ISSN 0265-5195

November 2000

Issue 109

Forthwrite

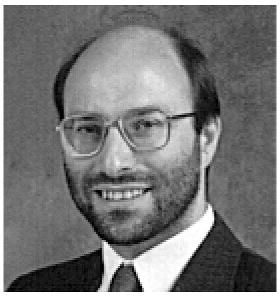
news events people reviews projects programming

BLT is not a Sandwich

More on Forth and Windows

November
2000

Issue 109



Editorial

Hope you like the new front cover and contents page. We've added some hyperlinks too for our electronic readers.

Dave Pochin returns with another in his series on working with Windows. This is becoming a valuable resource – for more, see http://www.sunterr.demon.co.uk , a busy site with over 7,000 visits. Wil Baden makes an appearance in this issue; the first of many I hope.

We welcome Triangle Digital Services Ltd. which has joined our new Corporate Membership programme. This provides member companies with benefits in marketing and recruitment.

Keith Matthews has retired as Treasurer and we have appointed Neville Joseph in his place – see elsewhere in this issue.

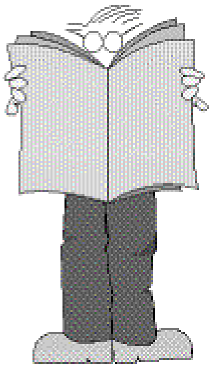
The on-line publication of Forthwrite continues. The download page has seen another 685 visits since the last issue.

Welcome to Roelf Toxopeus, a new member from Utrecht and a warm welcome back to old members Gerry Jackson and Henry McGeough, whose company hosts our web site.

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

Until next time, keep on Forthing,

[illegible]



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

Forth News

FIG UK

Visits to the download page for Forthwrite etc. are running at 300 per issue, so electronic distribution appears to have trebled Forthwrite's circulation.

FIG UK has been invited to submit an article to SIGPLan Notices, a regular publication of ACM, the US Association for Computing Machinery (http://info.acm.org/). SIGPLan stands for Special Interest Group for Programming Languages.

The article, entitled "Forth in the UK" is expected to appear in the December issue of SIGPlan.

Dutch Forth Users Group

Willem Ouwerkerk (chairman) reports that the Dutch Forth Users Group have updated their new web-site. Visit
http://www.forth.hccnet.nl/
for more information about the Dutch Forth Users Group, 8052-ANS-Forth, ByteForth for AT89x051 & AVR, a hardware course, Bamboe and the 'Egel workbook.

Embedded Forth - ANS addition

In a reply to a question on comp.lang.forth from Chris Jakeman on how modern compilers work with embedded systems, Elizabeth Rather of Forth Inc. writes,

"There's a proposed addition to ANS Forth covering the development of Forth on a target system … in a draft at
ftp://ftp.forth.com/pub/ANSForth/
in the files XCtext5.doc and Capp5.doc (also .pdf versions)."

Wil Baden Home Page

The following appeared on comp.lang.forth:

```
Announcing the Neil Bawd & Wil
Baden home page
```
http://home.earthlink.net/~neilbawd/

```
The HTML has been deduced from
plaintext source files.  There are
ten thousand lines of heavily
commented Standard Forth. The HTML
and plaintext are both there.  The
source for the program to markup
the plaintext is included - both
HTML and plaintext.
```

(Note Neil Bawd is an anagram for Wil Baden).

Object Oriented Forth Libraries

Contributors to the comp.lang.forth newsgroup are agreed on the importance of libraries in supporting object-oriented Forth. Anton Ertl is working on a portable library for manipulating data structures (like the Standard Template Library in C++) as part of a larger project. The library will be published once the project is complete.

An OOP version of Forth by Anton can be found at http://www.complang.tuwien.ac.at/forth/objects/

pbForth for Lego MindStorms

Ralph Hempel has announced that version 1.2.1 of pbForth for the Lego Mindstorms system has been released.

There is also a separately available GUI for pbForth which runs under Windows and Linux systems at http://www.hempeldesigngroup.com /lego/pbFORTH

StrongForth

Dr. Stephan Becher announces that the latest version of strongForth is now available for free download on *http://home.t-online.de/home/s.becher/forth/* StrongForth is an ANS Forth with strong static type checking.

Forth Newsgroup

The following two postings to comp.lang.forth from Howerd Oakford, a FIG UK member, make an excellent example of collaboration on the newsgroup.

Date: Wed, 30 Aug 2000
“Does anyone have a Forth version of the MD5 secure hash algorithm?.....”

Date: Mon, 4 Sep 2000
“Thanks to everyone who replied to my quest for MD5.
Ulrich Hoffman sent me a 32 bit Forth version which compiled and ran under Win32Forth first time!”

Presentations

On 19th August, Dr. C. H. Ting gave a presentation to the Silicon Valley chapter of the Forth Interest Group on a VHDL version of his P16 microprocessor core. Jeff Fox of UltraTechnology has created an html page with information from the lecture including the VHDL code. See http://www.ultratechnology.com/p16vhdl.htm

Jef Raskin also gave a presentation about the subject of his new book, the Humane Interface.

Jef Raskin's work on the Canon Cat was featured in June issue of Forthwrite. The ideas embodied in the Cat are developed further in his new book.

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

BLT is not a sandwich

Win32Forth and Windows Block Transfers

Dave Pochin

Another useful contribution from Dave for all Windows users.
Check the Forthwrite index for earlier items in this valuable series.

I'm still passing my time on rainy afternoons looking at the features of Win32Forth which allow me to use Windows functions.

Windows has a function BitBlt (bit block transfer) which can move blocks of bits between devices such as memory, display, printers and plotters etc..

Each device has a 'device context', which is a data structure of information about that particular device.

In turn each 'device context' has a 'handle', which is used to identify and access the 'device context', the handle is very often named as 'hdcnnn', where nnn is the name of the device.

The function BitBlt can do more than just transfer bits from a 'source' device to a 'destination' device, it can operate on both blocks of bits and in some cases the current pattern as well, as part of the transfer process depending on the value of a parameter `dwROP` (in Windows-speak), Win32Forth calls this value `blitmode` which is a little more obvious.

There are 256 possible values for `blitmode`, fifteen of those most commonly used are given special names;- `BLACKNESS`, `MERGECOPY`, `MERGEPAINT`, `PATINVERT`, `PATPAINT`, `PATCOPY`, `DSTINVERT`, `NOTSRCCOPY`, `NOTSRCERASE`, `SRCAND`, `SRCCOPY`, `SRCERASE`, `SRCINVERT`, `SRCPAINT` and `WHITENESS`.

You can make a good guess about the action of many of these `blitmodes`, but others are a little obscure.
For example, `PATPAINT` is described as. ' Combines the colours of the pattern with the colours of the inverted source rectangle using an `OR` operation. The result of this operation is combined with the colours of the destination rectangle using an `OR` operation.' Or if you prefer, as these operations are often defined in the texts in logical terms, `PATPAINT` then becomes `P|~S|D`.

The listing demonstrates directly two of the 15 `blitmodes`, (`DSTINVERT` and `SRCAND`) but can be easily modified, in the method `M: BitBlts:` , to show the remainder.

The display shows the effect of each `blitmode` in a separate row.
From left to right, the columns are :- The SOURCE block, a copy of the DESTINATION block, before the transfer, the name of the `blitmode` and the DESTINATION block after the transfer.

The Windows function is described by :-
```
        BitBlt ( hdcDest, xDest, yDest, xWidth, yHeight,
                 hdcSrc, xSrc, ySrc, dwROP)
```
As usual, to translate this to most Forths the parameter list has to be reversed and the Windows function called with `Call BitBlt`, but using Win32Forth we have been given a Method in the file dc.f , which makes life very easy.

```
:M BitBlt: ( blitmode, sourcex,y sourcedc, sizex,y destinationx,y -- )
```

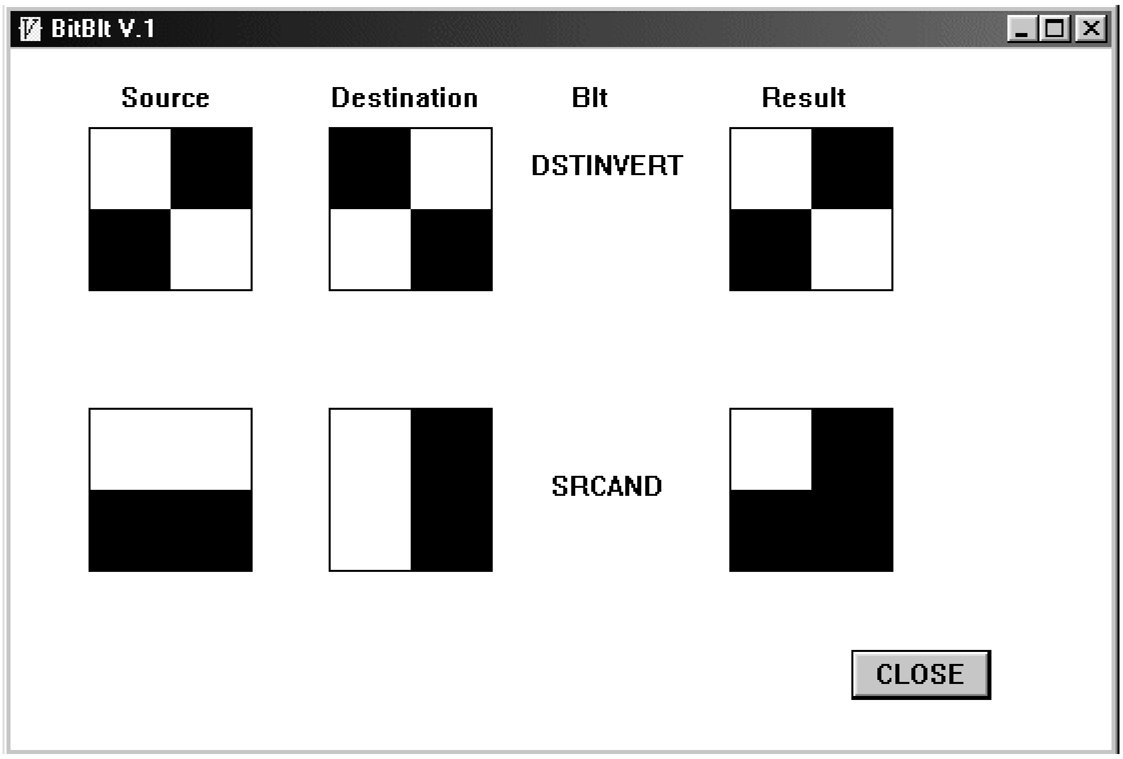
Notice that the parameters are not strictly in reverse order, as the body of the method does a little stack juggling to make everything work. Notice also the there is no Destination source in the parameter list, this again is worked into the body of the method by including the term `hdc`, which makes the screen the destination device.
If you want a different destination, you will not be able to use `:M BitBlt:`

In the listing, the source and destination blocks have been restricted to black and white and a plain light red brush has been used as the pattern, change any of these if you want to experiment.

If you want to see real uses of `:M BitBlt:` , the `blitmode SRCCOPY` (direct copy of the source block to the destination block) is used twice early in the example program Windemo.dc.

As usual, a good textbook is very helpful, not to mention a cup of strong coffee and a proper BLT sandwich.

```
 \ Examples of raster operations heavily based on DC.f
 \ including FillArea and BitBlt, see diagram below.
```



```
\ Define an Object that is a child of the Class Window
 :OBJECT Bltdemo <SUPER WINDOW

  ButtonControl Button_1    \ a button

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

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

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

 :M WindowTitle: ( -- title )
                z" BitBlt V.1 "
                ;M
```

```
:M StartSize:   ( -- width height )
               550 350
               ;M

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

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

\ Set up a Button and create Pens and Brushes.

```
:M On_Init:     ( -- )

               IDOK              SetID:   Button_1
               self              Start:   Button_1
               420 300 70 25     Move:    Button_1
               s" CLOSE"         SetText: Button_1
                                GetStyle: Button_1
               BS_DEFPUSHBUTTON OR
                                SetStyle: Button_1
             ;M

:M SetUps: { left top right bottom -- }
```

\ draw frames for blocks

```
           39  39 MoveTo: dc 120 39 LineTo: dc 120 120 LineTo: dc
           39 120 LineTo: dc  39 39 LineTo: dc

           159  39 MoveTo: dc 240 39 LineTo: dc 240 120 LineTo: dc
           159 120 LineTo: dc 159 39 LineTo: dc

           359  39 MoveTo: dc 440 39 LineTo: dc 440 120 LineTo: dc
           359 120 LineTo: dc 359 39 LineTo: dc

           39 179 MoveTo: dc 120 179 LineTo: dc 120 260 LineTo: dc
           39 260 LineTo: dc  39 179 LineTo: dc

           159 179 MoveTo: dc 240 179 LineTo: dc 240 260 LineTo: dc
           159 260 LineTo: dc 159 179 LineTo: dc

           359 179 MoveTo: dc 440 179 LineTo: dc 440 260 LineTo: dc
           359 260 LineTo: dc 359 179 LineTo: dc
```

\ Make the source, original destination and destination blocks

```
           80 40 120  80 Black FillArea: dc
           40 80  80 120 Black FillArea: dc
```

```
          NOTSRCCOPY 40 40 GetHandle: dc 80 80 160 40 BitBlt: dc
          SRCCOPY   160 40 GetHandle: dc 80 80 360 40 BitBlt: dc

           40 220 120 260 Black FillArea: dc
          200 180 240 260 Black FillArea: dc
          SRCCOPY 160 180 GetHandle: dc 80 80 360 180 BitBlt: dc

          \ Setup the text
           55  16 s" Source" TextOut: dc
          160  16 s" Destination" TextOut: dc
          280  16 s" Blt" TextOut: dc
          375  16 s" Result" TextOut: dc
          260  50 s" DSTINVERT" TextOut: dc
          270 210 s" SRCAND" TextOut: dc
      ;M

 :M BitBlts:
      \ Top row of display. Alternatively use any of
      \  BLACKNESS WHITENESS NOTSRCCOPY SRCCOPY MERGECOPY
      \  PATCOPY PATINVERT PATPAINT
         DSTINVERT 40 40 GetHandle: dc 80 80 360 40 BitBlt: dc

      \ Bottom row of display. Aternatively use any of
      \ SRCERASE SRCINVERT SRCPAINT MERGEPAINT NOTSRCERASE
        SRCAND 40 180 GetHandle: dc 80 80 360 180 BitBlt: dc
 ;M

:M On_Paint:  ( -- )            \ screen redraw procedure
      \ Use this brush as the current pattern
      LTRED BrushColor: dc
      SetUps: self
      BitBlts: self
 ;M

:M WM_COMMAND   ( hwnd msg wparam lparam -- res )
       OVER LOWORD ( Id )
       CASE
               IDOK OF
                       Close: self
                    ENDOF
       ENDCASE
       0
 ;M
;OBJECT

: Go          ( -- )
               Start: Bltdemo ;
 CR CR .( To run type ' Go ' <ENTER> ) CR CR
```

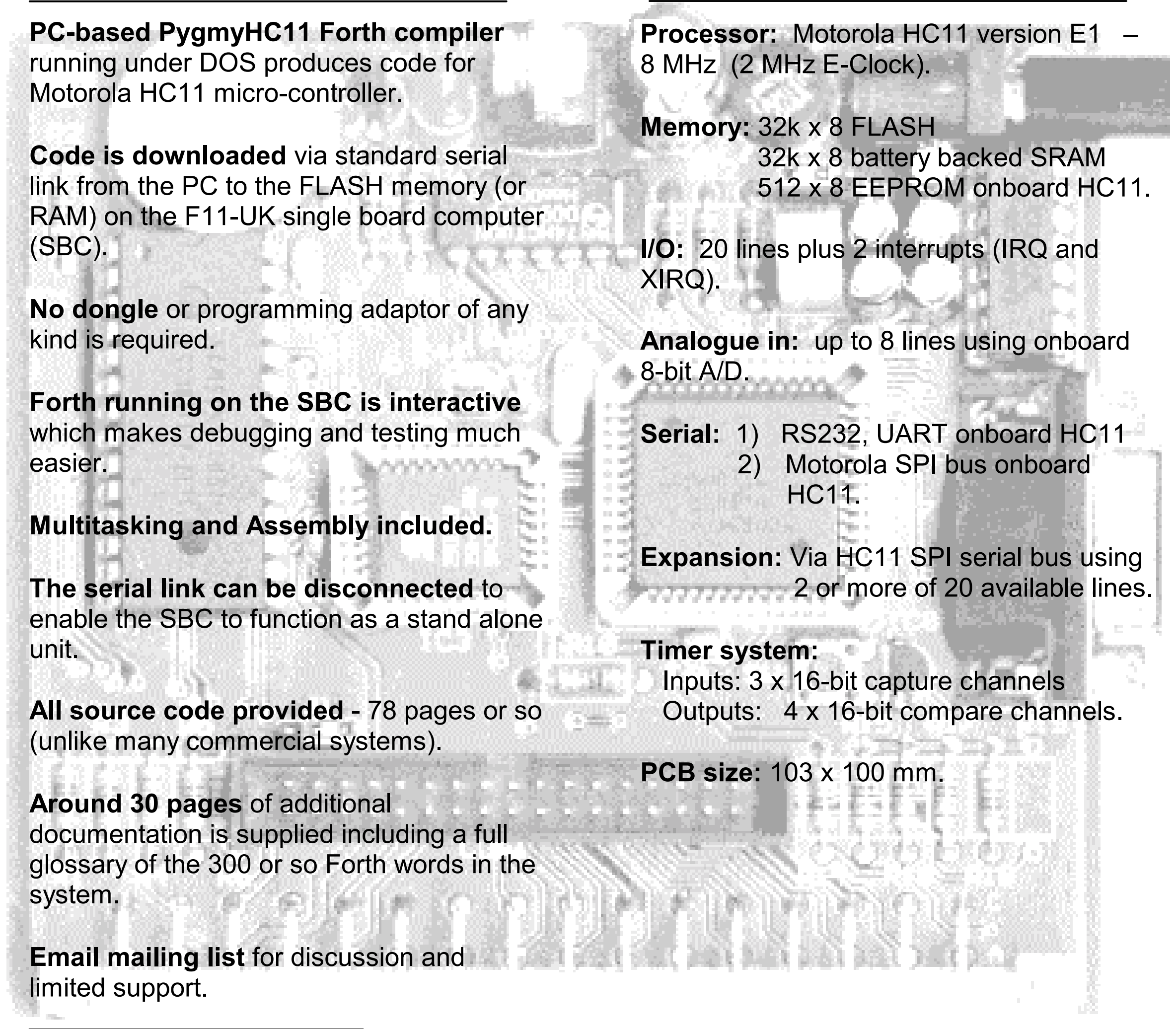
F11-UK

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:	mailto:jeremy.fowell@btinternet.com and 0121 440 1809

Wil Baden
neilbawd@earthlink.net

Permutation by Transposition Sequence

Wil Baden

Wil Baden is well-known for his articles in Forth Dimensions but his is his first appearance in Forthwrite. A recent discussion about permutations on the newsgroup inspired him to publish 2 versions of the “bell-changing” algorithm, where each permutation differs from the next by swapping just an adjacent pair of elements – see the results of the tests at the end of this article.

Although there are simpler ways to find all the permutations of a set, the “bell-changing” approach is valuable whenever there is a significant penalty in switching from one permutation to the next.

Also note the way `'th` is defined and used in the first version.

I suspect this is the first time Forthwrite has contained any Algol 60 code!

In an thread on comp.lang.forth about permutations, I wrote:

```
Here is the permutation method that I prefer. It
was "originated" by me, but I don't think I'm the
original person to do it.
```

I've found that the same sequence of transpositions is in the earliest published algorithms. That was 1961 or 1962. However they were not computed with the same technique.

I would not have suspected that they were the same if there had not been "bell-ringing" in the British Computer Journal article.

"*Generation of permutation sequences: Part 2*", *R. J. Ord-Smith, British Computer Journal, vol 14 num 2 May 1971*

> [ACM Algorithm 115A] retains a combinatorial interest because it generates a bell-ringing sequence. This algorithm is also noteworthy for the subtle manner of its skilful organization. In general array access dominates in an Algol procedure of this kind. Notwithstanding the extra array accesses caused by the use of both the signature vector d and an auxiliary vector p the elegant control exercised in the use of these is worthy of study by anyone who wishes to be known as a programmer.

This is the simplest of the six algorithms shown in the article. Written in Algol 60 using GOTO, it makes C look good.

```
PROCEDURE perm(x, n); VALUE n; INTEGER n; ARRAY x;
BEGIN OWN INTEGER ARRAY p,d[2:10]; INTEGER k, q; REAL t;
       IF first THEN BEGIN FOR k := 2 STEP 1 UNTIL n DO
       BEGIN p[k] := 0; d[k] := 1 END;
       first := FALSE
       END;
       k :=  0;
index: p[n := q := p[n] + d[n];
       IF q = n THEN BEGIN d[n] := -1;
                     GOTO loop
                     END;
       IF q <> 0 THEN GOTO transpose;
       d[n] := 1; k := k + 1;
loop: IF n > 2 THEN BEGIN n := n - 1;
                     GOTO index
                     END;
       q := 1; first := TRUE;
transpose: q := q + k; k := q + 1; t := x[q];
           x[q] := x[k]; x[k] := t
END of procedure perm;
```

A Forth Version
I suspected that the sequences were the same, but I had to compile and run the procedure to be sure. It was easy to convert to Forth provided a GOTO was available.

```
PERM                    ( x n -- )
```

Given array *x* of *n* elements, modify *x* for a permutation of the elements. Before using, FIRST should be TRUE. After the last permutation, FIRST will be set back to TRUE. The user displays the array.

This implementation uses a GOTO extension to Forth from the Neil Bawd web site at http://home.earthlink.net/~neilbawd/goto.html

Program Text 1

```
INCLUDE GOTO  \  The GOTO tool from Neil Bawd web site.

ONLY FORTH  ALSO LABELS

    : EXCHANGE  ( addr addr -- )  \ See SWAP@ in Bons Mots Nov 97
        DUP @ >R  OVER @ SWAP !  R> SWAP ! ;
    : 'th     ( n "addr" -- &addr[n] ) \ See TH in Bons Mots Aug 97
        S" CELLS " EVALUATE
        BL WORD COUNT EVALUATE
        S" + " EVALUATE
        ; IMMEDIATE

VARIABLE FIRST                 \  Flag for Initialization
FIRST ON                       \  See ON in Bons Mots Aug 97

CREATE P  10 1+ CELLS ALLOT    \  Auxiliary Vector
CREATE D  10 1+ CELLS ALLOT    \  Signature Vector

: PERM              ( x n -- )
    FIRST @ IF
        DUP 1+ 2 DO
            0 I 'th P !  1 I 'th D !
        LOOP
        FIRST OFF
    THEN
    0 SWAP                     ( x k n)
LABEL index
    DUP 'th P >R  DUP 'th D @  R@ @  +  ( x k n q)
    DUP R> !
    2DUP = IF
        'th D  -1 SWAP !       ( x k n)
        GOTO loop
    THEN                       ( x k n q)
    DUP IFGOTO transpose
    DROP                       ( x k n)
    1 OVER 'th D !
    >R 1+ R>
LABEL loop
    DUP 2 > IF
        1-
        GOTO index
    THEN
    FIRST ON
    1                          ( x k n q)
LABEL transpose
    NIP                        ( x k q)
    +
    CELLS +                    ( &x[q])
    DUP CELL+ EXCHANGE         ( )
    ;

ONLY FORTH
```

An ANS Forth Version

The Standard Forth revision below uses variable `Sequence-Number` instead of `FIRST` and auxiliary vectors. The maximum array size has been increased to 12.

`PERM` `( x n -- )`

Given array *x* of *n* elements, modify *x* for a permutation of the elements. Before using, `Sequence-Number` should be set to 0. `Sequence-Number` will be set back to 0 after the last permutation. The user displays the array.

Program Text 2

```
    : EXCHANGE  ( addr addr -- )
        DUP @ >R  OVER @ SWAP !  R> SWAP ! ;

VARIABLE Sequence-Number  0 Sequence-Number !

: PERM              ( array num -- )
    1 Sequence-Number +!
    Sequence-Number @ 2 ROT ?DO  ( array num)
        I /MOD                   ( array rem quot)
        OVER IF
            1 AND IF  I SWAP -  THEN  ( array num)
            CELLS +  DUP 1 CELLS -  EXCHANGE  ( )
            UNLOOP EXIT
        THEN                     ( array 0 quot)
        NIP                      ( array num)
        DUP 1 AND 0= IF  >R CELL+ R>  THEN
    -1 +LOOP
    CELLS +  DUP 1 CELLS -  EXCHANGE  ( )
    0 Sequence-Number ! ;
```

Testing

The following code is for the ANS Forth version but can be modified to suit the `GOTO` version.

Program Text 3

```
: IDUMP          ( addr length -- )
    BOUNDS ?DO  I @ .  1 CELLS +LOOP ;

CREATE X  12 CELLS ALLOT

MARKER ONCE
: INIT          ( -- )
```

```
    X  12 1+  0 DO    ( addr)
        I 1+  OVER !  CELL+
    LOOP DROP ;
INIT ONCE

: PERMUTATIONS      ( array n -- )
    0 Sequence-Number !
    BEGIN
        Sequence-Number @ OVER MOD 0= IF  CR  THEN
        2DUP CELLS IDUMP  2 SPACES
        2DUP PERM
    Sequence-Number @ 0= UNTIL
    2DROP ;

CR  X 3 PERMUTATIONS  CR  X 4 PERMUTATIONS  CR  X 5 PERMUTATIONS

1 2 3   2 1 3   2 3 1
3 2 1   3 1 2   1 3 2

1 2 3 4   2 1 3 4   2 3 1 4   2 3 4 1
3 2 4 1   3 2 1 4   3 1 2 4   1 3 2 4
1 3 4 2   3 1 4 2   3 4 1 2   3 4 2 1
4 3 2 1   4 3 1 2   4 1 3 2   1 4 3 2
1 4 2 3   4 1 2 3   4 2 1 3   4 2 3 1
2 4 3 1   2 4 1 3   2 1 4 3   1 2 4 3

1 2 3 4 5   2 1 3 4 5   2 3 1 4 5   2 3 4 1 5   2 3 4 5 1
3 2 4 5 1   3 2 4 1 5   3 2 1 4 5   3 1 2 4 5   1 3 2 4 5
1 3 4 2 5   3 1 4 2 5   3 4 1 2 5   3 4 2 1 5   3 4 2 5 1
3 4 5 2 1   3 4 5 1 2   3 4 1 5 2   3 1 4 5 2   1 3 4 5 2
1 4 3 5 2   4 1 3 5 2   4 3 1 5 2   4 3 5 1 2   4 3 5 2 1
4 3 2 5 1   4 3 2 1 5   4 3 1 2 5   4 1 3 2 5   1 4 3 2 5
1 4 2 3 5   4 1 2 3 5   4 2 1 3 5   4 2 3 1 5   4 2 3 5 1
2 4 3 5 1   2 4 3 1 5   2 4 1 3 5   2 1 4 3 5   1 2 4 3 5
1 2 4 5 3   2 1 4 5 3   2 4 1 5 3   2 4 5 1 3   2 4 5 3 1
4 2 5 3 1   4 2 5 1 3   4 2 1 5 3   4 1 2 5 3   1 4 2 5 3
1 4 5 2 3   4 1 5 2 3   4 5 1 2 3   4 5 2 1 3   4 5 2 3 1
4 5 3 2 1   4 5 3 1 2   4 5 1 3 2   4 1 5 3 2   1 4 5 3 2
1 5 4 3 2   5 1 4 3 2   5 4 1 3 2   5 4 3 1 2   5 4 3 2 1
5 4 2 3 1   5 4 2 1 3   5 4 1 2 3   5 1 4 2 3   1 5 4 2 3
1 5 2 4 3   5 1 2 4 3   5 2 1 4 3   5 2 4 1 3   5 2 4 3 1
2 5 4 3 1   2 5 4 1 3   2 5 1 4 3   2 1 5 4 3   1 2 5 4 3
1 2 5 3 4   2 1 5 3 4   2 5 1 3 4   2 5 3 1 4   2 5 3 4 1
5 2 3 4 1   5 2 3 1 4   5 2 1 3 4   5 1 2 3 4   1 5 2 3 4
1 5 3 2 4   5 1 3 2 4   5 3 1 2 4   5 3 2 1 4   5 3 2 4 1
5 3 4 2 1   5 3 4 1 2   5 3 1 4 2   5 1 3 4 2   1 5 3 4 2
1 3 5 4 2   3 1 5 4 2   3 5 1 4 2   3 5 4 1 2   3 5 4 2 1
3 5 2 4 1   3 5 2 1 4   3 5 1 2 4   3 1 5 2 4   1 3 5 2 4
1 3 2 5 4   3 1 2 5 4   3 2 1 5 4   3 2 5 1 4   3 2 5 4 1
2 3 5 4 1   2 3 5 1 4   2 3 1 5 4   2 1 3 5 4   1 2 3 5 4
```

Simple Forth Permutations

Chris Jakeman

Cutting small pieces from a larger piece in an efficient or optimal way is an activity well-suited to computing. As part of a study, I needed a way to generate all the possible permutations. I didn't get very far alone, so I asked for help on the newsgroup and was delighted to receive informed responses and quality code from Marcel Hendrix and Wil Baden (see another article in this issue).

It is embarrassing to admit that one source I did not consult was the Forthwrite Index. Perhaps I thought I knew it thoroughly, but when eventually I did examine it, there were 3 entries – Gordon Charlton in Feb 90 and Ed Hersom in Oct 91 and Apr 92.

I believe Ed's contribution in Oct 91 is classic. Stacks are a natural way to manage permutations and Ed's item is very short, uses both stacks effectively and does exactly what I wanted. Thanks, Ed.

Some points to note:

The usual Forth style is for a word to consume its arguments but this is not appropriate for recursive words, which leave similar arguments on the stack.

Many Forth people have criticised the use of `ROLL` as it commonly indicates poor factoring. This definition, however, is sufficient justification for the retention of `ROLL` all on its own.

As Ed pointed out, the number of permutations of `n` is `n!` (factorial `n`) so the number of permutations rises dramatically as `n` increases. As you can see from the table below, there is no need to worry about overflowing the Return Stack with excessive recursion. Indeed, permuting more than 8 items is likely to result in more answers than you might want to deal with.

n	1	2	3	4	5	6	7	8	9
n!	1	2	6	24	120	720	5040	40320	362880

Here is Ed’s solution, edited to suit.

```
: Perm ( <items to be permuted> #items -- <items to be permuted> #items )
  DUP 1 = IF
    CR >R             \ Problem can't get any smaller.
    .S R>             \ Solution is on stack.
  ELSE
    DUP 0 DO
      >R              \ Save the count.
      R@ 1- RECURSE \ Call Perm with reduced count to solve a smaller
problem
      ROLL            \ Pull from bottom of stack and repeat.
      R>
    LOOP
  THEN ;
```

and

```
11 22 33 3 Perm
```

prints the following 6 permutations

```
11 22 33
11 33 22
22 33 11
22 11 33
33 11 22
33 22 11  ok
```

euroFORTH 2000

This annual event is now being organised by FIG UK member Malcolm Bugler of Malvatronics Ltd. The following announcement was made to comp.lang.forth. Please note that the date and venue for this annual event have been changed (again).

From: Malcolm [MalcolmB@compuserve.com]
Subject: euroForth 2000 - It's really happening!

Dear all,

Well, euroForth 2000 is definitely ON!

This year an intrepid group of Forthers will meet over the weekend of 17th to 19th November 2000 in the superb Cheshire countryside at the elegant and stylish De Vere Mottram Hall Hotel in Prestbury, Cheshire, UK (01625 828135)

The conference will feature the normal well tried and tested format and run from Friday lunch to Sunday lunch, with an optional 4th day.

Details of conference rates, a booking form and more details on the location and facilities are also on the conference web site:-

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

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

Papers have been invited on, but not limited to, the following:

- Code Generation and Optimisation
- Message-based, object, and component systems
- GUI interfacing and abstraction
- Embedded Systems Techniques
- Virtual and Silicon Stack Machines

Conference delegates are welcome and encouraged to give papers on subjects related to the conference topics. Papers should be no more than 7 pages and should take between 20 and 25 minutes to deliver.
Workshop topics will be selected by acclamation at the conference, and we suggest that the following topics be considered:

- International Standards
- Cross Compiler Standards
- OOF Syntax Commonality

Conference chair:-

Malcolm Bugler,
c/o Malvatronics Ltd,
6 Drury Lane, Knutsford,
Cheshire, UK, WA16 6HA
tel: +44 1565-751400
fax: +44 1565-751440
malcolmb@malvatronics.co.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

See you all there!

Malcolm Bugler

Conference and accommodation prices

Resident Delegate (single room)	**£350**
Conference fee, 2 nights, 3 meals a day	
Resident Delegate (sharing)	**£290**
Conference fee, 2 nights, 3 meals a day	
Non-resident Delegate	**£210**
Conference fee, lunch and dinner	
Resident Guest (sharing with delegate)	**£240**
2 nights, 3 meals a day	

Chris Jakeman
cjakeman@bigfoot.com

Did you Know?
– Open Firmware

*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).*

Most of us have heard of Open Firmware – a use of Forth adopted by Sun, Motorola, IBM and Apple years ago – but could we explain it convincingly. Elizabeth Rather did just that recently on comp.lang.forth.

"Open Firmware was originally developed at Sun, and has been a feature of all Sun workstations since 1986. In 1994 it became a standard, IEEE1275, and was adopted as the "plug-and-play" architecture by the Power PC consortium (Apple, IBM, Motorola). It is used on all Macs built since 1995 (possibly earlier), and is quite common in non-Intel PCI-bus systems (e.g. SPARC, PowerPC).

Basically (I just taught some courses in this), there's a native Forth in PROM on the motherboard, which launches at power-up. It knows about some local devices, and then queries all possible bus addresses. If there's a peripheral at a particular address supported by an Open Firmware driver, it will have a recognizable code. The motherboard's Forth will then "suck" the driver from the PROM on the peripheral and compile it into executable form. When this is done for all devices, the program then knows the complete configuration of the system, and can boot the OS (e.g. Solaris, MacOS) and configure it. The whole operation takes a second or two, between power-up and launch of the OS.

What's on a peripheral is called "FCode". It's a byte-code representation of Forth source, and hence far more compact than ASCII source would be, even in compressed form. As source, it's also totally portable; the same card will work in a SPARC workstation or a PPC-based box. But once it's compiled by the motherboard into real executable code, it's fast (which it needs to be to be a usable device driver).

The real magic is that you can interrupt the process, preventing the OS from booting, at which time you're in Forth. You can use Forth's interactivity to diagnose hardware or other problems. The reason Open Firmware succeeded at Sun was they realized they could bring up new systems and peripherals much faster using it this way, not to mention facilitating diagnosis & repair of broken systems."

Elizabeth D. Rather FORTH Inc.

Chris Jakeman
01733 753489
cjakeman@bigfoot.com

AGM Report

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

Changes to Committee

Keith Matthews has retired from Treasurer after many years (see the Chairman's letter in this issue). Having reliable financial management is vital to the well-being of voluntary organisations like FIG UK and we are delighted that **Neville Joseph** has agreed to take over. Neville already does this sort of work for other organisations and has a good idea of what he's taking on.

Jeremy Fowell has become well-known for his work on F11-UK and he has now joined the Committee as an Ordinary Member, sort of Minister Without Portfolio.

Review of Last Year

The **F11-UK** was the big event of the past 12 months and we expect this to launch a number of interesting projects. The **Web-Forth** project has made little progress but we hope it will revive.

Two companies have taken up Corporate Membership during this period, **MPE Ltd.** and **Triangle Digital Services**.

We have several initiatives which other FIGs have yet to imitate. **IRC** is going well, with a good mix of regulars and visitors, mostly Forthers from overseas. The publication of **Forthwrite on the web-site** seems to be successful, with more downloads than we expected and no membership losses apparent. We have also started placing Forthwrite into several **universities** which may bring results in the long-term.

The finances, as reported in the last issue, are now in balance and look healthy.

Sylvia continues to manage our **Library**, which contains all the recent books and a complete set of Forthwrite and Forth Dimensions. As books like Thinking Forth go out of print, this resource is increasing in value to our international audience.

Most importantly, membership continues at or slightly above last year's level. Sadly, this year marks the disappearance of **International FIG** from the scene. One or two ex-FIG members have switched to FIG UK and we are beginning to publish some material that would have gone to Forth Dimensions.

During the year ahead, we hope to attract a few more of the active ex-members which will bring in some new ideas and opinions.

Plans for Next Year

There are several ideas for the **Web Site**, including hosting projects based on F11-UK, and we want to make it easier for visitors to place themselves on a mailing list for events etc., so that we can encourage them to become full members.

The **Forthwrite team** continues as before publishing all your valuable contributions. We strive to improve the magazine step by step – this issue is no exception.

Mr K. Matthews
20 Spindlebury
Cullompton
EX15 1SY

5A Riverfield Road
Staines, TW18 2EE
1st November 2000

Dear Keith,

Following your recent retirement as Treasurer, I want to express my sincere thanks for all the work you have done to help FIG UK over the years.

When you took over the job in 1987 we were in some trouble, as the previous treasurer had not been well, and the financial side of things was in a state of chaos. I know you inherited all the records in a large plastic bag and amongst the contents were a large number of unbanked and out of date cheques! Naturally, you took steps to get the old cheques replaced and as far as I remember the vast majority of members were happy to oblige. From that point on, you did everything a good treasurer could do and we always knew exactly what money was available.

You instituted a valuable system of accounting which to this day would allow us to refund a proportion of members subscriptions in the event of an inability to produce all the Forthwrites that have been paid for.

Like you, I have been involved with other organisations and I am very well aware of the vital, if often unseen, duties that a treasurer undertakes to ensure that everyone else can concentrate on their main objective. It is always a great comfort for any group to know that the financial side of things is in competent and wise hands.

I am sure all the members of FIG would like to join me in thanking you for the many years of hard work that you have so cheerfully given us.

Very best wishes for the future,

Chris Hainsworth,

Chairman, FIG UK

Forth in the UK

Chris Jakeman

FIG UK has been invited to submit an article to SIGPLan Notices, a regular publication of ACM, the US Association for Computing Machinery (*http://info.acm.org*). SIGPLan stands for Special Interest Group for Programming Languages.

The article is expected to appear in the December issue of SIGPlan.

FIG UK is the UK chapter of the Forth Interest Group, a not-for-profit organisation promoting the Forth computer language originally developed by Charles Moore for his personal use in 1968. This article reports the current status of Forth, its use in the UK and the role of FIG UK.

Forth – What is it?
For every PC on the planet there are 60-70 processors working in embedded systems to control videos, central heating or windscreen wipers. For example, when Federal Express drivers deliver a package, you sign for it not on paper but on a custom PDA. This is a package-tracking device with bar-code scanner, keyboard, display, comms. ports and about 2 Mb of databases for packages, postal codes, etc.. FedEx have been using this since 1986, enhancing it over several generations of processor[1]. The programming for this product is an example of Forth in action.

Another example of the good match between Forth and controller hardware is the ease with which Forth can be ported to a brand new controller board. Forth runs on over 100 processors (including radiation-hardened ones used in space missions) and prices to complete a commercial port are just £500[2] ($740).

Forth takes programming to extremes, dispensing with most of the baggage deemed essential in other languages. The syntax is so simple it can be described in one short sentence, “Forth words are separated by spaces and executed from left to right.” Forth practitioners agree that the overriding concern is to “keep it simple”.

[1] History reported by E.Rather, MD of Forth Inc.

[2] Typical quotation reported by S.Pelc, MD of MPE (July 2000)

This ruthless approach brings interesting benefits which are greatest in the area for which Forth was developed – embedded systems. The key advantages are: factoring, interactive development and Forth's extensible compiler.

Factoring
For example, to convert from packed binary-coded decimal to hexadecimal you might write a macro in ANS C:

```
#define BCDtoHEX(b) ((b)-6*((b)>>4)) /*  Packed BCD byte to char */
```

The equivalent in ANS Forth would be a word:

```
: BCD>Hex ( nibble-nibble -- byte )  DUP  4 RSHIFT  6 * - ;
```

where `( nibble-nibble -- byte )` is a comment indicating the changes to the topmost values on Forth's Data Stack and `4 RSHIFT` is equivalent to `>>4`. Note that Forth needs a minimum of variables like `b`, preferring instead to manipulate the values on the Stack with words like `DUP which` duplicates the top value.

The advantages from factoring will now become clear. `BCD>Hex` is a first-class word in Forth and can be tested as soon as it has been written with the command `1 BCDtoHEX .` which uses the `.` word to print the result of the conversion. Testing all the values from 0 to 99 can be done in a single line too.

In contrast, to test the C macro `BCDtoHEX`, you must write, compile, link and test a small program or possibly use a debugger to poke values into a register, step through the macro code and check the result. Neither solution is of the same order of convenience as Forth.

Forth makes it so easy to write and test short routines (known as factoring) that it leads to a very different style of programming.

Interactive Development
Embedded systems are usually developed on a PC host using a cross-compiler that delivers a program which is downloaded to the target and tested by running under a debugger. This is a lengthy and intrusive procedure which is quite unnecessary for Forth users.

Instead, because the Forth compiler is incremental, each word can be written on the host and immediately compiled and tested on the target (with no need for debugger or emulator) before proceeding to the next word. In a field where testing may consume more than 50% of the project timescale, this approach (known as "umbilical Forth") brings major benefits.

Extensible Compiler
The Forth compiler is open and uniquely extensible. Programmers are encouraged to provide their own extensions to match the application they are building. The array construct appears in most languages – sometimes the size

of the array is fixed, sometimes dynamic, the first element may be addressed as element no. 1 or no. 0 or the array may be associative. The data may be stored in RAM or on disk and bounds-checking may or may not be available.

In contrast, Forth has no array construct but instead provides simple tools to build whatever type of array is appropriate for your current application – providing a marked degree of freedom for the programmer. A number of high-performance extensions for object-oriented programming have been published, the simplest provides efficient inheritance and polymorphism in just 12 short lines of code[3].

This openness and flexibility has intrigued a generation of programmers.

Landmarks

Forth enjoyed a wave of popularity in the early days of micro-computers among programmers who found that it was much faster than BASIC. It provided some significant firsts in early applications such as the very first word processor on the IBM PC[4] - EasyWriter. Commercial Forths now sport high-performance native code compilers which retain all of Forth's advantages. The compiler currently delivering the highest performance code (using VFX[5] technology) is a match for the very best that C compilers can offer.

Forth is now an international standard, initially ANS Forth (1994) and subsequently accepted by ISO[6].

Open Firmware is another international standard[7] which uses Forth to provide the "plug and play" architecture in all PowerPC computers shipped by Apple[8], Sun[9] and IBM. The real magic is that you can interrupt the boot process, preventing the operating system from booting, at which time you're in Forth. You can use Forth's interactivity to diagnose hardware or other problems. Sun realised they could bring up new systems and peripherals much faster this way.

Modern Forth

Forth has kept pace with other developments in computing. Commercial products make good use of Windows for the development platform and deliver the unique "umbilical Forth" remote development.

---

[3] Bernd Paysan - http://www.jwdt.com/~paysan/mini-oof.html

[4] John Draper – http://webcrunchers.com/crunch/Play/ibmstory/home.html

[5] VFX compiler technology and benchmarks from MPE Ltd. - http://www.mpeltd.demon.co.uk/pfwvfx.htm

[6] ISO/IEC 15145:1977

[7] Open Firmware standard IEEE 1275 since 1994

[8] On Macs since 1995

[9] From the SPARCStation 1 in 1989

Smart-cards benefit more than most from Forth. Working in conjunction with Europay International (Europe's largest financial services company) and Forth Inc., MPE Ltd. designed the Open Terminal Architecture, a token (byte-code) technology supporting portable smart-card applications on Point-of-Sale terminals. It was implemented on a dozen or so different types of terminal, is in increasing use in Europe and has recently become an ISO standard[10].

As a comparison with Java, a very small Forth-based kernel was implemented on an 8051-based smart-card also for Europay. This system benchmarked 10 times faster than rival Java-card prototypes, and offered significantly more functionality[11].

In the Unix world, the GNU free software compiler gcc has made a substantial difference to people writing or using C compilers. The GNU project now boasts the Gforth development system alongside gcc. Gforth is a high-quality collaborative effort conforming to the ISO standard and delivering code of good performance on a variety of platforms.

Forth in the UK

The UK has a number of companies using Forth to serve the embedded systems market. Embedded applications for clients include ISDN routers, TCP/IP stacks, exposure systems for 70% of European newspaper plates[12] and desktop applications include construction software used for planning projects like Hong Kong's Chai Tak airport.

The annual euroFORTH conference is hosted by the UK on alternate years and provides a forum for academics and developers alike. Academic research in this country is currently under way at Bournemouth and Teeside Universities.

FIG UK

At the UK chapter of the Forth Interest Group, we celebrated our 20th anniversary last November and currently support over 100 members with a healthy mix of professionals and amateurs. We draw members from a number of other countries and have run several multi-national projects.

FIG was crucial in promoting Forth by disseminating free source long before the Internet and the Open Source movement. FIG UK has continued to support this activity, nowadays taking advantage of the low-cost and wide reach of the Internet. Whilst the main archive of Forth code resides at the International FIG web site[13], the UK web site[14] has introduced a number of innovations.

---

[10] OTA standard

[11] Results reported by E.Rather, MD of Forth Inc.

[12] Reported by S. Pelc, MD of MPE Ltd.

[13] International FIG at http://www.forth.org

These include a regular issue of national and international Forth News, the first Forth magazine to be published and indexed on-line (Forthwrite ISSN 0265-5195), links to UK training providers and the only regular Forth chat session, held monthly, which attracts callers from Europe and beyond.

A valuable non-Internet resource is the lending library for books, magazines and conference proceedings. FIG UK also endeavours to match members with Forth projects and jobs.

Outlook

What is the outlook for Forth in the UK? Its users are more self-sufficient than users of other languages and don't need a substantial Forth industry to keep up-to-date and continue to reap the benefits. However Forth has largely escaped the attention of academics and it is a shame that so few programmers are aware of the unique Forth approach. Nowadays, the biggest obstacle to using Forth is the shortage of programmers with Forth experience.

The key activity for FIG UK is therefore educational – finding new ways to present Forth so that potential users have an easy way to get to grips with it and judge its benefits for themselves. Two projects already under way address this issue:

1. F11-UK is a new professional quality low-cost controller board programmed in Forth from a PC. The instructions for building and using the kit are carefully designed to suit people with more enthusiasm than experience.
2. WebForth is a Java applet that allows users to experiment with Forth using a browser. Although previously held back by the "flakey" behaviour of applets and browsers, WebForth is embedded in a web site designed to teach, not Forth but programming itself.

Forth is too different to appeal to the mainstream of programmers. However those with an open mind will find Forth's distinctive approach not only fascinating but a remarkably productive tool in the ever-expanding world of embedded computers.

[14] FIG UK web site at http://forth.org.uk

Alan J M Wenham
01932 786440
101745.3615@compuserve.com

Vierte Dimension 3/00

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.

Letters and Communications

Michael Kalus	Michael Kalus, doctor, hobby Forther and a co-founder of the Forth Gesellschaft, looks back and delivers a glowing acknowledgement of Forth and the members of the Gesellschaft. Using Forth as a test instrument he was finally able to understand his computer.

Alan Turing, the Enigma

Joachim Merkel J.Merkel@ Tbx.berlinet.de	Joachim Merkel (J.Merkel@Tbx.berlinet.de) reviews the German version of Andrew Hodge's book concerning Alan Turing and the breaking of the German Enigma code in WW2

Minimodul 2M. A processor card including Forth

Hans Eckes, Karlsfeld	For several years the Patriot firm (www.ptsc.com) have had a Forth processor, the PSC-1000. The author gives a first impression of this and describes a PSC-1000 processor card with the appropriate Forth.

Directorate

Fred Behringer behringe@ mathematik. tu-muenchen.de	Fred Behringer appears as a new director replacing Friederich Prinz (who from now on remains as editor of Vierte Dimension). Fred gives a glimpse into his professional development and explains his intentions with regard to Forth Gesellschaft.

Other groups

Fred Behringer behringe@ mathematik.tu-muenchen.de	Fred's review of Forthwrite 106 and of Figleaf 19 and 20.

The way of stones

Soeren Tiedemann,

Freiburg

The author implements the children's game Ishido (Tsi Tao in China) on the MUP21 in machine Forth.

Forth annual meeting 2000 in Hamburg

Friederich Prinz

Friederich Prinz gives a report on this year's annual meeting of the Forth Gesellschaft in Hamburg. "It was not so well organised as the one last year in Oberammergau but it was nevertheless a success"

Currently from FIG Silicon Valley

Henry Vinerts

Henry Vinerts (translated by Thomas Beierlein) praises the work of Friederich Prinz, the editor of Vierte Dimension. In regard to the present troubles of the American Forth Dimensions, he suggests that a world-wide magazine should be published and that Fred Behringer should be charged with the task of translation of all the submitted articles into all the necessary languages. It is not known whether the suggestion is meant seriously!

Everything is Forth

Joerg Staben,

Hilden

The author further investigates Win32Forth, which its originator Tom Zimmer, originally wished to designate as F95. The complete package, Win32Forth includes the executable file WIN32FOR.EXE Joerg finds that basically Java, Open Boot, and Real3D are essentially Forth. Win32Forth recommends itself to the totally dedicated and involved programmer.

Mathematicians can also bite. A book preview

Fred Behringer

behringe@mathematik.tu-muenchen.de

A preview has appeared on the internet of "Axiomatic Economics Theories" by Victor J. Aguilar (http://www.axiomaticeconomics.com). Fred supplements this with his own personal comments.

Greatest common divisor without division, for ZF and TurboForth in 32-bit mode

Fred Behringer

behringe@ mathematik.tu-muenchen.de

This is Fred's expansion of his article originally published in Forthwrite 106.

A Web server in Forth

Bernd Paysan

Some 200 code words have been defined in order to implement HTTP/1.1 in GForth. This is the original German version of Bernd's paper which appears in Forthwrite 108

Review of the annual gathering 2000 of Forth Gesellschaft

Ewald Rieger

The number of attendees was down. The finances are in good order. The Internet is more strongly in evidence. The international relationships should be extended. Martin Bitter's Forth programmed RCX Lego-robot for school instruction should be detailed on the Web.

Deutsche Forth-Gesellschaft

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

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

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

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

or write to
Forth-Gesellschaft e.V.
Postfach 161204
18025 Rostock
Germany
Tel.: 0381-4007872

Letters

The Magazine Team are always pleased to get feedback and encouragement – see below, about the magazine from John and about the IRC sessions from Jim Lawless, who joined us from the USA for the first time in September.

John Tasgal

From: John Tasgal [john@tcl.prestel.co.uk]
Sent: 23 August 2000 10:53
Subject: Forthwrite

Hello Chris,

Thanks for your efforts with Forthwrite !

Another truly heavyweight edition, which I'm studying with great interest. Bernd's program is remarkable - I hadn't realised that's how a UNIX-based program would implement a web-server.

Best regards,

John

Jim Lawless

From: Jim Lawless [jimbo@radiks.net]
Sent: 03 September 2000 03:12
To: Chris Jakeman (Personal)
Subject: #FIGUK chat

Hi, Chris.

I enjoyed the all-too-brief #FIGUK chat today.

```
Jim Lawless                 | MailSend and MailGrab : Send and
jimbo@radiks.net            | receive Internet e-mail from batch
http://www.radiks.net/jimbo | programs or compiled EXE's.
```

FIG UK is always ready to help Forth users – we were able to help Klaus who is a member of Forth Gesellschaft.

Klaus Zobawa

Sent: 26 July 2000 21:15
To: Klaus Zobawa
Subject: RE: State machines

Hi Klaus,

> Sent: 01 January 1601 00:00
>
> my name is Klaus Zobawa and I am a member of the german FORTH Group.
> Mr Fred Behringer told me, that in one of the past Forthwrite Issue it is a
> Article about state machines. I have implemented a FORTH program to
> convert a graphic STM table in IF / ELSE structures.
> Can you mail me the article about State machines?
>
> Thank you.

The index on our web site lists the following:

```
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
```

Electronic versions of the first 3 have been lost in a computer crash, but I can scan them in and e-mail them to you.
I can also provide contact addresses for the authors if that would be useful.
Our authors are usually delighted to receive some interest in their efforts.

```
                                   ____/  / __   /   /   /   /   /
Bye for now                       /      /  /  _/   /   /   /  /
                                 __/    /  /__     /   /   /_/
Chris Jakeman                   /      /  /   /   /   /   /  \
                            __/     __/ ____/    ___/ __/    _\

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

Norman is one of our new on-line readers. Our web-site list of "Free Forths to Try" is not intended to include every freely published Forth (that's a job for the FAQ - Systems, which gives it only a dismissive mention) but to recommend a few mainstream Forths.

We passed on Norman's comments to the newsgroup where he got feedback, some from FIG UK members.

Norman Ngige

From: norman [norman@mciworld.com]
Sent: 22 August 2000 04:06
To: cjakeman@bigfoot.com
Subject: BBL/Abundance

Hi,

My name is Norman Ngige and I currently live in Dallas Texas. I have a question for you.

ftp://ftp.taygeta.com/pub/Forth/Archive/ibm/bbl/
http://mindprod.com/products.html#ABUNDANCE (Latest Version..)

The above are links to a forth know as BBL and a database written in that Forth known as ABUNDANCE. It was featured in BYTE magazine in 1986 and you even had an article on it in your magazine 'Forthwrite' at one time. (Aug '90 – Ed.) It's a very fast 32-bit forth with a 1Mbyte address space that is extremely well documented, and available without cost. Full permission to use it as you please is granted by the author. I understand that NASA and JPL even used parts of it for some of their missions.

What I find perplexing is that it is virtually unknown in the Forth community though I see regular mention of other DOS-based Forths like F-PC, eForth, hForth, and Pygmy. Now I'm relatively new to forth so there is a lot I don't know about Forth in particular and the Forth community in general, but it seems to me, that BBL is as fast and as powerful as the best of them, yet is doesn't appear on your list of 'Free Forth to Try', Could you shed some light on this for me?

Thanks,

Norman Ngige

Ralph Hempel has pioneered the use of Forth for programming the Lego Mindstorms system. We have reported books that include the use of pbForth, but it is good to hear that another book dedicated to Forth is ready for publication.

It also encouraging to hear that Forth was well received at Summer Camp - just the sort of exposure we need for the Forthers of the future.

Ralph Hempel

From: Ralph Hempel [rhempel@bmts.com]
To: cjakeman@bigfoot.com
Subject: RE: Forthwrite in PDF format

> Hi Ralph,

> I've been watching the development of pbForth with interest. This is
> exactly the sort of approach we need to show how easy it is to learn
> programming using Forth.

I did two kid's camps at Purdue University this summer, for grades 5,6,7 and they loved it. The parents were very impressed that their children were able to really program a robot!

The book should be available in October, I'll let you know exactly when.

> We would be delighted to report that your book is available, buy it for the
> FIG UK library and review it thereafter. I'm sure can find a FIG UK member
> who is using LEGO Mindstorms. The people most interested in robots log in
> to our monthly IRC session so I'll ask then. (You would be welcome to join
> us).

> As to an article - yes please. I can think of lots of questions to ask. We
> can accept almost any format, but ASCII is as good as any.

I don't think I can get anything together for you this month, but in my copious spare time :-) I'll see what I can do. The real story of pbForth is how easy it was to port from Dr. Koh's sources.....

I'll keep in touch, and I have released a new version of pbForth on my site.

Another thing I'm working on is a ForthPort in Tcl. The basic idea is that we can send canonical source code to the Tcl script, and it pumps out assembler source ready to build...

Cheers,

Ralph Hempel - P.Eng

Check out pbFORTH for LEGO Mindstorms at:
http://www.hempeldesigngroup.com/lego/pbFORTH

FIG UK Committee

Membership enquiries, renewals and changes of address to Doug.
Technical enquiries and anything for publication to Chris J..
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 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.