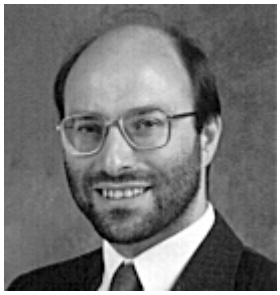
ISSN 0265-5195

Forthwrite FIGUK

Issue 105 January 2000



Editorial

Welcome to this Millennium-free issue. The FIG UK Reunion was a big success, read all about it in this issue. Jeremy has had some problems with the Hardware Project, F11-UK, but happily is back on track again - see his report. Several members have accepted the Clock Challenge but there's still time for others to join in.

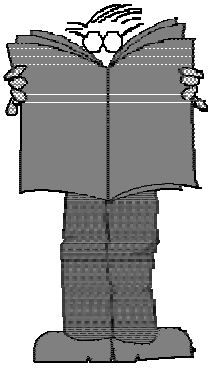
I'm especially pleased to publish another article from Dave Pochin on his explorations on Win32Forth. Together with his Internet guide "Getting Started", Dave is building a valuable resource for newcomers to both Forth and Windows.

Apologies to Alan Wenham for mixing him up with Alan Winfield yet again (!) in the last issue. Alan provides an expanded report on Vierte Dimension in this issue.

Welcome to new members Martin Eales, Rob Paterson, our second member from Ireland, Mick Dennis from Shannon, and Jan Coombs who is re-joining us.

Our IRC session last month was one of the most active yet, probably because it's at an earlier hour. Using IRC is not complicated and it's a good opportunity to chat and collect some tips so get organised in time for 9:00pm Sat 5th February.

Until next time, keep on Forthing,



Forth News

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

EUROPEAN NEWS

EuroFORTH 99

The 15th euroFORTH conference on the FORTH programming language and FORTH processors was held on September 17 - 20, 1999 at the Institute for Informatics and Automation of the Russian Academy of Sciences in St. Petersburg, Russia

Reuben Thomas of the Computer Laboratory at Cambridge University presented a paper called**:**

"Machine Forth for the ARM processor". The paper is available on the WWW and although MF has not won Reuben over entirely, he writes:

"MF's explicit use of an address latch is simple way to improve the performance of small Forth compilers that cannot afford to have an optimiser. Even more interesting are its non-destructive conditionals,
which could easily be used in traditional Forth.

In conclusion, Machine Forth's judicious mixture of novelty and classic simplicity merit careful study, though I for one will not be abandoning the traditional combination of Forth and assembler in its favour."

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

PUBLICATIONS

Electronic Design Online magazine, Nov 99 issue, includes an article by Tom Napier on the role of Forth in embedded systems. This is available online at

http://www.elecdesign.com/Pages/magpages/nov2299/embed/1122es1.htm

WEB SITES

Dutch Web Site

Willem Ouwerkerk invites visitors to the new web site for the Dutch Forth Users Group.

Forth gebruikers groep dutch users group

http://www.forth.hccnet.nl

Web Ring

The Web Ring approach to linking web-sites with a common interest appeared in the June issue of Forth News. By mid-October, 25 Forth sites had joined the Forth Web Ring including FIG UK, FIG Russia and International FIG.

FREE SYSTEMS

LEGO Robot Book Boosts Forth

Howard Shapiro reports. that he came across a book recently with an unexpected reference to Forth.

"The Unofficial Guide to LEGO MINDSTORMS Robots" by Jonathan Knudsen's

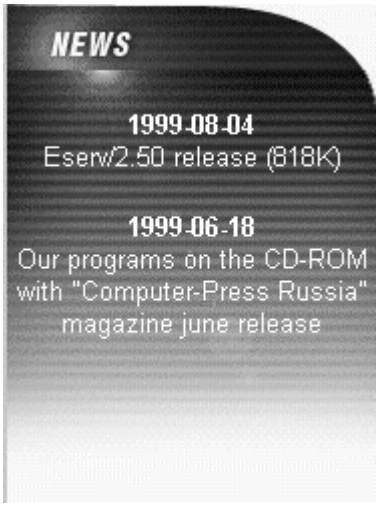
(O'Reilly & Associates,

1999, ISBN 1-56592-692-7).

This features a chapter on Ralph Hempel's pbFORTH for the LEGO system, with a brief introduction to Forth.

COMMERCIAL SYSTEMS

Eserv



Version 2 of the Eserv Forth Web-Server is now available from:

http://www.eserv.ru/

Jeremy Fowell
0121-440-1809
Jeremy.Fowell@btinternet.com

F11-UK, FIG UK Hardware Project

Jeremy Fowell

Jeremy reports on some unexpected problems and their solution.

Code Complete

The code for programming the FLASH memory on the F11-UK board is now completed.

Since you may wonder what I have been up to during the last few weeks of dark winter evenings, this is the story. It took about 15 pages of Forth and works like this:

We begin with all memory areas on the F11-UK board blank, except for one vital item which is the bootstrap loader onboard the HC11.

Bootstrap Loader

This is a small section of code built in to every chip, its function is to receive up to 512 bytes of code via the serial port and copy this to internal RAM starting at address $0000 and working up towards higher memory.

The bootloader is activated by switching into bootstrap mode. This is simply done by moving jumper link J2 to position A and pressing the reset switch. The bootloader is now running and waiting for the first byte from the serial port. This must be $FF in order to select the faster of two available baud rates, 1200 or 7812 baud.

7812 ?@!*?? yes, those engineers at Motorola really did select a non-standard rate. Fortunately 7680 baud, which is the nearest the PC can manage, works well enough.

Talker

We now download a talker program into the onboard RAM which will be responsible for loading PygmyHC11 into the external FLASH. Once the talker is loaded the bootloader passes execution to address $0000 to run the talker.

The complete talker does not fit into the onboard RAM so its first task is to load the second part into external RAM. When completed there are seven essential target functions available to the host computer:

1) Read from target memory and send to serial port,
2) Receive from serial port and write to buffer,
3) Move from buffer to target memory,
4) Wipe FLASH memory,
5) Lock FLASH memory,
6) Unlock FLASH memory and
7) Restart talker.

Normal Mode

When the downloaded code has been locked and verified the HC11 can be switched into normal mode by moving jumper link J2 to position B. Start a terminal program on the PC and press the reset switch and you should see the PygmyHC11 prompt.

The PC end is handled by a downloader program written in 80x86 Pygmy. This makes use of Frank Sergeant's serial port code in **SERIAL.SCR** which also has a simple terminal program for communicating with the target when PygmyHC11 is running.

There are talkers already available written in assembler, but they don't seem to do the sort of jobs we need. After looking at this code to see if it could be adapted I gave up and decided to write one in Forth. There were a few problems along the way.

Was Forth a better choice than assembler for the talker ? It's difficult to say, certainly the one or two real show-stoppers would have still occurred with assembler, and once the core was set up, adding the last few words in Forth was very easy.

Conclusion

We now have all the code to program the FLASH memory very easily, including words to guard against accidental erasure and verify that the contents are correct.

There are a few restrictions with FLASH however, you have to write to one or more complete sectors (128 bytes in this case) at a time and you cannot access the FLASH device while programming it. Also downloading and verifying at 9600 baud is a bit slow so we will need to develop a way of only updating those sectors where the code has changed. On the other hand we can also say goodbye (hopefully) to EPROM programming and the cost of a programmer.

Now let's get on with that documentation. This at least should be more or less routine.

Chris Jakeman
cjakeman@bigfoot.com

Did you Know? - EasyWriter

Chris Jakeman

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

Forth has a long and fascinating past but the 18 year old item reported here came to my notice just recently thanks to a posting by FIG UK member Leo Wong.

The History Books

The introduction of the first IBM PC on Aug 12, 1981, is a focal point of the best-selling biography of Bill Gates, "Hard Drive". It reports:

> "The basic machine introduced that day had one (5.25" floppy) disk drive, 16 Kb of RAM and came with a price tag of $1,565. With options, the price rose quickly as high as $6,000.
>
> IBM offered several application programs for the PC, including the popular spreadsheet program, VisiCalc, and a word-processing program called EasyWriter from Information Unlimited Software. Unbeknown to IBM, the infamous "phone phreak" Captain Crunch wrote EasyWriter, reportedly while serving a jail sentence after the feds caught him making free long-distance phone calls with his blue box. (Captain Crunch got his name when he discovered that a toy whistle included in boxes of the breakfast cereal of the same name emitted a tone that caused Ms Bell's circuitry to release a long-distance line to the caller.)"

You may have guessed by now that EasyWriter, the first word processor for the now ubiquitous PC, is a Forth application. Here is a precis - Captain Crunch (real name John Draper) tells the full story on his web site at

http://webcrunchers.com/crunch/Play/ibmstory/home.html

which is well worth reading in full.

Coding In Jail

Draper did indeed do a lot of work on day release from jail. As he puts it,

> "So, during the day, I would be coding EasyWriter, and just before I left work to go back to jail, I would get a complete listing of EasyWriter to take back with me. Then, I would examine the code for mistakes. I would be up late at night in my little cubicle, examining code, writing new code, getting ready to type it in when I returned to work the next day.
>
> It was a perfect coding environment, coding in jail."

Draper had developed a commercial Forth for the Apple II and needed a word-processor to write the documentation. EasyWriter was the result, being put to use even as it was being developed. After 2 months of work at this intensive pace, EasyWriter was good enough to show at the 4th West Coast Computer Faire.

The most popular editor at the time was Electric Pencil for CP/M computers and the market for post-CP/M machines was wide open.

Forth for Speed

> "Rumours were that Barny Stone was feverishly working on his own version of a word processor but was writing it in BASIC. Then, one day, Barny came over and played with EasyWriter for the first time. Because I wrote EasyWriter in FORTH, I didn't need or use Apple's DOS, so I developed my own file format for EasyWriter. So, when I booted up EasyWriter, it booted up very fast and loaded in less than 3 seconds. With Andy Hertzfield's help, I changed the disk interleaving so that disk reads and writes were twice as fast. When Barny played with it, he was floored by the awesome speed it scrolled, and how fast the disk accesses were. Then he told me that I was much further than he was and he gave up on his development effort after he saw EasyWriter."
>
> "During this time, I wrote a really cool FORTH debugger that allowed single stepping through FORTH code (totally unheard of in those days). I also wrote a de-compiler that would take the compiled FORTH code and re-generate source code. This was invaluable in tracing down some gnarly compiler problems in FORTH. You see, I was not only writing a word processor, but I was also developing the language on the fly as well. I even wrote a DOS (in Forth) to manage the EasyWriter text files, using a FAT (File

allocation table) and all that other gnarly Disk Operating System low level code. I found that FORTH allowed me total flexibility. If the language didn't have a feature, I implemented it. Simple as that."

Sales were way above expectations and development continued to accommodate the rash of new video cards which gave 80 mixed case characters across the Apple screen (instead of 40 chars and upper case only).

On the PC

"Around July 1981" Draper's company signed an agreement with IBM to port EasyWriter to the new computer and learned that it used the new 8086 cpu.

Now knowing the cpu, Draper bought one of the few 8086 computers then available and used the FIG-FORTH listing to implement Forth for it. He writes,

"It took us 2 weeks to get the Forth Kernel working. Eventually, the IBM computer arrived. Within 30 minutes, I had Forth up and running on the IBM-PC, partly because we just ported the ".HEX" text file over (just for fun). But surprisingly, when we ran FORTH, it just came up and ran, once we converted it to the .COM file. IBM'ers were totally blown away that a language could be operational so quickly. The next day, IBM's best software engineers were quizzing us on how we did it so fast."

So, thanks to Forth, they were able to deliver EasyWriter in time for the 12th August launch of the PC and secure their place in the history of computing.

Chris Jakeman
01733 753489
cjakeman@bigfoot.com

FIGUK

20th Anniversary Reunion

Twenty FIG people past and present found their way to the bar of the Royal Westminster Hotel on Saturday 13th.

Once we'd adopted name tags and looked around, we found that we had a good mixture of old and new members. From the early days of Forth we had Mike Beach, Chris Stephens (Comsol) and his old business partner Nic Vine.

At the opposite extreme, Jeff Penn and Federico de Ceballos represented the newest members. Federico had furthest to travel, calling in on his way from Spain to California to present a paper at FORML 99 !

We had some fun tracking down the "big names" from the past. We successfully found old-timer Keith Goldie-Morrison with just a week to go (not easy after so many years, but a distinctive name helps). Too late, as he had a wedding to attend. Sadly we still haven't located Dick de Grandis-Harrison.

Our guest of honour, journalist and Forth author Dick Pountain missed a good night out - it turned out that he had taken to his sick bed and forgotten all about us! Jeremy Fowell, Jack Brien and Keith Matthews sent more conventional apologies.

Alan Winfield told us some startling tales of research into robots for clearing mines.

Bill Stoddart was another face from the early days. With Gordon's support, he is now proposing a research project for a Forth that can run backwards.

Bill Powell, who started it all, turned up, as did ex-members Gordon Charlton and John Hayhow. Gordon and John enjoyed the evening so much they've have decided to re-join (along with Bill Stoddart).

Once we had all gathered, effectively taking over the hotel bar, we moved on to a nearby pub restaurant for dinner.

Conversation ranged from deep Forth debate to fool-proof recipes for instant ice-cream! Interesting to meet Andrew Haley after reading his authoritative posts on the Newsgroup and to chat with current members Mike Smart, Howerd Oakford and Steven Smith.

Stephen Pelc regaled us with stories from his trips to USA and brought along a charming MPE colleague trained in animal psychology. (I'm not sure what this says about FIG meetings.)

Several people wrote to say they enjoyed the event, so we hope to do it again for our 25th Anniversary in 2004.

Chris Jakeman
cjakeman@bigfoot.com

Clock Challenge

Chris Jakeman

Here is our second installment on the Clock Challenge.

Components

Without a time signal to feed into our F11-UK board, this challenge would be still-born. Fortunately, the modules in the Maplin catalogue are still available, though there have been small changes.

Two components are needed to deliver a signal ready for decoding - an aerial (or antenna) and a receiver module.

Part:	Antenna	Receiver	Carriage
Code:	MK72	MK68	-
Price incl VAT:	4.99	16.99	4.13
Purchase from www.maplin.co.uk & 01702 554000			

A third component is listed, a "microcontroller Decoder Module" which decodes the signal and sends the time and date down a serial line. We won't need this as we can program the F11-UK to do this sort of thing ... and much more.

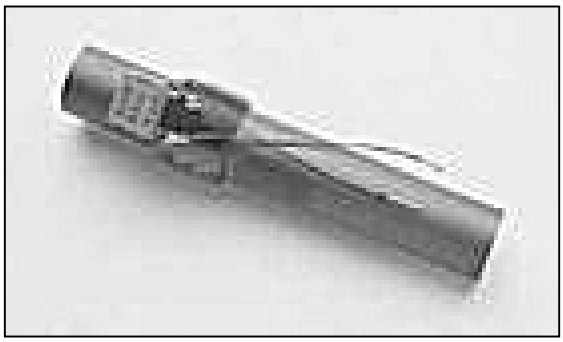
Signal

We don't need the F11-UK in order to see the signal. With the addition of a battery, an LED and a current-limiting resistor, the LED will flash once (sometimes twice) a second as the signal is detected.

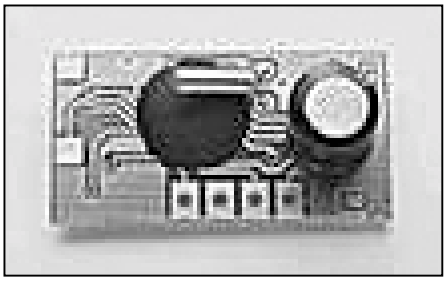
I have bought a set of these components from Maplin. As soon as the power was connected, the LED started to flash. It's quite fascinating to watch. A long half-second flash indicates the first second of the minute and the coded data follows in a mixture of short and long flashes thereafter. The details of the code given in the last issue show that there are two data bits. For most of the code, the second bit - Bit B - is unused, so a short flash indicates Bit A is off and a longer one indicates Bit A is on.

Bit B is currently used for the first few seconds, indicating the approximate difference between GMT and UTC (don't ask!), so there may be some double flashes during that period. The section of signal needed for decoding time and date is from second 17 to second 51 and Bit B is unused in this section.

I imagine that signal strength varies, being strongest when the antenna points towards the transmitter. I live about an hour's drive from Rugby, so the signal is strong and turning the antenna made no difference at all.

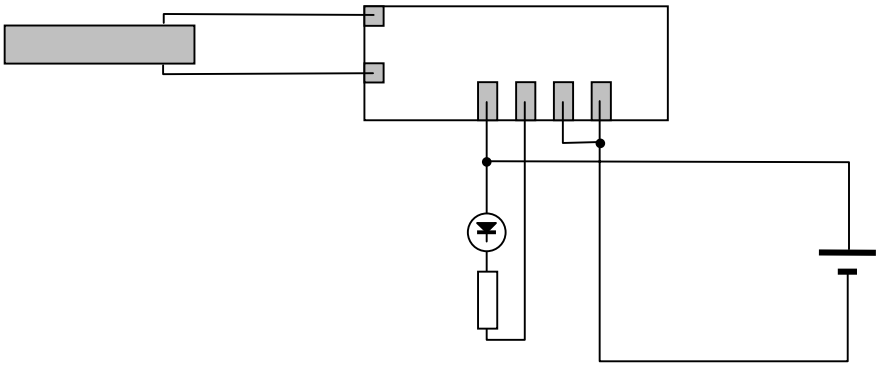


Construction



The receiver module sent by Maplin differs slightly from the one described on their web site. In particular, it has an "open collector" output which can drive up to 2mA (100 times more than the advertised module) and enough to light a small LED. It also has a "low power consumption" feature which initially led me to think the device was faulty. It turns out that if control pin LP2 is not linked to ground pin LP1, then no signal is output.

I used a 9 volt battery (PP9), a 10K resistor to limit the current to 0.9mA and a small low-current LED.



German Transmissions

Although the Rugby transmissions reach to Northern Europe, our German members would be better to work with the German service which uses a different frequency. Details can be found at http://www.eecis.udel.edu/~mills/ntp/dcf77.htm

Rising to the Challenge

Latest news (heard at our December IRC session) is that 2 other members have bought these components in readiness for the challenge and 2 more have agreed to take part. Our German colleagues are also looking into their equivalent service, so that challenge may even go international.

Alan J M Wenham
01932 786440
101745.3615@compuserve.com

Vierte Dimension 4/99

Alan Wenham

Alan 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 Alan.

Riddle - An attempt at finding a solution

Fred Behringer — A riddle was posed in VD 3/99 concerning the connection of three lights and three switches. Fred not only presents a solution but also takes the opportunity to discuss the exactness of assumptions in riddles, Forth, and with programming in general.

Reed-Solomon Error Correction - Part 1

Glenn Dixon — This is a translation by Fred of a paper originally from Forth Dimensions, Vol 20 No 4.

Patriot Scientific PSC 1000, a Java-, Forth-, and C-processor

Jens Wilke — This unit is marketed as a Java processor but can also be considered as a good Forth processor. Jens Wilke describes the operational structure and discusses how it can be adapted for Gforth.

BEGIN-UNTIL in 32 K on the 80386 in ZF-Assembler

Fred Behringer — One can implement structured programming (IF-THEN and so on) in ZF-Assembler but the jumps can only cover 128 bytes (forwards or backwards). Fred shows how the jumps may be extended to 32 kilobyte length. Only three high-level Forth words are necessary for extension of the assembler.

Hashing - Part 1

Friedrich Prinz

Fritz analyses the hashing mechanism in ZF. This study originated in 1994 and was further worked out within a tutorial circle in the Forth Group at Moers and, because of its size, has not been published up to now. Questions of optimal hashing have acquired great significance in our present time of very large Forth systems. This is the first part of a three-part series.

Eaker's CASE statement in assembler for ZF and Turbo Forth

Fred Behringer

Fred has expanded assembler in ZF and Turbo Forth "on the fly" so that one can use structured programming CASE constructs in assembler as well as in high-level definitions. He relates this to the BEGIN-UNTIL paper noted above and, in the same way, extends it to over 32 kilobytes.

Dear Chris,

I have changed the way in which I report on Vierte Dimension, the journal of our German Forth friends.

Fred Behringer has kindly agreed to send me a short summary of the most important aspects of the significant papers. I have translated these and Fred has then checked that I have extracted the proper meaning from them. We must be most grateful to him because some of the specialist details of Forth and of German grammar can be a touch obscure!

Alan Wenham

Dave Pochin
01905 723037
davep@sunterr.demon.co.uk

"(Sc)roll Up, (sc)roll Up, see Win32Forth scroll the Window."

Dave Pochin

This is the latest in a very valuable series from Dave, who allows us to share his explorations of Windows using Win32Forth - see also Issues 101 and 103. (I've been following this closely as I expect it save me a lot of time when I get around to programming for Windows. - Ed)

I have to confess to having a 'hang up' about scrolling techniques. There seems to be a great deal more work involved than should be necessary. In addition, there are no practical results for all my trouble unless I get involved with the file handling, the font styles and the screen handling as well.

However, as a beginner with Win32Forth, I must not get into a negative frame of mind. What is required is a keen analytical approach, never my strongest point !

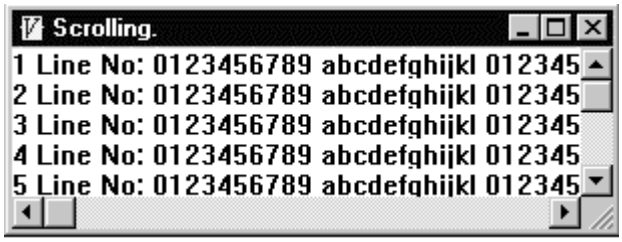
The approach here is to strip away all the trimmings and try to look only at the scrolling part of the task. This involves using token representations for files, font styles, and window characteristics; so there is a great deal of extra work to do before using any of the material in the listing as part of an application. On the other hand, there are only three Windows procedures to consider and there are no basic differences between those required for vertical and horizontal scrolling.

The problem can be tackled by :-

1. Setting out a reasonable specification for the demonstration.
2. Explaining some of the terms used by Windows.
3. Finding the parts of the scrolling procedure that Windows does not do for us.
4. Updating the content of the window in response to the scrolling action.
5. Observe and comment on the results.

1. Specification - write a short application to show scrolling.

1.1. Display a window with vertical and horizontal scroll bars.



1.2. Both scroll bars to show correctly the effects of ; Line Up, Line Down, Page Up, Page Down and the use of the scroll box.
1.3. The window initially to display a page consisting of 5 lines of text of approximately 40 characters a line.
1.4. The window to scroll vertically by use of the scroll box, and by increments of 5 lines, and by 1 line.
1.5. The window to scroll horizontally by use of the scroll box, and by increments of approximately 10 characters and by 1 character.

2. Terms used by Windows

The terms used in scrolling procedures include the following.

2.1 The Client Area of the window is the part of the window available to the user, i.e., excluding the Title and Scroll bars. The extent of the Client Area is described by two parameters, width `cxClient` and height `cyClient`. In practice, these values are obtainable from Windows, but in the listing they are defined as values.

2.2 The average size of a character is described by two parameters, width `cxChar` and height `cyChar`. These values are also obtainable from Windows, but once again in the listing they are defined as values.

2.3 The scroll bars are added to the window by modifying the window style, using the constants `WS_VSCROLL` and `WS_HSCROLL` in the `WindowStyle` method.

2.4 When a scroll bar is clicked or the scroll box is dragged the action is coded as a parameter in a message for Windows to process. The messages are called `WM_VSCROLL` and `WM_HSCROLL`. (**W**indow **M**essage **V**ertical ... etc). Two of the methods used in the listing take the names of these messages. Five scrolling parameters, `SB_LINEUP`, `SB_LINEDOWN`, `SB_PAGEUP`, `SB_PAGEDOWN` and `SB_THUMBTRACK` (**S**croll **B**ox _ .... etc) are used in the listing.

3. Scroll Bar Operations

3. Windows requires our program to look after three scroll bar operations :-

- Setting up the range of the scroll bar.
- Processing the scroll bar messages.
- Updating the position of the scroll box.

3.1 Setting the range of the scroll bar. Use the Windows procedure `SetScrollRange` which is defined as :- `SetScrollRange ( hwnd, nBar, nMin, nMax, bRedraw )` where `hwnd` is the window handle, `nBar` is either `SB_VERT` or `SB_HORZ`, `nMin` and `nMax` are the minimum and maximum

values of the range and bRedraw is TRUE if you want the scroll bar set to these values. In Win32Forth, reversing the order of the parameters gives :-

```
FALSE 100 0 SB_VERT Gethandle: self call SetScrollRange drop.
```

3.2 The scroll bars messages are processed by reading the value of the parameter (SB_LINEUP etc) of the message and using a case ... end case branching routine to alter the value of a variable (or a value) which is either VScrollPos or HScrollPos in the listing.

3.3 The Windows procedures SetScrollPos and GetScrollPos are used to set and obtain the position of the scroll box. If initially the position of the scroll box and the value of VScrollPos (or HScrollPos) are the same, representing the top (or left) of the scroll bar, then at the end of the processing of a WM_VSCROLL (or a WM_HSCROLL) message the new position of the scroll box obtained from VScrollPos (or HScrollPos) and the old position obtained from GetScrollPos can be compared and the position of the scroll box updated by using SetScrollPos and redrawing the window.

In Windows GetScrollPos is defined as :-

```
GetScrollPos ( hwnd, nBar ).
```

In the Win32Forth listing for the vertical scroll box position this becomes :-

```
SB_VERT Gethandle: self call GetScrollPos
```

Similarly in Windows SetScrollPos is defined as :-

```
SetScrollPos ( hwnd, nBar, nPos, nRedraw )
```

In the Win32Forth listing for the vertical scroll box this becomes :-

```
TRUE VScrollPos Gethandle: self call SetScrollPos drop
```

4. Updating the scroll box

Updating the scroll box is not enough.

When the window is redrawn the window content must be scrolled as well as the scroll box . To do this the values of VScrollPos and HScrollPos are used to calculate a new position within the file which must be displayed in the top left hand corner of the client window. In the listing this is done in part of the :M On_Paint: method by re-writing an artificial file of 25 lines of text.

5. Comments

5.1 The results could be improved in many ways. For example, by adjusting the vertical incremental routine it is possible to stop the scrolling when the ‘end of the file’ , the 25th line, is the bottom line in the window.

5.2 There are many places where experimenting with changes in the parameters in the listing can alter the performance.

5.3 If you wish to see scrolling work in a larger application, use the Winview editor to see Tom Zimmer's **Winview.f** itself; about a quarter of the way through you will see some of the detail required by a proper scrolling routine.

6. Code Listing

```
\ ScrollUp.F        Simple Scrolling
\ Modified as a Win32Forth OOP. D.R.Pochin

\ See Windows.f for details of the Window class
\ Modified from WINHELLO.F

\ Define an object "ScrollWindow" that is a
\ child object of class "Window"
\ Redefine the Window Title.
\ Add scroll bars and some text to scroll.

\ Create a new Object of the Class Window.
\ See Window.f for details.
:Object ScrollWindow  <Super Window

\ Set up some values.
   25 value NumLines      \ Set the number of lines of text.
   80 value NumCols       \ Set the number of columns of text.
    8 value xChar         \ Set a value to represent the average
                          \ width of a character.
   15 value yChar         \ Set a value to represent the average
                          \ height of a character.
                          \ NOTE. Both these are arbitrary.
                          \ They are simplifications of values which
                          \ should by obtained from the
                          \ characteristics of the font used.
  270 value xClient       \ Width of the client window.
   75 value yClient       \ Height of the client window.
                          \ NOTE:
                          \ Both these are arbitrary simplifications.
    0 value VScrollPos    \ Initial value for the position of the
                          \ vertical scroll box.
    0 value HScrollPos    \ Initial value for the position of the
                          \ horizontal scroll box.

:M WindowStyle: ( -- style )      \ Inherit WS_OVERLAPPEDWINDOW.
              WindowStyle: super  \ from M: WindowStyle in Window
                                  \ Class.
```

```
              WS_VSCROLL or       \ Add a vertical scroll bar.
              WS_HSCROLL or       \ Add a horizontal scroll bar.
              ;M

:M WindowTitle: ( -- Zstring )
                z" Scrolling."    \ New title for the window.
              ;M

:M StartSize: ( -- w h )          \ Set the width and height of our window.
                300 90
              ;M

:M StartPos:  ( -- x y )        \ Set the screen origin of our window.
              200 100
            ;M

:M On_Paint: ( -- )       \ All window refreshing is done by On_Paint:
              NumLines 0 \ Start loop limits to insert text.
              do xChar 0 HScrollPos - *      \ Set left edge of
                                             \ text.
                yChar 0 VScrollPos - i + *  \ Set line of text.
                                            \ Change the line number
                                            \ and space to the string
                                            \ temp$.
                i 1+ s>d <# 32 hold #s #> temp$ place
                                            \ Add more text to temp$.
                s" Line No: 0123456789 abcdefghijkl 0123456789"
                temp$ +place
                temp$ count TextOut: dc \ Print out temp$.
              loop                      \ Return to start of loop.
            ;M

:M On_Init: ( -- )      \ Things to do at the start of window
                        \ creation.
     On_Init: super     \ Do anything superclass needs.
                        \ Set the range of the vertical scroll bar.
     FALSE NumLines 0 SB_VERT Gethandle: self call SetScrollRange
       drop
                        \ Set the initial position of the vertical
                        \ scroll box.
     TRUE  VScrollPos SB_VERT Gethandle: self call SetScrollPos
       drop
                        \ Repeat for the horizontal scroll bar.
     FALSE NumCols 0  SB_HORZ Gethandle: self call SetScrollRange
       drop
     TRUE  HScrollPos SB_HORZ Gethandle: self call SetScrollPos
       drop
          ;M
```

```
:M WM_VSCROLL  ( h m w l -- res )    \ Control the Vertical scroll
                                     \ bar.
            \ Place the position of the box on the return stack.
             swap word-split >r
      case              \ Identify the scroll command.
        SB_LINEDOWN     \ Add 1 line to the value VScrollPos.
           of VScrollPos 1 + to VScrollPos
           endof
        SB_LINEUP       \ Subtract 1 line from the value VScrollPos.
           of VScrollPos 1 - to VScrollPos
           endof
        SB_PAGEDOWN     \ Set value VScrollPos down 5 lines.
           of VScrollPos  yClient yChar / +  to VScrollPos
           endof
        SB_PAGEUP       \ Set value VScrollPos up 5 lines.
           of VScrollPos yClient yChar / - to VScrollPos
           endof
        SB_THUMBTRACK   \ Fetch box position from return stack
                        \ and update the value VScrollPos.
           of r@ to VScrollPos
           endof

        \ Check that the value VScrollPos is within the page.
        0 VScrollPos NumLines min max to VScrollPos

        \ Check if value of VScrollPos <> position of scroll box,
        \ if so then reset scroll box.
        VScrollPos SB_VERT Gethandle: self Call GetScrollPos <>
        if
          TRUE VScrollPos SB_VERT Gethandle: self Call SetScrollPos
            drop
          Paint: self             \ Redraw the window.
        then
      endcase                     \ end case switch.
      0 ;M

:M WM_HSCROLL ( h m w l -- res )  \ Control the Horizontal scroll bar.
              \ Place the position of the box on the return stack.

         swap word-split  >r
     case                \ Identify the scroll command.
       SB_LINEDOWN       \ Add 1 column to the value HScrollPos.
          of HScrollPos 1 + to HScrollPos
          endof

       SB_LINEUP         \ Subtract 1 column from the value HScrollPos.
          of HScrollPos 1 - to HScrollPos
```

```
            endof

                  \ Set value HScrollPos down xClient/(4*xChar) columns.
         SB_PAGEDOWN
            of HScrollPos xClient xChar 4 * / + to HScrollPos
            endof

                  \ Set value HScrollPos up xClient/(4*xChar) columns.
         SB_PAGEUP
            of HScrollPos xClient xChar 4 * / - to HScrollPos
            endof

         SB_THUMBTRACK     \ Fetch box position from return stack
                           \ and update the value HScrollPos.
           of r@ to HScrollPos
           endof

         \ Check that the value HScrollPos is within the page.
         0 HScrollPos NumCols min max to HScrollPos

         \ Check if value of HScrollPos <> position of scroll box,
         \ if so then reset scroll box.
         HScrollPos SB_HORZ Gethandle: self Call GetScrollPos <>
         if
            TRUE HScrollPos SB_HORZ Gethandle: self Call SetScrollPos
              drop
            Paint: self            \ Redraw the window.
         then
       endcase                     \ end case switch.
         0 ;M

 :M On_Done:  ( -- )               \ things to do before program termination
              On_Done: super       \ then do things superclass needs
           ;M

;Object                            \ Complete the new object.

\ Words to start and finish running the program.
: DEMO        ( -- )               \ Start running the program.
              Start: ScrollWindow
           ;

: UNDEMO      ( -- )               \ Close the window.
              Close: ScrollWindow
           ;

\ Instructions to run.
   cr cr    .( Type 'DEMO' to run program, 'UNDEMO' to stop ) cr
```

Forth for Fun

Hugh Aguilar, Chris Jakeman, Willem Ouwerkerk, Friederich Prinz, Martin Bitter, Fred Behringer

The contributions here, from 4 countries, were inspired by some comments in a Forth Dimensions article on Code Cracking by Hugh Aguilar. There has been much emphasis in recent years, through the ANSI standard and comp.lang.forth, on the advantages that professionals gain from using Forth. The authors of this paper, whilst welcoming these advantages, remind us that Forth is also suitable for amateur programming and, in many ways, more suitable than other more complex languages.

Hugh Aguilar, in his Forth Dimensions article on code cracking, writes: "The author found that writing CrakPoly was fun, and that using it is fun, too. Also, designing and writing fun programs is good practice for working on commercial products.

C++, with its emphasis on GUIs and commercial development, requires too much work to be used in weekend projects. Because nobody programs as a leisure activity anymore and, in so doing, gets practice at programming, our professional programming is now described with terms like "death march project" and "anti-pattern". These apparently are the wages of professionalism."

This article will also be published in Forth Dimensions. A Dutch version will appear in 'Het Vijgeblaadje' (Dutch FIG), a German version, in Vierte Dimension (German FIG).

Why "Forth For Fun

Hugh Aguilar,
FIG International,
author of several articles in Forth Dimensions

1. Writing fun programs is fun. This is an end in itself.
2. Writing fun programs is more fun if other people get involved and contribute. The program becomes more useful and interesting this way.
3. Forth Dimensions and its cousins are practically the only magazines left these days that will publish articles about fun programs and which will provide source code. This is a reason in itself to use Forth.

4. Forth is an ideal language for writing fun programs because it is simple.
5. Portable Forth provides a command-line interface, a simpler alternative to writing a GUI interface. Forth is robust as well as simple, however, and can easily support extensions such as OOP and GUI.

6. Writing fun programs makes a person a better programmer. This has a positive effect on a person's professional work (Forth or otherwise).

7. Fun programs which initially don't seem to have any commercial value can evolve into something really good and worthwhile. If a lot of people start using the program, it may even be possible for the author to make some money by selling support and writing custom upgrades. This idea, of making money from public-domain open-source software, is Richard Stallman's.

I think that there are a lot of cool Forth programs out there that are languishing in obscurity because the author thinks that nobody cares about his program because it has no commercial potential. We need to encourage this unknown Forth programmer to write an article and have it published. People do want to read it.

Forth at home but not at work - why bother?

Chris Jakeman, Editor of Forthwrite, FIG UK

I've been dabbling in Forth for 15 years and been a member of FIG UK for nearly as long. I've never written a line of Forth code professionally and my use of Forth at work has been limited to one-liners like:

```
50 RANDOM 1+ .
```

to select the winner of a publicity draw with 50 contestants.

During those 15 years, I have managed small software teams doing difficult things in C, C++, Unix and Windows and Forth has been an important part of my continuing education.

Looking back, the pages of Forthwrite, the FIG UK magazine, are sprinkled with my various work-in-progress showing the results of experiments in language design (parsing, compiling and object-oriented code), string

searching, pattern-matching and memory management etc..

All this was done "for fun" in a spirit of research and yielded some new algorithms, (including a very fast case-insensitive string search which I have looked for but never found in any book). However it proved very useful in keeping one step ahead of my staff, who tend to come to me when they have run out of ideas (yes, very gratifying).

If you ask me, "Surely all this study could have been done in a mainstream language?" I would reply, "Yes, but only in Forth is the work short enough to fit into an article." It is principally because Forth is small, simple and fun (or you might say "primitive") that I have felt moved to explore binary search or memory management. On a Unix computer, these facilities are provided and taken for granted.

Forth, for me, has been fun. Its application in my professional life has been peripheral but real nonetheless. I shall continue to have fun with Forth and to enjoy the contact I get with other Forthers who usually turn out to be more interesting people than most.

Play and Learn

Willem Ouwerkerk
Editor of
Het Vijgeblaadje,
Chairman of
HCC-Forth-gebruikersgroup
(Dutch Forth Users)

I always had fun in creating my own tools and toys. As a kid and then an adolescent I modelled, designed and built my own boats and racing cars. Nowadays I am able to use Forth systems mostly of my own design and a reasonable knowledge of electronics to create my own toys.

My interest in the behaviour of man and animals can be expressed using Forth. I design machines (robots), controlled by Forth with their behaviour built into the software. Here I use the same methods as before. Forth allows me to experimentally (intuitively) fathom a problem. These robots are merely entertaining, but I enjoy inventing and creating them very much. Although they have no practical value, the smile and astonishment of the visitors to an exhibition is worth the trouble of building the robots.

Through articles for 'Het Vijgeblaadje', manuals and other books which I spread via the Dutch Forth Users Group, I am trying with varying success to share this enthusiasm with others. The 'Egel-workbook is a result of this. Also I

am trying to keep an element of fun in my professional work also. An elegantly designed piece of soft- and/or hardware gives me much pleasure. Especially if it is done with a minimum use of resources.

A large part of the job in implementing a new Forth means re-inventing the wheel. Huge portions of Forth are known territory, but using the existing hardware optimally is an art in itself. I very much enjoy using it to fathom a processor and keep it under control with software I wrote myself (called ByteForth). Personally I think that ByteForth is at least as good as professional systems in making best use of scarce resources, even though some compiler-writers will be smarter than I.

It's strange that many do-it-yourself Forth implementations suffer from poor documentation. I always write documentation for each implementation and use it regularly myself! Maybe it's because I use my own systems to create robots and do various jobs for others. A neatly made handbook looks good and it makes the (non)professional system useful for others also.

Forthers help Forthers

Friederich Prinz
Editor of
Vierte Dimension,
Director of Forth-Gesellschaft
(German FIG),
winner of 1995
Swap-Dragon Award

Every Forther has asked himself and other Forthers why they have chosen Forth and no other language, and a virtual machine of the early seventies as their favourite means of programming. They all know the answer.

Once a discussion has started and the usual pros and cons have been exchanged, one always reaches a point where one consideration appears which I for one am very fond of: the outlook of the individuals who are exploring Forth.

Friederich Prinz, Editor of Vierte Dimension, Director of Forth-Gesellschaft (German FIG), Winner of 1995 Swap-Dragon Award

These people are using the computer to solve problems. The problems they solve, however, are often not their own but ones posed to them by others who are looking for help. Once asked, most Forthers tend to forget time and space and their own interests and start spending days and nights on a problem which is not their own. In doing so they sometimes produce neat little tools and sometimes big and sophisticated Forth systems. Moreover, generously and without asking for consideration, they almost always

distribute their findings to others who might find them useful.

It seems to me that Forthers are especially used to thinking in small steps and they act accordingly. Of course, Forthers also know the methods of reasoning on a large scale as well as the technique of division of labour widely praised in computer science. They also, however, have a mind for being responsible for the small things which are contained in the big problems they analyse. Forthers feel responsible for the essentials of the problems they come in contact with.

I must add that I have rarely known any Forther who considers himself too great to deal with "trivial" problems. Of course one can find such people but in my eyes they are too petty to be regarded as participants in a big project. The biggest project that Forth and Forthers face is to make the things we enjoy known and available to as many people as they can reach.

Forth is fun and great fun comes from Forthers.

Forth - or the art of motorcycle repair

Martin Bitter
Forth-Gesellschaft (German FIG),
author of a number of papers iin Vierte Dimension,
active co-chatter on IRC channel #FIGUK

I drive my car to school every workday, and I must say that I totally dislike driving a car. I enjoy to ride my old motorbike (almost as old as I) much more, over the soft curves of country roads. What bliss, to accelerate out of a curve with a slightly spinning rear wheel! (For those, who are curious: DKW RT 175, 9 brake HP, built in October 1953.) When I got the bike 15 years ago, it came in three large boxes - it was a "basket case." Fortunately, in those days I did not know that the rule of restoration says:

"Never buy a dismantled motorcycle!"

In spite of everything, I brought the machine back to running order and now I know every one of its bolts and nuts. Really - every one! And all of its functions. But I have never become a DKW expert who can recognise at a glance the year and model or relate the history of the firm and the names of the designers.

It was fun to tinker with such a machine and it is fun to drive it. Nevertheless, I drive to work with my car - it is more practical, more comfortable, and (although hard to admit) more dependable. At the same time, I have little

interest to tinker or to work on my car; I leave that to a workshop I can trust.

It is very much like that with Forth:

With a great deal of pleasure I use the ZF Forth that was developed by Tom Zimmer a long time ago and that is presently cultivated by the Forth group in Moers. I don't exactly know every byte of this version of Forth, but if I were to seriously try to learn, it would be possible. To decipher the structure of this Forth (and Tom Zimmer's ways of programming in those days) would be very satisfying, and all I would need for the task would be ZF Forth itself.

I like the learning possibilities that Forth offers when I try to figure out its internals, and I can use the new knowledge to help me unravel the secrets of DOS.

When I first connected with Forth, I knew none of the "golden rules of programming" nor the mantras of computer science like

"always separate code from data ;-)."

The only rule was:

"Computer runs = good, computer hangs = bad!"

Of course, several books helped me further; among the first were both of Leo Brodie's classics. Much more help however came from studying the source code or extracts and snippets of code from other Forth programmers that were published in Vierte Dimension.

Different programmers showed there their thoughts and their code and I absorbed everything like a sponge.

The Forths that I use are inexpensive (there are a few others that are bargains). As an ambitious hobbyist I cannot spend much money for a computer language program, and, on the other hand, I do not stand to make much money with Forth either.

So it is just for fun! But sometimes Forth helps me to help my students or to solve problems for which I cannot obtain any ready-made solutions.

I have learned much from the code written by others, even when the code has not always been perfect. It is my unshakeable conviction that everyone who writes Forth code, whether it be for fun and pleasure or for earning a living, should publish. The work of "polishing" an article will be more than compensated for by the feeling to have shared something with the Forth society.

Skills boost from recreational Forth

"Publish or perish" is a slogan widely known among scientists. So it's not the search for truth that counts, but the fear of perishing? No kidding. Modern university ranking and evaluation is mostly based on counting papers in the Science Citation Index. This is one side of life, the professional one. The other side is the legitimate desire of a creative individual to do things he likes even if they don't pursue any purpose.

Fred Behringer
FIG US, FIG UK,
Forth-Gesellschaft
(German FIG),
winner of 1999
Swap-Dragon Award

Like many Forthers, I'm an expert in my own field. I need no Forth for showing my own expertise. Forth is too narrow a field for that. I value Forth for recreational purposes, for leisure activities not too far away from the topics of my own profession. I like doing things without purpose. I've chosen Forth to do so. Not by coincidence. I was once looking for a close-to-the-machine language to make the digital part of a hybrid computer automatically maintain the analogue part of the machine. FORTRAN and ALGOL were no help and Assembler was tedious. So I invented my own language, DISPRA (Dialogsystemsprache).

This was in 1969. Later I saw that Forth could have been THE language for that purpose but I was then already working in other fields of science. However, I became a Forth addict. Today, the professional side of my life has passed and I have discovered that Forth is also good for retirement. I can remain active without the inhuman degree of competition needed in serious scientific work.

I am the ubiquitous hobbyist, the "enthusiast who has the tendency to re-invent the wheel". I like re-invention. It's the only way that leads me to understand things. Also, I'm the one who doesn't hesitate to construct "me-too" compilers. And I didn't care, for instance, whether or not Transputer Forth could have any future, neither in the past nor at present. I am curious. I like learning. I like learning by doing. I've learnt things I couldn't have learnt from books on their own. I don't have any objection against "considering Forth as sort of a religion".

I don't consider Forth as a language. For me, it is an idea. I like a Linux-like evolution of projects done by a group of enthusiasts on the 'Net. It reminds me of work done within the international scientific community, only that Forth, for me, can be done on a more leisure-oriented basis. I don't like to spend any money for any Forth system, however cleverly designed. I like Stallman's open source idea. Science has lived on it for more than two hundred years. Imagine a mathematician publishing a new theorem without presenting a proof! I like to get in contact with other Forthers who like ... see above. I like .... Forth.

Letters

Dick Pountain

From: Dickp96@aol.com
Sent: 16 November 1999 17:26
To: cjakeman@bigfoot.com
Subject: Re: FIG UK 20th Anniversary

Chris - I'm terribly sorry about that. I was in bed with some variety of stomach flu and didn't even check my diary until Monday, by which time it was too late to email you. Still I'm glad a good time was had.

Dick

Bill Stoddart

Hi Chris

The reunion was brilliant. Wonderful to meet you and find FIG in such good hands. It's very nice of you to want to revive my papers, but I think I'll leave them in peace. I hope to be back on the Forth scene though: I may need a computer that runs backwards for one of my research projects in Formal Methods. We had a little conversation about this over dinner. Gordon Charlton, who was too tired to speak, conveyed the details to me telepathically.

As a computer runs a program it throws away information. For example x := 3 throws away the previous value of x. Forth's stack based architecture throws away less information that other architectures, and we can use is at the basis of a reversible computing engine. We define Forth primitives which have the usual functionality but which transfer any information they throw away to a history stack. For example DROP now transfers the stack top to the history stack. We also need ~DROP (DROP run

continued

backwards) which moves the top item on the history stack back to the parameter stack.

We develop, like this, a Forth virtual machine in which every "code" definition has both a backward and forward version, and by some ingenious implementation technique we contrive a Forth system with completely reversible code.

There is a new command: -> "guard"

-> (flag --)

If flag is true continue forward execution, otherwise go into reverse. And a new control structure defined with CHOOSE and END

Example:

CHOOSE WORD1 WORD2 ... WORDN END

If running forwards select any one of the choices WORD1 to WORDN and continue execution. If running backwards make a choice that has not been tried before, and run forwards again. If all choices have been tried continue running backwards.

The idea is to allow a backtracking search style of programming, as in Prolog. Andrew Haley gave me a web reference for the reversible programming community:

ftp://ftp.netcom.com/pub/hb/hbaker/ReverseGC.html

(Some of them know about Forth and sing its praises).

Warm Regards

Bill

Gordon Charlton

From: Gordon Charlton
[gordon@charlton.demon.co.uk]
Sent: 20 November 1999 16:06

Dear Chris,

First of all, thank you for dragging me to the FIG UK reunion. I enjoyed it very much. I also enjoyed issue 104 of Forthwrite, the first I have read in a long while, and its articles about the past and future of Forth, and the Hardware Project and associated challenge, which set me thinking...

It has been about five years since I last wrote for Forthwrite, and during that time I have been exploring some of my other interests. Now, five years later, my Forth requirements have changed a great deal. My
first computer, a Jupiter Ace, could do virtually nothing except run Forth. That suited me well: I was straight out of college, and eager to explore computer programming. I spent many happy hours crouched over the little white plastic box, with a bag of frozen peas on top to stop it overheating.

Soon I progressed to an Atari ST, and MPE's excellent GEM Forth. It had a lot of things I continue to require from a Forth - an easy, intuitive editor, a well documented manual and a straight forward approach that made it simple to understand. It allowed me to continue what Steve Pelc calls "a personal exploration of the language", much of which was published in Forthwrite, and it allowed me to take control over my dot matrix printer.

This I needed, because the word processor I had available to me did not use anything like the full capabilities of the printer. But it was a simple enough matter to embed mark-up commands into the document I was writing, and run it through a simple parser written in GEM Forth.

continued

Now I no longer need that sort of control. I have a couple of Macs, and any control I need to exercise over my machines is available without recourse to Forth. Equally I am not particularly interested these days in personal exploration of programming languages, and the amount of understanding that I would require to write applications that make reasonable use of the Mac Operating System means that such use is rather beyond me.

So where could Forth fit into my life? Not on a Mac. The commercial Forths available are outside of my budget and perhaps I am rather jaded, but having used a proprietary Forth, the shareware offerings for the Mac are too quirky for my taste, and too spartan in the programming environment they offer.

I rather hanker for a little computer. One that I can understand in its entirety. The FIG-UK Hardware project, the F11-UK, comes some way to what I imagine, but not far enough. For one, I have no particular wish to wield a soldering iron. My facility for that sort of task is strictly limited, and I would anticipate that, despite my best efforts, I would destroy more components than I would successfully solder into place.

Secondly, my primary computer, an iMac, does not have a serial port. Yes, I could buy a USB to serial adaptor, but that is another expense. Thirdly there does not appear to be any prospect of Mac software with which to talk to the F11-UK.

And then there is the shareware Forth. To be fair I do not have any experience of Pygmy Forth, but I see that, for instance, Frank Sergeant has eschewed DO .. LOOP in favour of FOR .. NEXT. Oh dear, quirkiness looms. Now that we have a workable standard, what is wrong with adhering to it?

continued

OK, so if I could go to a shop and buy a "Little Forth Box" (and I mean *little* - I see no reason why it need be larger than a credit card or thicker than a USB port, a processor, a battery, some RAM, some PROM and an interface chip or two) and plug it into my iMac, and install a piece of software that opens up a monitor window onto the machine inside a programmers' text editor (I am thinking of the Mac's BBEdit, which is ideally suited to the job, and sufficiently extensible) then what would I do with it?

Well, I hear that there are little chips available that fit inside light switches and the like that control simple devices and can talk to one another. Networking my house sounds like an admirable project, and, if I could buy a little Forth box, maybe I could buy a similar-sized box that would let it talk to my household appliance network. And maybe another card would let it talk to my telephone, and hence the Internet. Or perhaps I could sit it amidst some Lego Technic or Meccano, and have a little robot scurrying about my floor. I am sure that would amuse the socks off my eight year old son, and give him an excellent introduction to computer programming.

Paul Bennett once told me the most excellent maxim - One Process, One Processor. Much as we all love our big desktop computers with memory by the bucket-load and clock speeds to die for, they are not always the right tool for the job, and when the right tool is a simple little machine, Forth is the right tool for programming it. And, of course, that would mean that, as a hobbyist, I would finally be using Forth for what it was intended for, for process control, and for slipping into the tiny spaces where other languages cannot squeeze.

We have heard much about the computerised house of the future over the years, and this is where I see one possible future for Forth.

continued

So here is my challenge: At the moment the only person I know of with a "house of the future" is Bill Gates (excuse me for a moment while I go and wash my mouth with carbolic soap) but I think with Forth we could prove that you don't have to be the richest man in the world to have process control in the home.

With the right tool at our fingertips, and with the imagination and flair of the individual membership of FIG-UK, and the experience and expertise of the corporate members in embedded systems I could have my Little Forth Box, and what better way to meet Steve Pelc's goal for the next five years, of spreading the Forth word and producing a generation of Forth programmers?

Yours,

Gordon Charlton

Forthwrite Index

Jack Brien maintains a set of 3 indexes to Forthwrite on the FIG UK web site. These indexes are sorted by date, by author and by subject and are updated with each issue. The subject index is repeated in Forthwrite annually, with the new entries highlighted.

Back issues of Forthwrite may be borrowed from the Library without charge, so this is a good way for a new member to catch up on topics of special interest.

Forthwrite Subject Index 1990-1999

---

algorithms	Bennett, Paul	94-06 Fuzz, fibs and forms
algorithms	Bennett, Paul	95-06 Fractionally angular
algorithms	Charlton, Gordon	93-04 Backwards (psychic programming)
algorithms	Charlton, Gordon	95-06 Easter Sunday
algorithms	Hersom, Ed	92-10 Advanced course
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	Pochin, David	94-10 First attempts at Fuzzy Logic
algorithms	**Ramsay, Chris**	**99-08 Forth and Genetic Programming**
applications	Anderson, Joe	98-07 Forth In Space
applications	Brien, Jack	91-02 Typing tutor (code)
applications	Franin, Julio	92-08 Torsion measurement system
applications	Green, Roedy	90-08 Abundance (database)
applications	Grey, Nigel	91-06 Big Blue on the move IBM CAD (review)
applications	Kendall, Les	91-02 Terminal emulator for PC (code)
applications	Smith, Graham	91-02 Logic gates
applications	Stephens, Chris	93-08 Seven thousand networked micros
applications	**Trueblood, Mike**	**99-11 Radio Clock**
arithmetic	Bennett, Paul	97-02 From the 'Net - Square Roots (code)
arithmetic	Brown, Jack	92-10 Floored v symmetric division (tutorial)
arithmetic	Filbey, Gil	91-04 Tutorial
arithmetic	Filbey, Gil	91-12 Mixed point arithmetic (tutorial)
arithmetic	Filbey, Gil	92-02 Mixed point arithmetic (tutorial)
arithmetic	Filbey, Gil	92-04 Mixed point arithmetic (tutorial)
arithmetic	Filbey, Gil	93-02 Floating point
arithmetic	Filbey, Gil	95-02 Cube roots
arithmetic	Haley, Andrew	91-04 Function approx. by Chebyshev series
arithmetic	Hersom, Ed	98-07 Quad (Fixed-Point) Arithmetic
arithmetic	Jakeman, Chris	90-12 A high-level /MOD (code)
arithmetic	Payne, John	91-12 Fixed point arithmetic (word set)
arithmetic	Preston, Philip	91-02 Multi-cell arithmetic (code)
arrays	Brien, Jack	92-02 Ways with arrays (code)
arrays	Jakeman, Chris	90-08 Arrays and records (code)
assembly	Tanner, P.	96-05 Linking machine code modules with Forth

---

block tools	Charlton, Gordon	94-04	One-screen library load (code)
block tools	Filbey, Gil	91-02	Bits and loading blocks (tutorial)
block tools	Hainsworth, Chris	91-02	Editing blocks (tutorial)
bons mots	Bezemer, Hans	97-08	Th
bons mots	Eckert, Brad	97-08	On Off On? Off?
bons mots	Elvey, Dwight	98-01	Setting bits with MASK
bons mots	Hersom, Ed	97-11	NVars [H] [D]
bons mots	Hoyt, Ben	98-03	PLACE is to COUNT as ! is to @
bons mots	Luke, Gary	97-08	Tally
bons mots	Payne, John	97-11	3rd Swap@ Sgn #>ASCII
bons mots	van Norman, Rick	98-03	MANY for debugging
bons mots	Wenham, Alan	97-11	Z
bons mots	Wenham, Alan	98-01	Printing binary with .SB U1B. U2B.
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	Bennett, Paul	91-04	High level FOR..NEXT (code)
control flow	Brien, Jack	91-02	Extended ANS structures (F83 code)
control flow	Brien, Jack	94-06	Extending ANSI control structures
control flow	Brien, Jack	95-06	Portable control structures
control flow	Carpenter, R.H.S.	92-12	Flow-charting method
control flow	Charlton, Gordon	90-04	Universal delimiter (code)
control flow	Charlton, Gordon	95-06	Trouble with DO
control flow	Jakeman, Chris	96-05	If and begin - ANS style
control flow	Preston, Philip	93-06	Shortcuts and drop-outs
database	Filbey, Gil	91-08	FIG UK database (tutorial)
database	Filbey, Gil	91-08	FIG UK database (tutorial)
design	Allwright, R.E.	95-06	Pagination
design	Bennett, Paul	94-08	Taking exception ...
design	Brien, Jack	91-10	Return stack ENTER ISNOW and aliasing
design	Brien, Jack	99-01	Working with Wordlists
design	**Brien, Jack**	**99-06**	**Handling Literals**
design	Charlton, Gordon	93-04	Upside down
design	Flynn, Chris	94-10	Numerical input
design	Hersom, Ed	92-10	NVARS
design	Hersom, Ed	94-08	Simple user interface
design	Jakeman, Chris	95-06	From the 'net
design	Matthews, John	94-02	On his September lecture
design	Payne, John	90-12	Simpler Forth (comment)
design	Smart, Mike	93-10	Computer Shopper Programmer's Challenge
design	Telfer, Graham	96-05	The specification method hunt
design	**Telfer, Graham**	**99-06**	**Skeletons - Designing a Recursive Application**
design	Thomas, Reuben	92-06	Forth lifestyle
dynamic data	Charlton, Gordon	90-04	Dynamic words (code)
dynamic data	Charlton, Gordon	94-06	Work, rest and play
editing tools	Brien, Jack	95-06	Full screen editor
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)

---

graphics	Charlton, Gordon	93-10	Not drawing a line
graphics	Filbey, Gil	90-04	Plotting spirals (tutorial)
graphics	Payne, John	92-08	Flood fill
graphics	Payne, John	93-10	How Bresenham's line drawing alg. works
graphics	**Pochin, Dave**	**99-08**	**Figuring it out with Win32Forth**
graphics	**Pochin, Dave**	**00-01**	**"See Win32Forth scroll the Window"**
hardware	Bennett, Paul	96-07	Chuck's chips
hardware	Fowell, Jeremy	92-08	P20 chip, part 1/2
hardware	Fowell, Jeremy	92-10	P20 chip, part 2/2
hardware	Fowell, Jeremy	99-01	FIG UK Hardware Project
hardware	**Fowell, Jeremy**	**99-04**	**FIG UK Hardware Project - Progress**
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	**Heuvel, Leendert**	**99-04**	**The 'Egel Coursebook**
hardware	Koopman, Philip	90-10	RTX 4000 (publicity)
history	**Behringer, Fred**	**99-11**	**Swap Dragon**
history	**Brien, Jack**	**99-11**	**FIG UK - The Last 20 Years**
history	Hainsworth, Chris	99-01	Forthwrite Issue No. 1 revisited
history	**Jakeman, Chris**	**00-01**	**Did you Know? - EasyWriter**
history	Powell, Bill	99-01	The Birth of FIG UK
history	Rather, Elizabeth	95-04	The evolution of Forth
history	Rather, Elizabeth	95-12	The Forth approach to operating systems
humour	Allwright, Ray	98-05	A Story of Cowboys
humour	Jakeman, Chris	96-05	From the 'net - a drinking song
humour	Payne, John	90-12	A program that works the French way
humour	Smith, Graham	95-06	Book titles
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
interfacing	Bennett, Paul	99-01	Writing the World - 2
interfacing	Robinson, Dave	91-08	Mouse handling (F83 code)
internals	Allwright, Ray	98-03	From the 'Net - Minimal word sets
internals	**Allwright, Ray**	**99-04**	**From the 'Net - Turnkey Apps and Docs**
internals	Bennett, Paul	92-10	Top-down development of a Forth system
internals	Bennett, Paul	93-04	QUIT, the story continues...
internals	Brien, Jack	97-08	Building a new inner interpreter
internals	Charlton, Gordon	91-02	A replacement for DO .. LOOP (code)
internals	Flynn, Chris	91-06	Forth engine on 68000
internals	Hainsworth, Chris	90-02	Kiss and run (exploring F-PC)
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
interpreting	Brien, Jack	96-11	Implementing an outer interpreter
interpreting	Jakeman, Chris	95-10	From the 'net - text interpreter
interview	**Moore, Charles**	**99-06**	**1xForth**
library	Hainsworth, Sylvia	91-04	FIG UK library bulletin

---

library	Hainsworth, Sylvia	98-05	Purchases and current publications
library	Jakeman, Chris	96-11	Library assets
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
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
presentation	Bennett, Paul	91-06	Manual documentation (code)
presentation	Brien, Jack	90-02	Locals and more (discussion)
presentation	Brien, Jack	91-02	GIST for indexing source (code)
presentation	Brien, Jack	94-10	Readable Forth
presentation	Charlton, Gordon	93-12	StackFlow
presentation	Charlton, Gordon	97-02	From the 'Net - StackFlow
presentation	Matthews, Keith	90-12	Stack diagrams (explored)
presentation	Tanner, P.H.	94-12	Post indentation
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	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
puzzles	Hainsworth, Chris	90-06	Forth brain teasers
random nos.	Filbey, Gil	93-06	Visualising random numbers on Apple II
random nos.	Filbey, Gil	93-08	Testing for randomness
random nos.	Jakeman, Chris	93-06	Random numbers
random nos.	Payne, John	93-08	More on random numbers
review	**Anderson, Joe**	**99-06**	**Forth for Virtual Reality**
review	Bennett, Paul	97-11	EuroForth '97 Conference
review	Bennett, Paul	98-11	EuroForth '98 Conference
review	Charlton, Gordon	94-10	Riding the wild 'net
review	Charlton, Gordon	95-02	Report from EuroForth '94
review	Flynn, Chris	98-10	A Hard Day Garbage Collecting
review	Fowell, Jeremy	98-05	Forth Programmers' Handbook
review	Jakeman, Chris	98-05	Genetix - The Inside Story
review	Jakeman, Chris	98-10	jeForth
review	**Jakeman, Chris**	**00-01**	**FIG UK 20th Anniversary Reunion**
review	Payne, John	98-07	FORML Proceedings 94 & 95
review	Wenham, Alan	98-01	Vierte Dimension
review	Wenham, Alan	99-01	Vierte Dimension
review	**Wenham, Alan**	**99-11**	**Vierte Dimension**
review	**Wenham, Alan**	**00-01**	**Vierte Dimension 4/99**
roots	Behringer, Fred	98-03	Square roots once more
roots	Behringer, Fred	98-05	Cubic roots without division
roots	Brien, Jack	97-11	From the Net - More on square roots
roots	Charlton, Gordon	90-10	Square root (code)

---

roots	**Jakeman, Chris**	**00-01 Cube Roots Again**
roots	**Jakeman, Chris**	**00-01 From the 'Net - Cube Roots**
roots	Trapp, John	91-02 Square-roots for double/floating point
roots	Wilson, R.J.	90-08 Root of rational numbers (code)
searching	Charlton, Gordon	90-12 A faster string search (code)
searching	Charlton, Gordon	91-10 A binary search (code)
searching	Charlton, Gordon	93-02 Shift-AND string search (code)
searching	Charlton, Gordon	94-02 Best string search (code)
searching	Hersom, Ed	91-12 Recursive BINSEARCH (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	Barr, Stan	95-12 A third stack
stacks	Charlton, Gordon	94-04 Stacrobaticus exotica
stacks	Filbey, Gil	94-08 Stacks (tutorial)
stacks	Hersom, Ed	97-11 Multi-precision Stack Operators
stacks	Jakeman, Chris	95-08 Stack manipulation
stacks	Joseph, Neville	95-10 Stack manipulation
stacks	Preston, Philip	92-12 Stocking fillers - stacks & write-only
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	Borrell, Richard	98-03 Deferred Language Translation
strings	Brien, Jack	93-06 Comment on Blockl & Tack
strings	Brien, Jack	96-07 String handling
strings	Charlton, Gordon	91-04 A string pattern matcher (code)
strings	Charlton, Gordon	93-04 ANSI and portability - STRLIT (code)
strings	Charlton, Gordon	93-06 Similarity
strings	Jakeman, Chris	95-12 From the 'net - please
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	Leibniz, David	91-02 String stack routine (code)
strings	MacLean, Ruaridh	91-02 High level DIGIT (tutorial)
strings	Oakford, Howerd	98-11 Multiple Language Programs Made Easy
strings	Payne, John	92-04 Text processing
strings	Preston, Philip	92-10 TACK and BLOCKL
structures	Brien, Jack	98-01 Building Forth Structures
systems	Behringer, Fred	97-08 Forth for the Transputer
systems	Bennett, Paul	92-02 Pygmy Forth (review)
systems	Besemer, Hans	98-05 4th - The Alternative Compiler
systems	Green, Roedy	90-08 BBL Forth (review)
systems	Hersom, Ed	93-02 Pocket Forth (review)
systems	Jakeman, Chris	99-01 Web Forth Project
systems	**Jakeman, Chris**	**99-06 Web Forth Project**
