Forth is an appropriate tool for this kind of work and delivers fast results.
This is an edited translation of the paper presented at the Forth Tagung 2000 conference in Hamburg and available at http://www.jwdt.com/~paysan/httpd-en.html.

### *Introduction*

Since I have always given bigFORTH/MINOS-related presentations in the last few years, I'll do something with Gforth this time. Gforth is another tool you can do neat things with, and in contrast to what you here elsewhere, Forth is suitable for almost anything. Even a web-server.

In this age of the “new economy”, the Internet is important. Everybody is “in there” except Forth, which hides in the embedded control niche. There isn’t any serious reason for that. The following code was created in just a few hours of work and mostly operates on strings. The old prejudice, that Forth was good at biting bits, but has troubles with strings, is thus disproved.

### Motivation

What do you need a web-server for in Forth? Forth is used for measurement and control in remote locations such as the sea-bed or the crater of a volcano. Less remotely, Forth may be used in a refrigerator and, if that stops working, things soon get messy. So a communication thingy is built in.

How much better would it be if instead of “some communication thingy built in”, there was a standard protocol. HTTP is accessible from the web-cafe in Mallorca, or from mobile yuppie toys such as PDAs or cell phones. Perhaps one should build such a web-server into each oven and into the bath, so that people can use their cell phone on holidays to check repeatedly (every three minutes?) if they really turned their oven off.

Anyway, the customer, boss or whoever buys the product, wants to hear that there is some “Internet-thingy” build in, especially if one isn't in e-Business already. And the costs must be zero too.

But let's take this slowly, step by step.

### *A Web Server, Step by Step*

Actually, you have to study the RFC[5] documents. The RFCs in question are RFC 945 (HTTP/1.0) and RFC 2068 (HTTP/1.1), which both refer to other RFCs. Since these documents alone are much longer than the source code presented below (and reading them would take longer than writing the sources), we will defer that for later. The web server thus won't be 100% RFC-conforming (i.e. implement all features), and conforms only as far as necessary for a typical client like Netscape. However additions are easy to achieve.

A typical HTTP-Request looks like this:

```
GET /index.html HTTP/1.1
Host: www.paysan.nom
Connection: close
```

(Note the empty line at the end). And the response is

```
HTTP/1.1 200 OK
Date: Tue, 11 Apr 2000 22:27:42 GMT
Server: Apache/1.3.12 (Unix) (SuSE/Linux)
Connection: close
Content-Type: text/html

<HTML>
...
```

This looks quite trivial, so let's start. The web server should run under Unix/Linux. That takes one problem out of our hands - how we get to our socket - since that's what *inetd*, the Internet daemon, does for us. We only need to tell it on which port our web server expects data, and enter that into the file `/etc/inetd.conf`:

```
# Gforth web server
gforth stream tcp nowait.10000 wwwrun /usr/users/bernd/bin/httpd
```

---

[5] RFC: Request For Comments -Internet standards documents are all named like this.

We won't replace the default web server just yet (something might not work straight away), so we shall need a new port and that one goes into the file `/etc/services`:

```
gforth  4444/tcp # Gforth web server
```

When we do a restart or a `killall -HUP inetd`, *inetd* will realize the changes and start our web server for all requests on port 4444. What we need next is an executable program. Gforth supports scripting with `#!`, as is common for scripting languages in Unix. In the line below, the blank is significant:

```
#! /usr/local/bin/gforth

warnings off
```

We had better disable any warnings. Let's load a small string library:

```
include string.fs
```

We shall need a few variables for the URL requested from the server, the arguments, posted arguments, protocol and states.

```
Variable url       \ stores the URL (string)
Variable posted    \ stores arguments of POST (string)
Variable url-args  \ stores arguments in the URL (string)
Variable protocol  \ stores the protocol (string)
Variable data      \ true, when data is returned
Variable active    \ true for POST
Variable command?  \ true in the request line
```

A request consist of two parts, the request line and the header. Spaces are separators. The first word in a line is a “token” indicating the protocol, the rest of the line, or one/two words are parameters.

Since we can process a request only once the whole header has been parsed, we save all the information. Therefore we define two small words which take a word representing the rest of a line and store it in a string variable:

```
: get ( addr - )  name rot $! ;
: get-rest ( addr - )
  source >in @ /string dup >in +! rot $! ;
```

As told above, we have header values and request commands. To interpret them, we define two wordlists:

```
wordlist constant values
wordlist constant commands
```

But before we can really start, the URL might contain spaces and other special characters, what to do with them? HTTP advises to transmit these special

characters in the form `%xx`, where `xx` are two hex digits. We thus must replace these characters in the finished URL:

```
\ HTTP URL rework

: rework-% ( add - ) { url }  base @ >r hex
  0 url $@len 0 ?DO
      url $@ drop I + c@ dup '% = IF
          drop 0. url $@ I 1+ /string
          2 min dup >r >number r> swap - >r 2drop
      ELSE  0 >r  THEN  over url $@ drop + c!  1+
  r> 1+ +LOOP  url $!len
  r> base ! ;
```

So, that's done. But stop! URLs consist of two parts: path and the optional arguments. Separator is '?'. So first split the string into two parts:

```
: rework-? ( addr - )
    dup >r $@ '? $split url-args $! nip r> $!len ;
```

So we've defined the basics and can start. Each requests fetches a URL and the protocol, splits the URL into path and arguments and replaces the special character glyphs by the real characters (but those in the arguments remain as we don't yet know what should happen to them). Finally, we must switch over to another vocabulary, since the header follows after the request.

```
: >values  values 1 set-order command? off ;
: get-url ( - ) url get protocol get-rest
  url rework-? url rework-% >values ;
```

So now we can define the commands. According to the RFC, we only need `GET` and `HEAD`, `POST` is then a bonus.

```
commands set-current

: GET   get-url data on  active off ;
: POST  get-url data on  active on  ;
: HEAD  get-url data off active off ;
```

And now for the header values. Since we need a string variable for each value, and otherwise want only to store the string, we build that with `CREATE DOES>`. Again: we need a variable *and* a word, which stores the rest of the line there. In two different vocabularies; the latter with a colon behind.

Fortunately, Gforth provides `NEXTNAME`, an appropriate tool for this. We construct exactly the string we need and call `VARIABLE` and `CREATE` afterwards.

```
: value:  ( - )  name
  definitions 2dup 1- nextname Variable
  values set-current nextname here cell - Create ,
  definitions  DOES> @ get-rest ;
```

And now we set to work and define all the necessary variables:

```
value: User-Agent:
value: Pragma:
value: Host:
value: Accept:
value: Accept-Encoding:
value: Accept-Language:
value: Accept-Charset:
value: Via:
value: X-Forwarded-For:
value: Cache-Control:
value: Connection:
value: Referer:
value: Content-Type:
value: Content-Length:
```

There are some more (see RFC), but these are all we need for the moment.

### *Parsing a Request*

Now we must parse the request. This should be completely trivial, we could just let the Forth interpreter chew it but for two little caveats:

1. Each line ends with `CR LF`, while Gforth under Unix expects lines to end with an `LF` only. We thus must remove the `CR`. And

2. each header ends with an empty line, not some executable Forth word. We must therefore read line by line with `refill`, remove `CR`s from the line end, and then check if the line was empty.

```
Variable maxnum

: ?cr ( - )
  #tib @ 1 >= IF  source 1- + c@ #cr = #tib +!  THEN ;

: refill-loop ( - flag )
  BEGIN  refill ?cr  WHILE  interpret  >in @ 0=  UNTIL
  true  ELSE  maxnum off false  THEN ;
```

So, the key things are done now. Since we can't let the Forth interpreter loose on the raw input stream *stdin*, we pre-process the stream ourselves. We initialize a

few variables which we need to interpret anyway, and steal some code from INCLUDED:

```
: get-input ( - flag ior )
  s" /nosuchfile" url $!  s" HTTP/1.0" protocol $!
  s" close" connection $!
  infile-id push-file loadfile !  loadline off  blk off
  commands 1 set-order  command? on  ['] refill-loop catch
```

Waiiiit! The request isn't complete yet. The method POST, which was added as bonus, expects the data now. The length fortunately is stored as base 10 number in the field "Content-Length:".

```
  active @ IF  s" " posted $! Content-Length $@ snumber? drop
      posted $!len  posted $@ infile-id read-file throw drop
  THEN
  only forth also  pop-file ;
```

### *Answer a Request*

OK, we've handled a request, and now we must respond. The path of the URL is unfortunately not as we want it; we want to be somehow Apache-compatible, i.e. we have a ``global document root'' and a variable in the home directory of each user, where he can put his personal home page. Thus we can't do anything else but look at the URL again and finally check, if the requested file really is available:

```
Variable htmldir

: rework-htmldir ( addr u - addr' u' / ior )
  htmldir $!
  htmldir $@ 1 min s" " compare 0=
  IF    s" /.html-data" htmldir dup $@ 2dup '/ scan
        nip - nip $ins
  ELSE  s" /usr/local/httpd/htdocs/" htmldir 0 $ins  THEN
  htmldir $@ 1- 0 max + c@ '/ = htmldir $@len 0= or
  IF  s" index.html" htmldir dup $@len $ins  THEN
  htmldir $@ file-status nip ?dup ?EXIT
  htmldir $@ ;
```

Next, we must decide how the client should render the file - i.e. which MIME type it has. The file suffix is all we need to decide, so we extract it next.

```
: >mime ( addr u - mime u' )  2dup tuck over + 1- ?DO
  I c@ '. = ?LEAVE  1-  -1 +LOOP  /string ;
```

Normally, we'd transfer the file as is to the client (transparent). Then you tell the client how long the file is (otherwise, we'd have to close the connection after each request). We open a file, find its size and report that to the client.

```
: >file ( addr u - size fd )
  r/o bin open-file throw >r
  r@ file-size throw drop
  ." Accept-Ranges: bytes" cr
  ." Content-Length: " dup 0 .r cr r> ;

: transparent ( size fd - ) { fd }
  $4000 allocate throw swap dup 0 ?DO
      2dup over swap $4000 min fd read-file throw type
      $4000 - $4000 +LOOP  drop
  free fd close-file throw throw ;
```

We do all the work with `transparent`, using `TYPE` to send the file in chunks to support “keep-alive” connections, which modern web browsers prefer. The creation of a new connection is significantly more “expensive” than to continue with an established one. We benefit on our side also, since starting Gforth again isn't for free either. If the connection is keep-alive, we return that, reduce `maxnum` by one, and report to the client how often he may issue further requests. When it's the last request, or no further are pending, we send that back, too.

```
: .connection ( - )
  ." Connection: "
  connection $@ s" Keep-Alive" compare 0= maxnum @ 0> and
  IF  connection $@ type cr
      ." Keep-Alive: timeout=15, max=" maxnum @ 0 .r cr
      -1 maxnum +!  ELSE  ." close" cr maxnum off  THEN ;
```

Now we just need some means to recognise MIME file suffixes and send the appropriate transmissions. For the response, we must also first send a header. We build it from back to front here, since the top definitions add their stuff ahead. To make the association between file suffixes and MIME types easy, we simply define one word per suffix. That gets the MIME type as string. `transparent:` does all that for all the file types that are handled using `transparent:`

```
: transparent: ( addr u - ) Create  here over 1+ allot place
  DOES>  >r  >file
  .connection
  ." Content-Type: "  r> count type cr cr
  data @ IF  transparent  ELSE  nip close-file throw  THEN ;
```

There are hundreds of MIME types, but who wants to enter all of them? Nothing could be easier than this, we steal the MIME types that are already known to the system, say from `/etc/mime.types`. The file lists the mime type on the left paired with the file suffixes on the right (sometimes none).

```
: mime-read ( addr u - )  r/o open-file throw
  push-file loadfile !  0 loadline ! blk off
```

```
BEGIN  refill  WHILE  name
    BEGIN  >in @ >r name nip  WHILE
        r> >in ! 2dup transparent:  REPEAT
    2drop rdrop
REPEAT  loadfile @ close-file pop-file throw ;
```

One more thing we need: for active content we want to use server side scripting (in Forth, of course). Since we don't know the size of these requests in advance, we don't report it but close the connection instead. That relieves us of the problem of cleaning up the trash the user is creating with his active content (that's Forth code!).

```
: lastrequest
  ." Connection: close" cr maxnum off
  ." Content-Type: text/html" cr cr ;
```

So let's start with the definition of MIME types. Get a new wordlist. Active content ends with `shtml` and is `included`. We provide a few special types and the rest we get from the system file mentioned above. For unknown file types, we need a default type, `text/plain`.

```
wordlist constant mime
mime set-current

: shtml ( addr u - )  lastrequest
  data @ IF  included  ELSE  2drop  THEN ;

s" application/pgp-signature" transparent: sig
s" application/x-bzip2" transparent: bz2
s" application/x-gzip" transparent: gz
s" /etc/mime.types" mime-read

definitions

s" text/plain" transparent: txt
```

### *Error Reports*

Sometimes a request goes wrong. We must be prepared for that and respond with an appropriate error message to the client. The client wants to know which protocol we speak, what happened (or if everything is OK), who we are, and in the error case, a error report in plain text (coded in HTML) would be nice:

```
: .server ( - )  ." Server: Gforth httpd/0.1 ("
  s" os-class" environment? IF  type  THEN  ." )" cr ;
```

```
: .ok  ( - ) ." HTTP/1.1 200 OK" cr .server ;


: html-error ( n addr u - )
  ." HTTP/1.1 " 2 pick . 2dup type cr .server
  2 pick &405 = IF ." Allow: GET, HEAD, POST" cr  THEN
  lastrequest
  ." <HTML><HEAD><TITLE>" 2 pick . 2dup type
  ." </TITLE></HEAD>" cr
  ." <BODY><H1>" type drop ." </H1>" cr ;


: .trailer ( - )
  ." <HR><ADDRESS>Gforth httpd 0.1</ADDRESS>" cr
  ." </BODY></HTML>" cr ;


: .nok ( - ) command? @ IF  &405 s" Method Not Allowed"
  ELSE  &400 s" Bad Request"  THEN  html-error
  ." <P>Your browser sent a request that this server "
  ." could not understand.</P>" cr
  ." <P>Invalid request in: <CODE>"
  error-stack cell+ 2@ swap type
  ." </CODE></P>" cr .trailer ;


: .nofile ( - ) &404 s" Not Found" html-error
  ." <P>The requested URL <CODE>" url $@ type
  ." </CODE> was not found on this server</P>" cr .trailer ;
```

### *Top Level Definitions*

We are almost done now. We simply glue together all the pieces above to process a request in sequence - first fetch the input, then transform the URL, recognize the MIME type, work on it including error exits and default paths. We need to flush the output, so that the next request doesn't stall. And do that all over again  times, until we reach the last request.

```
: http ( - )  get-input  IF  .nok  ELSE
  IF  url $@ 1 /string rework-htmldir
      dup 0< IF  drop .nofile
      ELSE  .ok  2dup >mime mime search-wordlist
          0= IF  ['] txt  THEN  catch IF  maxnum off THEN
  THEN  THEN  THEN  outfile-id flush-file throw ;
```

```
: httpd  ( n - )  maxnum !
  BEGIN  ['] http catch  maxnum @ 0= or  UNTIL ;
```

To make Gforth run that at the start, we patch the boot message and then save the result as a new system image.

```
script? [IF]  :noname &100 httpd bye ; is bootmessage  [THEN]
```

### *Scripting*

As a special bonus, we can process active content. That's really simple: We just write our HTML file as usual and indicate the Forth code with “<$” and “$> ” (the space for the closing parenthesis is certainly intentional!). Let's define two words, `$>` , and to get the whole thing started, `<HTML>`:

```
: $> ( - )
  BEGIN  source >in @ /string s" <$" search  0= WHILE
      type cr refill  0= UNTIL  EXIT  THEN
  nip source >in @ /string rot - dup 2 + >in +! type ;
: <HTML> ( - )  ." <HTML>" $> ;
```

That's quite enough, we don't need more. The rest is all done by Forth, as in the following example:

```
<HTML>
<HEAD>
<TITLE>GForth <$ version-string type $>  presents</TITLE>
</HEAD>
<BODY>
<H1>Computing Primes</H1><$ 25 Constant #prim $>
<P>The first <$ #prim . $> primes are: <$


: prim?  0 over 2 max 2 ?DO  over I mod 0= or  LOOP  nip 0= ;


: prims ( n - )  0 swap  2
  swap 0 DO  dup prim? IF  swap  IF  ." , "  THEN  true swap
  dup 0 .r 1+  1  ELSE  1+  0  THEN
  +LOOP  drop ;


#prim prims $> .</P>
</BODY>
</HTML>
```

### *Outlook*

That was a few hundred lines of code - far too much. I have delivered an “almost” complete Apache clone. That won't be necessary for the sea-bed or the refrigerator. Error handling is ballast, too. And if you restrict to single connection (performance isn't the goal), you can ignore all the protocol variables. One MIME type (text/html) is sufficient -- we keep the images on another server. There is some hope that one can get a working HTTP protocol with server-side scripting in one screen.

### *Appendix: String Functions*

Certainly we need some string functions, it doesn't work without. The following string library stores strings in ordinary variables, which then contain a pointer to a counted string stored allocated from the heap. Instead of a count byte, there's a whole count cell, sufficient for all normal use.

The string library originates from bigFORTH and I've ported it to Gforth (ANS Forth). But now we consider the details of the functions. First we need two words bigFORTH already provides:

```
: delete   ( addr u n - )
  over min >r  r@ - ( left over )  dup 0>
  IF  2dup swap dup  r@ +  -rot swap move  THEN  + r> bl fill ;
```

delete deletes the first `n` bytes from a buffer and fills the rest at the end with blanks.

```
: insert   ( string length buffer size - )
  rot over min >r  r@ - ( left over )
  over dup r@ +  rot move   r> move  ;
```

`insert` inserts as string at the front of a buffer. The remaining bytes are moved on. Now we can really start:

```
: $padding ( n - n' )
  [ 6 cells ] Literal + [ -4 cells ] Literal and ;
```

To avoid exhausting our memory management, there are only certain string sizes; `$padding` takes care of rounding up to multiplies of four cells.

```
: $! ( addr1 u addr2 - )
  dup @ IF  dup @ free throw  THEN
  over $padding allocate throw over ! @
  over >r  rot over cell+  r> move 2dup ! + cell+ bl swap c! ;
```

`$!` stores a string at an address; if there was a string there already, that string will be lost.

```
: $@ ( addr1 - addr2 u )  @ dup cell+ swap @ ;
```

`$@` returns the stored string.

```
: $@len ( addr - u )  @ @ ;
```

`$@len` returns just the length of a string.

```
: $!len ( u addr - )
  over $padding over @ swap resize throw over ! @ ! ;
```

`$!len` changes the length of a string. Therefore we must change the memory area and adjust address and count cell as well.

```
: $del ( addr off u - )   >r >r dup $@ r> /string r@ delete
  dup $@len r> - swap $!len ;
```

`$del` deletes `u` bytes from a string with offset `off` .

```
: $ins ( addr1 u addr2 off - ) >r
  2dup dup $@len rot + swap $!len  $@ 1+ r> /string insert ;
```

`$ins` inserts a string at offset `off`.

```
: $+! ( addr1 u addr2 - ) dup $@len $ins ;
```

`$+!` appends a string to another.

```
: $off ( addr - )  dup @ free throw off ;
```

`$off` releases a string.

As a bonus, there are functions to split strings up.

```
: $split ( addr u char - addr1 u1 addr2 u2 )
  >r 2dup r> scan dup >r dup IF  1 /string  THEN
  2swap r> - 2swap ;
```

`$split` divides a string into two, with one char as separator (e.g. '?' for arguments)

```
: $iter ( .. $addr char xt - .. ) { char xt }
  $@ BEGIN  dup  WHILE  char $split >r >r xt execute r> r>
     REPEAT  2drop ;
```

`$iter` takes a string apart piece for piece, also with a character as separator. For each part a passed execution token (`xt`) will be called. With this you can take apart arguments -- separated with '&' - with ease.

---

Bernd Paysan is a member of German FIG – Forth Gesellschaft – and posts regularly to comp.lang.forth.

# *More on International FIG*

After the brief report on International FIG in the last Forthwrite, you may be wondering how things have developed. In the absence of any official communication from the Board of Directors, we have to fall back on a comp.lang.forth message from Elizabeth Rather of Forth Inc. (4-Aug) which is worth reporting in full:

FIG isn't dead, but it's on life support right now. I forwarded this message to the rest of the board hoping others would respond. Please regard this as a personal response to the question, not an official statement of the board of FIG (of which I am a member).

FIG is suffering from a severe lack of support, both in terms of money and also time/energy. For the last several years it has been running at a deficit financially, continuing to operate thanks to the dedication and financial support of Skip and Trace Carter of Taygeta. They tried everything they knew to improve membership, but it dwindled. Marlin gave up trying to pull FD together 4 times a year because of lack of good articles.

Ironically, the decline in FIG membership and support doesn't necessarily reflect a decline in Forth usage: only a tiny fraction of Forth users are members of FIG. I have tried to explore ways of making FIG more relevant to the Forth programmers who are professionals using Forth in their jobs, as well as the hobbyists who have been the FIG mainstays, but it's a difficult chicken-egg solution. Few professional Forth programmers have time to write articles about their work, so they look at FD and see nothing from their peers and go away.

Last winter, exhaustion set in, and Skip resigned as President, requesting that the office be moved from Taygeta. It is in the process of being moved back into John Hall's keeping, but it's unclear how to really set up for handling sales, etc. Marlin has also resigned as FD editor. We have at least one possible volunteer to help, and suggestions regarding electronic publication. We have a final edition of FD ready to publish very soon (Marlin has had it finished since last winter, but it has been awaiting official notice of such things as a new board election and FORML).

If FIG is important to you, now is the time to step forward and see what you can contribute, not only in terms of money, but also logistical support and effort to make things happen.
Please respond to <board@forth.org>.

Thanks,
Elizabeth

It is sad to see such a valuable organisation brought low and I am sure we all want to see International FIG overcome their difficulties, recover the key position they hold and resume their support of Forth users worldwide.

You may be wondering why they should be in crisis with around 700 members where FIG UK and Forth Gesellschaft are comfortable with 100-200. I suspect (and this is only a personal interpretation) that International FIG have financial commitments for staff and premises and the reducing number of members has tipped them into a deficit. The current paralysis may be because they cannot agree on how to proceed.

Two points I ought to make:

1. FIG UK (and I suspect Forth Gesellschaft too) do not have financial commitments of this kind. Our costs are in proportion to our numbers and we have been careful to keep it that way.
2. FIG UK no longer has a declining membership and numbers have been steady for the past two years.

As I hope the pages of Forthwrite make clear, Forth itself continues to be used, to grow and develop and FIG UK is very well placed to promote the interests of Forth users throughout the UK.

Chris Jakeman

---

**More Editorial**

In the last issue, Dave Pochin's "Floating Point Fudge" article came some test cases for checking the code (as mentioned in the article), which I left out by accident. If anyone would like copies of these, please contact Dave directly.

# *Dutch Forth Users Group*

Reading Dutch is easier than you might think. And as Forth is an international language, reading Dutch code is easier still for a Forth enthusiast. Are you interested? Why not subscribe to

## HCC-Forth-gebruikersgroep

For only 20 guilders a year (£6.30), we will send you 5 to 6 copies of our "fig-leaf" broadsheet 'Het Vijgeblaadje' . This includes all our activities, progress reports on software and hardware projects and news of our in-house products.

To join, contact our Chairman:

Willem Ouwerkerk
Boulevard Heuvelink 126
6828 KW Arnhem, The Netherlands
E-Mail: w.ouwerkerk@kader.hobby.nl

The easiest way to pay is to post a 20 Guilder note direct to Willem.

# Letters

Graeme Dunbar has a new book to report and Robin Francis is a new member and tells us of his interests. Meanwhile Nicholas Nemtsev from Russia is one of our new on-line readers who has comments on an item by Fred Behringer in the first issue of Forthwrite to be published on-line.

**Graeme Dunbar**

From: g.r.a.dunbar@eee.rgu.ac.uk
Sent: 27 June 2000
To: F11-UK Mailing List
Subject: Book and Software

A rep from Prentice Hall came to see me a few weeks back and I told him I was looking for books on the 68HC12 to support our courses. He sent me a copy of:

"Design of Embedded Systems Using 68HC12/11 Microcontrollers" by Richard Haskell, Prentice Hall 1999. ISBN 0 13 083208 1.

It uses a version of Forth called WHYP that comes on a floppy disc with the book. This looks like a useful way of learning about the hardware and machine code of the 68HC11 and 68HC12 with a Forth "wrapper". It reminds me a bit of Loeliger's and John Matthews' books in the way in which the code evolves from assembly language beginnings.

I haven't had time to try WHYP out properly, but it look promising. The hardware side of the book looks good and does not shy away from modules like the SPI interface (as provided by F11-UK) and the timers.

Unfortunately it is only the Forth that prevents me re-commending it as a book on our courses as it does not match up with our syllabi.

Regards, Graeme

**Robin Francis**

From: remfrancis@rdplus.net
Sent: 07 July 2000

Hi Chris,

Many thanks for the welcome. My interest has been purely academic up to date although I am now taking a more lively interest in learning Forth. It was the young son of a friend (living in South Africa) who said he was interested in programming robots that woke me up from my slumbers and sent me looking at the Forth.org web site for him. So while he is over here with his parents I am giving him my copy of Brodie plus F83 to make a start. That is how I found out about FIG UK. I shall make a gift subscription if he takes to it as I suspect he will (Now there's a good idea! Ed.)

My main interest is intellectual history and I have also wondered if anybody has thought of starting a computer museum which would include a software museum. My visit to Bletchley Park was encouraging in that respect. It is certainly the right place to start one and maybe that is what they are going to do. It is rather a long way from where I am but I shall make sure of going back and see how they are getting on. If they do, you must make sure they have all kinds of Forth in their software museum. Then I might be able to run that complex 3D stuff on the 8086 I have just been reading about.

I shall come back when I have found my way about a bit more. I had no idea there was such a lively Forth community still in being. What is it about TILs that is so intellectually fascinating?

Regards for now,

Robin Francis

**Nicholas Nemtsev**

From: nn@vdk.psc.ru
Sent: 10 August

Dear Chris,

I have read the article "32-bit GCD without Division in ZF and Turbo Forth" by Fred Behringer in Forthwrite's issue # 106. Obviously, Fred is treating 0/a, where a is non-zero, as an exceptional case along with a/0 and 0/0. However, I think it is better to assign the value <a> to the GCD in cases where 0/a, instead of Fred's exception-signalling 0, in order to be able to continue with the calculation in mind since it is not really necessary to consider 0/a as an exceptional case.

The following is my version of a GCD program (InfoForth, 16-bit operands, postfix x86 assembler).

```
CODE GCD ( N1 N2 -- GCD)
     BX POP AX POP
     CX, CX XOR CX INC DX, CX MOV
     AX, AX OR  1$ JNS AX NEG  ( ABS A)
  1$: BX, BX OR  2$ JNS BX NEG  ( ABS B)
  2$: AX, AX OR  10$ JZ  BX, BX OR 10$ JZ    ( zero? )
  3$: AL, # 1 TEST 4$ JNZ AX, # 1 SAR CX, # 1 SAL 3$ JMP
  4$: BL, # 1 TEST 5$ JNZ BX, # 1 SAR DX, # 1 SAL 4$ JMP
  5$: CX, DX CMP 6$ JLE  CX, DX MOV
  6$: AX, BX CMP 9$ JE 7$ JA AX, BX XCHG
  7$: AX, BX SUB
  8$: AL, # 1 TEST 6$ JNZ AX, # 1 SAR 8$ JMP
  9$: CX MUL AX PUSH NEXT,
  10$: AX, BX ADD  9$ JNZ
       AX INC 9$ JMP
 END-CODE
```

Best regards,
Network Administrator
MUP "Gorvodokanal", Pskov
Nicholas Nemtsev

## FIG UK Services to Members

**Magazine** Forthwrite is our regular magazine, which has been in publication for more than 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** Jack Brien maintains our web site at http://forth.org.uk. He 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.