ISSN 0265-5195

April
2001

Issue 111

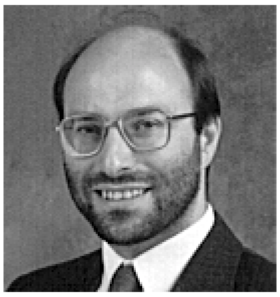
Forthwrite

news events people reviews projects programming

Six Easy Fonts
An exercise in Win32Forth

FIGUK magazine:

April
2001
Issue 111



Editorial

Here we are at last.

We welcome Leo Wong back into print – his subject may be lightweight but his treatment repays study. Thanks to Leo and Fred Behringer, Ed Hersom's permutations turn up twice more in this issue – surely a record! Dave Pochin also returns with yet another contribution on using the Windows environment. This issue contains some start-of-year material held over from January.

Do take a close look at the Index printed at the end of this issue. It shows how much good work has been published and also reveals some gaps to be filled. The Letters section also includes some interesting suggestions.

Welcome to three new members, Gianluca Massotti of London who is studying 3D graphics, Boris Fennema from Ireland and Erik Johansson, who is our third member from Sweden.

Look out for the June issue with David Abraham's review of the new Mindstorms book, a report of my visit to the German FIG 2-day Conference and details of Forth on the NEAR space probe.

Don't forget the monthly IRC session. There's now a countdown reminder on our web site.

I'll finish this time by thanking those many members who accompany their renewals with words of encouragement. It's great to know our efforts are appreciated.

[illegible]

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

Forth News

FORTH PUBLICATIONS

John Hall has revised the FIG web site, http://www.forth.org and has organised a new searchable database of Forth publications. There are 138 entries so far and he is looking for help particularly from authors who have articles in electronic format - contact: jdhall@mac.com

BETA TESTERS NEEDED

In a posting to comp.lang.forth(clf), Albert Lee Mitchell states:

"We are on the cusp of releasing our first family of tethered Forths, the 8051 family, under the LGPL license. Anyone interested in being a Beta tester?" contact: *alm@amresearch.com*

At the moment we are considering a port of amrFORTH to the Motorola 6805/6808 microcontrollers if the demand is sufficient.

http://*www.amresearch.com*

PERSONAL FORTH ROBOT

Don Golding of Angelus Research announced on clf:

"… the introduction of our new personal robot:: Bugsy AI. It uses our real-time Artificial Intelligence control system and of course is a Forth-based machine."

http://www.angelusresearch.com

SIGPLAN

An article by FIG UK member Julian Noble on jump tables and finite state machines will be published in June 2001 issue of ACM's SIGPLan magazine. Chris Jakeman's article on Forth in the UK was published in the December issue of SIGPLan, pages 19-21 of Vol.35 No.12

NEW OWNERSHIP

MegaWolf Inc. http://www.megawolf.com has bought the MacForth products from Forth Inc.. MegaWolf produces hardware and software for the Macintosh and is a long-time user of MacForth. It plans to continue the long history of MacForth, starting with a free upgrade of PMF for existing users.

For the complete press release, see:
http://www.macforth.com/pressrelease1.html

VERSION 0.30 OF PFE

Guido Draheim has announced the release of beta release of vesion 0.30 of the Portable Forth Environment (pfe). Download from:

http://pfe.sourceforge.net

GFORTH

Anton Ertl has begun work on a peephole optimiser for GForth. As a first step he has switched to in-line implementations for elements like literals and DOES> as this less compact style is more suitable.
http://www.complang.tuwien.ac.at/anton/home.html

App-Watch

Continuing the applications theme from the last issue, word has come in about another 3 applications in Forth – expect details in a future issue.

The first is a set of commercial robots used to handle delicate electronic equipment.

The second is a British software application used by many large corporations including Microsoft.

The third is an implementation for Windows of the Unix “cron” tool which runs background tasks at specified times. It is written and configured in Forth and hails from Russia.

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

Six Easy Fonts
An exercise in Win32Forth

Dave Pochin

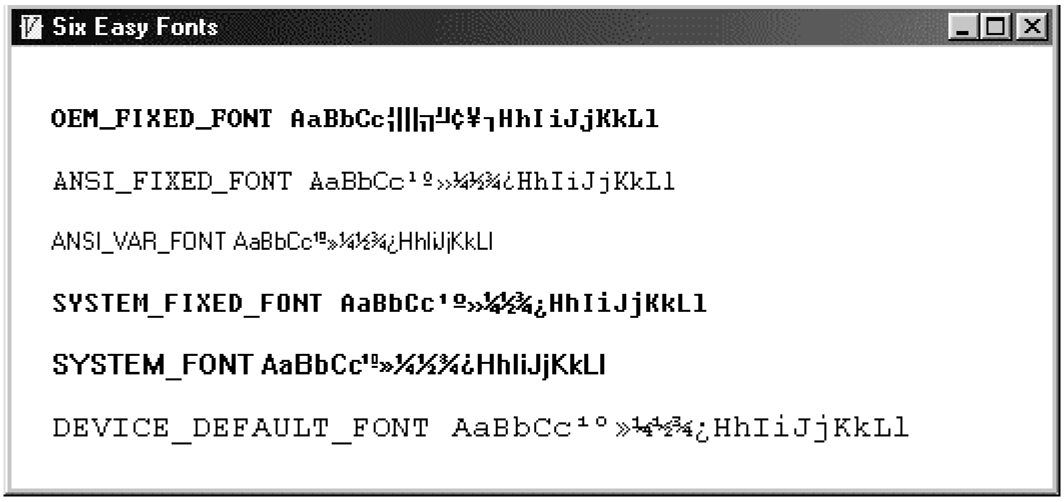
Another useful article in Dave's growing series on using Win32Forth to get the better of Windows.

Fonts and Forth, Forth and Fonts - not a combination I've given much thought to, let alone used. When I need fonts, I've got dozens to choose from in the good old Word Processor using Windows !!! But that's just the point. I've made a fuss about using the Windows part of Win32Forth, why ignore the poor old fonts, because, quite frankly it looks like a big job for very little return, but the beginnings don't seem too terrifying.

The Windows system provides for six Stock Fonts, each of which is slightly different.

- `ANSI_FIXED_FONT`

  "ANSI" refers to the character set, "FIXED" to the pitch between letters.
- `ANSI_VAR_FONT`
- `DEVICE_DEFAULT_FONT`
- `OEM_FIXED_FONT`
- `SYSTEM_FIXED_FONT`
- `SYSTEM_FONT`



```
OEM_FIXED_FONT AaBbCc╡║╢╖╜¢¥┐HhIiJjKkLl

ANSI_FIXED_FONT AaBbCc¹º»¼½¾¿HhIiJjKkLl

ANSI_VAR_FONT AaBbCc¹º»¼½¾¿HhIiJjKkLl

SYSTEM_FIXED_FONT AaBbCc¹º»¼½¾¿HhIiJjKkLl

SYSTEM_FONT AaBbCc¹º»¼½¾¿HhIiJjKkLl

DEVICE_DEFAULT_FONT AaBbCc¹º»¼½¾¿HhIiJjKkLl
```

Depending on your particular set up, some of these fonts may appear on the display exactly the same, or not appear at all, so don't blame Microsoft.

These Stock Fonts are part of a large number of Stock Objects available in Windows, and Win32Forth provides a very simple method to call them in the file **DC.f.**

```
    :M SelectStockObject:  ( id -- oldobj )
                     GetStockObject: self  SelectObject: self
    ;M
```

All that is needed is to have the identity of the required font on the stack before calling the function, and to drop the old object identity off the stack afterwards. Note, if you read some of the texts, you will see it is common to store this old object identity as a variable, so you can swap back again later and the listings are full of ‘`new_font`’ and ‘`old_font`’ statements, which just cause confusion at first.

Having selected your font, then carry on printing strings as usual, in the listing I’ve used the TextOut method, again from the file **DC.f**

```
    :M TextOut:      ( x y addr len -- )
                     swap rel>abs 2swap swap hDC Call TextOut ?win-error
    ;M
```

So in the listing you will find;

```
    OEM_FIXED_FONT SelectStockObject: dc DROP
    20 30 s" OEM_FIXED_FONT AaBbCc¹º"1/41/23/4¿HhIiJjKkLl" TextOut: dc
```

- which are two of the prettiest lines of Forth in all of Win32Forth, short, and self contained, no messing about reversing stack parameters to make Windows look like Forth or vice-versa.

The size of the window in the listing is large enough to accommodate each of the font strings. Where indicated, just replace each font title in turn.

As an example of using stock fonts in a real application, look at the two defining words, `: system-fixed-font ... ;` and `: small-font ... ;` in the section ‘ Font Selection ‘ about half way through the file **Window.f**

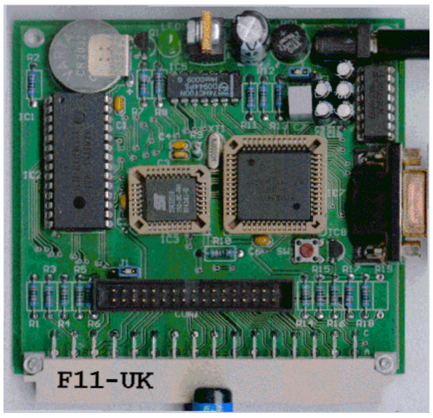
Using these Stock Fonts is almost as easy as picking them from a Menu Bar. Talking of Menu Bars, if you select Display from the Win32Forth console bar, you’ll find an item ‘ List of Fonts in System   .FONTS ‘, which will scroll through the list of fonts available. It’s a couple of pages long !!!! Hmmm !!!

The following listing will open a similar window to the one above and display one of the six fonts. To execute it, enter

```
    START: FONTDEMO
```

Listing

```
ANEW PROGRAM

\ Define an Object that is a super object of the Class "Window".
:OBJECT Fontdemo <SUPER WINDOW

:M ClassInit:   ( -- )                 \ Things to do at window creation.
                ClassInit: SUPER       \ Do anything the class needs.
                ;M

:M WindowStyle: ( -- style )           \ Inherit the style from the class.
                WindowStyle: SUPER
               ;M

:M WindowTitle: ( -- title )           \ Title for the window.
                z" Six Easy Fonts        One example only"
                ;M

:M StartSize:   ( -- width height )    \ Set width and height of window
                520 80
                ;M

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

:M Close:       ( -- )                 \ Do anything the class needs.
                Close: SUPER
                ;M

:M On_Paint:  ( -- )                   \ screen redraw procedure
   /*
   For the other five fonts, replace ' OEM_FIXED_FONT ' in
   both the following two lines with any of :-
     ANSI_FIXED_FONT  ANSI_VAR_FONT  SYSTEM_FIXED_FONT
     SYSTEM_FONT  DEVICE_DEFAULT_FONT
   */
              OEM_FIXED_FONT SelectStockObject: dc drop
              20 30
              s" OEM_FIXED_FONT AaBbCc¹º1/41/23/4¿HhIiJjKkLl"
              TextOut: dc
              ;M

;OBJECT          \ Complete the definition of the new object.
```



0121 440 1809
jeremy.fowell@btinternet.com

F11-UK FIG Hardware Project

The following messages were extracted from the F11 UK mailing list. Although the kit is complete, enhancements and improvements continue.

From Jeremy Fowell

The incremental compile project for the F11-UK board is taking shape. It didn't take very long to sort most of it out except that I had a little trouble with checksums. These are now used to check that the code downloaded to the F11-UK target board is the same as the file on the hard disk.

This is much faster than the original VERIFY. I thought you might like to see some of the code.

The final words for the PC side are as follows.(CHECKSUM is also needed on the HC11 side and should run unchanged).

```
: CHECKSUM     ( a # -- u )
              ( Calc 16-bit checksum  u  for  #  bytes
                starting at address a)

              0 SWAP   FOR  SWAP COUNT ROT +  NEXT  NIP  ;


: FILE-CHECKSUM  ( name -- u )
              ( Calc 16-bit checksum  u  for contents of
                binary file whose name is stored in a
                counted string at address = name.)

              0 SWAP  FILE-OPEN  0
              BEGIN                              ( u)
                  128-READ DUP                   ( u a # #)
              WHILE
                  TUCK CHECKSUM SWAP             ( u u1 #)
                  >FIN+!  + ( update file pointer)  ( u+)
              REPEAT
              2DROP   FILE-CLOSE  ;
```

It's quite handy to be able to get the checksum for a file stored on the hard disk. `CHECKSUM` shows the simplicity of the `FOR ... NEXT` loop. These words have now been added to `11DOWNLOAD.4TH` where you can also find the code for `128-READ` and `>FIN+!`.

The following word is used to compare the file containing the most recently compiled PygmyHC11 code with the previous version. The result enables us to determine where the modified code starts, and is the basis for incremental compiling.

```
: COMPARE-FILES  ( a1 a2 -- u1 | T )
              ( Compare two files stored on disk whose
              names are held in counted strings at  a1
              and  a2.  If the files are different return
              u1  the number of bytes from the start of
              the file to the first sector that differs.
              Return true if files are identical.

              The 2 files must be in the current DOS
              directory.  Sectors refer to target FLASH
              memory, see  DOWNLOAD.)
```

I haven't included all the code for this word as it's a bit long.

Sadly January came and went in the blink of an eye, even Christmas seemed a bit short this year. The main reason being that I was unexpectedly asked to fit in a project to design a timer to go inside the terminal box on an industrial motor.

It turned out to be a great exercise in Forth philosophy, stripping out all non-essential parts. I used the smallest available PIC (the 8-pin 508) and got the software down to 82 bytes of assembler. The PCB is under 45 x 45mm. The temperature and EMC environment are bad, but since somebody else paid for all of this I can't say much more about it (maybe a good thing).

About 1/3 of the time was spent trying to sort out the difference between reality and the various pieces of documentation which were written by somebody on another planet. PygmyHC11 was sorely missed.

I look forward to quieter times ahead.

Paul Aksterstam (formerly de Bak), wrote in from Sweden with the solution to a vexing problem which was welcomed by Jeremy and Martin Bitter.

Paul Bak:
After a lot of searching the internet, I have finally come across information that sheds light on the problem I had with uploading PygmyHC11 to the F11-UK from DOS under Windows (either full-screen or a DOS box).

The symptoms were as follows (referring to the F11-UK User Guide, Ver 1.0): When downloading code as per section 10, all worked successfully down to section 10.10, then a problem occurred. I never got the messages in section

10.13. Starting over, I monitored the signals from the serial port and noticed that *nothing* was being transmitted from the PC serial port during the
whole procedure!

It turns out that Microsoft's virtual communication device (VCD.VXD) in Windows 98 Second Edition and Windows Millennium Edition has a bug that incorrectly initializes the default state of the virtual COM ports used by MS-DOS virtual machines. The Microsoft article Q252184 entitled 'MS-DOS-Based Programs Unable to Initialize COM Ports on Computers with ACPI Support' describes the problem and provides a solution.

Exposing the bug

If you are running one of these Windows versions and want to see if your PC suffers from this bug, do the following:

1. Start the PC to an MS-DOS prompt (i.e. not running Windows).
Type MODE COM1:19200 (or MODE COM2:19200), and then press Enter.
If the command runs successfully do the next step.

2. Re-start Windows. Open an MS-DOS window and again type MODE COM1:19200 (or MODE COM2:19200), and press enter. If you receive the error message:

'Function not supported on this computer'

then the bug is present.

Fixing the bug

The bug can be permanently fixed by a small change in the Windows registry. Use either the manual fix or the quick fix I supplied separately.

This fix disables the power management of the COM ports. I had no problem communicating with the F11-UK board from a DOS window after the modification. Even Telix for DOS (my favorite comms program) worked again!

Regards,

Paul Akterstam

Jeremy Fowell again:

This is most welcome news, and obviously involved quite a bit of work.

I have Windows 98 (edition unknown) running on my desktop PC and the Millennium Edition on a laptop, so I will give both a try and report back.

Martin Bitter:

Thank you, Paul, this helped me a lot!

Regards,

Martin

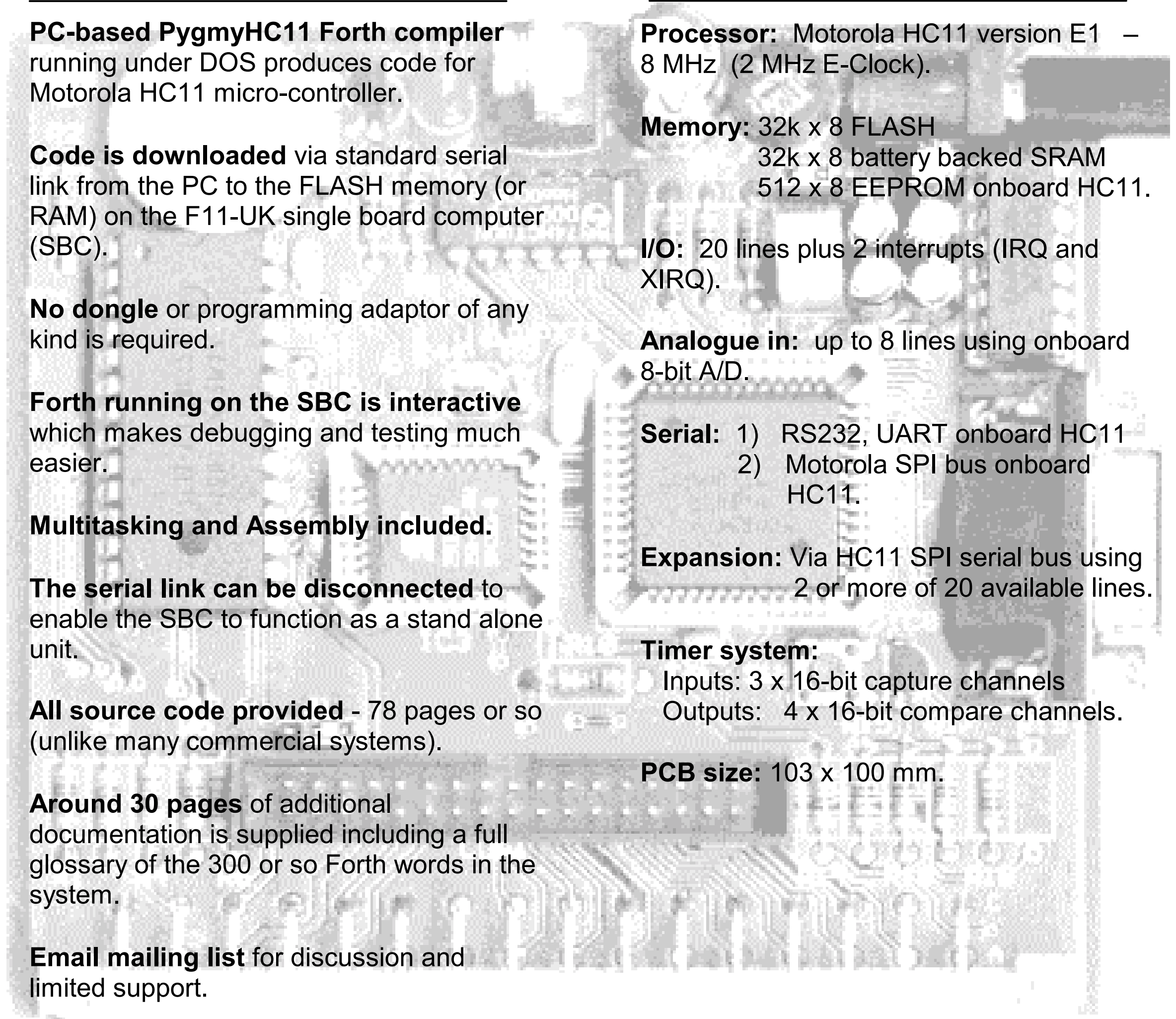
F11-UK

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

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

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

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



Software

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

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

No dongle or programming adaptor of any kind is required.

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

Multitasking and Assembly included.

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

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

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

Email mailing list for discussion and limited support.

Hardware:

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

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

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

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

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

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

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

PCB size: 103 x 100 mm.

Price to FIG UK members:	£47.0 plus postage and packing (£2 UK, £4 overseas) plus $25.0 (US Dollars) for registration of 80x86 Pygmy Forth with the author Frank Sergeant.
Delivery:	ex-stock.
More information:	jeremy.fowell@btinternet.com and 0121 440 1809

euroFORTH 2001

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

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

FIG UK member Bill Stoddart (W.J.Stoddart@tees.ac.uk) is the Program Chair and invites papers on both academic and business topics. Dates for submissions are listed at http://dec.bournemouth.ac.uk/forth/euro/ef01.html

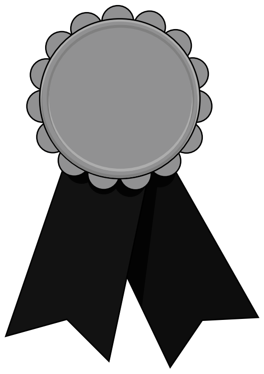
euroFORTH is the only international conference on the programming language Forth, its underlying principles and its innovative potential for product development.

Conference chair:-

Dr. Bill Stoddart
School of Computing &
Mathematics,
University of Teesside,
Middlesbrough, Cleveland.
TS1 3BA
Tel: +44 (0)1642 342 673
Fax: +44 (0)1642 230 527
W.J.Stoddart@tees.ac.uk

Program chair:-

Dr. Peter Knaggs,
Bournemouth University,
Talbot Campus, Fern Barrow,
Poole, Dorset,
UK, BH12 5BB
tel: +44 1202 595625
fax: +44 1202 595314
pknaggs@bournemouth.ac.uk



Nominations for the FIG UK Awards - 2000

The FIG UK Awards of 1999 were won by Jeremy Fowell and Alan Wenham. These awards are given to encourage effort and recognise achievement.
Please take the time to look back over the past year and send in your personal nominations for 2000.

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 2000. 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 2000. 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. 2000 are eligible (except the judges).

Fred Behringer
behringe@mathematik.tu-muenchen.de

Generating Combinations

Fred Behringer

subtitled - “Printing All Combinations of K 1's in an N-Bit Word in High-Level Forth”.

Editor’s Comment: I don’t understand some of the mathematical terms in the second paragraph but the bit patterns reveal what is being done and how.
In the listing, note the way `2-NTH` looks up the answer in the table, a classic technique which is far easier in Forth than most languages.

This note was inspired by Ed Hersom's program on "Simple Permutations" as presented by Chris Jakeman in issue 110 of Forthwrite. As with Ed's, this note is in high-level Forth (ignoring any efficiency considerations) and it relies on `RECURSE`. It deals with systematically generating all combinations of k distinct items taken at a time from a set of n items, irrespective of permutations - to be precise, all instances of k 1's in an n-bit binary word.

The problem arose when I tried to convert the disjunctive standard representation of an n-variable boolean function into its multilinear arithmetic equivalent. One field of application is Reliability Theory with its multicomponent systems of either malfunctioning or well-working parts. Another field of conceivable application is a multisensor robot. Imagine taking appropriate measures to the signals coming from Ralph Hempel's Lego robots as extended to, say, 16 homebrew sensors following Michael Gasperi's Web proposals.

Algorithm: From right to left, fill-up the n-bit "word" with the required number, let's say k, of 1's.

Given any combination of k 1's in the word, e.g. 11000101, shift the first 1 from the left which has a leading 0, one bit to the left, here 11001001, and restart the process with the smaller problem (the "subtree") of keeping the "right-hand part" of the word, here `....1001`, unchanged, i.e., take `00001001` as an intermediate starting word, the previous "node" as the root of the subtree of the remaining (forward) recursions, in our example.

Illustration: The following sequence of combinations of three 1's in a six-bit word, as obtained by the program to follow, will (hopefully) illustrate the way the algorithm works. The sequence should be read from left to right, line by line.

```
1      2      3      4      5      6      7      8      9      10
000111 001011 010011 100011 001101 010101 100101 011001 101001 110001
11     12     13     14     15     16     17     18     19     20
001110 010110 100110 011010 101010 110010 011100 101100 110100 111000
```

Note that "words" 2 to 4 have 011 as their fixed intermediate part and the remaining problem is the one of finding all "combinations" of one 1 in three bits (the leftmost three). "Words" 5 to 7 take up the recurrence (of shifting the leftmost 1 by one bit to the left and restarting a smaller problem) at the right-hand part (i.e. 000011) of "word" 2, retaining 000101 as the "fixed part".

Implementation: The filling-up with 1's is done by the else-part of the first `RECURSE`. This is controlled by the parameter "count" which had previously been placed on stack. The shifting to the left of the leftmost 1 led by a 0 is done by the if-part of the first `RECURSE`. Backtracking in case of a 1 being shifted out of the "word" to the left is done by the second `RECURSE`. Actually, this is to be considered in the negative sense: the second `RECURSE` stands for resuming recursion. For this purpose, the part of the word that remains when a 1 is shifted out to the left, irrespective of the number of 1's stored in the parameter "count", is saved in the variable `ACCU`. When backtracking, the content of `ACCU` is compared with the parameter "accu" at each stage of the backtracking process. The second `RECURSE` is not executed, i.e. another step of backtracking is added, as long as the content of `ACCU` does not yet match the value of the parameter "accu" at the particular stage. The overall backtracking process stops, i.e., there will be no more restarting forward recursions, once the very first triple of parameters (level=0 count=0 accu=0), the root of the entire tree, is met again.

Remark: Clearly, the depth of recursion will never exceed the number of bits in the "word". The main purpose of backtracking is to get to the parameters "level" and "count" of the intermediate stage of restart signalled by the current content of `ACCU`. The human eye is capable of immediately seeing where the leftmost 1 with a leading 0 is placed. Assuming that, by low-level bit counting or other devices, the program could be designed to execute this task in "one step", avoiding several steps of backtracking, there would still remain the problem of the number of (forward) recursions becoming excessively large. There is no such problem in the program to follow since there we have an alternating succession of forward and backward recursions.

Time required: The number of ordered sets of k 1's in n bits is given by the respective binomial coefficient (the product of the largest k integers smaller than, or equal to, n, devided by the product of the first k positive integers). The largest number of feasible bits (wordlength) in my program is 16_d. So, clearly, the largest number of combinations is reached for k = 8 while n = 16, amounting to $12{,}870_d$. Hence, that number of combinations to be computed can be regarded the worst case for timing considerations. On a 486/66 machine, the worst case just mentioned needed about 1 second in case of no display, and 48 seconds when the individual combinations of 8 1's in 16 bits were displayed on screen.

Listing:

```
HEX
HERE
  1 ,   2 ,   4 ,   8 ,   10 ,   20 ,   40 ,   80 ,
100 , 200 , 400 , 800 , 1000 , 2000 , 4000 , 8000 , 10000 ,
                          \ 10000 is provided for the case of an
                          \ excessive 1 in a 16 bit situation.
: 2-NTH ( n -- 2^n )      \ 0 <= n <= 1F
   [ DUP ] LITERAL        \ [ DUP ] and DROP because of Turbo Forth's
   SWAP CELLS + @ ; DROP  \ stack balance error preventing mechanism

VARIABLE #BITS            \ "Wordlength"
VARIABLE #ONES            \ Number of 1's to be created
VARIABLE ACCU             \ Powers of 2 accumulated for backtracking

: (K-IN-N) ( level count accu -- )
   2 PICK #BITS @         \ lv cn ac lv (#b)
   #ONES @ - 3 PICK +     \ lv cn ac lv (#b)-(#o)+cn ; any bit room
   <= IF                  \ left ? Otherwise go to previous stage.
     OVER #ONES @ =       \ Enough 1's collected in count ?
     IF
       BASE @ OVER 2 BASE ! CR 0\ Print next group of #ONES 1's in a 'word'
       <# #BITS @ 0 DO # LOOP    \ of #BITS bits, filled-up by leading 0's.
       #> TYPE SPACE BASE !      \ Print in base two and restore base.
       3DUP 2 PICK 1- 2-NTH -    \ Replace leftmost 1 by 0 , save value
       DUP ACCU !                \ (powers of 2) in the variable ACCU ,
       ROT 1+ -ROT 2 PICK 1-     \ and shift said 1
       2-NTH +                   \ one bit to the left.
     ELSE
       3DUP ROT 1+ ROT 1+ ROT    \ lv cn ac lv+1 cn+1 ac
       5 PICK 2-NTH +            \ lv cn ac lv+1 cn+1 ac-new
     THEN
     RECURSE
   THEN
   3DUP + + 0<>                  \ Stop backtracking ?
   IF
     DUP ACCU @ =                \ Continue backtracking
     IF                          \ if "node" value not yet reached.
       3DUP 2 PICK 1- 2-NTH -    \ Resume recurrence, i.e. replace leftmost
       DUP ACCU !                \ 1 by 0, save value (powers of 2) in
                                 \ ACCU,
       ROT 1+ -ROT 2 PICK 1-     \ shift said 1
       2-NTH +                   \ one bit to the left,
       RECURSE                   \ and continue process of recurrence.
     THEN
   THEN                          \ Backtrack to previous stage
   2DROP DROP ;                  \ and drop parameters.
```

```
: K-IN-N ( #ones #bits -- )
   DUP 1 10 BETWEEN NOT         \ Error if "wordlength" out of range
   ABORT" Input error" #BITS !
   DUP 1 #BITS @ BETWEEN NOT    \ Error if number of 1's out of range
   ABORT" Input error" #ONES !
   0. 0 (K-IN-N) ;
```

Chris Jakeman
cjakeman@bigfoot.com

Did you Know?
– NEAR Space Probe

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).

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

All the instruments and the command and data handling system
were programmed in Forth. This event will be covered in detail in our next issue.

Source – John Hayes, John Hopkins University

FIGUK

Moving On

We heard recently that Chris and Sylvia Hainsworth have decided to emigrate to Spain. As Chris put it, "We have been out to the Costa Blanca for our holidays for the last few years, staying with Sylvia's sister and brother in law, and we decided that we really liked the area, the climate and the way of life. Some of our friends moved there last year and others are planning to go fairly soon so it seemed a good idea for us to join the club.

Place in the sun

For years I had assumed that Spain was all like Benidorm around the coast and a total desert elsewhere. In fact, the area around Javea is very pretty along the coast and inland is all green valleys and mountains. We don't yet know exactly where we will settle but probably a few miles inland in the Orba or Jalon valley."

Chris and Sylvia have done a great deal for Forth in the UK over the years and we hope to print an item about that shortly. In the meantime, we wish them lots of happiness in the sun.

New Chairman

I am delighted to report that Jeremy Fowell has agreed to take over as Chairman of FIG UK. Jeremy has been an Ordinary Member of the Committee and has already done much for FIG UK, principally through the Hardware Project. I am sure his energy and enthusiasm will serve us well in the years ahead.

Librarian wanted

However we still need a new Librarian to take over from Sylvia, who has given house room to the Library since its inception and provided an efficient and cheerful service.

The Library has recently been trimmed down to 5 metres of shelf space. It is a valuable and unique resource, being the only Forth Lending library and the largest collection of Forth material anywhere.

If you might like to volunteer to house and run the Library, please contact Jeremy or me as soon as possible, so that Sylvia can finish her packing! (As Editor, I would love to start a regular series with a short piece exploring an item from the library in most issues.)

Look forward to hearing from you,

[illegible]

Alan J M Wenham
01932 786440
101745.3615@compuserve.com

Vierte Dimension 1/01

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.

General

Martin Bitter has replaced Friederich Prinz temporarily as editor for this volume and appeals to the membership for articles. There are four partly provocative letters with excellent responses and seven communications concerning Forth happenings. The FIG-UK advert for membership appears again in this volume.

UUENCODE and UUDECODE

Wil Baden

Fritz Prinz has taken up an Internet contribution by Wil Baden, Martin Bitter has translated it. Code is given for a Forth implementation.

Riddle - Number Representation

Fred Behringer

behringe@mathematik.tu-muenchen.de

Fred gives three numbers:

10001111110
10000101010
100000101000100

which appear to be binary but are in fact the same number in different
bases. Which bases? What number?

Other Groups

Fred Behringer

behringe@mathematik.tu-muenchen.de

Fred reviews the Dutch “Figleaf”, volume 22 and 23

Reports from across the Pond

Henry Vinerts

(these details may well already accessible to FIG-UK members via the Internet)

Three reports from Henry about the regular meetings of Silicon Valley FIG. John Hall seems to have enough energy to arouse FIG International to new life. Annual contributions, however, are still under review, since up to now no further volumes of Forth Dimensions have appeared. No FORML conference is taking place this year.

Henry reports in detail on Chuck Moore's report. None of the former Board of Directors there appeared at the SVFIG meeting.

John Hall, acting president, says that Forth is not yet dead but he but that he can well do without "constructive criticism" from whatever side.

A Salutary Lesson

Joerg Staben

Joerg reviews the years from 1984 up to now and considers that Forth has not been developed in accordance with users' wishes with the inevitable result that it is not widely utilised.

Five Year's Later - a Positive Statement

Joerg Staben

This is related to the item above. Joerg reviews progress and trends and concludes that programming is now event-orientated and visual. He considers that programming is not Forth any more and that nothing can be done about it.

No response from the RCX Microprocessor

Fred Behringer

behringe@mathematik.tu-muenchen.de

Fred describes a method using Forth (he uses TurboForth) and Assembler to generate a short .COM file and he uses this to solve the problem of the infrared transmitter shutting down to the stand-by mode after 5 seconds of inactivity of the RCX brick.

Win32Forth and Graphics

Joerg Staben

Joerg gives several explanations concerning Win32Forth, GDI, OpenGL, 2D-Graphic, and 3D-Graphic.

The Russian Method of Multiplication

Martin Bitter

Martin describes the Russian peasant method of multiplication, known to the Western world for over one hundred years - and known to the Egyptians since 1800 B.C. Keep dividing the multiplier by 2 and multiplying the multiplicand by 2 until the multiplier equals 1, and keep track of remainders, summing the remainders up and adding the sum to the last multiplicand obtained by the said succession of multiplications by 2.

In Fourth Place

Martin Bitter

Martin gives a very lively report on the Lego robot competition in which his own school achieved fourth place out of 70 competitors.

Neal Bridges is well-known for his Quartus Forth for the Palm Pilot. This extract from comp.lang.forth shows how Forth can be used on a web-server to synthesize a minimal web page.

```
Subject: Is it possible to use Forth for CGI?

GForth works fine for CGI.  One example -- I'm using it at
http://www.quartus.net, on the front page, to display the date in Roman
format.

Two things were required: 'warnings off' at the start of the script, 'flush'
after text output, and of course 'bye' at the end of the script.  Here's an
abbreviated example script (hello.cgi):

#! /home/user/gforth-0.4.0/gforth
warnings off

.( Content-Type: text/html) cr cr
.( <html><head><body>)
.( Hello!)
.( </body></html>)

flush
bye
--
Neal Bridges  <http://www.quartus.net> Quartus Handheld Software!
```

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.

Leo Wong
hello@albany.net

Solving a Riddle

Leo Wong

The following riddle was posted by Steve Graham (js.graham@home.com) to newsgroups discussing neural nets, APL, AWK, BASIC, Beta, COBOL, Dylan, Forth, MUMPS and Lisp. Although not an obvious candidate for solving riddles, Forth is flexible enough and Leo shows us how.

There are 5 houses in 5 different colors. In each house lives a person with a different nationality. The 5 owners drink a certain type of beverage, smoke a certain brand of cigar, and keep a certain pet. No owners have the same pet, smoke the same brand of cigar or drink the same beverage.

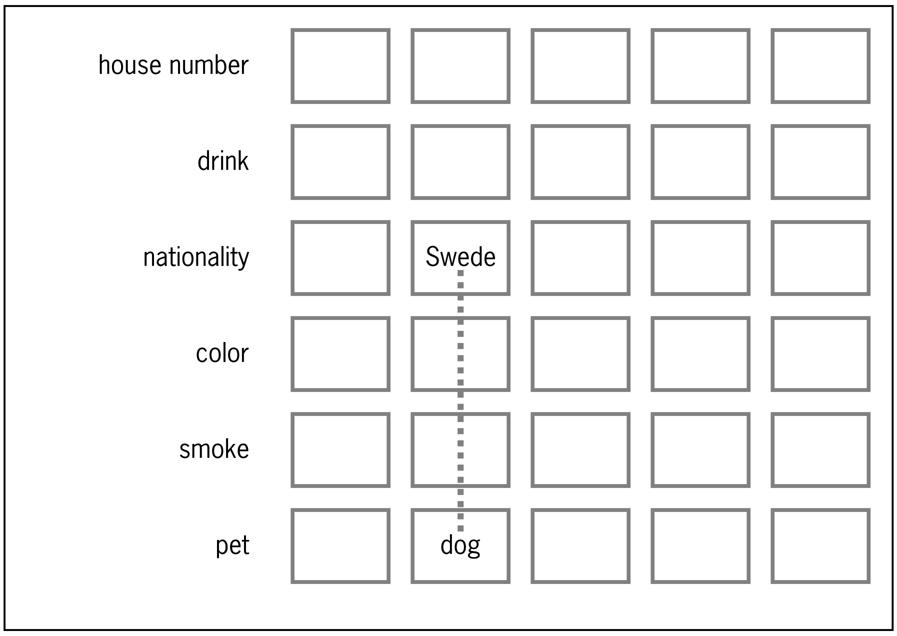
The question is: "Who owns the fish?"

Hints:

1. The Brit lives in the red house.
2. The Swede keeps dogs as pets.
3. The Dane drinks tea.
4. The green house is on the left of the white house.
5. The green house's owner drinks coffee.
6. The person who smokes Pall Mall rears birds.
7. The owner of the yellow house smokes Dunhill.
8. The man living in the center house drinks milk.
9. The Norwegian lives in the first house.
10. The man who smokes Blends lives next to the one who keeps cats.
11. The man who keeps the horse lives next to the man who smokes Dunhill.
12. The owner who smokes Bluemasters drinks beer.
13. The German smokes Prince.
14. The Norwegian lives next to the blue house.
15. The man who smokes Blends has a neighbor who drinks water.

Editor: The brute force way to solve this is by visiting all the permutations, exiting as soon as all the constraints are met and announcing the solution. Leo's program is more subtle, recognising that there are 24,883,200,000 (120^5) possible arrangements and using the hints to *solve* the riddle, instead of just checking to see if a particular permutation *has solved* the riddle. Leo's comments are in quotes.

He first modelled the problem using 30 cards laid out in rows, one for each category. "When a hint linked two categories (for example, "Swede" and "dog") I used scotch tape to keep them at the right distance apart so that placing "Swede" in the nationality row (ie. finding its position) would place "dog" in the pets row. After I had thus "modelled" the riddle, I solved it by hand in considerably fewer than 20 minutes."



“One of the strengths and weaknesses of Forth is that (aside from common problems solved by standard Forth words), it doesn't come with ready-made solutions or approaches. Having come up with a method that I liked, I had to teach it to Forth.”

Editor: This problem is suited to the Prolog language. Note that Prolog-like solvers have been successfully implemented in Forth and, indeed, the necessary back-tracking mechanisms can be implemented more efficiently in Forth than in a conventional language.

With Leo’s approach, storing the data in the right way simplifies the program dramatically.

His code repays some study, not just because he solves the problem in an interesting way, but also because of the Forth techniques he uses including:

- building data structures at compile time,
- using character cells to store numeric data where this is more convenient.
- using macros (sparingly) to make the code more readable

“One thing I noticed as I moved the cards around was that it was no use arranging, say, the colors if, in the row above (in this case, nationalities), each alternative did not at least have its own place (the Dane and the Swede cannot live in the same house) - though not necessarily its right place. This notion I incorporated in the word `placed`, which tells the program to consider the next row only if the "elements" in the row above are each in a house, correct house or not. This ensures that a key constraint is met (only one element of each category to a house) and dispenses with much inner looping.”

“So my strategy (not necessarily the best possible) had three tactics:

1. Reduce the number of possibilities.
2. Apply the hints along the way rather than using them at the end to check if the solution is correct.
3. Go to the next row only if the row above was "placed".

Tactics 2 and 3 turned out to be highly effective.”

“`permute` computes 120 permutations of the five houses and commas them into the chars array called `perms`. Each number stand for a house.” A dump of `perms` shows

```
120 5   4 3 2 1 0   3 4 2 1 0   2 4 3 1 0   4 2 3 1 0  etc..
```

“`board` is a pad for tallying how many of a category's elements are in each house. For `placed` to return true, each house should have one element, or in other words, since there are five houses and each category has five elements, no house should have no elements. So for a particular category, `placed` first clears the board by filling with `0`, then fetches the house number of each element and increments the tally for that house number, and finally looks to see if any house has a zero tally. If it does, the elements in that category are not satisfactory and inner loops of `riddle` are skipped. Note that board is allotted 6 chars because Hint 4 requires that “white” is placed beyond “`green`” which might be placed in house 4.

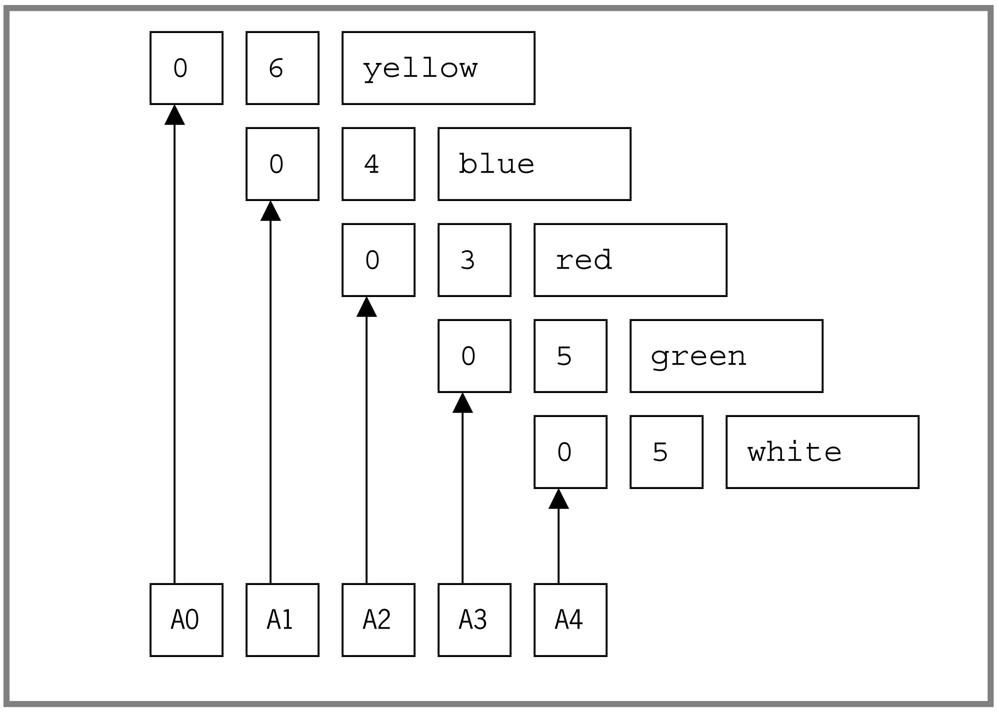
```
: n!  ( n -- n! )  dup 2 < if drop 1 exit then  dup 1- recurse * ;
0 value #items
: ,perm   ( -- )  #items 0 ?do i pick c, loop ;
: perm  ( <items> #items -- <items> #items )
   dup 1 = if  >r ,perm r>
   else dup 0 do >r  r@ 1- recurse  roll r> loop then
;
: drops  ( n -- )  0 ?do drop loop ;  \ Also used by riddle

: permutations
   create  ( <items> #items -- )
     dup n! ,  dup chars c,  dup to #items
     perm drops
   does>  ( -- #items a #perms )  dup @ >r
     cell+ count swap r> ;

0 1 2 3 4  5 permutations perms

: string,  ( a u -- )  dup c, 0 do
                       count c, loop drop ;
: spells  ( a u -- a' )  create here >r  0 c,  string,  r> ;
: ,s  ( x1 ... xn n -- )  begin ?dup while dup roll , 1- repeat ;
: category  ( x1 x2 x3 x4 x5 -- )  create  5 ,s ;
```

`perm` is based on Ed Hersom’s offering in the Nov. 2000 issue

`count` is commonly used to prepare parameters for type. In `permutations` and in `string,` Leo uses it to index through a sequence of bytes.

```
\ colors
s" yellow" spells yellow
s" blue" spells blue
s" red" spells red
s" green" spells green
s" white" spells white  category colors
```

`spells` compiles the text preceded by a character-sized cell to hold the current house number for the element. These are compiled into a list (see A0-A4) by `category`.



The phrase `2 milk c!` (as used later) records the statement in Hint 8.

```
\ nationalities
s" Brit" spells brit
s" Dane" spells dane
s" Norwegian" spells norwegian
s" German" spells german
s" Swede" spells swede  category nationalities

\ drinks
s" beer" spells beer
s" milk" spells milk
s" tea" spells tea
s" coffee" spells coffee
s" water" spells water  category drinks
```

```
\ smokes
s" Blaumeister" spells blaumeister
s" blends" spells blends
s" Dunhill" spells dunhill
s" Prince" spells prince
s" Pall Mall" spells pallmall  category smokes

\ pets
s" birds" spells birds
s" cats" spells cats
s" dogs" spells dogs
s" fish" spells fish
s" horse" spells horse  category pets
```

The following hints record fixed allocations.

```
2 milk c!                \ hint 8
0 norwegian c!           \ hint 9
norwegian c@ 1+ blue c!  \ hint 14
```

The other hints depend on more complex relationships as follows.

```
: colors!  ( permutation -- colors )
   count red c!
   count yellow c!
   c@ dup green c!  1+ white c!  \ hint 4
   colors ;
```

so that `colors!` allocates houses to red, yellow and green colours and also follows Hint 4 which links the houses that are green and white. The blue colour is not included as this was specified by Hint 14 above.

`perms{ colors! ... }perms` is used below to solve the riddle and this expands to

```
perms 0 do dup colors! ... over + loop
```

`colors!` pulls values from the sequence of 120 x 5 values in `perms` and then tests each of the 120 results with `placed`.

```
: nationalities!  ( permutation -- nationalities )
   count dane c!  count german c!  c@ swede c!
   red c@ brit c!         \ hint 1
   nationalities ;
```

```
: drinks!  ( permutation -- drinks )
   count beer c!  c@ water c!
   dane c@ tea c!          \ hint 3
   green c@ coffee c!      \ hint 4
   drinks ;

: smokes!  ( permutation -- smokes )
   count blends c!  c@ pallmall c!
   yellow c@ dunhill c!    \ hint 7
   beer c@ blaumeister c!  \ hint 12
   german c@ prince c!     \ hint 13
   smokes ;

: pets!  ( permutation -- pets )
   count cats c!  count fish c!  c@ horse c!
   swede c@ dogs c!      \ hint 2
   pallmall c@ birds c!  \ hint 6
   pets ;

create board 6 chars allot
: c++  ( a -- )  dup c@ 1+ swap c! ;          \ Increment char at address
: scan  ( ca1 u1 c -- ca2 u2 )
   >r
   begin dup while over c@ r@ <> while 1 /string repeat then
   r> drop ;
: cut  ( c ca u -- n )  rot scan nip ;       \ n=remaining chars including c
: placed  ( category -- ? )
   board 5 0 fill
   5 0 do dup @ c@ chars board + c++ cell+ loop drop
   0 board 5 cut 0= ;

: macro  ( "name <char> ccc<char>" -- )
  \ by Wil Baden
   : char parse
   postpone sliteral
   postpone evaluate
   postpone ;
   immediate
;
```

?no is a convenience word to save typing and created by macro.

macro (a word supplied by Wil Baden) just inserts the text between the delimiters. It's also used later on.

Some of the hints cannot be expressed with the fixed rules or simple rules above, so these are listed below and tested as the last step. The hints which are commented out could be tested in the same way. This would implement the brute force method and require each permutation of the 24,883,200,000 to be tested until the solution was found. Leo points out that a hint like "Norwegian in the 1st house" could be used to reduce the number of permutations in a category to fewer than 120.

```
macro ?no " ( a1 a2 -- )  - if false exit then"
: constraints  ( -- ? )
\  (  1 ) brit c@ red c@ ?no
\  (  2 ) swede c@ dogs c@ ?no
\  (  3 ) dane c@ tea c@ ?no
\  (  4 ) green c@ white c@ 1- ?no
\  (  5 ) green c@ coffee c@ ?no
\  (  6 ) pallmall c@ birds c@ ?no
\  (  7 ) yellow c@ dunhill c@ ?no
\  (  8 ) milk c@ 2 ?no
\  (  9 ) norwegian c@ 0 ?no
   ( 10 ) blends c@ cats c@ - abs 1 ?no
   ( 11 ) horse c@ dunhill c@ - abs 1 ?no
\  ( 12 ) blaumeister c@ beer c@ ?no
\  ( 13 ) german c@ prince c@ ?no
\  ( 14 ) norwegian c@ blue c@ - abs 1 ?no
   ( 15 ) blends c@ water c@ - abs 1 ?no
   true ;

: .spell  ( a -- )  count type space ;

: .nth  ( n category -- )
   5 0 do
     2dup @ count rot = if .spell leave else drop then cell+
   loop 2drop ;

: .solution  ( -- )
   CR ." The " fish c@ nationalities .nth ." owns the fish." ;

macro perms{ " ( -- n a a )  perms 0 do dup"
macro }perms " ( n a -- )  over + loop 2drop"
macro unloops " ( n -- )  begin ?dup while unloop 1- repeat"

: riddle  ( -- )
   perms{ colors!  placed if
      perms{ nationalities!  placed if
         perms{ drinks!  placed if
           perms{ smokes!  placed if
              perms{ pets!  placed if
                 constraints if
                        .solution  10 drops  5 unloops  exit
                     then
                  then
               }perms then
            }perms then
         }perms then
      }perms then
   }perms ;
```

Leo also supplied some words to show the contents of the matrix of cards:

```
: .row  ( category -- )
   cr
   5 0 do dup
      5 0 do dup @ count j =
         if count dup >r type 12 r> - spaces
         else drop then
      cell+ loop drop
   loop drop ;
: .matrix  ( -- )  \ For Jean Grezel
   nationalities .row
   colors .row
   drinks .row
   smokes .row
   pets .row
;
```

To see the solutions offered in some of the other languages, see
http://members.home.net/js.graham/einstein/

From: Lance Collins [collinsl@bigpond.net.au]

Subject: FIG in Melbourne, Australia

For the last few years Melbourne FIG has become a social club for a group of old friends. We meet on the first Friday of each month for a couple of hours and chat about computer and technology related topics. We have not seriously discussed Forth in the last ten years.

We are down to about ten members but usually they all turn up. Our meeting place is a community centre with a large meeting room but somehow we always cram into the kitchen and have a two hour supper!!.

Four of us (including myself) still program seriously and the language we use is Delphi.

Best wishes to FIG UK

Lance Collins

Letters

The Magazine Team are always pleased to get feedback and encouragement. Here we have news from Andrew Holt on his use of Forth at Sun (no, not Open Firmware this time), an update from Howerd Oakford on the protocol project he presented at euroFORTH and an idea from Graham Telfer. We also had a substantial contribution from Ian Thain, a visitor to our web site and a professional Forth programmer.

Andrew Holt

From: Andrew Holt [andrew.holt@uk.sun.com]
Subject: Learning from Java

I have never mailed anything to Forthwrite before, so here is my first attempt.

I have been using Forth, on & off, for about 15 years, having trod the well-worn path of implementing my own Forth(s).

I am currently doing some Java work and am using a Forth-like language (Fiji by Jack Woehr) as an integration & test tool. (Version 1.1 is on http://www.softwoehr.com , version 1.2 is currently in beta and can be had by mailing Jack.)

Using this caused me to reflect on the future of Forth, and software evelopment in general, so here goes.

Software development is beginning to see the fulfilment of the promise that object-oriented languages offer. More & more quality Java class libraries are becoming available covering a huge diversity of features. For example, there are classes for accessing IMAP mail, and complete SQL databases. (Most of the library stuff I use comes from Sun, off the http://java.sun.com web site but you can find others at places like http://www.gamelan.com .)

The philosophy (espoused by many luminaries in the Forth community) of making better & better wheels is no longer viable, or worthwhile. Some time ago someone mentioned an idea to me that he termed, in a non-pejorative way, "the lazy programmer'.

The simple idea is that "the best line of code is the one that somebody else writes for you". The strength of OOP is that if you find a class that provides the functionality you require but certain methods are poorly implemented you can 'overload' those with your own implementations. If the underlying class changes, so lang as the interface stays the same, nothing else needs to change.

So, selecting from Java class libraries, and glueing the whole thing together with an interactive, interpreted tool is a powerful development model. Fiji is not perfect & does lack some useful features (exception handling, for example) but does provide a useful, platform independent framework. (Tried it on Solaris & MacOS X Beta.

By the wayI still find it amazing that I can compile code on a Sun SPARC box and run it on an Apple PowerPC box.

Anyway, if you have the latest run-time from the Sun web site, it should work OK, but as my home is a Windows-free zone I can't vouch for it personally.

In conclusion it is my belief that if Forth is going to have a future outside of the embedded world this kind of tool is it, otherwise it will become an interesting historical curiosity.

For my own personal projects, I will probably be using Fiji. For Sun projects I already am, currently using it as a test/integration tool. It may make into the actual product if I can address some of its limitations (specifically exception handling).

I have a feeling that I may stir up some sort of response ;)

Regards,

Andrew

Graham Telfer

Thanks for the last Forthwrite. As usual, I enjoyed reading it. I've recently switched over to Win32-4th from Aztec.

I was browsing the Extreme Programming Wiki Wiki Wiki Web FAQ (http://c2.com/cgi/wiki?WikiWikiWebFaq) and thought the idea of an editable page was really good. Letting people contribute and edit material using a Wiki might be a way to rapidly build up excellent reference and tutorial material. Backed up by the magazine on-line, FIG UK could play a leading role.

Maybe some of the developers of Forth would place copies of their manuals and guides on the Wiki. Then people could offer examples of usage or expand the information directly.

I'm sure many other ideas will come to mind. Keep up the good work

Yours,

Graham

Ian Thain

From: Ian H Thain [ian@thain.com]
Subject: Re: Going Forth

Many thanks for your informative reply to my note to Peter Knaggs. I have taken some time to browse around the FIG UK site, which is well laid-out and very presentable, and you are all to be congratulated. At last there's a UK Forth site to which one can point people without feeling faintly embarrassed, and for this relief, much thanks! :)

It's good to see that there are indeed still one or two Forth projects running, but even the most dedicated FIGger (?) would be hard put to claim that Forth has ever been anything more than a minority and peripheral language, and this grieves me more than somewhat. In short, if Forth is ever to take its rightful place as a mainstream programming language (which I am sure we all feel it has every right to do), then it needs a real, professional, marketing job done on it, which will require time, commitment, and - not least - money.

Now, I should say that we probably have as good an opportunity in front of us now as we have ever had. The big telcos have paid about ten times what they should have done for their 3G phone licences, and to the best of my knowledge (which is admittedly partial) there is a painfully obvious dearth of 3G applications around to provide a means of getting their money back, let alone of making a profit. The City investors are throwing decidedly old-fashioned looks towards the likes of Valance and Bonfield, and there is the sound of quite a large flock of telecom chickens coming home to roost.

Forth's portability, minimalist architecture, and speedy development environment make it an absolute peach for all those sweating telco bosses - if only they'd ever heard of us! If we, as a Forth community, pulled out the stops and got ourselves organised, I have little doubt that we could produce those vital apps faster and better than anybody else around, and for the first time ever, perhaps put Forth on the map. (And put a shilling or two in our pockets as well).

But problem No.1 is that we are simply not that well organised.

The biggest issue facing commerical Forth today is the job market - bringing together clients and Forth programmers. However impressive the demo on the exhibition stand may be, the problem I keep getting thrown at me - and for which I really have no adequate answer - is, "So where are all the Forth programmers?"

No technical director or IT manager can be blamed for rejecting Forth in favour of C, C++, or VB, because he knows perfectly well that C programmers are two a penny, and Forth programmers are about as common as hens' teeth. If he needs a C programmer to maintain his code, he needs only to put an ad in the paper and he'll have the CVs of a dozen well qualified applicants in his email the next day. I cannot remember *ever* seeing a job ad for a Forth programmer. I have recently scanned several computer staff agencies both here and in the USA, using Forth as the search keyword, and not a single Forth job did I find anywhere. This is not the sign of a thriving language.

[FIG UK holds a register of members available for projects and positions and forwards details of opportunities to these members - Ed.]

After all the arguments in favour of Forth, the real reason I always come back to it is that I enjoy it; programming in Forth has, quite simply, been so much *fun*. If I had to face a programming future without Forth, I'd probably be tempted to give up computers now and grow spuds for a living.

Best regards,

Ian

Howerd Oakford

From: Howerd Oakford [spred@freenetname.co.uk]
Subject: Re: euroFORTH 2000

Work is hectic at the moment. The company are trying to finish their latest phone by the end of March. This is excellent experience for me, and is being put to good use in my PPP program - I can now open a PPP connection with any Web or WAP site that I have come across (providing I have the relevant passwords). I have UDP output completed, and some rudimentary UDP input (i.e. not fragments), and this and TCP are getting closer...

Perhaps in a few more weeks, I can publish PPP.com on my web site, and you can have a link to the real thing...

Regards,

Howerd

Forthwrite Index

Jack Brien maintains a set of 3 indexes to Forthwrite on the FIG UK web site updating them with each issue. These indexes are sorted by date, by author and by subject going back to 1990. The subject index is repeated in the magazine annually, with the 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, how about writing an article yourself?

Forthwrite Subject Index 1990-2000

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	Kendall, Les	01-01	XML and Forth
applications	Matthews, John	01-01	Forth as Preferred Development Environment
applications	Wong, Leo	01-04	Solving a Riddle
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**
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
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
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	99-08	Figuring it out with Win32Forth
graphics	**Pochin, Dave**	**00-01**	**"See Win32Forth scroll the Window"**
graphics	**Pochin, Dave**	**00-11**	**"BLT is not a Sandwich"**
graphics	**Pochin, Dave**	**01-04**	**Six Easy Fonts**
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**
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**
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
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
interfacing	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**
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
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
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
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, Federico**	**00-04**	**21st FORML Conference**
review	**Wenham, Alan**	**00-04**	**Vierte Dimension 1/00**
review	**Wenham, Alan**	**00-06**	**Vierte Dimension 2/00**
review	**Jakeman, Chris**	**00-08**	**euroForth '989 Conference**
review	**Jakeman, Chris**	**00-11**	**Forth in the UK**
review	**Wenham, Alan**	**00-11**	**Vierte Dimension 3/00**
review	**Ives, Robert**	**01-01**	**"Forth Application Techniques"**
review	**Oakford, Howerd**	**01-01**	**euroFORTH 2000 Conference report**
review	**Wenham, Alan**	**01-01**	**Vierte Dimension 4/00**
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	Stocking 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**
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
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	Jakeman, Chris	99-11	Clock Challenge
tutorial	**Jakeman, Chris**	**00-01**	**Clock Challenge - 2nd instalment**
tutorial	**Brien, Jack**	**00-04**	**All you need to know about STATE, IMMEDIATE and POSTPONE**
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

FIG UK Committee

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

FIG UK Web Site

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

FIG UK Membership

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

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

Forthwrite Deliveries

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

Copyright

FIG UK

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://forth.org.uk. 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.