

More Graphics with Win32Forth

FIGUK magazine:
An Interview with Barry Culver
From the 'Net
euroFORTH Conference Report
Word Completion for Quikwriter Project
Using Wordlists for Many[
Across the Big Teich
Vierte Dimension 3/2002

events

euroFORTH 2002 – Conference Report

..... **23**

AGM Report **31**

news

Forth News **2**

reviews

Across the Big Teich
35

Vierte Dimension 3/2002 **40**

programming

Rectangles in Win32Forth **5**

Using Wordlists for Many[..... **32**

people

An Interview with Barry Culver . **13**

From the 'Net **18**

**Nominations for the
FIG UK Awards** **30**

projects

**Word Completion for Quikwriter
Project**

..... **27**



Editorial

Forth News provides a good check on the health of the Forth community. In this issue, we report significant new programs in networking, ColorForth for Windows, HolonX, a new MISC design, updates to many public domain tools, a complete wordset for floating point and other items - a great way to round off 2002.

See inside for the promised interview with Barry Culver and another key item in the Win32Forth series from Dave Pochin.

We welcome new member Philip Eaton who has long experience in Forth and an interest in restoring old arcade games.

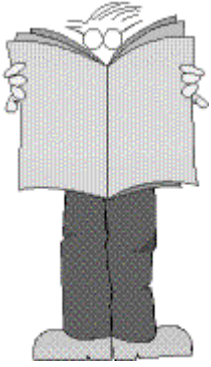
With great regret, we report the death of Ed Hersom. Ed was a member for 17 years contributing interesting items to Forthwrite.

May I commend Chuck's videos to you? (Now that I have ADSL, downloading a 100MB presentation is quite feasible.) Chuck is a good speaker with a great sense of humour, see <http://www.ultratechnology.com>

PS. Don't forget the monthly IRC session. Our next one is Saturday 1st February on the IRC server called "IRCNet", channel #FIGUK from 9:00pm.

Until next time, keep on Forthing,

Chris Jakeman



Forth News

Forth Events

In the previous issue, we reported that Bernd Paysan had contributed a Forth entry to the contest run by the 2002 International Conference on Functional Programming. Anton Ertl has analysed the results at <http://icfpcontest.cse.ogi.edu/scoring/table4s.html> to find that "Bernd's "busy-bee" (entry 30) is 10th out of 168 entries. Congratulations! This is especially remarkable since Bernd worked alone and did not make use of all the available time."

Bernd writes about his entry at <http://www.jwdt.com/~paysan/icfp.html>

Forth Applications

CWeed v2.1

Howerd Oakford has improved CWeed, a well-proven program for tidying up the white-space in source files. It can convert a C source file to a prescribed layout standard, but is also useful for removing tabs, trailing spaces and double lines from any text file. Also PC <-> Unix format conversion, display of control characters etc...

The download contains complete source for Win32Forth; see <http://www.inventio.co.uk/Cweedexe.htm>

Mail and News

With the aid of his minimal proposed pipes and sockets wordset, Marcel Hendrix has implemented short programs to send email, to receive email and receive news. The wordset and programs can be found at <http://home.iae.nl/users/mhx/pipes&socks.html>. The code is for iForth on WinNT 4.0 and Linux 2.0. iForth is a commercial product; but licences without cost will be considered if the community benefits.

Email

4ePost is a new mailer from Jos v.d.Ven which is able to exchange email or news in ASCII using the Internet. Its special features are the speed and management of spam.

This is an "open project" and contributions are encouraged.

4ePost requires Windows, Internet Explorer and Win32Forth. See <http://home.planet.nl/~josv/m4ePost.html>

Updates for 3D Chess

Jos v.d.Ven has posted updates to several Win32Forth programs downloadable from <http://home.planet.nl/~josv/msources.html> including 3D Chess, Toolset and Scene v2.2

Forth Resources

Floating Point Words

Brad Eckert has provided an extension for 16-bit and 32-bit ANS Forths. It provides 32-bit or 64-bit mantissas and 15-bit or 31-bit exponents, depending on your cell size. He also indicates 11 words which are candidates for programming in assembler. This is the first extension I have seen in this rather difficult area and a major contribution to the public domain. See <http://www.tinyboot.com/float.txt>

HolonX

Wolf Wejgaard is well-known for his Holon system for Forth which handles source text in a tree structure of modules, groups and words. Whereas previous Holon systems have been dedicated to specific targets

Holonforth[®]

(x86/MSDOS, 68HC11, JavaVM), in the new HolonX the structured source text is "loaded" to a linear text file ready for an external compiler/interpreter. If you load the whole application, the complete source text is contained in this one file. You can also choose to load only one module, one group, or just one word. Internally, source is saved in an XML format. Wolf offers HolonX as a tool to experiment with structured source. See <http://holonforth.com/tools/holonx.htm>.

^Forth to C Translator

David Williams has upgraded the ^Forth to C translator for PFE to include floating point words, with a few other code generation revisions. The

translator enables you to write in Forth and compile to C. You can also mix C code snippets in your Forth. See <http://www-personal.umich.edu/~williams/archive/forth/hatforth/dir.html>

Forth at VUB

Vrije Universiteit Brussel is the Free University of Belgium. The Principles of Programming Languages is one of the courses run by the Computer Science department which includes exercises in Pascal, Java, Smalltalk, Prolog and Forth. Phil Burke's ANS pForth is used for the exercises. See the course syllabus at <http://progmc30.vub.ac.be/FV's%20Courses/CMP213/> and pForth at <http://www.softsynth.com/pforth/>

Forth for Music

Jodell Bumatai has launched the SVFIG Musical Forth Words Project in conjunction with students of Cogswell College. Windows, Linux and Mac will be supported. Forth Programmers interested in music are welcome to join this project. See <http://www.dolfina.org/tutorials/goforth.htm>

Non-commercial Systems

ColorForth for Windows

Richard Collins has programmed a variant of ColorForth for Windows (works on Win2000). Now you can explore without having to boot from floppy. Download the 20K (!) executable from <http://homepages.paradise.net.nz/rscollins/c4/>

b16 FPGA Processor

Bernd Paysan has designed a 16-bit MISC processor for the public domain with components obtained by negotiation with Hans Eckes (hanseckes@addcom.de). The architecture is inspired by the c18 chip and by ColorForth from Charles Moore and provides 32 instructions. The design comes with sample code and a simple programming environment and programs are downloaded via a serial line. See

<http://www.jwdt.com/~paysan/b16.html>

Forth for iPaq PDA

Francois Vignon has offered his Forth implementation to others who might be interested (reach him at

him at f.vignon@ifrance.com)

Three other people have registered their application of Forth on iPaq at

<http://www.handhelds.org/z/wiki/HandheldsPeople>



Win32Forth v6.01

A new release is available which includes access to sub-directories. See http://sourceforge.net/project/showfiles.php?group_id=55294

kForth Updates

Krishna Myneni reports that kForth v1.0.13 is now available for many versions of Linux including Mandrake 9.0. See

<http://ccreweb.org/software/kforth/kforth.html>

Francois Vignon has offered his Forth implementation to others who might be interested (reach him at f.vignon@ifrance.com). Three other people have registered their application of Forth on iPaq at

<http://www.handhelds.org/z/wiki/HandheldsPeople>

PicForth v0.7 Released

Samuel Tardieu has greatly improved the compiler and documentation for this cross-compiler. PicForth generates code for the Microchip PIC 16F87x microcontroller family and is hosted on Gforth. For downloads and examples, see

<http://www.rfc1149.net/devel/picforth>

Commercial Systems

New Product SwiftX-SC

Forth Inc. have announced a new product, SwiftX-SC, a version of SwiftX for the Atmel AT90SC flash-based smart cards (based on the AVR mpu). This product was exhibited in a preliminary form at the Cartes smart card show in Paris in October, and is now ready for shipment. See

<http://www.forth.com>

Rectangles in Win32Forth

Dave Pochin

Dave has been sharing his discoveries on using Win32Forth to tame the Windows monster for some time. Material supporting this unique series can be found at <http://www.sunterr.demon.co.uk> .

Working with the graphics in Win32Forth can be frustrating. Here is a listing and some examples for working with rectangles.

I recently wished to fill a rectangular area of the screen with a colour. The class WinDC in the file **DC.F** has two methods that are suitable, FillRect and FillArea.

The method FillRect requires a local variable rectangle and makes a call to FillRect(), a Windows function.

The method FillArea uses the method SetRect and a rectangle called FillRect and then calls the Windows function FillRect().

So FillRect: is a method in WinDC
FillRect is an instance of the class rectangle, and
FillRect() is a Windows function.

Too many FillRect's for me. Here we go, defer the job and get back on the learning curve.

At the end of the Win32Forth file **Class.F** is a definition of the class Rectangle which contains a structure Record: and a number of methods.

FillArea (left top right bottom color_object --)

Uses the instance of Rectangle called FillRect, declared at the beginning of WinDC, SetRect: and .AddrOf from the class Rectangle.

FillArea will certainly do the job, but if I ever need more than one rectangle, I will have to repeatedly reset the rectangle FillRect, which could make debugging difficult.

FillRect (color_object rectangle --)

By using FillRect, I can have as many rectangles as my heart desires, providing I have their addresses, but each will still require the co-ordinates to be set. No problem, use the class Rectangle.

- a) declare a rectangle called nrect using Rectangle nrect
- b) set it up using left top right bottom SetRect: nrect
- c) find its address using nrect.AddrOf

Like every instruction in WinDC, the required graphic can only be displayed by calling a Windows function. So both FillRect and FillArea end by making a call to the Windows function FillRect(). This requires the conversion of the relative address used by Forth to the absolute address used by Windows, by using the term rel>abs and the handle of the device context hdc.

All done, now back to work. But no, a quick check in the reference texts shows there are fifteen Windows functions dealing with rectangles. I really don't want to look at these now, but since I've started....

These Windows functions are of two types, those which connect a rectangle to the screen and require a call to the device context, and those which manipulate the rectangle structures.

All the Windows rectangle functions return a value, these are either an indicator of success/failure of the function or of an operation giving a true/false result.

The listing below is heavily commented so that it can be easily changed for any required variations. Any Windows functions that call the device context use GetHandle: DC , and all the return values have been dropped, the success/failure returns should really be ?win-error and the values of the true/false returns are indicated where appropriate.

Each figure in the screen-shot shows the result of the operations in each section of the listing, and hopefully, only a few additional notes are needed.

Section 1

FillRect and FillArea are both Win32Forth methods from the class WinDC but, in general, the Windows functions have to be used, as there are no Forth equivalents. All three fill functions use a brush, line2* uses a stock brush and WinDC, line3* uses Windows and one of 30-odd system colours defined as constants. (COLOR_INFOBK is the colour of the background of the ToolTip control, to find the other constants use constants COLOR_ and use any term with this prefix, for example COLOR_DESKTOP or COLOR_3DLIGHT. The result will depend on how the desktop is set up, all these constants require 1+ as shown. Interchange lines marked 1*, 2*, and 3* to see the effect).

The Windows function InflateRect is used to change the size of a rectangle; the function CopyRect to copy one rectangle to another and the function EqualRect to compare two rectangles. The effect of these functions is demonstrated in the figure below.

Section 2

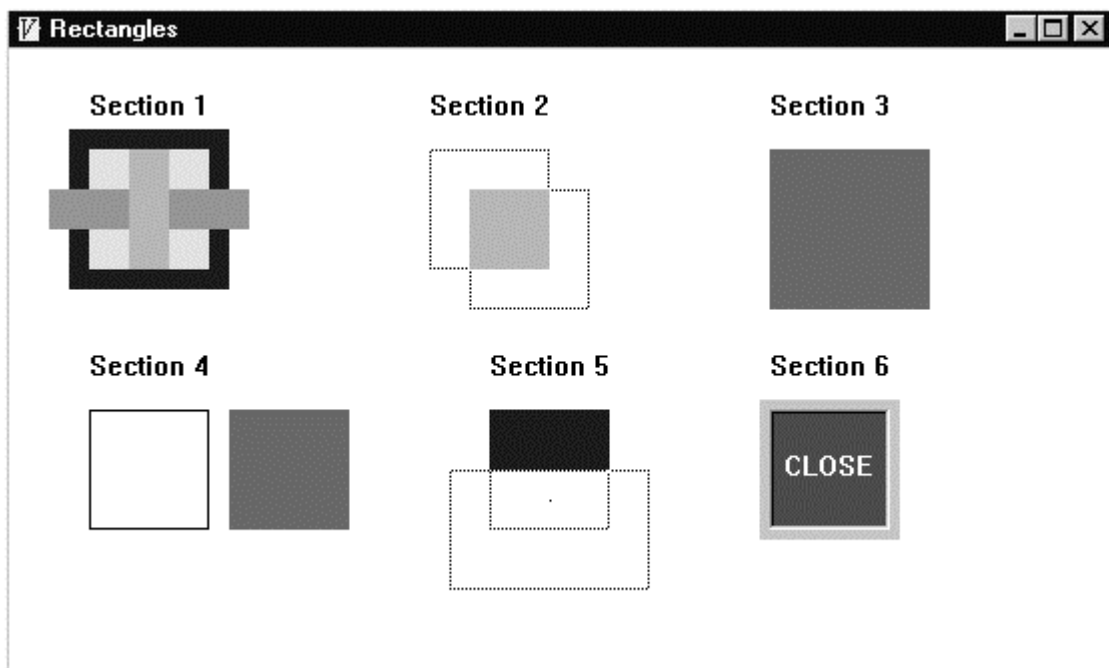
In this section two overlapping rectangles are set and made visible by using the Windows function DrawFocusRect. The rectangle is outlined with a series of one pixel dots; as this is an XOR operation. The rectangle will show against all

backgrounds and, if the function is repeated, the rectangle is no longer visible on the display.

The intersection of the two rectangles can be set to a third rectangle by using the `IntersectRect` function.

Section 3

In this section two overlapping rectangles are drawn as before, and are then used to set a third rectangle representing their union using the `UnionRect` function. The result is not the union of two arbitrary figures as these rectangle operations will only succeed if the result is a rectangle.



Section 4

An alternative way of marking the rectangle is to use the function `FrameRect` instead of `DrawFocusRect`; `FrameRect` outlines the rectangle with a specified brush. Rectangles can be offset to new positions using the `OffsetRect` function. This changes the co-ordinates in the structure and can be tested by commenting out lines (4*) and (5*) and noting that the frame is not offset.

Section 5

As well as the operations to form the union and intersection of two rectangles, a subtraction operation is available. Because these operations succeed only when they result in a rectangle, a demonstration based on overlapping rectangles as in sections 2 or 3, would give one of the rectangles as the result.

Rectangles can be erased, (their structure values set to 0) either by using the Windows function `SetRectEmpty` or, as here, by using the method `EraseRect` :

from the class `Rectangle` and emptiness can be tested by using the function `IsEmpty`.

Another useful test is the function `PtInRect`; it is not necessary to set the point as in this section before using the test.

Section 6

Another Windows rectangle function is `DrawEdge`. In its Win32Forth form it may be written `DrawEdge (Flag Edge Rectangle Handle -- flg)`.

The flag can be any of 19 constants with the prefix `BF_`.

The edge can be any of 4 constants with the prefix `BDR_` followed by 11 letters or of the 4 constants with the prefix `EDGE_`. A Windows reference guide or similar should give a full explanation of these constants. With a little more time, better ways of producing the figure shown could be found.

Q. What can be done with all this information?

A. Make a new dancing rectangle demonstration, or maybe a simple game.

Q. If I make an array of elements based on Section 6, do I have to spend time developing ToolBars?

A. No, no, not now. There is work to do.

```
\ Rectangles.F  Experiments with rectangles

anew program

:OBJECT Rectdemo <SUPER WINDOW

\ Set up three instances of the class Rectangle
Rectangle arect
Rectangle brect
Rectangle crect

:M ClassInit: ( -- )
    ClassInit: super
;M

:M ExWindowStyle: ( -- style )
    ExWindowStyle: SUPER
;M

:M WindowStyle: ( -- style )
    WindowStyle: SUPER
    WS_BORDER OR
    WS_OVERLAPPED OR
;M

:M WindowTitle: ( -- title )
```

```

        z" Rectangles "
        ;M

:M StartSize:  ( -- width height )
               550 310 ;M

:M StartPos:   ( -- x y )
               100 100
               ;M

:M Close:      ( -- )
               Close: SUPER
               ;M

:M On_Init:    ( -- )

        \ Set two rectangles using rectangle class ( left top right bottom -- )
        40 50 100 110 SetRect: arect
        40 150 100 210 SetRect: brect
        ;M

:M On_Paint:   ( -- )          \ screen redraw procedure

\ Section 1: FillRect(), InflateRect(), CopyRect(), EqualRect().
40 20 s" Section 1" TextOut: dc

\ Use FillArea: from dc { left top right bottom color_object -- }
30 40 110 120 LTBLUE FillArea: dc

\ Use FillRect: from dc { color_object rectangle -- }
( 1* ) LTYELLOW arect.AddrOf FillRect: dc

\ alternative, use System colours and call to Windows FillRect()
\ ( 2* ) BLACK_BRUSH SelectStockObject: dc drop arect.AddrOf
\      rel>abs GetHandle: dc Call FillRect drop

\ ( 3* ) COLOR_INFOBK 1+ arect.AddrOf rel>abs GetHandle: dc Call
\      FillRect drop

\ Alter the size of arect ( dy dx rectangle -- )
-20 20 arect.AddrOf rel>abs Call InflateRect drop
LTGREEN arect.AddrOf FillRect: dc

\ Copy arect to crect
arect.AddrOf rel>abs crect.AddrOf rel>abs Call CopyRect drop
arect.AddrOf rel>abs crect.AddrOf rel>abs Call EqualRect drop
( True )

\ Alter the size of crect ( dy dx rectangle -- )

```

```
20 -40 crect.AddrOf rel>abs Call InflateRect drop
arect.AddrOf rel>abs crect.AddrOf rel>abs Call EqualRect drop
( False )
```

```
LTCYAN crect.AddrOf FillRect: dc
```

```
\ Section 2: DrawFocusRect(), IntersectRect()
```

```
210 20 s" Section 2" TextOut: dc
```

```
\ Reset arect and brect in a new position and outline both.
```

```
210 50 270 110 SetRect: arect
```

```
230 70 290 130 SetRect: brect
```

```
arect.AddrOf rel>abs GetHandle: dc Call DrawFocusRect drop
```

```
brect.AddrOf rel>abs GetHandle: dc Call DrawFocusRect drop
```

```
\ Let crect be the intersection of arect and brect
```

```
brect.AddrOf rel>abs arect.AddrOf rel>abs crect.AddrOf rel>abs
```

```
Call IntersectRect drop
```

```
\ Show crect
```

```
LTCYAN crect.AddrOf FillRect: dc
```

```
\ Section 3: DrawFocusRect(), UnionRect()
```

```
380 20 s" Section 3" TextOut: dc
```

```
\ Reset arect and brect in a new position and outline both.
```

```
380 50 440 110 SetRect: arect
```

```
400 70 460 130 SetRect: brect
```

```
arect.AddrOf rel>abs GetHandle: dc Call DrawFocusRect drop
```

```
brect.AddrOf rel>abs GetHandle: dc Call DrawFocusRect drop
```

```
\ Let crect be the union of arect and brect
```

```
brect.AddrOf rel>abs arect.AddrOf rel>abs crect.AddrOf rel>abs
```

```
Call UnionRect drop
```

```
\ Show crect
```

```
LTMAGENTA crect.AddrOf FillRect: dc
```

```
\ Section 4: FrameRect(), OffsetRect(), InvertRect()
```

```
40 150 s" Section 4" TextOut: dc
```

```
\ Reset brect and frame it with the black brush
```

```
40 180 100 240 SetRect: brect
```

```
BLACK_BRUSH GetStockObject: dc
```

```
brect.AddrOf rel>abs GetHandle: dc Call FrameRect drop
```

```
\ Offset brect to a new position, Fill with a colour and invert it.
```

```
0 70 brect.AddrOf rel>abs Call OffsetRect drop
```

```
( 4* ) LTGREEN brect.AddrOf FillRect: dc
```

```
( 5* ) brect.AddrOf rel>abs GetHandle: dc Call InvertRect drop
```

```
\ Section 5: SubtractRect(), PtInRect(), IsRectEmpty()
```

```
240 150 s" Section 5" TextOut: dc
```

```
\ Reset arect and brect in a new position and outline both.
```

```
240 180 300 240 SetRect: arect
```

```
220 210 320 270 SetRect: brect
```

```
arect.AddrOf rel>abs GetHandle: dc Call DrawFocusRect drop
```

```
brect.AddrOf rel>abs GetHandle: dc Call DrawFocusRect drop
```

```
\ Subtract brect from arect, use crect as the result
```

```
brect.AddrOf rel>abs arect.AddrOf rel>abs crect.AddrOf rel>abs
```

```
Call SubtractRect drop
```

```
\ Fill crect
```

```
LTBLUE crect.AddrOf FillRect: dc
```

```
\ Erase arect and test for empty
```

```
EraseRect: arect
```

```
arect.AddrOf rel>abs Call IsRectEmpty drop ( True )
```

```
\ Test for brect empty
```

```
brect.AddrOf rel>abs Call IsRectEmpty drop ( False )
```

```
\ Set a point in brect
```

```
270 225 BLACK SetPixel: dc ( x y color_object -- )
```

```
\ Test for point in brect { y x rect -- }
```

```
225 270 brect.AddrOf rel>abs Call PtInRect drop ( True )
```

```
\ Section 6: DrawEdge()
```

```
380 150 s" Section 6" TextOut: dc
```

```
\ Reset arect and brect in a new position and fill both.
```

```
375 175 445 245 SetRect: brect
```

```
380 180 440 240 SetRect: arect
```

```
LTGRAY brect.AddrOf FillRect: dc
```

```
LTRED arect.AddrOf FillRect: dc
```

```
\ Use DrawEdge() and decorate rectangle.
```

```
BF_RECT EDGE_SUNKEN arect.AddrOf rel>abs
```

```
GetHandle: dc Call DrawEdge drop
```

```
LTRED SetBkColor: dc
```

```
WHITE SetTextColor: dc
```

```
387 200 s" CLOSE" TextOut: dc
```

```
;M
```

```
:M WM_LBUTTONDOWN
```

```

    set-mousexy
    mousey mousex brect.AddrOf rel>abs Call PtInRect
    if
      Close: self
    then
    0 ;M

;OBJECT

: DEMO      ( -- )
            Start: Rectdemo
            ;
cr cr .( Type DEMO to run )

```

Library Donations

Two books have been donated to the FIG UK Library.

Forth Techniques includes a useful reference for Forth standards prior to ANS Forth. 1985, Olney and Benson, Pan Books, 0-330-28961-6

All About Forth has a section clearly explaining the indirect threading mechanism. 1990, MVP-Forth Series, Glen Haydon

Culver Consultancy - An Interview with Barry Culver

Culver Consultancy is the latest company to take out corporate membership with FIG UK (see http://www.figuk.plus.com/corporate_membership.htm). Barry gives us an insight into his work with Forth.

I formed Culver Consultancy in 1998 as a one-man business offering software and some hardware design consultancy and specialising in embedded systems for instrumentation.

Customers come to us by word of mouth, supplemented by occasional mailings, presence on Triangle's list of experts, our web site at <http://www.culverconsultancy.co.uk> and the Applegate directory at <http://www.applegate.co.uk> and now <http://www.fig-uk.org>.

Dual Gas Differential Analyser

This product is a sophisticated high-end dual gas analyser system for research applications in bio-science. It features a colour display with a Windows-style interface. The instrument is powered by a 68332 processor, with an 8051 for interfacing.

The software for the 68332, incorporating screen drivers, floating point maths, a DOS-compatible filing system using PC cards, a high speed data processing and logging system was designed and written by Culver Consultancy Ltd.

This product is manufactured by ADC BioScientific Ltd. at <http://www.adc.co.uk/>



About 60% of our business is Forth, though it used to be more. Where I get to decide, Forth is usually my first choice, except for very small jobs (like baby PICs) where assembler is more appropriate, or for Windows where there is a lot of user interface - it is hard to beat the MS IDEs for Windows GUI development, though VB is a dreadful language!

"Forth is usually my first choice"

The immediate attraction of Forth was the ability to try out the code that you had just written immediately and even a tiny kernel gives you

so much functionality in a tiny amount of space. I started using Forth on small micros - suddenly I could write code and test hardware far more quickly - there was no going back!

Having used Forth for so long, it is a bit like an old pair of slippers - very comfortable to work with. Once you start thinking in a Forth-like way, the limitations of other languages/development environments tend to get in the way.

I started off using a long-dead Rockwell product - a 6502 with a couple of K of Forth kernel on-chip. I could build a simple controller with this beastie in no time, and used it in a couple of lab equipment products. I have done a lot with Triangle cards - these are great for short runs and quick turn-around jobs - I still use them a regularly, though I find the TDS development environment is a bit too slow and awkward for large programs. The MPE cross-compilers (8051 and 68K) are used for several products that I designed for one client. Their fast compilation is offset by complications with the cross-compiler; it is hard to use defining words and the documentation didn't help. I understand that newer versions are better in this regard.

While contracting, I was using a heavily customised Forth based on F83, using multiple segments to squeeze as much code and data as possible into the DOS environment. I still use this Forth regularly for product maintenance and it is probably my favourite Forth despite being 16 bits and DOS - it has great debugging facilities. Unfortunately, it is very much a proprietary implementation so I could not use it for anything else without rather complicated licensing issues!

Recently, I have been playing with an evaluation copy of the MPE VFX compiler for Windows. To try it out properly, I have been writing a piece of code to retrieve data from DAT tapes. I haven't had much spare time, so things are progressing very slowly - but it looks very promising so far. The speed is amazing (especially on a fast modern PC), but debugging is a pain unless you turn off the optimiser!

"the speed is amazing"

You asked about collaborating with other Forth users. When I get stuck, I mostly turn to the 'Net - it's a fabulous resource and so vast that someone, somewhere has tackled the same problem before.

I worked on a big project for a couple of years as a contractor and I think there were five other Forth programmers working on it. But I think at least three of them no longer write Forth software. I don't think that there are many of us around, and sadly, I suspect that we are a dying breed.

The world is changing, and in the software world, more and more software is written using rapid development environments to reduce costs. The traditional Forth benefits of efficiency and instant feedback are becoming less relevant, given that even the smaller micros are being developed with the C/C++ language in mind and come with the appropriate development tools. Given enough complexity and processing power, you can make a C++ programming environment that is as immediate as a simple Forth one... I guess that the challenge to the Forth community is to keep up with all this. I worry that Forth is fast becoming an enthusiasts' language.

I don't want to be too pessimistic, as I really love using Forth. The reason for joining FIG UK was to see what other are doing, and maybe swap ideas.

My project with MPE VFX Forth is certainly the most exciting recent project - too bad I have had so little time to spare on it. The learning curve is really steep for me, as this is programming under Win32 in the raw - a new environment, and of course leaving it for weeks on end does not help. But when you get something working there is a real sense of achievement.

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 Windows or DOS 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.

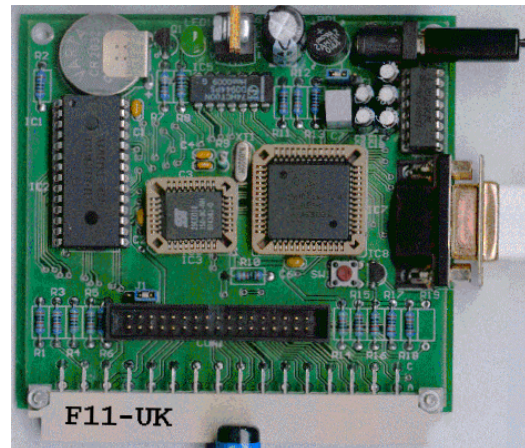
Forth running on the SBC is interactive which makes debugging and testing much easier.

Multitasking and Assembly included.

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

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



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 & 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.

Joining the F11-UK Mailing List:

Graeme Dunbar, our list moderator, has reported that the list has received a number of requests of uncertain origin to join it. Since spamming is a potential problem the banner on the List's home page has been altered to ask prospective members to identify themselves first. If you would like to join please apply to:

<http://groups.yahoo.com/group/fig-forth-uk/>

Activity on the Mailing List has been episodic, with a burst of messages kicked off recently by Garth Wilson. Jeremy has continued to refine the F11-UK kit based on feedback from our users and, following a survey posted by Graeme Dunbar, we are starting to discuss potential extender boards to further extend the variety of devices that can be connected.

From the 'Net

Chris Jakeman

Here are a number of short items from the comp.lang.forth newsgroup which are well worth re-publishing. You can explore the entire archive of the newsgroup by visiting <http://www.google.com>

BCD to Binary

Bernd Paysan posted this to comp.lang.forth in a thread about "Commenting". This routine converts a 2-digit BCD (binary-coded decimal) integer to binary. For example, in BCD, the bit-pattern 0101 0101 represents 55. To convert to binary, extract the top half-byte, multiply by 10 and divide by 16. This gives the bit-pattern 0011 0111 representing 55 in binary.

$$(5*16 * 10 / 16) + 5 = 50 + 5 = 55$$

A previous poster had already pointed out that it is quicker to use `2/` than `*/` to calculate $10/16 = 5/8 = 1/2 + 1/8$ and that's used below.

```
: *10/16 ( n1 -- n2 )      \ much faster than 10 16 */
                          2/ dup 2/ 2/ + ;
: tens    ( bcdxy -- bcdx0 ) $F0 and ;
: ones    ( bcdxy -- bcd0y ) $0F and ;
: bcd>bin ( bcd -- n )      dup tens *10/16 swap ones + ;
```

Originally `bcd>bin` was defined as a single word:

```
: BCD>BIN ( uc -- uc' )
  DUP   OF0 AND   2/ DUP 2/ 2/ +   SWAP   OF AND   + ;
```

but Bernd has split it into 3 more words to show the value of factoring into small definitions. Not only is Bernd's version self-documenting, but it makes `tens` and `ones` available for use in other BCD words.

Aligning Addresses

Anton Ertl posted this to comp.lang.forth in a thread about "Rounding up Integers to the next power of 2". Two versions were offered for aligning addresses to 4-byte boundaries with a challenge as follows:

```
: align ( n -- n' )  1- 3 OR 1+ ;
: align ( n -- n' )  NEGATE -4 AND NEGATE ;
```

"If anybody can top these two, feel free to join in!"

Anton responded to the challenge with:

```
: aligned ( c-addr -- a-addr ) 3 + -4 and ;
```

which has only two operations. Moreover the literals are often for free in native code.

Meta-Programming

Bernd Paysan responded to another challenge on the newsgroup regarding meta-programming (ie programs that write programs) based on an example in C++ using templates.

```
> Indeed, I would enjoy seeing someone tackle a Forth equivalent to  
> the Bubble Sort example given at the above link, if anyone is  
> feeling particularly "meta".
```

```
: lift ( addr -- addr' )  
  dup 2@ < IF dup 2@ swap 2 pick 2! THEN cell+ ;  
  
: bubble<N> ( n -- )  
  >r : r>  
    1 swap 1- ?DO  
      postpone dup  
      I 0 ?DO postpone lift LOOP  
      postpone drop  
      -1 +LOOP  
      postpone drop  
  postpone ;  
;  
  
4 bubble<N> bubble4
```

This line creates a word to bubble sort an array of 4 integers in memory. The usual loop is unrolled for maximum performance:

```
see bubble4  
: BUBBLE4  
  DUP LIFT LIFT LIFT DROP DUP LIFT LIFT DROP DUP LIFT DROP DROP ;
```

and may be used as follows:

```
Create test 3 , 1 , 4 , 2 ,  
test bubble4
```

For more on Template Meta-Programming in C++, see
<http://osl.iu.edu/~tveldhui/papers/Template-Metaprograms/meta-art.html>

Stashing on Return Stack

ANS Forth introduces two word pairs which work with a variable number of values on the Data Stack. These are SAVE-INPUT .. RESTORE-INPUT and GET-ORDER .. SET-ORDER.

For example, SAVE-INPUT ($x_n \dots x_1$ n --) where x_1 to x_n describe the current state of the input source specification for later use by RESTORE-INPUT.

Typically the values extracted by SAVE-INPUT or GET-ORDER are stashed on the Return Stack so that settings can be changed temporarily and later restored by retrieving the data from the Return Stack.

A useful pair of words to do this are N>R and R>N. These are not mentioned in the ANS Forth document (or Gforth or Win32Forth), but can be found in VFX and SwiftForth.

After posting a query on the newsgroup, Wil Baden provided this pair of definitions:

```
: N>R      (  $x_n \dots x_1$  n -- )( R: --  $x_1 \dots x_n$  n )
  dup      (  $x_n \dots x_i$  n n-i)
  BEGIN dup WHILE
    ROT >R
    1-
  REPEAT      ( n 0)
  DROP >R ;

: NR>      ( --  $x_n \dots x_1$  n )( R: --  $x_1 \dots x_n$  n )
  R> dup      ( n n-i)
  BEGIN dup WHILE      (  $x_n \dots x_i$  n n-i)
    R> ROT ROT
    1-
  REPEAT      (  $x_n \dots x_1$  n 0)
  DROP ;
```

Interpreted DO

Responding to a comment about executing DO .. LOOP at the command line, Wil Baden pointed out that any ANS Forth can interpret constructs which must be compiled, such as DO .. LOOP and IF .. THEN by using the word MARKER (which ANS introduced). He suggests the following scheme to add to your toolbox.

All Standard Forths can have an interpreted DO

Here's one way...

```
: :GO S" MARKER NONCE : (GO) " EVALUATE ; IMMEDIATE
: GO S" (GO) NONCE " EVALUATE ; IMMEDIATE

:GO 91 65 DO I EMIT LOOP ; GO
```

which prints

ABCDEFGHIJKLMNOPQRSTUVWXYZ ok

Garbled Messages

Jerry Avins replied to a posting about decoding messages by hand and included this personal item, which deserved repeating here.

The night that terrorists machine-gunned the waiting area at Lod airport (near Tel Aviv), I was at JFK airport waiting to pick up a friend. Once flights at Lod were resumed, people ready to go were sent out on any available flight. Some of course were dead, many in hospital.

Those of us waiting had no way to know who had survived. Who were arriving, and on what flights, were known only through teletyped passenger lists sent to the El-Al counter in JFK. A few of those lists came through with framing errors; some may remember the kind of garbling that creates.

I asked for and got the first garbled list and tried to make sense of it. By writing down the bit patterns of what I saw, imagining the missing framing bits, and guessing the shift, I come up with a few possibilities for each letter. Only one letter string seemed namelike, so I had a firm grip on the shift (until it changed), leaving only little uncertainty for each letter.

The general murmur behind me as I worked was that a harmless mystic was amusing himself. Until, that is, a woman shouted, "That's my cousin!" Then everyone wanted to see the list so far, and if whipping could have made me work faster, they would have done it. Some parts of some lists were beyond my pencil-and-paper method, but I retrieved most of the names in the end.

Other people copied the decoded names and circulated the lists. Each name was crossed off as it was recognized. The "names" not recognized came back to me for rework, and alternate interpretations made a few of them recognizable.

"Give me your ASCII, your EBCDIC, your huddled masses of bits yearning to be read ..."

Solution to Puzzle

Here is a small puzzle from Michael Gassanenko, published in the previous issue.

A word with this behaviour is listed in the ANS Forth core word-set. What is its name?

```
: X?
  2DUP > TUCK INVERT AND >R AND R> OR ;
```

Congratulations to Fred Behringer who was the first to offer a correct solution – **MAX**

Fred also provides his analysis¹ as follows.

Writing down the consecutive stack operations shows that the compound operation is taking exactly two items from the stack, neither more nor less, and is placing one, and only one, resulting item on the stack if executed.

Here is my analysis in shorthand notation, with a and b any single precision signed integer available on your machine:

$w := (a > b)$	$w = -1$ if $a > b$	$w = 0$ if $a \leq b$
a b a b	2DUP	
a b w	>	
a w b w	TUCK	
a w b $-(1+w)$	INVERT	easily seen
a w $b*(1+w)$	AND	easily seen
a w	>R	
$-a*w$	AND	easily seen
$-a*w$ $b*(1+w)$	R>	
= a	0	if $w = -1$, i.e. if $a > b$
= 0	b	if $w = 0$, i.e. if $a \leq b$
Hence,		
$-a*w \vee b*(1+w)$		OR
= a if $a > b$		
	= max(a,b)	
= b if $a \leq b$		

¹ Before he retired, Fred was Professor of Operations Research (Applied Mathematics) at Technische Universitaet Muenchen).

Note that there is but a very restricted number of ANS core words with two entries and one output. So empirically it should not be too difficult to have an "intelligent guess" and try it with an appropriate number of test values, yielding a high degree of likelihood. However, this way one can never be sure to have covered any and all cases.

Mathematics, as the essence of precise reasoning, is needed to turn belief into certainty.

Forth Inside

From the Editor:

As Forth is ideally suited to embedded applications, it tends to be invisible. Forth insiders are aware of the best-known applications, as mentioned in Forth in the UK (Nov 2000 issue and at <http://www.fig-uk.org>).

I plan to start an occasional section in the magazine entitled "Forth Inside" to give little-known applications an airing and also provide another opportunity for Forth professionals users to share their experiences.

I have some unpublished applications already on file but I would be grateful for news of any commercial applications in UK or abroad which have never had a mention in Forthwrite.

euroFORTH 2002 – Conference Report

Bill Stoddart

The papers described here are available from the FIG UK library and electronic versions may also be downloaded from <http://www.complang.tuwien.ac.at/anton/euroforth2002/papers/>

Organised by Anton Ertl and hosted by the Technical University of Vienna, the conference began with lunch in the Restaurant Habigoff, a former hat shop which has retained its 19th century interior. As well as presentations and workshops, we enjoyed walks in the town centre (just adjacent to the university), in the Vienna Woods, and in the Prater, with its famous beer garden, spectacular funfare and wooded paths. Our conference dinner was at the Vienna Town Hall.



There were eight presentations and two workshops.

David Gregg,
John Waldren

Primitive Sequences in General Purpose Forth Programs continued the work from EuroFORTH 2001 on exploration of threaded code optimisation in Gforth via "super instructions" A number of applications have been analysed to check for common instruction sequences and the

effects of optimising the most common sequences from each application has been tested on the others. Both static and dynamic analysis has been investigated, with the clear result that static analysis provides the best overall performance increase whilst dynamic analysis, which gives greater weight to frequently used sequences (e.g. inner loops), can be very effective when applied back on the application which provided the analysis.

Anton Ertl

The Evolution of Vmgen describes a virtual machine generation tool which creates C code definitions for virtual machine implementation from operation descriptions. As the tool has evolved, new features allow more user configuration, the ability to describe effects on multiple stacks, handling of "super instructions" and abstraction away from the mechanics of the threaded code implementation, e.g for describing the embedding of literal values.

Jaanus Pöial

Stack Effect Calculus with Typed Wildcards, Polymorphism and Inheritance is a framework for static type checking, refining the stack effect calculus which Jaanus introduced in the early 1990's. If a, b, c, d are stack argument lists we start with some rules for combining type signatures such as:

$$(a \text{ -- } b \ c) (c \text{ -- } d) = (a \text{ -- } b \ d)$$

meaning that if an operation which removes argument a and leaves b c is followed by one which removes c and leaves d, the overall effect is to take a and leave b and d. New ideas included a more general interpretation for the effects of operations that operate on any type (wildcards) e.g. dup swap, with rules formulated to allow for inheritance and polymorphism.

Frank Zeyda

Implementing Sets for Reversible Computation is a general implementation of sets in which sets are held as ordered arrays. Sets of integers or strings are ordered in the obvious ways. Sets of sets are ordered first by size, then by comparing successive elements. Sets of pairs are ordered by their first elements. For data items other than integers reference semantics are used such that a set is a list of references to data items on the heap.

The purpose of these sets is to support "reversible computation" efficiently - see paper below.

Federico de Ceballos

UDP/IP over Ethernet for 8-Bit Microcontrollers presents a tiny UDP stack in Forth for the CS8900 Ethernet Controller, allowing simple connectivity between micro-controllers. In a second talk, **Forth for the QNX Realtime**

Platform, Federico described an alternative operating system for the PC and

embedded targets: QNX Neutrino is a real time operating system with a relatively small micro-kernel architecture which promises simple device driver and interrupt programming as well as the ability to deploy applications on single processors or across clusters of processor without special coding. Slightly disappointed by the fact the colorForth did not run on his computer, Federico has been looking for a more appropriate environment for his Forth applications than that provided by Linux or Windows. Further developments are awaited...

Bill Stoddart

Efficient "Reversibility" with Guards and Choice

I presented a paper on mechanisms for reversible computation in the context of native compiled Forth for the Pentium. Two new features are added to Forth called choice and guard. A choice construct has the form `<CHOICE A [] B ... CHOICE>` . This makes a provisional choice between A, B etc.. The guard construct, now written as `-->`, takes a flag from the stack. If true, execution continues ahead; if false, execution reverses back to the most recent choice which has a still unexplored alternative. The presentation was illustrated with a classic Knight's Tour case study - where the goal is to visit every square on a chess board using the Knight's moves and visiting only unvisited squares.

[The purpose of Reversibility is to

Anton Ertl

Super-instructions in Gforth Anton Ertl gave a talk demonstrating the implementation of super instructions in Gforth. Instruction sequences are recognised below the level of comma (compilation). `lit 5 +` is converted to `lit+ 5`. Gforth's built-in disassembler was used to demonstrate the code generated, including cases where attempts at optimisation are frustrated by the compiler "register spilling".

We had a short workshop on parsing set expressions with demo by Daniel Ciesinger, and an extended workshop on applying static typing.

Many thanks to Anton Ertl for organising a wonderful conference and being a perfect host.

Nick Nelson and Micro-Ross have offered to organise the 19th euroFORTH at Ross on Wye with possible visits to laundries and car plants to see Forth in serious action! Many thanks to them and see you there.

Bill

euroFORTH 2003 is held in the UK. For announcements, join the mailing list at euroforth-subscribe@yahoogroups.com.

Word Completion for Quikwriter Project

Chris Jakeman

In support of the Quikwriter project, I undertook to experiment with algorithms to save keystrokes. Here is a report of progress so far.

Introduction

The aim is to make the computer easier to use by offering word completion to reduce the number of keystrokes needed to create text. This is intended to help disabled users of the Quikwriter keyboard, but it might help any keyboard user. The algorithm described here is called MinType and could be implemented by the F11-UK Forth board which modifies the stream of keystrokes being sent from the keyboard to the PC.

Experimental Platform

The experiments are carried out using Microsoft Word programmed in VBA. Programming Word is not without problems but it has several advantages, including a very fast search back through a document. It also lets us display any extra characters inserted in colour (shown below with underline added).

Since this project was started, many of us have become familiar with "predictive text" on mobile phones. This works very differently from MinType to overcome the problems of using just 10 keys to enter the whole alphabet with punctuation. Some phones also provide word completion, but MinType should do much better by working with the current document backed up by a personal vocabulary.

Non-Intrusive Operation

Several similar schemes can be downloaded for Windows PCs. The most promising ones also had awkward interfaces. In contrast, MinType succeeds in being pretty unobtrusive. To benefit from MinType, you will have to press a single well-chosen key. If you choose to ignore the "expansions" that MinType offers, then just keep typing and the expansion will disappear.

Predictive Operation

MinType watches what you type. As soon as a word has grown to 3 characters long, MinType searches for a suitable expansion and adds the expansion in red. If the expansion is appropriate, you can accept it by MinType key - Ctrl-Space seems most effective for this. If not, type another character and MinType will offer an alternative expansion. For example:

expansion
experiment
experimental

Here the user types "exp" as the stem of "experiment" and is offered "ansion". He ignores the offer and types "e". The offer is withdrawn and replaced by "riments". Again the user ignores the offer and types "r". The offer is replaced by "imental" which the user accepts by pressing the MinType key. Finally MinType appends a space character.

Including the MinType character, the user typed 6 characters and saved 7.

The current experiments are designed to record how good MinType is at offering the expansion you wanted and how many keystrokes can be saved in doing real work. We record:

- the accuracy - whether MinType could offer the correct expansion.
- the performance - the proportion of expansions that are rejected.
- the saving - the proportion of keystrokes the user saves.

Making Good Predictions

MinType uses two searches to find an expansion for the root you have typed.

The primary search looks back through the current document to find the nearest word which shares the same root. If you reject the expansion MinType has offered it notes the rejected expansion on a list and searches further back for an alternative. This "context search" is more likely to find the desired expansion than searching a standard vocabulary because it contains fewer inappropriate words. The scope of the search is limited to about 10,000 characters to avoid delays when searching large documents.

If MinType finds nothing appropriate using the primary search, it switches to searching a personal vocabulary. Words are added to this vocabulary as follows. If you accept a MinType expansion, then the expanded word is automatically appended to a file of words with the same 3-letter root. This file ends up containing a list of words that have been used at least twice. MinType searches these files backwards from the end, thus finding the more recently used words first. This personal vocabulary grows from nothing to contain the words which you used recently.

Word Endings

MinType has to treat the ending of words with special care as an English root may take multiple endings such as:

expand expands expanding expanded expanse expansion expansions

It does this by recognising common endings and offering the whole expansion with the ending highlighted and requiring two key-presses of Ctrl-Space to adopt both parts.

Increasing the Savings

This scheme offers real savings but 4 keystrokes are still required for every expanded word. This limits the savings achievable.

The simplest way to improve on this is to offer the user expansions after he types just the first keystroke. These expansions are taken from a form completed by user with single entries for the letters "A" to "Z" and "a" to "z". The user can choose to have these offered automatically (which might be intrusive) or when he presses the Ctrl-Space key.

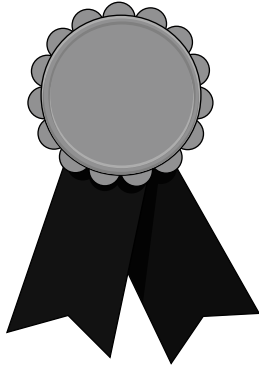
A more interesting option is to store pairs of words in the personal vocabulary. As soon as the user has accepted a word expansion, he can be offered the next word at once. This is especially useful for titles such as "Forth Interest Group" which would require just 6 keystrokes.

Writing Source

The Context Search is especially suitable for writing source code as this tends to repeat words defined earlier in the file. I look forward to trying this with Forth.

Experimental Results

MinType is currently accumulating a Personal Vocabulary based on my personal use of Word. It saves statistics as the vocabulary grows and I expect to see trends as more words are added. These will be reported in a future issue.



Nominations for the FIG UK Awards - 2002

The FIG UK Awards of 2001 were won by Chris Hainsworth and Dave Pochin. These awards are given to encourage effort and recognise achievement.

Please take the time to look back over the past year and send in your personal nominations for 2002.

Free membership

To nominate your candidate, send in a note of who, in your opinion, most deserves an award and why. The recipient of each award will receive a place in the FIG UK web-site's Hall Of Fame, a mention in Forthwrite and ***a year's free membership.***

Achievement

The Achievement Award is given to the member who has made the best contribution towards Forth during 2002. The contribution may be a presented paper, a library of code or an idea which inspires others. Whatever form it takes, the contribution must support the goals of FIG UK.

Forthwrite

The Forthwrite Award is given to the member who has made the best contribution to Forthwrite magazine during 2002. The contribution may be judged on quality of writing, tutorial potential, entertainment value or other criteria which the Forthwrite Team deem appropriate.

The awards are judged by the officers of FIG UK. All who are members on 31st Dec. 2002 are eligible (except the judges).



01733 352373
cjakeman@bigfoot.com

AGM Report

Doug Neale offered his hospitality once again – thanks Doug and to Mrs. Neale too.

Changes to Committee

No changes this year - all our officers are willing to serve for the next year.

Review of Past Year

Our web-site continues to be key resource. Jenny has added a site map and a search facility and it now provides the world's first global index of Forth source code.

Our list of web-site subscribers (who have signed up to be notified of changes) is also growing. **IRC** also continues to go well, with a good mix of regulars and visitors, including Forthers from overseas.

The **Forthwrite** Team continues to publish regularly with a wide range of material and we are grateful to the newest members of our team, Henry Vinerts and Joe Anderson.

The finances, as reported in the last issue, are still in balance and look healthy. Membership continues to be stable with around 110 members, many of whom are very active.

Plans for Next Year

The big event in 2003 is euroFORTH, to be held in the UK this time. We want FIG UK to play a role this time round and will encourage all our members to attend.

Using Wordlists for Many[

Jenny Brien and Chris Jakeman

Following the definition of `Many:` in the previous issue, here is an alternative definition, `Many[`, which makes use of a short wordlist. As wordlists often seem to be overlooked by Forth programmers, this technique is worth exploring.

An algorithm driven from a set of data is often simpler to maintain than the equivalent sequence of `if .. else .. then`s. This is usually called data-driven programming and Forth lends itself to this style. Examples include State Machines² and Brad Rodriguez BNF parser³.

`Many[` is provided to apply a word repeatedly until it reaches the terminator `"]`". For example:

```
: enums4 ( n -- n+1 ) DUP CONSTANT 1+ ;

1 Many[ enums  jan feb mar apr
                may jun jul aug      ( comment starts
                comment ends )      \ more comment
                sep oct nov dec ] drop
```

In this article, a special short wordlist is created to hold the 3 words which were recognised above and treated specially by `Many[`. These are the comment words `"(`, `"\"` and the terminator `"]`". (`"]`" is perhaps a better choice than the `";`" used by `Many:`). Some Forth users have special comment words, eg `(* .. *)` and it would be easy to add these to the list without changing the definition of `Many[` in any way.

```
wordlist constant ManyWordlist \ Create an empty wordlist

get-current \ Save current compilation wordlist
  ManyWordList set-current
  : ( postpone ( ;
  : \ postpone \ ;
  : ] -1 throw ;
set-current \ Restore saved compilation wordlist
```

² Graeme Dunbar in Forthwrite Jul/Aug/Oct 1998 and Julian Noble in Forth Dimensions Sep 1998.

³ 1988 and documented by Bernd Paysan at <http://www.jwtd.com/~paysan/screenful.html>

⁴ "enum" is a keyword in some programming languages being short for "enumeration" and meaning a numbered list.

The definition of `]` is explained later. Note that `(` and `\` do exactly what they have always done. ANS Forth provides `wordlist` to create a list identified by an integer known as a "wid". `set-current` is used to change the wordlist into which new words are compiled.

Jenny has provided two parsing words which factor out some useful functionality. `NextWord` signals the end of the input stream in a way which can be tested easily. Although ANS Forth provides a comprehensive mechanism for rewinding the input stream (`SAVE-INPUT N>R` and `NR> RESTORE-INPUT`), this is overkill when the need is occasionally to re-read a single word and Jenny provides `UnWord` for this purpose.

```

: NextWord ( -- ca u ) get next word, REFILLing if needed
  begin          \ if at end of input then ca u = 0 0
    bl word count
  dup 0= while
    2drop
  refill 0= until
    0 0          \ If the REFILL fails at end-of-file
  then
;
: UnWord ( n -- ) set >IN to before last word parsed
          \ n is length of word
    1+          \ Allow for delimiter
  >in @ swap -  \ Calc characters to unread
    0 max >in ! \ Extreme case - no blanks
;

```

Now we can define `Many[` which uses the standard `search-wordlist` to look up each word from the input stream in `ManyWordList`. It is important that `Many[` should not leave anything on the Data Stack when executing an `xt`. In our `enums` example this would give the wrong result.

```

: Many[ ( -- )
  ' >r begin          \ Keep stack clean for CATCH
  NextWord dup 0= abort" ] missing after Many[" \ End-of-file
  tuck              \ Save length for UnWord
  ManyWordlist search-wordlist if
    nip            \ Leave xt found for CATCH to execute
  else
    UnWord r@      \ Rewind input, get saved xt for CATCH
  then
  catch until r> drop
;

```

Note the use of `CATCH` instead of `EXECUTE`. When `Many[` finds the terminator `]` and executes it, `]` throws an exception which gets us out of the loop. This might be seen as an abuse of the `CATCH .. THROW` mechanism, which is provided to

handle exceptions and should not be invoked during normal operation. With this in mind, Jenny has provided an alternative using an exit flag as follows.

First, we redefine]

```
: ( ( ExitFlag - ExitFlag ) postpone ( ;
: \ ( ExitFlag - ExitFlag ) postpone \ ;
: ] ( ExitFlag - ExitFlag ) 0= ;
```

Then Many[becomes:

```
: Many[ ( -- )
  ' >r begin                                \ Keep stack clean for EXECUTE
  NextWord dup 0= abort" ] missing after Many[" \ End-of-file
  tuck                                       \ Save length for UnWord
  ManyWordlist search-wordlist if        \ If listed word
  nip false swap execute                   \ Bury exit flag
  else
  UnWord r@ execute false                 \ Rewind input, do not exit
  then
  until r> drop
;
```

Across the Big Teich

Henry Vinerts

This material was prepared for Vierte Dimension by Henry Vinerts, and printed by kind permission of Forth Gesellschaft (German FIG)

FIG Silicon Valley Chapter Meeting - Sep 2002

Greetings!

After a six-week break, SVFIG was "back-to-school" again. I believe I have mentioned before that Cogswell College (<http://www.cogswell.edu>), one of the oldest polytechnic colleges in the San Francisco area, offers a pleasant and convenient place for our meetings. It seems to me that it is one of the few remaining examples of "Small is beautiful," disregarding, of course, the ample parking space around the building.

Forthers are not any different from 'normal' people when it comes to arriving on time, and quite often even the scheduled speakers are not there at 10 o'clock, in which case Dr. Ting fills in. This time Dr. Ting was absent but Henrik Thurfjell was ready to describe the development of a product that he and John Peters had carried from the concept sketched on the back of a napkin to a demonstrably workable stage. It is an electrician's instrument for tracing circuits in buildings.

Inputs from current-sensing probes on the wires at the circuit breakers are processed through a board with an 8051 chip speaking modified AMR Forth. Voice-message outputs go by radio to the technician in the building, as he walks around changing the loads in the various circuits by turning on lights, appliances, or plugging lamps into outlets. Thus a usual two-person job can be done in less time by one. Henrik's talk generated plenty of discussion and idea-swapping with the audience to last the two hours until lunch.

In the afternoon, Tim Duncan, the director of the Digital Audio Technology department at Cogswell College, described a course program that he had proposed for the curriculum, which would favor Forth over C and LISP among the languages preferred by musicians working with digital audio. He invited SVFIG to offer help and ideas for the Forth instruction that would go with the course. I believe that thanks to Tim Duncan's partiality to Forth we have enjoyed Cogswell's hospitality for over six years now.

To top off the day, at 3pm most of us took off for a scheduled tour of the Computer History Museum, which is located next to the prominent airplane hangar in nearby Moffett Field/NASA Ames Research Park.

The Computer History Museum was established here in 1996, taking over most of the collection previously housed in a museum in Boston⁵ Presently only about 10% of the over 3500 artifacts are visible to the public, but expansion to a larger building is in the plans. Our curator turned out to be an old Forther, LaFarr Stuart (cf. Forth Dimensions, May/June 1980. p.2), who made our visit especially interesting and nostalgic. For you, my friends, I suggest a visit via <http://www.computerhistory.org/exhibits/>, and going further to highlights/ you can pick your favorites among the MITS Altair 8800, the Cray 1A, the Wehrmacht's Enigma, the 1972 Xerox PARC Alto, the Apple-1, or a number of others. Better yet, visit us at SVFIG and we'll take you to the museum.

A final news item overheard at the meeting: Chuck Moore and his wife have sold their house in Silicon Valley and moved up to Sierra City, above the snow line in the California foothills. Happy hiking and skiing, Chuck!

⁵ The guidebooks misled me during a holiday in Boston last year. I paid good money to find the exhibits missing. Glad it's got a good home - Ed.

Hello, Friends,

This report will be short, as was our October meeting. I have told you before, that without Dr. Ting and George Perry the SVFIG engine runs on fewer than the full complement of cylinders, or as we say in Yankee parlance: "It misses." Ting was out of town and George, being "under the weather," begged off from running the meeting and left as soon as he knew that Jay McKnight, being the "tallest in our midst", would keep the show going.

Anyway, John Peters held on to a few of us in the morning, showing us how he could quickly communicate over the Internet with all of the aficionados of Win32Forth. If there are any such Forthers out there who have not yet joined in polishing Forth for Windows, by all means they should get in touch with John. The afternoon group grew close to 20 to listen to Al Mitchell, who abandoned Windows two years ago. He was here with physical evidence of the Forth Stamp that he is developing, and I have written to you about that in the July Forthwrite. A few key words is all I will mention now: PIC, C8051F017 and C8051F300 from Cygnal, Linux, GForth, GUI in TCL, amrBASIC, fuzzy logic. Al's Forth Stamp is up to 500 times faster than Parallax's Basic Stamp. Al's own AMRForth rev.6 will be on his website in a month, and you should visit <http://www.amresearch.com/> if you wish to know more about BASIC running on Forth.

We quit the meeting early to go to the Vintage Computer Fair at nearby Moffett Field. Perhaps a couple of dozen exhibitors there with all kinds of early machines. My first look at the APL keyboard and KIM-1 running a chess program. I met Hans Franke next to a sign drawing attention to <http://www.gfhr.de/>

(Gesellschaft fuer historische Rechenanlagen), and he told me all about the good computer stuff one can find on Schillerstrasse in Munich. I'll have to ask Dr. Behringer to tell me about that.

That's it for this month. Tschuess!

Henry

FIG Silicon Valley Chapter Meeting - Nov 2002

Hello, Forth friends!

"The second annual FIG Convention was a big success with 250 FORTH users, dealers, and enthusiasts attending a full day of sessions on FORTH and FORTH-related subjects. The Villa Hotel in San Mateo, CA, provided the setting this year."

I am quoting from Volume II No.5 of Forth Dimensions, the issue for January/February 1981. Twenty-two years later the Villa Hotel is still there, the tradition to hold Forth Days in November at Thanksgiving time is still carried on by Silicon Valley Forth Interest Group, and, although the group of attendees has shrunk to about a tenth of the above-mentioned size, Charles Moore still delights them with talks about his brainchild and, if I may say so, brain-grandchildren, such as ColorForth.

I regret that I could attend only the morning session of the SVFIG Forth-Day meeting on November 16th and had to miss Dr. Ting's traditional lunch-time barbecue, as well as Chuck's Fireside chat at the end of the day. I did hear Jeff Fox's presentation on his and Soren Tiedemann's Forth GUIs and OSs, both of which duplicate most of the common Windows features with 400 to 600 words of Forth code. I quote Jeff: *"It is faster to write your own code than go to Windows and try to find the desired function. When you do all in Forth yourself, you have control over all abstractions. Don't use more layers of abstractions than you need."* Readers can find more information

about Jeff's Aha at <http://www.ultratechnology.com/aha.htm>
and Soren's Allegra compilers at <http://www.planet-interkom.de/soeren.tiedemann>

Dr. Ting followed Jeff with a demonstration of F# (F-sharp, as in music) running in Chinese on a Windows XP machine. The platform has been chosen by the Taiwanese, with whom Dr. Ting has collaborated for some time now, but F# comes from his own eForth, after he had tried and rejected Win32Forth as being too complicated.

The project involves generation of Chinese characters and development of Forth that non-English speaking Chinese, especially young children, can use as introduction to programming and to Forth itself. By redefinition with "alias" Ting has given Chinese names to all Forth words that are needed, so that even 3rd-grade schoolchildren can already construct their own programs. A great tool for those who are handicapped by lack of English! To quote Dr. Ting: "Forth is the best language for any foreign tongue."

I wish I could end with a quote from the day's talk by Chuck, but, since I missed it, let me repeat what is written in the Forth Dimensions of 1981:

"Following a panel session on the FORML conference at Asilomar, Charles Moore of FORTH, Inc., closed the morning session with a reminder that it is the very flexibility and versatility of FORTH which will cause more problems as more people become acquainted with it." I am hoping that this will generate some comments from our readers.

Sincerely,
Henry



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

Vierte Dimension 3/2002

Joe Anderson

Joe provides a look at the latest issue of the German FIG magazine. To borrow a copy or to arrange for a translation of an individual article, please call Joe.

Editorial

Friederich Prinz

Instead of an editorial, this time something light-hearted on operating systems.

Readers' Letters

Klaus Sobawa

Eleven inputs about this and that, from Leg-REX ("hard" and "soft") about the estimated computational capacity of the Universe as a "system", up to hiring of micro-controllers.

News from FIG Silicon Valley

Henry Vinerts

Three reports on the Forth activities of the US continuation group from Silicon Valley and a request from the Editor of Vierte Dimension (Fritz Prinz) to Henry, to continue as heretofore with his stories depicted in such personal colours.

MicroCore

Klaus Schleisiek

Klaus.Schleisiek@hamburg.de

"An open-source, scalable, dual-stack, Harvard processor synthesiable VHDL for FPGAs".

RCX on line

Adolf Krüger and

Michael Kalus

It was really not easy to see why the Lego-RCX had to be controlled over an IR transmitter, if two wires should suffice. In Martin Bitter's school activities more and more RCXs had

to be loaded at the same time, and then each interfered with the others. The two authors have developed a splendid circuit with an opto-coupler, which fits into the RCX and solves the problem.

The Ushi Working Group

Willem Ouwerkerk

Presentation by Willen, the president of the Dutch Forth Group, on the recent Forth conference in Garmisch-Partenkirchen about the Dutch home-built robot project. Translated by Fred Behringer.

TO - A function with many possibilities

Albert Nijhof

Presentation by Albert, the editor of the Dutch Forth magazine "Vijgeblaadje", on the recent Forth conference in Garmisch-Partenkirchen about a generalisation of the concepts understood by VALUE in ANS-Forth. For the extension DOES> has to be adapted. Translated by Fred Behringer.

A call to Assembly (2/3)

Julian Noble

The second part of a tutorial by the author on the easy use of assembler in Forth. Translated by Fred Behringer. The article appeared in its original version in Forthwrite 114.

VD List of Contents (Part II)

Fred Behringer

behringe@mathematik.tu-muenchen.de

The second part of Fred's comprehensive list of all articles that have been published up to now in Vierte Dimension (6 pages). The list is in subject groups, and within each subject group arranged in date order of appearance, and containing in addition authors and title. The whole lay-out of this list has been modelled on the Index List from Forthwrite.

Instant

Jens Wilke

Contribution from the author on the Forth conference in Garmisch-Partenkirchen:
The Instant project is a development of the cross-compiler, that enables ANS-Forth compatible programmes to run via a cross-compiler without having to be modified.

FINDRAMD.COM - Assembler programming in Forth

Fred Behringer

behringe@mathematik.tu-muenchen.de

This is once more entry in Fred's "Column for language migrants". The program FINDRAMD.EXE is to be found on a Windows98 emergency boot diskette and it enables the assignment of a drive letter to a created RAM-disk. The hard disk on the machine may have several partitions and in order to copy necessary files to the emergency RAM-disk one must

know the relevant drive letters. FINDRAMD.EXE does all this and is 6855 bytes long. FORTH-assembler can be quickly used to generate a program FINDRAMD.COM which is only 20 bytes long ! (It should in fairness be pointed out that the FINDRAMD.EXE program does include error trapping and error messages.)

MuP21/F21-Bootprocess

Soeren Tiedemann

The author discusses the installation, memory map and bus-logic, 8-bit bootmode, boot routines, and software of the above processor in what is intended to be a series of articles.

Pontifex

Friederich Prinz
Friederich.Prinz@t-online.de

Pontifex refers to the building of a bridge between Heaven and Earth. Fritz lightheartedly discusses a demo version of a program for the "virtual construction of bridges" for would-be bridge constructors available by download from <http://www.chronilogic.com> .



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.

Forthwrite Index

Jenny Brien maintains a set of 3 indexes to Forthwrite on the FIG UK web site and updates them with each new issue. These indexes are sorted by date, by author and by subject going back to 1990. The subject index is published in the magazine annually (below), with this year's new entries highlighted.

Back issues of Forthwrite may be borrowed from the Library without charge, so this is a good way to catch up on topics of special interest. If you spot a topic that has not been adequately covered, please drop a line to the Editor.

Forthwrite Subject Index 1990-2002

Subject	Author	Date	Title
algorithms	Hersom, Ed	92-10	Advanced course
algorithms	Charlton, Gordon	93-04	Backwards (psychic programming)
algorithms	Hersom, Ed	93-04	Trees & splines
algorithms	Hill, Will	93-06	Solving with Newton-Raphson
algorithms	Payne, John	93-12	Approximate pattern matching
algorithms	Bennett, Paul	94-06	Fuzz, fibs and forms
algorithms	Pochin, David	94-10	First attempts at Fuzzy Logic
algorithms	Bennett, Paul	95-06	Fractionally angular
algorithms	Charlton, Gordon	95-06	Easter Sunday
algorithms	Ramsay, Chris	99-08	Forth and Genetic Programming
applications	Green, Roedy	90-08	Abundance (database)
applications	Brien, Jack	91-02	Typing tutor (code)
applications	Kendall, Les	91-02	Terminal emulator for PC (code)
applications	Smith, Graham	91-02	Logic gates
applications	Grey, Nigel	91-06	Big Blue on the move IBM CAD (review)
applications	Franin, Julio	92-08	Torsion measurement system
applications	Stephens, Chris	93-08	Seven thousand networked micros
applications	Anderson, Joe	98-07	Forth In Space
applications	Trueblood, Mike	99-11	Radio Clock
applications	Bennett, Paul	00-08	Logging on - statistically speaking
applications	Paysan, Bernd	00-08	A Web-Server in Forth
applications	Matthews, John	01-01	Forth as Preferred Development Environment
applications	Kendall, Les	01-01	XML and Forth
applications	Wong, Leo	01-04	Solving a Riddle
applications	Brien, Jenny	01-07	"Quikwriter" proposal

applications	Anderson, Joe	01-07	Forth for NEAR Spacecraft
applications	Fowell, Jeremy	01-09	"Quikwriter" Project Launch
applications	Brien, Jenny	02-01	JenX revisited - A Simple XML Parser
applications	Brien, Jenny	02-04	Flickwriter Project
applications	Paysan, Bernd	02-09	Competitive Programming
arithmetic	Jakeman, Chris	90-12	A high-level /MOD (code)
arithmetic	Preston, Philip	91-02	Multi-cell arithmetic (code)
arithmetic	Filbey, Gil	91-04	Tutorial
arithmetic	Haley, Andrew	91-04	Function approx. by Chebyshev series
arithmetic	Filbey, Gil	91-12	Mixed point arithmetic (tutorial)
arithmetic	Payne, John	91-12	Fixed point arithmetic (word set)
arithmetic	Filbey, Gil	92-02	Mixed point arithmetic (tutorial)
arithmetic	Filbey, Gil	92-04	Mixed point arithmetic (tutorial)
arithmetic	Brown, Jack	92-10	Floored v symmetric division (tutorial)
arithmetic	Filbey, Gil	93-02	Floating point
arithmetic	Filbey, Gil	95-02	Cube roots
arithmetic	Bennett, Paul	97-02	From the 'Net - Square Roots (code)
arithmetic	Hersom, Ed	98-07	Quad (Fixed-Point) Arithmetic
arithmetic	Behringer, Fred	00-04	32-bit GCD without Division
arithmetic	Pochin, Dave	00-06	Floating Decimal Fudge
arithmetic	Jakeman, Chris	02-09	Linear Interpolation
arrays	Jakeman, Chris	90-08	Arrays and records (code)
arrays	Brien, Jack	92-02	Ways with arrays (code)
assembly	Tanner, P.	96-05	Linking machine code modules with Forth
block tools	Filbey, Gil	91-02	Bits and loading blocks (tutorial)
block tools	Hainsworth, Chris	91-02	Editing blocks (tutorial)
block tools	Charlton, Gordon	94-04	One-screen library load (code)
bons mots	Bezemer, Hans	97-08	Th
bons mots	Eckert, Brad	97-08	On Off On? Off?
bons mots	Luke, Gary	97-08	Tally
bons mots	Hersom, Ed	97-11	NVars [H] [D]
bons mots	Payne, John	97-11	3rd Swap@ Sgn #>ASCII
bons mots	Wenham, Alan	97-11	Z
bons mots	Elvey, Dwight	98-01	Setting bits with MASK
bons mots	Wenham, Alan	98-01	Printing binary with .SB U1B. U2B.
bons mots	Hoyt, Ben	98-03	PLACE is to COUNT as ! is to @
bons mots	van Norman, Rick	98-03	MANY for debugging
bons mots	Wong, Leo	98-05	Laying down values with COURSE
concurrency	Charlton, Gordon	91-10	Co-routine monitors (code)
concurrency	Charlton, Gordon	94-04	One-screen concurrent Forth (code)
control flow	Charlton, Gordon	90-04	Universal delimiter (code)

control flow	Brien, Jack	91-02	Extended ANS structures (F83 code)
control flow	Bennett, Paul	91-04	High level FOR..NEXT (code)
control flow	Carpenter, R.H.S.	92-12	Flow-charting method
control flow	Preston, Philip	93-06	Shortcuts and drop-outs
control flow	Brien, Jack	94-06	Extending ANSI control structures
control flow	Brien, Jack	95-06	Portable control structures
control flow	Charlton, Gordon	95-06	Trouble with DO
control flow	Jakeman, Chris	96-05	If and begin - ANS style
database	Filbey, Gil	91-08	FIG UK database (tutorial)
database	Filbey, Gil	91-08	FIG UK database (tutorial)
design	Payne, John	90-12	Simpler Forth (comment)
design	Brien, Jack	91-10	Return stack ENTER ISNOW and aliasing
design	Thomas, Reuben	92-06	Forth lifestyle
design	Hersom, Ed	92-10	NVARS
design	Charlton, Gordon	93-04	Upside down
design	Smart, Mike	93-10	Computer Shopper Programmer's Challenge
design	Matthews, John	94-02	On his September lecture
design	Bennett, Paul	94-08	Taking exception ...
design	Hersom, Ed	94-08	Simple user interface
design	Flynn, Chris	94-10	Numerical input
design	Allwright, R.E.	95-06	Pagination
design	Jakeman, Chris	95-06	From the 'net
design	Telfer, Graham	96-05	The specification method hunt
design	Brien, Jack	99-01	Working with Wordlists
design	Brien, Jack	99-06	Handling Literals
design	Telfer, Graham	99-06	Skeletons - Designing a Recursive Application
design	Telfer, Graham	02-07	Expanding the Use of the Stack
dynamic data	Charlton, Gordon	90-04	Dynamic words (code)
dynamic data	Charlton, Gordon	94-06	Work, rest and play
editing tools	Jakeman, Chris	90-02	Search and replace 1/2 (code)
editing tools	Jakeman, Chris	90-04	Search and replace 2/2 (code)
editing tools	Lake, Mike	91-02	Full screen editor in one screen (code)
editing tools	Brien, Jack	95-06	Full screen editor
editorial	Hainsworth, Chris	91-04	Forthtalk and EuroFORML report
editorial	Jakeman, Chris	92-08	Soapbox - "Do it yourself"
editorial	Payne, John	92-12	Fat, thin or inflatable?
editorial	Wilson, R.J.	93-06	Seeing trees in the wood
editorial	Rush, Peter	95-02	Honeywell Forth Bulletin Board
editorial	Jakeman, Chris	96-05	From the 'net - perceptions
editorial	Hersom, Ed	96-07	Why Forth?
editorial	Jakeman, Chris	96-11	Sell-by-date

editorial	Jakeman, Chris	97-02	FIG UK joins the World Wide Web
editorial	Jakeman, Chris	97-02	Welcome Disk
editorial	Brien, Jack	97-08	FIG UK Web Site
encryption	Greenwood, Mike	98-03	File Encryption
exceptions	Charlton, Gordon	91-04	CATCH and THROW (code)
exceptions	Jakeman, Chris	93-10	Portable CATCH and QUIT (code)
exceptions	Jakeman, Chris	93-10	Using CATCH and QUIT (code)
FANSI project	Bennett, Paul	90-06	Time for a new FIG Forth (comment)
FANSI project	Charlton, Gordon	90-10	High-level /MOD using recursion (code)
FANSI project	Charlton, Gordon	90-10	High-level multiply (code)
FANSI project	Flynn, Chris	90-10	Discussion on REQUIRES
FANSI project	Hainsworth, Chris	90-10	FANSI that (proposal)
FANSI project	Bennett, Paul	90-12	FANSI environs (proposal)
FANSI project	Flynn, Chris	90-12	Response to design proposals (comment)
FANSI project	Payne, John	90-12	Response to design proposals (comment)
FANSI project	Charlton, Gordon	91-06	FANSI definitions (code)
FANSI project	Charlton, Gordon	91-08	FANSI bloomers (code)
FANSI project	Payne, John	91-08	Notes on FANSI (code)
FANSI project	Bennett, Paul	91-10	Report on FANSI
FANSI project	Charlton, Gordon	91-12	FANSI vocabularies (proposal)
FANSI project	Brien, Jack	92-02	FANSI (comment)
FANSI project	Payne, John	92-02	FANSI (comment)
FANSI project	Preston, Philip	92-02	FANSI (comment)
FANSI project	Payne, John	92-12	FANSI QUIT
file tools	Brien, Jack	91-02	Loading dependant source (code)
file tools	Jakeman, Chris	93-02	File access, part 1 (code)
file tools	Jakeman, Chris	93-04	File access, part 2 (code)
file tools	Jakeman, Chris	93-06	File access, part 3 (code)
file tools	Jakeman, Chris	93-08	File access, part 4 (code)
file tools	Brien, Jack	95-10	Hierarchical screen filing
file tools	Wong, Leo	98-10	ANS File Words for Pygmy Forth
file tools	Behringer, Fred	99-01	ANS File Words for Turbo Forth - 1
fractions	Charlton, Gordon	90-02	Vulgar words (code)
fractions	Wilson, R.J.	90-04	Rational numbers (code)
fractions	Wilson, R.J.	90-06	Transcendental rationale (code)
fractions	Charlton, Gordon	90-10	Rational approximation (comment)
futures	Jakeman, Chris	94-08	Telescript (comment)
futures	Jakeman, Chris	94-10	Some future directions (editorial)
futures	Jakeman, Chris	96-11	Forth and Java (comp.lang.forth)
futures	Pelc, Stephen	99-11	FIG UK - The Next 20 Years
futures	Jakeman, Chris	02-01	The Semantic Web

graphics	Filbey, Gil	90-04	Plotting spirals (tutorial)
graphics	Charlton, Gordon	92-06	Turtle graphics
graphics	Payne, John	92-08	Flood fill
graphics	Charlton, Gordon	93-08	Drawing a line
graphics	Charlton, Gordon	93-10	Not drawing a line
graphics	Payne, John	93-10	How Bresenham's line drawing alg. works
graphics	Pochin, Dave	00-11	"BLT is not a Sandwich"
hardware	Koopman, Philip	90-10	RTX 4000 (publicity)
hardware	Fowell, Jeremy	92-08	P20 chip, part 1/2
hardware	Fowell, Jeremy	92-10	P20 chip, part 2/2
hardware	Bennett, Paul	96-07	Chuck's chips
hardware	Fowell, Jeremy	99-01	FIG UK Hardware Project
hardware	Fowell, Jeremy	99-04	FIG UK Hardware Project - Progress
hardware	Heuvel, Leendert	99-04	The 'Egel Coursebook
hardware	Fowell, Jeremy	99-08	FIG UK Hardware Project - Progress
hardware	Fowell, Jeremy	99-11	FIG UK Hardware Project - Progress
hardware	Fowell, Jeremy	00-01	F11-UK Hardware Project - Progress
hardware	Fowell, Jeremy	00-04	F11-UK Hardware Project - Progress
hardware	Fowell, Jeremy	00-08	F11-UK Hardware Project - Launch
hardware	Jakeman, Chris	01-01	F11-UK Hardware Project - Progress
hardware	Jakeman, Chris	01-04	F11-UK Hardware Project - Progress
history	Rather, Elizabeth	95-04	The evolution of Forth
history	Rather, Elizabeth	95-12	The Forth approach to operating systems
history	Hainsworth, Chris	99-01	Forthwrite Issue No. 1 revisited
history	Powell, Bill	99-01	The Birth of FIG UK
history	Behringer, Fred	99-11	Swap Dragon
history	Brien, Jack	99-11	FIG UK - The Last 20 Years
history	Jakeman, Chris	00-01	Did you Know? - EasyWriter
history	Jakeman, Chris	00-04	From the 'Net, Forth for Novell
history	Crook, Neal	00-06	The Canon Cat
history	Jakeman, Chris	00-06	Did you Know? - Forth OS
history	Jakeman, Chris	00-08	Computer Conservation
history	Jakeman, Chris	00-08	Did you Know? - Forth v C
history	Jakeman, Chris	00-11	Did you Know? - Open Firmware
history	Jakeman, Chris	01-09	Did you Know? - smart cards
history	Jakeman, Chris	01-11	Did you Know? - large Forth projects
history	Jakeman, Chris	02-04	Did you Know? - Forth Help Nobel Prize Winners
history	Moore, Charles	02-09	Forth - The Early Years
humour	Payne, John	90-12	A program that works the French way
humour	Smith, Graham	95-06	Book titles
humour	Jakeman, Chris	96-05	From the 'net - a drinking song

humour	Allwright, Ray	98-05 A Story of Cowboys
humour	Gassanenko, Michael	02-01 From the 'Net - the non-English view
interfacing	Robinson, Dave	91-08 Mouse handling (F83 code)
interfacing	Bennett, Paul	98-05 Reading the World - 1
interfacing	Bennett, Paul	98-07 Reading the World - 2
interfacing	Bennett, Paul	98-10 Writing the World - 1
internals	Bennett, Paul	99-01 Writing the World - 2
internals	Hainsworth, Chris	90-02 Kiss and run (exploring F-PC)
internals	Charlton, Gordon	91-02 A replacement for DO .. LOOP (code)
internals	Flynn, Chris	91-06 Forth engine on 68000
internals	Bennett, Paul	92-10 Top-down development of a Forth system
internals	Bennett, Paul	93-04 QUIT, the story continues...
internals	Preston, Philip	93-12 RatForth - ANS on F83
internals	Preston, Philip	94-02 Ratforth revised etc.
internals	Preston, Philip	94-06 Redefining colon
internals	Preston, Philip	94-10 Simulating EVALUATE
internals	Preston, Philip	95-10 Variables, values & locals
internals	Wenham, Alan	95-12 Meandering Forth
internals	Brien, Jack	97-08 Building a new inner interpreter
internals	Allwright, Ray	98-03 From the 'Net - Minimal word sets
internals	Allwright, Ray	99-04 From the 'Net - Turnkey Apps and Docs
internals	Tasgal, John	00-04 An Introduction to Machine Forth
internals	Brien, Jenny	01-09 Treating Data as Source
interpreting	Jakeman, Chris	95-10 From the 'net - text interpreter
interpreting	Brien, Jack	96-11 Implementing an outer interpreter
interview	Moore, Charles	99-06 1xForth
interview	Lawless, Jim	01-11 An interview with Tom Zimmer
interview	Morrison, George	01-11 Charles Moore interview on Slashdot
library	Hainsworth, Sylvia	91-04 FIG UK library bulletin
library	Jakeman, Chris	96-11 Library assets
library	Hainsworth, Sylvia	98-05 Purchases and current publications
logic	Behringer, Fred	01-07 Arithmetized Logic in Forth
MCFAs	Brien, Jack	90-08 Comment
objects	Jakeman, Chris	94-12 Objects and so forth
objects	Jakeman, Chris	98-11 OOF - A Minimal Approach
objects	Prinz, Friederich	99-01 Counting Fruits the Classic Way
objects	Jakeman, Chris	02-01 A Safer Mini-OOF
performance	Jakeman, Chris	98-01 From the 'Net - Speed Demons
permutations	Charlton, Gordon	90-02 Permutations, a new algorithm (code)
permutations	Hersom, Ed	91-10 Permutations (code)

permutations	Hersom, Ed	92-04	Permutations/combinations
permutations	Baden, Wil	00-11	Permutation by Transposition Sequence ACM 115A
permutations	Jakeman, Chris	00-11	Simple Forth Permutations
permutations	Behringer, Fred	01-04	Generating Combinations
presentation	Brien, Jack	90-02	Locals and more (discussion)
presentation	Matthews, Keith	90-12	Stack diagrams (explored)
presentation	Brien, Jack	91-02	GIST for indexing source (code)
presentation	Bennett, Paul	91-06	Manual documentation (code)
presentation	Charlton, Gordon	93-12	StackFlow
presentation	Brien, Jack	94-10	Readable Forth
presentation	Tanner, P.H.	94-12	Post indentation
presentation	Charlton, Gordon	97-02	From the 'Net - StackFlow
probability	Filbey, Gil	90-12	Probability of common birthdays
probability	Filbey, Gil	90-12	Random thoughts on probability
probability	Payne, John	90-12	Probability of common birthdays
publications	Haley, Andrew	91-12	FORML 87, 88 & 89 (review)
puzzles	Hainsworth, Chris	90-06	Forth brain teasers
puzzles	Charlton, Gordon	90-12	Name that word
puzzles	Charlton, Gordon	91-02	Puzzle answers (code)
puzzles	Filbey, Gil	92-10	Tethered goat puzzle, part 1/2
puzzles	Filbey, Gil	92-10	Tethered goat puzzle, part 2/2
random nos.	Filbey, Gil	93-06	Visualising random numbers on Apple II
random nos.	Jakeman, Chris	93-06	Random numbers
random nos.	Filbey, Gil	93-08	Testing for randomness
random nos.	Payne, John	93-08	More on random numbers
review	Charlton, Gordon	94-10	Riding the wild 'net
review	Charlton, Gordon	95-02	Report from EuroForth '94
review	Bennett, Paul	97-11	EuroForth '97 Conference
review	Wenham, Alan	98-01	Vierte Dimension
review	Fowell, Jeremy	98-05	Forth Programmers' Handbook
review	Jakeman, Chris	98-05	Genetix - The Inside Story
review	Payne, John	98-07	FORML Proceedings 94 & 95
review	Flynn, Chris	98-10	A Hard Day Garbage Collecting
review	Jakeman, Chris	98-10	jeForth
review	Bennett, Paul	98-11	euroForth '98 Conference
review	Wenham, Alan	99-01	Vierte Dimension
review	Anderson, Joe	99-06	Forth for Virtual Reality
review	Wenham, Alan	99-11	Vierte Dimension
review	Jakeman, Chris	00-01	FIG UK 20th Anniversary Reunion
review	Wenham, Alan	00-01	Vierte Dimension 4/99
review	de Ceballos,	00-04	21st FORML Conference

	Federico	
review	Wenham, Alan	00-04 Vierte Dimension 1/00
review	Wenham, Alan	00-06 Vierte Dimension 2/00
review	Jakeman, Chris	00-08 euroForth '99 Conference
review	Wenham, Alan	00-11 Vierte Dimension 3/00
review	Jakeman, Chris	00-11 Forth in the UK
review	Wenham, Alan	01-01 Vierte Dimension 4/00
review	Ives, Robert	01-01 "Forth Application Techniques"
review	Oakford, Howerd	01-01 euroFORTH 2000 Conference report
review	Jakeman, Chris	01-07 Gesellschaft 2001 Conference report
review	Abrahams, David	01-07 "Extreme Mindstorms" book
review	Bennett, Paul	01-07 3 Free Forths and an OS too!
review	Wenham, Alan	01-09 Vierte Dimension 2/01
review	Wenham, Alan	01-11 Vierte Dimension 3/01
review	Vinerts, Henry	02-01 Across the Big Teich
review	Jakeman, Chris	02-04 From the 'Net
review	Oakford, Howerd	02-04 euroFORTH 2001 Conference report
review	Vinerts, Henry	02-04 Across the Big Teich
review	Wenham, Alan	02-04 Vierte Dimension 4/01
review	Behringer, Fred	02-07 German FIG Annual Conference
review	Fennema, Boris	02-07 "Write Your Own Programming Language Using C++"
review	Fennema, Boris	02-07 "The Practice of Programming"
review	Moore, Charles	02-07 An Interview with Chuck Moore
review	Vinerts, Henry	02-07 Across the Big Teich
review	Wenham, Alan	02-07 Vierte Dimension 1/02
review	Vinerts, Henry	02-09 Across the Big Teich
review	Rodriguez, Brad	02-09 Choosing Forth
roots	Wilson, R.J.	90-08 Root of rational numbers (code)
roots	Charlton, Gordon	90-10 Square root (code)
roots	Trapp, John	91-02 Square-roots for double/floating point
roots	Brien, Jack	97-11 From the Net - More on square roots
roots	Behringer, Fred	98-03 Square roots once more
roots	Behringer, Fred	98-05 Cubic roots without division
roots	Jakeman, Chris	00-04 Cube Roots Again
roots	Jakeman, Chris	00-04 From the 'Net - Cube Roots
roots	Jakeman, Chris	00-06 From the 'Net, Cube Roots
searching	Charlton, Gordon	90-12 A faster string search (code)
searching	Charlton, Gordon	91-10 A binary search (code)
searching	Hersom, Ed	91-12 Recursive BINSEARCH (code)
searching	Charlton, Gordon	93-02 Shift-AND string search (code)
searching	Charlton, Gordon	94-02 Best string search (code)

searching	Jakeman, Chris	95-06	Linear search
sets	Charlton, Gordon	90-06	Set manipulation (code)
sorting	Charlton, Gordon	90-08	Radix, an extravagant sort (code)
sorting	Charlton, Gordon	90-10	Sorting strings with qsort (code)
sorting	Charlton, Gordon	91-10	Heapsort (code)
stacks	Preston, Philip	92-12	Stacking fillers - stacks & write-only
stacks	Charlton, Gordon	94-04	Stacrobaticus exotica
stacks	Filbey, Gil	94-08	Stacks (tutorial)
stacks	Jakeman, Chris	95-08	Stack manipulation
stacks	Joseph, Neville	95-10	Stack manipulation
stacks	Barr, Stan	95-12	A third stack
stacks	Hersom, Ed	97-11	Multi-precision Stack Operators
standards	Jakeman, Chris	91-06	Portable code (code)
state machines	Charlton, Gordon	90-10	Variables for state machines (code)
state machines	Dunbar, Graeme	98-07	Finite State Machines 1/3
state machines	Dunbar, Graeme	98-10	Finite State Machines 2/3
state machines	Dunbar, Graeme	99-08	Finite State Machines 3a
strings	Leibniz, David	91-02	String stack routine (code)
strings	MacLean, Ruaridh	91-02	High level DIGIT (tutorial)
strings	Charlton, Gordon	91-04	A string pattern matcher (code)
strings	Payne, John	92-04	Text processing
strings	Preston, Philip	92-10	TACK and BLOCKL
strings	Charlton, Gordon	93-04	ANSI and portability - STRLIT (code)
strings	Brien, Jack	93-06	Comment on Blockl & Tack
strings	Charlton, Gordon	93-06	Similarity
strings	Jakeman, Chris	95-12	From the 'net - please
strings	Brien, Jack	96-07	String handling
strings	Jakeman, Chris	97-02	Pattern matching - 1/3 (tutorial)
strings	Jakeman, Chris	97-08	Pattern matching - 2/3 (FoSM with Forth)
strings	Jakeman, Chris	97-11	Pattern matching 3/3 (Rex)
strings	Borrell, Richard	98-03	Deferred Language Translation
strings	Oakford, Howerd	98-11	Multiple Language Programs Made Easy
structures	Brien, Jack	98-01	Building Forth Structures
systems	Green, Roedy	90-08	BBL Forth (review)
systems	Bennett, Paul	92-02	Pygmy Forth (review)
systems	Tanner, Philip	92-04	As in a glass darkly
systems	Hersom, Ed	93-02	Pocket Forth (review)
systems	Tanner, P.H.	93-06	URForth (review)
systems	Payne, John	95-02	A 32-bit Forth for Windows (review)
systems	Stephens, Chris	95-02	Forth for the Transputer (review)
systems	Behringer, Fred	97-08	Forth for the Transputer

systems	Worthington, Thom.	98-01	Aztec - A Forth For Windows '95
systems	Besemer, Hans	98-05	4th - The Alternative Compiler
systems	Jakeman, Chris	99-01	Web Forth Project
systems	Lancaster, Garry	99-04	Forth for the Z88
systems	Jakeman, Chris	99-06	Web Forth Project
systems	Ouwerkerk, Willem	99-08	ByteForth for MCS51 cpu's
systems	Tasgal, John	00-06	An Introduction to Color Forth
systems	Tasgal, John	00-06	The BMP Example
systems	Zimmer, Tom	01-09	4-bit Forth
systems	Eckert, Brad	01-11	Tiny Open Firmware
systems	Myneni, Krishna	02-04	Special Features of kForth 1/2
systems	Myneni, Krishna	02-07	Special Features of kForth 2/2
tools	Jakeman, Chris	90-06	Patch programming aid (code)
tools	Jakeman, Chris	90-10	Run-time operators (code)
tools	Preston, Philip	91-12	ALIAS ALIAS ALIAS (F83 code)
tools	Jakeman, Chris	92-12	Also and -Also (code)
tools	Charlton, Gordon	93-04	Wrong way round!
tools	Bennett, Paul	93-06	+MOD! (LOG?) and commenting words
tools	Brien, Jack	93-10	Utilities for F83 on Amstrad PCW
tools	Jakeman, Chris	93-12	Shell (code)
tools	Bennett, Paul	94-02	Spooling and browsing
tools	Jakeman, Chris	94-02	.Call and Assert (code)
tools	Jakeman, Chris	94-04	Check (code)
tools	Flynn, Chris	94-06	Conditional compilation
tools	Preston, Philip	94-08	More fun with EVALUATE
tools	Charlton, Gordon	94-12	16-bit cyclic redundancy checksums
tools	Franin, Julio	95-02	MC51 Forth debugging
tools	Smith, Graham	95-06	MARK
tools	Jakeman, Chris	95-08	Limit variables (code)
tools	Abrahams, David	95-10	General purpose utilities for F-PC
tools	Stott, Barrie	97-02	Stack checking (code)
tools	Jakeman, Chris	99-06	From the 'Net - Iterative Interpretation
tools	Wong, Leo	02-09	Iteration with Many:
tutorial	Charlton, Gordon	92-04	Two geese and a car
tutorial	Brown, Jack	92-06	An indefinite loop example
tutorial	Filbey, Gil	92-12	Escape codes and printing
tutorial	Filbey, Gil	93-02	A conjuring trick
tutorial	Hainsworth, Chris	93-02	Shallow end
tutorial	Filbey, Gil	93-04	Some old words revisited
tutorial	Filbey, Gil	93-10	Floating point
tutorial	Charlton, Gordon	93-12	Create .. does> ..

tutorial	Filbey, Gil	93-12	Postfix
tutorial	Filbey, Gil	94-02	Editorial & Tu
tutorial	Filbey, Gil	94-12	Floating point
tutorial	Filbey, Gil	95-08	Immediacy
tutorial	Filbey, Gil	95-10	Editorial
tutorial	Telfer, Graham	98-07	Wondrous Numbers
tutorial	Jakeman, Chris	98-11	jeForth Project
tutorial	Pochin, Dave	99-01	Forth for the New Year
tutorial	Pochin, Dave	99-01	Guide to Getting Started
tutorial	Pochin, Dave	99-04	Getting Stuck Into Win32Forth
tutorial	Pochin, Dave	99-08	Figuring it out with Win32Forth
tutorial	Jakeman, Chris	99-11	Clock Challenge
tutorial	Pochin, Dave	00-01	"See Win32Forth scroll the Window"
tutorial	Jakeman, Chris	00-01	Clock Challenge - 2nd installment
tutorial	Brien, Jack	00-04	All you need to know about STATE, IMMEDIATE and POSTPONE
tutorial	Pochin, Dave	01-04	Six Easy Fonts
tutorial	Noble, Julian	01-09	A Call to Assembly 1/3
tutorial	Pochin, Dave	01-09	Win32Forth Fonts
tutorial	Noble, Julian	01-11	A Call to Assembly 2/3
tutorial	Pochin, Dave	02-01	The End of the Line
tutorial	Noble, Julian	02-01	A Call to Assembly 3/3
tutorial	Telfer, Graham	02-04	Seven Times Five Equals Eleven
vectoring	Charlton, Gordon	90-10	Resolving forward references (code)
vectoring	Jakeman, Chris	91-02	Deferred words (code)
vectoring	Preston, Philip	91-04	Forgettable vectors and smart compiling
vectoring	Bennett, Paul	92-10	Vectoring with DOER and MAKE
vectoring	Allwright, Ray	97-11	From the Net - Defer and Is



Chairman **Jeremy Fowell,** 11 Hitches Lane, EDGEBASTON B15 2LS
0121 440 1809 jeremy.fowell@btinternet.com

Secretary **Doug Neale,** 58 Woodland Way, MORDEN SM4 4DS
020 8542 2747 dneale@w58wmorden.demon.co.uk

Editor **Chris Jakeman,** 50 Grimshaw Road, PETERBOROUGH PE1 4ET
01733 352373 cjakeman@bigfoot.com

Treasurer **Neville Joseph,** Marlowe House, Hale Road, WENDOVER HP22 6NE
01296 62 3167 naj@najoseph.demon.co.uk

Webmaster **Jenny Brien,** Windy Hill, Drumkeen, BALLINAMALLARD,
Co. Fermanagh BT94 2HJ
02866 388 253 webmaster@figuk.plus.com

Librarian **Graeme Dunbar** Electrical Engineering, The Robert Gordon University,
Schoolhill, ABERDEEN AB10 1FR
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

Copyright of each individual article rests with its author. Publication implies permission for FIG UK to

reproduce the material in a variety of forms and media including through the Internet.



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.