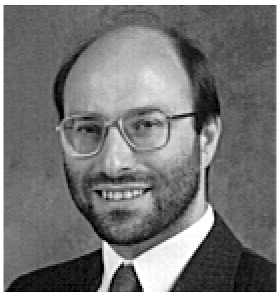
ISS 0265-5195
April
2002
Issue 116

Forthwrite

news events people reviews projects programming

Special Features of kForth

April
2002
Issue 116



Editorial

I am sad to report the recent death of Keith Matthews after a short illness. Keith served as our Treasurer for many years and was a great guy to work with.

Welcome to new members R .Wilks from Exeter and Krishna Myneni from Hunstville USA – see the article on his kForth in this issue.

Howerd's report on November's euroForth is very encouraging and I, for one, am planning a paper for euroForth 2002.

As usual, we have news from Germany and from Silicon Valley.

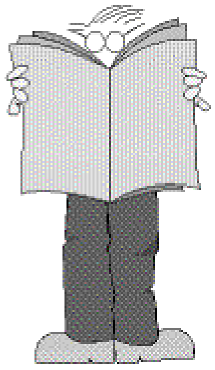
Our web site has been attracting a remarkable number of downloads – see article. There's a lot more interest in Forth than you might imagine.

Finally, congratulations to our Award Winners – see inside for details.

PS. Don't forget the monthly IRC session. Our next one is Saturday 4th May on the IRC server called "IRCNet", channel #FIGUK from 9:00pm.

Until next time, keep on Forthing,

[illegible]

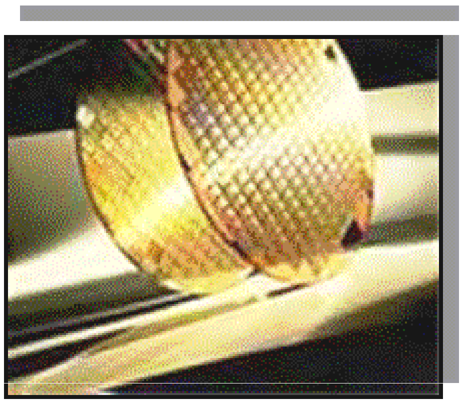


Forth News

Commerical Systems

IGNITE

PTSC is now allowing developers to download the fully functional IGNITE processor core and a complete development tools suite for evaluation and pre-production development. Ignite is a very fast stack processor based on the shBoom design by Chuck Moore



which runs many Forth primitives directly.

See http://www.ptsc.com/

"PTSC's 32-bit IGNITE™ processor cores outperform every rival RISC processor core available today—bar none."

UltraTechnology

Jeff Fox reports that his site at http://www.ultratechnology.com has accumulated over one million hits and a total of 162 GB was downloaded last month.

Forth Editor

ED for Windows (ED4W) does syntax highlighting, colouring, etc., for Forth words. See http://www.getsoft.com

Non-commercial Systems

Mac OS X Forth

Daniel Engeler has released a new native FORTH for Mac OS X under the Gnu Public Licence.

It's a high speed, small CLI application featuring stack caching, seamless integration of assembler and built-in help system. The author would appreciate feedback. See

http://www.ee.ethz.ch/~danengel/d/

Win32Forth

John Peters has set up a mail list for Win32Forth. You can join the list by sending a BLANK email to:

Win32Forth-subscribe@topica.com

and learn more about it at:
http://www.topica.com/lists/Win32Forth

PFE

The latest version of PFE (32.7x) provides a choice between indirect threading and subroutine calls. This is only available for the i386 processor family so far. A total of 4 threading models are supported, reducing the time for the reported benchmark to 55% of the time for indirect threading.

Forth Resources

ANS Technical Committee

After 5 years, ANSI requires the Technical Committee to revise the published standard. E.Rather reports that “It became clear that the members didn't have enough time to do the extensive work necessary to complete a revision and take it through the necessary public reviews, so we ratified Forth94, and it will be due for re-evaluation in 2005.

However, a new TC will have to be organized and qualified. ANSI raised the annual dues from $200 to $800 (without notice) and most members of the TC resigned in protest, so the TC was officially disbanded for having too few members.”

String Packages

Two references provided on comp.lang.forth are:

String-manipulating library from SP-Forth (spf.sourceforge.net) is available from CVS or here:

http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/spf/devel/~ac/lib/STR2.F

The basic string words in kForth can be found at: ftp://ccreweb.org/software/kforth/examples/strings.4th

krishnamyneni@compuserve.com

Special Features of kForth

Krishna Myneni and David P. Wallace

kForth was originally written, as many Forth's are, to provide user-programming facilities within another application. However kForth includes two unusual features which are reported in this 2-part paper, which was prepared for JFAR, the Journal of Forth Applications and Research.
Krishna Myeni is a member of FIG UK living in the USA.

We discuss two special and non-standard features of the kForth interpreter, a program based largely[1] on ANS Forth. Firstly we demonstrate how a limited form of data typing and type checking can catch a significant set of Forth programming errors, with almost no modification to standard Forth code. Secondly, we discuss the benefits and restrictions imposed by using a dynamically growable dictionary. The two new features of our Forth system require the addition of two new words: `A@` and `?ALLOT`.

Introduction

kForth[2] is a compact interpreter, largely based on ANS Forth[3]. However, several new features, which are not a part of the ANS standard, have been introduced in kForth. In this paper we discuss two of these non-standard features:

- Data typing and type checking
- Dynamic dictionary

A third non-standard feature, called deferred execution, will not be discussed here since it does not affect the structure of Forth code. We give the motivation for including these features in kForth, particularly as it relates to our original goal of writing an interpreter to be embedded into other applications. A parallel

[1] It lacks the standard `HERE C,` and `,` in order to support a dynamic data space, as explained in the next issue.

[2] K. Myneni, kForth User's Guide, (http://ccreweb.org, 2001). kForth and XYPLOT are released under the GNU General Public License. Source code, executables, and documentation for both of these programs may be obtained from http://ccreweb.org. Over 40 sample programs are available for download. kForth runs on x86 compatible processors operating under Linux or the various Win32 systems (Windows 95/98/NT/2000).

[3] ANSI X3.215 -1994, American National Standards for Information Systems - Programming Languages - Forth, (American National Standards Institute, New York, NY 1994).

goal was to allow code written for kForth to be easily ported to other ANS Forth systems — kForth may be viewed as a subset of ANS Forth. The two new words, `A@` and `?ALLOT`, introduced in kForth for supporting the new features described in this paper, have simple ANS Forth-compatible definitions. We describe how the two new features are implemented and illustrate their use with examples.

Brief History of kForth

kForth[4] has its origin as an embedded interpreter for the application XYPLOT, a plotting and data manipulation utility. One useful feature of XYPLOT is its expression evaluator, which parses simple algebraic expressions and applies the operations to an entire data set. For example, the expression y*2 multiplies all of the y values of an (x,y) data set by 2.

In its early stages of development, the expression evaluator consisted of a parser which broke down the expression into a vector of "op-codes", and an execution loop which performed the sequence of operations. A data stack held the intermediate values of the calculation. Thus, the beginnings of a stack based interpreter was written and incorporated into XYPLOT. Subsequently the expression evaluator was developed into a fully-featured interpreter that allowed the main application to be extended with modules written in Forth source code.

Forth was chosen as the language for the interpreter rather than developing a custom application language for several reasons:

- It is relatively easy to interpret stack-based code.
- One of the authors had several years of experience with Forth programming.
- Forth provides a wide range of functionality, from low level bitwise operations to high level floating point operations.
- Forth provides a foundation for constructing an application-specific language.
- The use of an established language such as Forth reduces the need to write extensive documentation, which would be required for any custom language.

In addition to its use as an embedded interpreter, kForth also functions as a standalone Forth computing environment. At present, kForth features a vocabulary of over 240 words, with 116 core words, 26 core extension words, and more than 60 words from the option sets for ANS Forth.

Data Typing and Type Checking

The rationale for data typing in kForth is to provide error checking. Unlike the statically-typed Forth system, strongForth[5], which defines a hierarchy of data

[4] Not to be confused with the earlier kForth by Guy Kelly, basis of the VPPlanner spreadsheet.

[5] S. Becher, Introduction to strongForth, (http://home.t-online.de/home/s.becher/forth/, 2000).

types, the implementation of kForth is limited to just two data types: `ADDR` for addresses and `IVAL` for all other data. Words listed in Table 1 verify that the address operand on the data stack (or return stack) has the proper type, i.e. type `ADDR`, to avoid processor exceptions caused by an invalid memory access. Although a signal handler might be used to catch such exceptions, if the interpreter is to be embedded into another application, this method would preclude the main application from having its own handler for such exceptions.

It is important to note that type checking in kForth is performed at run-time by associating data types with elements on the stack. This form of type checking is known as dynamic type checking[6]. Compilers for traditional languages such as C perform type checking at compile-time, a method known as static type checking.

Although implementation of static type checking in Forth has been discussed previously[7], it requires augmentation of the language itself to provide a means of specifying argument types to a word - the strongForth language provides a clever way to do this using the common stack diagram comment. In kForth, however, our motivation is not to implement strict data typing, but to use data typing to catch typical run-time errors, with virtually no modifications to the Forth language.

```
C@ C! W@ W! @ !
A@ 2@ 2! SF@ SF! DF@
DF! +! FILL ERASE COUNT TYPE
CMOVE CMOVE> LOOP +LOOP FIND '
ACCEPT OPEN READ WRITE NUMBER?
SYSTEM
CHDIR
```

Table 1: Words which perform address type checking in kForth

Two kinds of errors are likely to be caught by our limited method of type checking:

- The number of parameters on the data stack or the return stack is incorrect and one of the parameters is an address.
- Parameters on the data stack or the return stack are in the wrong order and one of the parameters is an address.

These are typical conditions created by problem code in Forth. Take the following simple example of code with incorrect ordering:

```
variable v
v 3 !              \ should be  3 v !
```

The following error message is displayed:

[6] A. V. Aho, R. Sethi, and J. D. Ullman, Compilers: Principles, Techniques, and Tools (Addison-Wesley, Reading, MA, 1988).

[7] B. Stoddart and P. J. Knaggs, Type Inference in Stack Based Languages, (Formal Aspects of Computing, 3: 1-000, 1992).

```
VM Error(1): Not data type ADDR
```

An operand of type `ADDR` was expected on top of the stack by `!` and not found. The kForth virtual machine (VM) quitted execution and returned an error code (the error message is actually displayed by the outer interpreter). The state of the stack at the time of the error may be examined by `.S` to diagnose the problem, since the VM performs `QUIT` rather than `ABORT`.
Corruption of the return stack can also be detected by run-time type checking. For example, a common problem is to push an item onto the return stack and forget to pop the item before the word returns. An extreme example is:

```
: BAD 3 >r ;
```

Execution of `BAD` results in

```
VM Error(5): Return stack corrupt.
```

The error is detected by checking the data type of the item on top of the return stack upon return from `BAD`. Since it does not have type `ADDR`, for a valid return address, the VM returns an error indicating corruption of the return stack. The VM executes `ABORT` when the return stack is found to be corrupt.

Now consider a more subtle coding error involving the return stack:

```
: bswap ( n1 n2 n3 n4 -- n2 n1 n3 n4 )
2r> swap 2>r ;          \ should be  2>r swap 2r>
1 2 3 4 bswap
```

Entering the above statements into three untyped Forth systems produced different results. One system displayed an error while the other two responded with `ok`. With the latter, examination of the stack showed that the arguments were unchanged. It should be noted that all of the systems we tried displayed an error if the word `bswap` was used inside of another word. A variant of the above example was also tested:

```
: test 10 0 do i . 2r> swap 2>r loop ;
1 2 3 4 test
```

It is interesting to observe the number of loop iterations executed by the various untyped Forth systems. At least one went into an infinite loop. Worse, the other systems executed this code without complaint, returning with `ok` - the only indication of a problem being the values printed by `i .` the diagnostic print of the loop index.

kForth detects problems with both of these code examples using type checking of the return stack. Upon every iteration, the words `LOOP` and `+LOOP` test for the presence of a branch address on the return stack via type checking. The price of detecting these kinds of problems is the added overhead for maintaining type

information and performing type checking. We measured the impact of type checking in kForth on execution performance and found that it caused about a 15% increase in the time to execute standard benchmarks[8].
By design, kForth implements data typing so that it is almost entirely transparent to the user or programmer. Every cell in the data stack has associated with it one of two types: `IVAL` or `ADDR`. Data types are stored in a separate Type Stack, shown in Figure 1, which is manipulated in parallel with the data stack. kForth does not provide words for direct manipulation of the Type Stack. Instead, intrinsic words which operate on the data stack perform corresponding operations on the Type Stack. Consider the behavior of `ROT`:

```
( n1 n2 n3 -- n2 n3 n1 )
```

If `n1` is of type `ADDR`, and `n2` and `n3` are of type `IVAL`, as shown here in Figure 1, `ROT` also rotates the type stack so that the top element has type `ADDR` after the operation. Other words which affect the data stack also manipulate the type

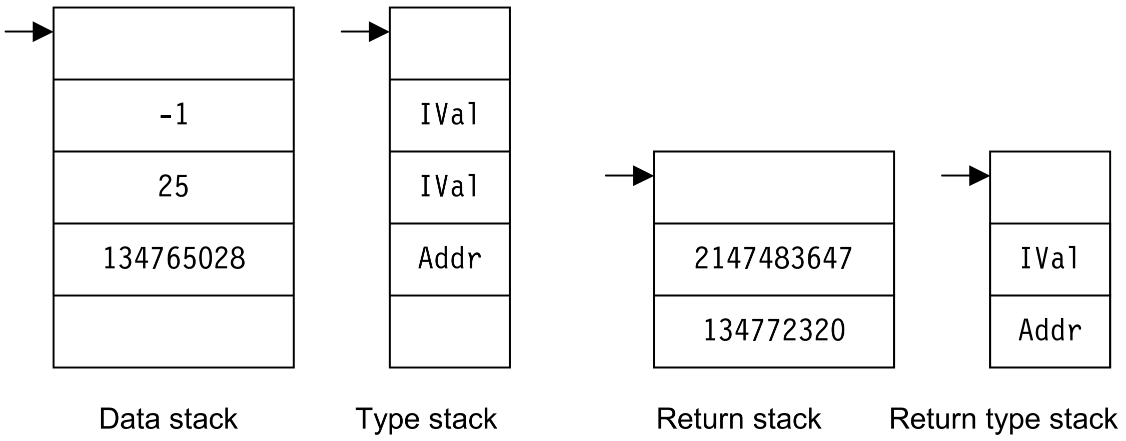


Figure 1: The kForth Stacks
The stack pointers are always aligned for each pair of stacks.

stack in an analogous manner. The behavior of two particular words, with respect to data typing, should be noted: `+` and `-` . Consider the case where an offset must be added to an address using `+` :

```
( a1 n -- a2 ) or ( n a1 -- a2 )
```

We expect that the result of `+` should produce an address if either the first operand or the second operand is an address, and indeed this is the typing rule observed by `+` in kForth. The behavior of `-` is different:

```
( a1 n -- a2 ) and ( n1 a -- n2 )
```

[8] A. Ertl, Performance of Forth Systems, http://www.complang.tuwien.ac.at/forth/performance.html, 1996). Standard benchmark programs used are: bubble-sort, matrix-mult, fib, and sieve.

We may subtract an offset from address a1 to obtain address a2; however, it is not sensible to expect that by subtracting address a1 from integer n1, we will obtain a valid memory address. Therefore the data type of the result depends on the ordering of the data types of the operands for - .

But the programmer need not be aware of these typing rules — for these cases, sensible Forth code produces sensible data typing, enabling subsequent error checking. The typing rules for + and - are implemented in an efficient manner and require little computing overhead in the virtual machine.

The return stack has an associated type stack, called the Return Type Stack, also shown in Figure 1. In transfers from the data stack to the return stack and vice-versa the data types are also transferred between the Type Stack and the Return Type Stack. As with the Type Stack, direct manipulation of the Return Type Stack is not permitted. Intrinsic words that modify the return stack also modify the Return Type Stack.

```
>R R> R@ 2>R 2R> 2R@
DO ?DO LOOP +LOOP
UNLOOP I J
QUIT EXECUTE EXIT
```

Table 2: Words which use the Return Type Stack

In addition to the VM itself, words which explicitly perform type checking using the Return Type Stack are LOOP and +LOOP. The loop index words, I and J, place an item on the data stack with the same type as the starting loop index. It is therefore possible to loop over an address range and use an index word to place an item of type ADDR on the stack. The following example illustrates this point:

```
      create tb1 20 allot
      : byte_sum ( -- n | compute the sum of bytes in table tb1 )
            0 tb1 20 + tb1 do i c@ + loop ;
```

Use of C@ on the index value returned by I is valid since the starting index has type ADDR. The above code is no different from that used in an untyped system, once again demonstrating the transparency of data typing in kForth.

Next, we discuss the only known instance in which the programmer must be aware of data typing in kForth: fetching address values from memory onto the stack. An address is fetched from memory using A@ instead of @ . The word A@ retrieves the same value as @ , but it also sets the data type of the stack cell containing the value to type ADDR. In contrast, @ sets the data type to IVAL. The following example illustrates the use of A@ :

```
      variable current_table
      create tb1 20 cells allot
      tb1 current_table !
      : @n ( n -- m | fetch the n^th element of the current table )
            cells current_table a@ + @ ;
```

The variable current table holds the address of a table, set to tb1 in the example. The address of the table is fetched onto the stack by current table a@

rather than by current table `@` , as in an untyped Forth system. Notice that `!` is used to store an address value to a memory location. Data has associated type information only while it resides on one of the two stacks (the data stack or the return stack). Type information is not retained for data stored at other memory locations. The need to provide a new word, `A@` , in the basic Forth dictionary may seem undesirable; however, it is a relatively small price to pay for the benefits of address type checking, which have been illustrated above. Use of `A@` also makes clear to the reader of the code that an address is being fetched rather than an other kind of data value. This section concludes with the following point:

Porting kForth code to an untyped Forth system requires that `A@` be defined to be synonymous with `@`.

The second half of this article discusses a dynamic version of `ALLOT` and will appear in the next issue of Forthwrite - Ed.

Krishna Myneni is a physicist and self-taught programmer who delights in devising new experiments, piecing them together out of odds and ends, and orchestrating the pieces with software. He is a long-time Forth user and proponent, much to the amusement of his colleagues.

From the 'Net

Here are some items from the newsgroup comp.lang.forth which are worth repeating.

Mike Losh, who collaborated on the FIG UK WebForth project, gave his view on comp.lang.forth recently of possible opportunities for Forth on the web

Due to changes with @Home, my personal webpage now resides at: http://mywebpages.comcast.net/mlosh/

I have only done minor updates for my new email and so on. This site contains my original eForth for Java and a GPL version of WebForth where the FIG UK members contributed a significant amount of code. There is also an HTML copy of my Forth Dimensions article about the jeForth applet for anyone who wants to more background info about it.

Please remember that this project started when Java and the Web were still relatively young (primarily '96 to '98, IIRC). A Forth in a Java applet perhaps has some value as a demonstration/education platform for learning about Forth, but any serious student of Forth is better served with either a native implementation for his or her hardware platform or by an OS-hosted Forth that exploits the operating system services. The Java applet approach that I started and others extended is still too limiting for really practical & useful applications.

If I were to design a Forth today for specifically implementing Web Services (the information technology du jour), I would use a totally different approach. For example, on Unix/Linux, I could imagine a daemon process (written in a native Forth or perhaps C) that listens to a TCP/IP port and executes a modified outer interpreter that understands some of the web service protocols (SOAP, RPC-XML) instead of a teletype-style UI. Through that process, remote users (human or other apps) could invoke Forth words to do whatever that system could do. Perhaps dynamically create new Forth words, which then automatically become new web services. Of course there would be significant security and authentication challenges compared to the traditional Forth outer interpreter.

Compared to my earlier efforts, such a "soapForth" or ".NetForth" might be really useful to dynamically build and tear down temporary web services in flexible real-world applications. It could carve out a niche among all the other .Net and other SOAP languages and products. But I am not working on such a thing. I think this entire web services concept needs a few more spins before it starts to generate more benefits than hype.

Forth has been memorably described as the quickest way to explore new hardware because of its support for interactive use. Ironically, the same reason is given below to explain the success of Forth in identifying and working round bugs in Windows.

John Dunn wrote
Is anyone doing Windows applications in Forth anymore? Not embedded system work or CE work, but real, user interface intensive Windows apps? If so, which Forth?

Both Forth, Inc. SwiftForth and MPE FVX look impressive, but the Windows interface is still daunting and not at all like the elegant simplicity that is Forth itself. iForth looks good too, especially the price and open licence. But as far as I can tell, there is no Windows GUI interface at all.

As a long time Forth programmer, who turned to the dark side some 3 years ago, to Visual Basic, and now (mostly because VB is about to become an orphan) taking another look at Forth for Windows...I would really appreciate any insight anyone here who is actually using Forth in a Windows app can share.

Glenn Dixon responded:
My son Tom and I have done several small Windows apps (with GUI), one medium one, and one huge one that includes fancy features--hardware cards with DSPs, web-deployed control and status screens, distributed web and and LAN EXEs, database, low-level sound card and Winsock interfaces, etc. One GUI includes drag-and-drop icon-based flowchart programming.

We use Swiftforth and have been very satisfied with it. At one point we seriously tried to move the app to Visual Basic, and again to Delphi, but found that after you dip below the visual interface the underlying environment was more difficult to work with than Forth.

This is because when you do a sophisticated Windows app, you spend most of your time programming, not in the language you chose, but in the language of the Windows API. It is a world of incomplete and incorrect documentation, bugs in API calls, poor migration from version to version, and a dozen different interfaces. Forth gives you a window (pun?) into this world and allows you to play with API calls like no other language. We spent A LOT of time doing this.

Jeff Fox responded to an inquiry about "stack machine" processors. I wonder how many readers realise that Chuck's chips are asynchronous (ie unclocked). This is an approach which has promised a great deal but proved impractical for larger processors.

> Has anyone ever tried an unclocked (asynchronous) stack machine design.
> I can see major advantages in this especially if call and data stack
> operations can be overlapped.

The Novix had three hardware busses, main memory, parameter stack, and return stack. This allowed it to execute operations on all three busses in the same clock cycle. But the chip used an external clock so the CPU and memory access was not asynchronous.

Every chip that Chuck Moore has designed since 1990 has been asynchronous. The real-time I/O coprocessors (if any) are externally clocked, the CPU have all been asynchronous and free running.

His latest designs have used dynamic logic in addition to being asynchronous. This lets them run faster, but it also means that if you put something on the stack and wait a while that it will go away. ;-)

Chuck uses a lot of hardware tricks. Some of his library components have fewer gates than the ones other people use. He simply leaves out a number of components that other people need. He has an unusual complementary bit pattern on the bus. He does not use stack pointers (or has not, he could if someone needed it). His designs run every instruction on every clock cycle. He has regular horizontal and vertical structures (very powerful design feature). Every transistor is individually tuned. And his design tools assist in this by having correct thermal modeling, smart layout assistance, extensive simulation with full analogue and thermal modelling of every junction on the chip, design run checks, simulation scripting, and is fast and efficient enough to run the CAD software on a small computer.

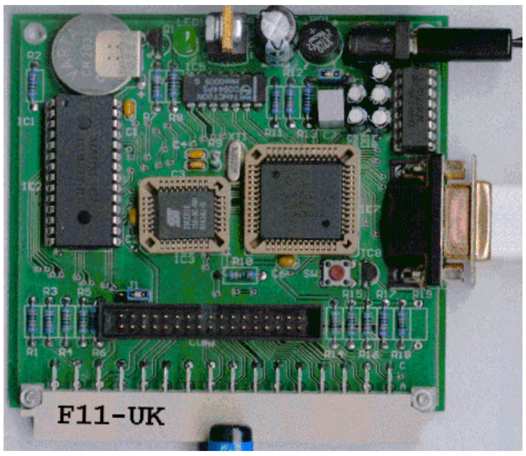
F11-UK

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

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

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

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



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.

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.

Hardware:

Processor:
Motorola HC11 version E1 - 8 MHz (2 MHz E-Clock).
Memory:
32k x 8 FLASH
32k x 8 battery backed SRAM
512 x 8 EEPROM onboard HC11.
I/O:
20 lines plus 2 interrupts (IRQ & XIRQ).
Analogue in:
up to 8 lines using onboard 8-bit A/D.
Serial:
1. RS232, UART onboard HC11
2. Motorola SPI bus onboard HC11.
Expansion:
Via HC11 SPI serial bus using
2 or more of 20 available lines.
Timer system:
Inputs: 3 x 16-bit capture channels
Outputs: 4 x 16-bit compare channels.
PCB size: 103 x 100 mm.

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

Flickwriter Project

Jenny Brien

This is a team project. Although based around the F11UK processor board, you don't need access to a board to make a valuable contribution. There is plenty of opportunity for Forth programming - just contact Jenny.

It's been a while since I wrote anything on this, but it seems some people are still interested, so here's the story so far.

I do some voluntary work with handicapped people (some with Cerebral Palsy - some with hand deformities) and noticed how difficult it was for them to use standard keyboards and mice. A number of current alternatives were also either unsuitable or too expensive.

I'm looking for an input system that could, with practice, match handwriting speed, but would work under the most adverse conditions. Think of doing it in the dark, whilst being driven over a bumpy road, using one hand and wearing a boxing glove. It struck me that a solution to this problem might also have uses in other areas, for example in wearable computers.

The 8-way "pizza key" idea is based on the Quikwriting system for palm computers (see http://www.mrl.nyu.edu/~perlin/demos/quikwriting.html). Later I discovered that the Medical Engineering Resource Unit had the same idea. It seems they have implemented it as a Windows mouse driver, using a PIC to convert joystick output to a PS2 mouse cable. See in particular http://www.meru.org.uk/casestudies/mary.html. I haven't yet seen this software in action.

For the present we're concentrating on a hardware keyboard substitute, sending the necessary key codes down a standard keyboard cable.

There are mouse-substitution and short-cut key programs available, which Chris is investigating, to run on the PC itself. There's still a lot of scope for experiment, though. The joystick gives the equivalent of a 32-key pad (64 if we count the less likely out-and-back combinations) That's not bad, but it still gives a lot of shifting to reproduce all the combinations possible on a standard keyboard.

The basic design principle is to only move/hold one thing at a time. Given that restriction, it should be easy to produce an input device to use the same interface for someone with different disabilities. I think with this restirction it is possible to do both keyboard and mouse substitution with only one extra switch which when you turn on when you want to point to something, and off when you have pointed to it. The next gesture on the joystick tells what you want to do with your selection - click, double-click, drag, shift-control-drag or whatever. It should be possible to speed input by allowing shifts to occur in parallel with input, which suggests the use of two pizza keys.

I wonder about the hardware that controls several PCs with one keyboard, mouse and monitor. What system does that use? Does there have to be a software driver running on each PC? I just have this vision of being able to control any computer on a network from something no bigger than a Palm, which carries all my personal data, prompts and interface preferences.

Forth on a Phone

I note from comp.lang.forth that Chris Double (chris@double.co.nz) is "using Forth on a Nokia 9210 cellphone under the Symbian OS".

Is anyone else programming Forth on a phone?

euroFORTH 2001 – The Report

Howerd Oakford

The November conference extended beyond Europe bringing in visitors from USA including Chuck Moore, inventor of Forth. We are indebted to Howerd who not only presented an interesting paper, but also provided us with this detailed report.

I counted no less than 5 papers by FIG UK members among the presentations. All these papers are downloadable from http://dec.bournemouth.ac.uk/forth/euro/ef01.html and are also available in printed form from our Library.

This was my second EuroForth at Schloss Dagstuhl. "Schloss" is usually translated as "castle", which for many English people conjures up images of a grassy hill with the remains of a few stone walls just visible at the top. A better translation would be "palace", with marble floors, chandeliers, and antiques beyond the dreams of avarice. There is the music room (with grand piano, cello and violin), the "haunted" meeting room (a lady in white appears at midnight, allegedly - I think I have just started this rumour!) complete with a very large chiming clock, and the "comfy sofa" lounge - scene of many late night discussions.

But that is only a part of Palace Dagstuhl - there is also the new section, consisting of the simple, but complete, accommodation (each room has an Ethernet port, phone and alarm clock), the computer room, library and meeting rooms.

There are two more ingredients that make any conference at Palace Dagstuhl special : the food and the "culture". The food is excellent - a never-ending supply of coffee and cakes, and delicious meals. The "culture" is difficult to define, and exists because of a number of simple things. Once you are inside Palace Dagstuhl there are no locks (the main entrance doors have electronic PIN keypads). There are supplies of drinks and snacks, which you pay for at the end of your stay by ticking off boxes on a piece of paper. There is a selection of mountain bikes which you are free to use to climb the cliff overlooking the site (which has the required ruins on the top of a grassy hill). And a communal sauna...

The overall effect is a feeling of being "at home", relaxed, and looked after, in a way that allows you to concentrate completely on the conference itself.

Bernd Paysan

A Web-Server in Forth No modern computer language is complete without support for the Web, and this paper shows how Forth can process HTTP requests, and become a Web Server. Like most of the high level Web protocols, HTTP uses ASCII text terminated by a CR, and therefore requires many string processing words. We all know that you can do anything in Forth, but its nice to prove it!

Malcom Bugler

Forth versus the Beast The interface between software and hardware is rarely more fragile than at power-up, power-down and reset. Watchdogs live in this twighlight zone, where software must run on hardware that is not quite ready for it, and hardware must make sure that it doesn't. When this nightmare world goes wrong, Forth can provide a guiding light...

Nick Nelson

A Windows driver program written in Forth Few real-time programmers can mention this ubiquitous OS without spitting, so it is good to see its foundations being chipped away, especially now NT and 2000 are beginning to make life difficult for those of us who like hardware.

Alan M Robertson

CANed Objects Extensions to the Forth compiler allow sensors to reveal their innermost secrets - and the dream of auto-configured sensors comes a step closer to reality.

Chuck Moore

The c18 colorForth Compiler A cross compiler in 3 blocks, for a chip that (theoretically) runs 2400 MIPS at 20mW. Unique.

M. Anton Ertl

Threaded Code Variations and Optimizations A thorough comparison of several different threading schemes, complete with real execution speeds and code sizes. A lot of work has been done here!

David Gregg, M. Anton Ertl, John Waldron

The Common Case in Forth Programs Another comprehensive study of Forth "under the hood", this time analysing just how often words are called in real Forth programs.

Bill Stoddart

Using Communicating State Machines to Design an Interrupt Driven Task Scheduler This one really got us all thinking. Formal Methods in computer science which can actually be understood by programmers! Perhaps now we can all prove that Forth is good!

Jenny Brien

Treating Data as Source An XML parser illustrates simple techniques to extend Forth's standard tools for handling source to handling data (see September issue of Forthwrite).

Daniel Ciesinger

OO Package for embedded control A proposal for a standard ObjectOriented extension wordset, suitable for embedded systems. Some clever design decisions provide the abstraction of OO by using the flexibility of Forth.

Manfred von Thun, presented by Reuben Thomas

Joy: Forth's Functional Cousin I am seeing a pattern here : take any field of modern computer science, express it in Forth, and suddenly mere mortals can understand it. In this case the Joy of Functional Programming (pun intended) is exposed to the common people.

Howerd Oakford

colorForth and the Art of the Impossible My first impressions of colorForth. Chuck's latest Forth has a very steep "unlearning curve" - it is not an "operating system", it does not have layers, it does not use ASCII. The list of features which are conspicuous by their absence is quite long. If you want to make money out of computer programming, jump on the current bandwagon and exploit its complexity. If you want to understand computer programming, look at Chuck's code and try to grasp its awesome simplicity.

Chuck Moore	**25x Emulator** An emulator in under 3 blocks for a chip that (theoretically) runs 60,000 MIPS at 500mW. Neat.
Reuben Thomas	**The Mite VM: bridging the complexity gulf** A new virtual machine, an attempt to "bridge the cultural divide" between the complexity of Java, and the minimalism of machineForth. Gives the "Forth bridge" a whole new meaning!
Federico de Ceballos	**A Minimal Development Environment for the AVR processor** An 8 bit virtual machine, a suite of compilers and a Windows interface all add up to a simple, reliable system for teaching programming. I am optimistic that a whole new generation of programmers will be produced who react in horror to the complexity of conventional environments . . .
Klaus Schleisiek	**An Open Source VHDL "micro core"** Hardware defined by software. Design your own Forth processor - choose the data and instruction width, then choose your FPGA chip... Awesome.

I could not finish this report without a brief mention of the workshops:

- Open Firmware
- Object Oriented Forth
- Security and Encryption
- Abstraction
- Internet Protocols
- Forth Hardware

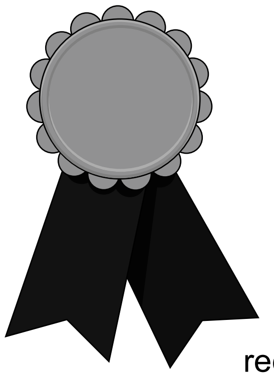
which were a rare opportunity to discuss these and other more diverse topics in more detail with some very smart people!

I would like to thank Chuck Moore for his two presentations at the conference, the time he spent explaining the design of colorForth and OKAD II, and of course for discovering and developing Forth in the first place...

Thanks are also due to Nicole Probst and Angelika Mueller at Schloss Dagstuhl, all of the presenters and participants, and of course to Bill Stoddart and Peter Knaggs for organising the event and editing the papers. Well done!

Another great euroFORTH. See you next time!

euroFORTH 2002 is in September, provisonally booked at the Vienna University of Technology. For announcements, join the mailing list at euroforth-subscribe@yahoogroups.com.



Presenting The FIG UK Awards of 2001

These awards are given to encourage effort and recognise achievement. The FIG UK Awards of 2000 were won by Keith Matthews and John Tasgal.

Free membership

To everyone who sent in their nominations - "thank you". Looking back, a lot of good work was done during 2001 and our judges, the officers of FIG UK, have now chosen two winners. They each receive:

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

Achievement

Chris Hainsworth: for 20 years of enthusiastic support for FIG UK, most recently as chairman.

Forthwrite

Dave Pochin: for his many (11 so far) articles most recently on using Windows from Win32Forth.

We congratulate Chris and Dave on winning - enjoy your year of free membership!

The FIG-UK Library

Graeme Dunbar

The Group's Library now has a new home in the Archive Room of the School of Engineering at the Robert Gordon University in Aberdeen. On behalf of all members of the group I give our thanks to Sylvia Hainsworth as our out-going Librarian, for many years of excellent support and service and to Doug Neale for the not inconsiderable task of packing the library up and sending it on to me.

Your new Librarian has been a member of FIG-UK since 1984 when a colleague introduced me to the group. I am a lecturer in Electrical Engineering and through my work on the School library advisory panel I hope to be able to call upon my colleagues in the University library to keep me right and give advice.

I expect that very few people have actually seen the library on its shelves. Five boxes of books sounded like an enormous amount, but on the shelves it looks quite modest. Nonetheless we have a good range of books, journals and conference proceedings. Most of the books would probably suit Forth novices as many of our more experienced members will already have a selection of them on their own shelves. The catalogue is really a history of Forth publishing and sadly new books are being added to it only infrequently. It would not be boasting to say that we will try to keep a copy of every new book on Forth published.

The latest additions to the library are replacement copies of "Forth Application Techniques" by Elizabeth Rather and "Forth Programmers Handbook" by Edward Conklin and E. Rather direct from Forth Inc.. Forth Inc. also offers a free trial version of Swift Forth to use with the books.

Over the years some of our copies have gone missing and as they are now out of print cannot be replaced unless we can find second-hand copies or donations from other members. They include some of the more specialist and advanced books that were never printed in large numbers. There has not been time to do a full check so I will be publishing pleas for donations in a future issue.

The situation regarding journals and conference proceeding is much brighter. We have a complete run of Forthwrite, with any missing issues having been made up from Doug Neale's stock of spares and my own copies.

The procedure for borrowing is simply to send me your request, giving your membership number. To give all members a fair chance of getting hold of material of interest to them there will be a limit of two books out on loan to any borrower at a time. The normal loan period will be three months unless another member makes a request. When you return the book or books please enclose stamps to the value on the package when you received it. This keeps money transactions to a minimum. Apart from the purchase of new books, the library should be largely self-funding if this scheme runs smoothly.

It would be very helpful if borrowers could spare the time to write a short review. This would be especially useful for new or uncommon books, but for the old standards even just a few lines and comments could well be of benefit to other potential readers. Depending on the book and the review this could be an article for Forthwrite or perhaps posted on our web page.

The tutorial "Forth Application Techniques" from Forth Inc. went missing soon after Robert Ives' review last year (see Jan 2001) but Graeme has since bought a replacement copy. As well as providing excellent hands-on material for basic concepts, it also includes advanced topics like word-lists and multi-tasking.

gtelfer@po.synapse.ne.jp

Seven Times Five Equals Eleven

Graham Telfer

Graham describes his development process for a set of routines which explore modular arithmetic.

Introduction

Most people look at `7*5 = 11` and immediately say it's wrong. They say the answer is 35. Arithmetic is one of those things we learn in school, use everyday, and never think twice about again. So, when does `7*5=11`? The answer is when using a clock. Or you could do it this way:

```
7*5 = 35
35/12 = 2 rem 11
```

or even

```
: ClockArith  ( n,n--)   * 12 mod  4.R ;
```

Reach for your Clock

We are going to advance the clock's hour hand 7 times in jumps of 5 hours each time. Doing this gives us this sequence:

```
Jumps    Clockface reads
  0          5
  1         10
  2          3
  3          8
  4          1
  5          6
  6         11
```

When we multiply, we expect `7*5` to give the same result as `5*7`. The fact that we are doing clock arithmetic should not change that. Here is the sequence for `5*7`.

```
Jumps    Clockface reads
  0          7
  1          2
  2          9
  3          4
  4         11
```

which ends up with the same result as before, **11**.

When I did this the first time two things struck me. The first was that there is a rollover to contend with. Every time the clock hand passes 12 we go back to 1 again.

The second thing was that here is a lovely example of a loop in action.

Time for Forth

This is an apparently simple problem and so the amount of code should reflect that. If we feel the code is getting too big then the chances are we do not understand the problem as well as we thought we did; or we are trying to solve a different problem.

Words and Names

Forth relies on words. These are the equivalent to procedures in other languages. Every language tutorial stresses finding good names for procedures, but in Forth naming takes on greater significance. With my wordsmith's hat on, let's consider a top level word for the program which is what the user will see.

Howeve even with my thesaurus this is proving difficult. I want to give a feeling of both the clock and the arithmetic. A decision must be made and I am going for `TickTock*`.

```
: TickTock* ( n,n--)   ... do something ... ;
```

Now all we have to do is fill in the blank space.

The Program

Looking back at the initial sequences, we can see two features. One is the idea of repeating a task, (counting the number of times we have jumped) and the second is the addition to the current hour hand value of a constant amount.

Let's replace `do something` with a name. Looking in my thesaurus again I come across `Progression`. It's perfect! `TickTock*` makes a `Progression` round the clock (and sounds more regal than advance).

```
: TickTock* ( n,n--)   Progression ;
```

To make a progression we need to progress one step at a time. That step needs to be inside a loop that keeps track of how many steps we have taken.

Suddenly a lot of terms are popping up. We don't want to get confused about the exact meaning of each:

`TickTock*`	is the top level word
`Progression`	is the loop
`Progress`	is one complete step of the loop

Forth gives us the choice of four looping structures. One of these keeps an automatic check on the number of times we've looped. It has the form:

```
   n n Do ... Loop
```

A bonus is that we can display the index from inside the loop with the built-in word `I`.

Helping Yourself

This is not a big program but anything that helps to understand what is happening is worth doing. When I'm in the early stages of sketching out words, I find that plugging in actual values into the stack diagrams instead of just `n1,n2` helps me think about what I want to happen.

Choose a standard way to describe the type of words. I always use upper case letters at the beginning of Forth words. For variables and constants I always start with an underscore: `_Name` .

When a variable or number on the stack has changed value then I put a tick at the end: `_Name'`.

Stack or Variables

Purists use the stack by tradition. Values on the stack are effectively unnamed local variables. Named variables are global. In many cases this works very well, and Forth provides many words to manipulate the stack.

I think that the stack can be a distraction at the time of developing a program. You have enough on your mind trying to get the logic and function right without getting bogged down ordering values on the stack. The stack manipulation words themselves can also get in the way of understanding the essence of what is happening inside a word. For this program, I'm going to use named variables.

```
Variable _Jump
Variable _ClockFace
```

Progression

`Progression` is the loop. We expect to be calculating `7*5, where 7` is the number of times around the loop and the `5 i`s the amount to jump each time. We can store that safely away leaving the `7` as the upper loop limit.

```
: Progression  ( 7,5--) (7,0 as loop limits)
   _Jump !  0 Do Progress Loop ;
```

Progress

`Progress` needs to add the value of `_Jump` to the `_ClockFace`. The only problem is to check for the rollover.

```
: Progress  ( _Jump, _ClockFace -- _ClockFace')
   _Jump @  _ClockFace @  +  RollOver?  _ClockFace ! ;
```

Having the addition naked so to speak looks messy and so we can take it out and give it a word of its own:

```
: +Jump  ( _Jump,_ClockFace -- _ClockFace')
   _Jump @  _ClockFace @  + ;
```

Rolling Over
Checking for a rollover means deciding to do something or do nothing to the clock's current value. If the clock's value goes over 12 then we do something; else we do nothing.

The number 12 doesn't tell us anything about what it is. Let's hide it behind a constant. Just calling it `Twelve` doesn't give any more information than the number. The number is the high point on the clock. Let's call this constant `_ClockHigh`.

```
12 Constant _ClockHigh
```

Now we have the number to be tested we need to make the test. This is `RollOver?` If the number being tested is greater than 12 how do we correct it? The word `Mod` does this for us.

```
: RollOver?  ( _ClockFace -- _ClockFace or _ClockFace')
   _ClockHigh Mod ;
```

Displaying the Results
We've given no thought yet to displaying the results. I considered having a full clock face but a quick look at the Windows programming involved dissuaded me. For this project I want to follow the layout of Figure 1.

First of all there is a title line. Underneath this, on each pass we want to display the number of advances and the curent hour hand value.

The Title
This can be a simple string of text.

```
: ClockTitle  (--)  Cr Cr ." Jumps     Clockface reads" ;
```

Showing the Progression
Remember the `do loop` lets us display the current index value using the word `I` and we want to take advantage of this. We must display each clockface as it is calculated inside the loop after we have displayed the index.
Two words can do this.

```
      : ShowTimes  ( Index --) 3 .R ;
      : ShowClockFace  ( _ClockFace --) 11 .R ;
```

All they need to do is get into the right position under the title and then display the times round and the current clockface.

Keeping things as simple as possible I've used the word `.R`. This offsets by the number of fixed width spaces given before displaying the number found on top of the stack.

`Progression` now looks like this:

```
: Progression  ( 7,5--) (7,0 as loop limits)
  0 Do Cr I ShowTimes Progress Loop ;
```

The clock face can be displayed inside `Progress`. Making this change means `Progress` looks like this:

```
: Progress  ( _Jump, _ClockFace -- _ClockFace')
   +Jump RollOver? ShowClockFace _ClockFace ! ;
```

*Time for TickTock**

Everything seems fine and so we can put write that top-level word.

The word must make a progression, but before that it should display the text. The word we wrote to do this was `ClockTitle`. Many programs leave untidy bits after they finish. It's only polite to tidy up afterwards. Here we just need to reset the `_ClockFace` to `0` and move the Forth's “OK” down using `Cr`.

```
: TidyUp (  n --)  0 _ClockFace ! Cr ;

: TickTock*  ( n1,n2 --)   ClockTitle Progression TidyUp ;
```

ARRGH!

It doesn't work! Tracking down bugs is a laborious but inevitable part of programming. So where is this little bug? Look at the code: `Rollover!` I'm storing the `_ClockFace` but immediatle afterwards I need to display the clock face in `Progress`, but there's nothing there.

Moving the `_ClockFace !` into `Progress` and providing a `DisplayCopy` for `ShowClockFace` should solve the problem.

Now try again.

Test Results

Type the two numbers (positive ones, please) you want to multiply and a space, then type the word `TickTock*` and press the Enter key

```
ok
7 5 TickTock*

Times   Clock Reads
  0          5
  1         10
  2          3
  3          8
  4          1
  5          6
  6         11

5 7 TickTock*

Times   Clock Reads
  0          7
  1          2
  2          9
  3          4
  4         11

5 13 TickTock*

Times   Clock Reads
  0          1
  1          2
  2          3
  3          4
  4          5
```

The Complete Code

```
    .( Clock Arithmetic Program)

    \ Helper Words
    12 Constant _ClockHigh  \ Rollover value on this clock
    Variable _Jump
    Variable _ClockFace   0 _ClockFace ! \ set the clock

    \ Display Words

    : ClockTitle  ( --)  Cr Cr ." Times   Clock Reads" ;
    : ShowTimes  ( --)  3 .R ;
    : ShowClockFace  ( --)   11 .R ;
    : DisplayCopy  ( _ClockFace -- _ClockFace, ClockFace) Dup ;

    \ Main Words
```

```
: TidyUp  ( n --)   0 _ClockFace ! Cr ;
: _Warn Cr Cr ." Sorry, clocks don't use negative values" Cr
   ." and so don't enter them" Cr ; _Warn
: Instructions  ( --) Cr
   ." Type the two numbers you want to multiply"
   ."  and a space," Cr
   ." then type the word TickTock* and press the Enter key" Cr ;

Instructions

: +Jump  ( -- _ClockFace') _Jump @ _ClockFace @ + ;
: RollOver?  ( _ClockFace -- _ClockFace or _ClockFace')
   _ClockHigh Mod ;
: Progress   ( --)
   +Jump RollOver? DisplayCopy ShowClockFace _ClockFace ! ;
: Progression  ( n,n--)  _Jump ! 0 Do Cr I ShowTimes  Progress Loop
;
```
: TickTock* (n,n--) ClockTitle Progression TidyUp ;

FIGUK

FIG UK Downloads

Visits to the FIG UK web site have been running at such high numbers (1000/month) that Jenny, our webmaster, has carried out a detailed analysis, especially for the downloads of Forthwrite.

After filtering out any unsuccessful attempts, over a 5-day period in January, more than 40 issues were downloaded each day.

These downloads were not especially biased towards the latest issue as might be expected, but spread over all the 10 issues we currently have on-line. I think this means that people are hearing about Forth, discovering the site and downloading the issues that sound most interesting. The numbers involved suggest that there is far more interest in Forth than might be thought just from the size of the FIG UK membership.

The challenge for us is to make it as easy as possible for some of these 1000 visitors to get started in Forth.

Henry Vinerts
Volvovid@aol.com

Across the Big Teich

Henry Vinerts

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

FIG Silicon Valley Chapter Meeting - Nov. 2001

Greetings from Silicon Valley!

"If the computer had been invented in China, what problems would English-speaking people have to surmount in order to use it?" Thus begins Timothy Huang's article in June 1984 issue of Dr. Dobb's Journal. Would you like me to send you the procedure for extracting the cube root by bead arithmetic (from "How to Use the Chinese Abacus" by Kwa Tak Ming)? I am just kidding, I know that in today's world it is more important to know how to underscore text in Visual Basic or in Win32Forth.

That almost sums up our topics in the first SVFIG meeting of year 2002.

I don't know what happened to Huang's First Chinese Forth, but now, almost eighteen years later, Dr. Ting's continuation of the same effort seems like a case of deja vu or "Alles schon dagewesen," as Ben Akiba would have said in German. Except, with his customary attention to detail and his infinite capacity for learning, Ting has brought Bezier curves and B-splines into the mechanics of drawing Chinese characters, arriving at yet another interesting subject for one of his lectures. As is usually the case, Ting's talk stimulated the participation of the audience, especially of those who enthusiastically wished to share their knowledge in related and unrelated subjects. And there is a lot of collective knowledge in that one room on the fourth Saturday of each month.

I can't say whether it was good or bad that this month we were back in the old room without direct Web access. Less entertainment..., more undivided attention to John Peters' campaign of improving Win32Forth, especially when he was inviting everyone to crash it. I wonder what Dave Pochin would have said, seeing what to me looked like a couple of dozen farm boys in delightful excitement, having come upon an unattended farm tractor. The "Windows monster" is very inviting, everyone wants to drive it, to show off his skills. "Don't panic, press F1 now!" "Let me try, I think I know how to do it!"

The boys did not even ask for an afternoon break, and the tractor still had 15 potential drivers at 4 P.M., when it was time to go home. Both copies of Forthwrite #115, which I had received just in time for the meeting, had circulated among the group; there were even a few "readers" of the batch of Vierte Dimensions that I had brought along. I am disappointed to report, however, that when Alan Furman tried to bring Hans Eckes' "Forth stamp" article (VD 4/2000) to the attention of the group, their interests were somewhere else, or perhaps the mindset of the SVFIG has changed more than I would like to believe.

I would like to end with a humorous note, but, since I am free to write whatever is on my mind, I'll save it for next time and, instead, share a quote from Warren S. McCulloch with those who consider themselves old Forthers:

"There is a utility in death because ... the world goes on changing and we can't keep up with it."

Keep smiling,

Henry

FIG Silicon Valley Chapter Meeting - Feb. 2002

My report of the SVFIG meeting of February 23rd can be summarized in Forth-like manner as **`JSB > DLL`** and I trust that it will hold true for generations to come.

It means that the works of J.S. Bach are greater than all of the dynamic link libraries that have been produced by thousands of monkeys (for lack of a better word). It also reinforces my belief that most of the worthwhile and enduring creative works stem from the genius and productive capacity contained in a single mind.

You see, in the morning session Dr. Ting delighted an audience of about ten of us with an organ performance of Bach's "Art of Fugue," the organ being a 4.77 MHz XT, programmed in Forth and
running four speakers through Ting's own hand-made oscillator card. In this 1988 package Ting used only 4 channels, so some of the 15 fugues were abbreviated, but the music sounded very realistic, which, as Ting explained, is due to the fact that pipe organ output can quite well be matched with square waves.

The whole concert fits on four 360KB floppy disks, but for those
who wish to hear it again on their modern machines, Ting has cut a 600 MB CD, which, albeit with a few stray notes, can provide one with almost an hour's enjoyment of Bach's gloomy grandeur. Amazing stuff! Now I'd rather go back to read my old copy of "Goedel, Escher, Bach," than continue with this report...

Let it suffice to report that the committee which was supposed to work on modifications of Win32Forth in the afternoon session was strangely absent. A number of listeners left at lunch, to be replaced by a few who had missed the concert because the WWW had demanded their morning; yet, nothing new was said or done about Win32Forth. Bob Smith was dutifully there, being one of the original contributors to F-PC, but I think that since Bob is an accomplished organist and accordionist, the morning was the main reason for his presence. Perhaps, unless we retrieve the single mind of Tom Zimmer from Texas, not much will happen to make us listen to Win32Forth instead of Bach.

Dave Jaffe gave a short description of his latest Forth programs
for threshold acquisition in switch debouncing circuits and for converting parallel-port output lines into serial streams, and then it was question-and-answer time for the remaining few who are looking for ways to pursue their chosen tasks with a minimum of lost time in the today's information jungle.

Happy hunting to you all,

Henry

Alan J M Wenham
01932 786440
101745.3615@compuserve.com

Vierte Dimension 4/01

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

Among other things, the editor Martin Bitter reported that Ralph Hempel had placed a new version of pbForth on the Internet at http://www.hempeldesigngroup/lego/pbForth/index.html. Martin is also setting up a literature service (Martin Bitter, Fred Behringer).
An interesting extract concerning Forth is published on the homepage of Bernd-M.Stejskal at http://www.stejskal.de/web/computer/forth/index.html. He also noted that fewer and fewer authors were writing for VD.

Riddles

Fred Behringer
behringe@mathematik.tu-muenchen.de

Fred Behringer discusses solutions to his VD2/2000 riddle.

MickerForth-MACRO4th.asm

Wolfgang Allinger
All@business.kbbs.org

In connection with use on a microcontroller, in this case an 8051, the author needed 32-bit arithmetic. The assembler routines he found in the literature were not clear. He constructed 35 Forth-like assembler macros and used these in conjunction with a stack with three 32-bit entries and a work register.

Reviews

Fred Behringer
behringe@mathematik.tu-muenchen.de

Fred Behringer reviews the Dutch Figleaf for August 2001 and Forthwrite 112.

Tower forever- a further approach

Michael Kalus and Adolf Krueger

As reported earlier (Forthwrite 114, p.33), Fred Behringer and Martin Bitter have devised a means to prevent the Lego robot IR transmitter from disconnecting after 5 seconds of inactivity. However, the transmitter then remains switched on until the battery is removed. Michael and Adolf make use of the DTR-signal to ensure that the transmitter is again disconnected from the battery as soon as the terminal program (eg HyperTerminal) ends or the plug of the COM-link is removed. A circuit diagram is given together with pin-outs of the 9-pin serial cable and photographs of the circuit board.

Syntax of Decision Tables in Forth

Klaus Zobawa
Klaus.Zobawa@t-online.de

This was a paper read at the 2001 AGM of Forth Gesellschaft (Forthwrite 112 p.17). The problem originated from a surgical diagnostic device which was required to be provided with some form of computer intelligence. An H8/300 micro controller was added and a decision table technique was chosen to be the best solution. Programming such decision structures by way of nested `IF ELSE THEN`s is difficult to do and bound to fail if later modifications and easy maintenance are needed. The author uses Forth to imitate the immediate structure of the graphical representation of a decision table. A vertical double stroke becomes the Forth word which signals the entry of an output field, etc..

The Forth "stamp" - the prototype

Hans Eckes
hanseckes@addcom.de

The "BASIC -postage stamp" is familiar in Germany as a development microcontroller of postage stamp size with BASIC as the programming language. Hans reports on a similar project with the PSC100 from Patriot as processor and a 32-bit F83 Forth with assembler, disassembler, and multitasker as programming environments. Block diagrams and schematics are included.

Quartus Forth - first experiences

Wolfgang Allinger

Wolfgang wanted to program a service overlay for a Palm PDA. Quartus is a 16-bit Forth for Palm PDAs, available as shareware. This note relates his preliminary experience of using it.

On the Lego Mindstorm's infra-red data transmission

Michael Kalus and Adolf Krueger
Adolf.Krueger@t-online.de

This is a very good analysis of data transmission from the PC to the infrared transmitter and then from there to the RCX (Lego Robot building block H8/300) and back again. What the authors found out was that there is no way of constructing a multiplex process to operate several robots simultaneously without mutual disturbance and this cannot be done if only the original hardware and firmware supplied is used. Other means will be needed.

Code definitions without CODE and ENDCODE

Fred Behringer
behringe@mathematik.tu-muenchen.de

This is Fred's "Column for language migrants". He shows the ease with which anyone can, in Forth, construct in-line assembler code in colon definitions `ASM[ ... ]FORTH`. Quite incidentally, Fred has developed two words `ASM]FORTH ... FORTH[ASM` with which one can easily change from assembler code (as inside a `CODE` definition) to high level Forth code and back again. `ASM]FORTH` and `FORTH[ASM` replace `]FORTH` and `ASM[` completely and can be used in a `CODE` or a colon definition in any order as often as needed.

Hardcode-Assembler "brute-force" for the Lego RCX

Martin Bitter
martin.bitter@forth-ev.de

The versatility of the RCX building block is limited by the Lego software to those possibilities (not many) foreseen by Lego. "Limitations through the software are limitations of personal freedom! Forth can help here!" The display segments (on the building block) are addressed by the setting of individual bits in the RCX -workspace. Martin suggests, through development of additional CODE definitions in pbForth on the RCX, how to generate a rudimentary cross-assembler on the PC.

Chris Jakeman
cjakeman@bigfoot.com

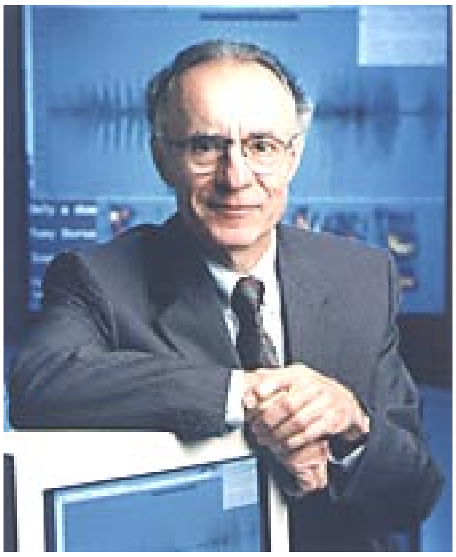
Did you Know? Forth Helps Nobel Prize Winners

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).

It was 1971 when the first standalone Forth that we'd all recognize was developed. It was used to do data acquisition and control for the 36' (11m) radio telescope belonging to NRAO[9] on Kitt Peak Mountain, Arizona.

http://www.bell-labs.com/user/feature/archives/penzias/ reports:

"Penzias also has been honoured for his pioneering work in interstellar chemistry, discovering the presence of key chemicals among the stars. Using his pioneering techniques to observe millimeter-wave radio spectra emanating from space, Penzias and his colleagues identified carbon monoxide and several other simple molecules in the dusty clouds in interstellar space.



Among other finds, the team pinpointed the nuclear composition of the constituent atoms of these molecules--the remnants of burned-out stars and the raw materials for new ones. This work gave astronomers an important new window into stellar composition and life cycles, and has grown into a flourishing branch of astronomy.

Penzias, Wilson and their Bell Labs co-worker Keith Jefferts discovered the existence of deuterium (heavy hydrogen) in outer space in 1973, providing additional clues to the birth of the universe."

Penzias and Wilson had previously[10] made the discovery of cosmic microwave radiation in interstellar space which won them the Nobel Prize.

Source – Bell Labs and Elizabeth Rather, Forth Inc

[9] National Radio Astronomy Observatory

[10] 1963, see http://www.bell-labs.com/project/feature/archives/cosmology/

Letters

The Magazine Team are always pleased to get feedback and encouragement. The first letter was published in the latest PCW magazine following a retrospective on the Jupiter Ace computer.

Chris Jakeman

Dear Editor,

I found your article on "Planet Jupiter", March issue quite fascinating but was sad to find you included the line "Forth **was** a compact and fast programming language".

Forth continues to serve the embedded computer market wherever it is important to make the most of scarce resources, such as spacecraft and mobile equipment. For example, the Titanic wreck was discovered using a submersible controlled by Forth.

Forth has been an ISO standard for many years and the interactive systems produced by commercial Forths are as fast as the non-interactive ones produced by C. The record for performance is currently held by MPE Ltd, a UK company. More details can be found at the Forth Interest Group web-site http://www.fig-uk.org

Another F11UK processor board has been sold, this one to Thierry Charlier de Chily who struggled with a faulty chip. Here is his reponse which is re-printed with his permission.

Thierry Charlier de Chily

From <thierry@charlierdechily.org>
Hi,

I have spent the last weeks totally concentrated on trying to make "my" F11 board running. After a pit stop, Jeremy fixed it and according to his message the board is now up and running. I am longing to receive it.

I want to thank Jeremy for his assistance, Graeme Dunbar and Paul Akterstam for their contribution and I want to thank all of you (on the F11UK mailing list) for your patience.

Time has come to introduce myself.

Me: I am French, 37 years old. I live in the south of France at 20 km of Cannes. I work in Sophia Antipolis, which, despite its name, is located in France, close to Cannes. It is a Technopole ("large urban centre with teaching and research facilities to support development of hi-tech industries"). I am a project manager. Formerly, I was a Unix/C programmer.

Forth and me: a friend of mine gave me a small introduction to Forth 18 years ago. He owned an Amstrad 6128 and I an Apple II. Compared to Basic (GWBASIC), the programming language taught at the University at the same time, Forth was a real shock to me. I bought and read "Forth – Salman, Tisserand and Toulout – Eyrolles editions" and "Débutez en Forth – Leo Brodie – Eyrolles editions" (I guess that it's the translation of "Starting Forth"). Unfortunately, I did not put these readings in practice. Every 3 or 4 years I read these books again (I have bought and read "Turbo Forth – REM Corp editions" too) and try to find an idea of application to implement in Forth. Nothing came. Thanks to robotics, time has come to practice Forth.

Robotics and me: Robotics is a recent hobby for me. Few years ago, my eyes fell on a Mindstorms Lego box. It was quite expensive. I had to wait a while and I bought one box last year in the sales. Just for fun, I started to thing to participate to a local robotics contest (http://195.83.41.66/robotik2.html) with a Lego Robot. The contest wasn't that easy (the rules were those of the French Robotics Cup - http://www.robotik.com) and because of a lack of time I did not participate but I swore to participate in June 2002. I read a lot and I chose a cheaper technology (Microchip PIC) in order to be able to build a team of "cooperative" robots with just a few bucks. I made some basic experiments in assembler and searched for a high-level programming language.

I realised at the end of 2001 that I was very late in my planning. I tried to find a "high level" micro-controller to speed up the software design phase and the coding phase. The price was no longer an issue for me (though it still was for my wife :-)) because cooperative robots were forbidden in the new rules. So I ordered an OOPIC which is in fact a PIC with a built-in Object Oriented kernel and library. The library includes servos, pmw, blinking LED... (www.oopic2.com).

At the same time, I discovered the F11 board. This board was cheaper than the other 68HC11 boards available on the market, it was an opportunity to practise Forth and Jeremy and I were on the same wavelength. So I registered with FIG-UK and ordered the board despite the fact that I couldn't find any Forth Robotics vocabulary on the Net.

I am always late on my planning, but I am glad to have an opportunity to practise Forth. The other good news is that the rules of the contest have changed. This year, in order to have more participants (only 3 teams last year), it will be an Introduction To Robotics (the French Robotics Cup's rules are quite elitist). They will lend a robot, so no design and mechanical issues. I have one more year to get ready...

Chairman	**Jeremy Fowell**,	11 Hitches Lane, EDGEBASTON B15 2LS 0121 440 1809 jeremy.fowell@btinternet.com
Secretary	**Doug Neale,**	58 Woodland Way, MORDEN SM4 4DS 020 8542 2747 dneale@w58wmorden.demon.co.uk
Editor	**Chris Jakeman,**	50 Grimshaw Road, PETERBOROUGH PE1 4ET 01733 352373 cjakeman@bigfoot.com
Treasurer	**Neville Joseph,**	Marlowe House, Hale Road, WENDOVER HP22 6NE 01296 62 3167 naj@najoseph.demon.co.uk
Webmaster	**Jenny Brien,**	Windy Hill, Drumkeen, BALLINAMALLARD, Co. Fermanagh BT94 2HJ 02866 388 253 webmaster@figuk.plus.com
Librarian	**Graeme Dunbar**	Electrical Engineering, The Robert Gordon University, Schoolhill, ABERDEEN AB10 1FR 01651 882207 g.r.a.dunbar@rgu.ac.uk

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

FIG UK Web Site

For indexes to Forthwrite, the FIG UK Library and much more, see http://www.fig-uk.org

FIG UK Membership

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

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

Forthwrite Deliveries

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

Copyright

FIG UK Services to Members

Magazine	Forthwrite is our regular magazine, which has been in publication for over 100 issues. Most of the contributions come from our own members and Chris Jakeman, the Editor, is always ready to assist new authors wishing to share their experiences of the Forth world.
Library	Our library provides a service unmatched by any other FIG chapter. Not only are all the major books available, but also conference proceedings, back-issues of Forthwrite and also of the magazine of International FIG, Forth Dimensions. The price of a loan is simply the cost of postage out and back.
Web Site	Jenny Brien maintains our web site at http://www.fig-uk.org. She publishes details of FIG UK projects, a regularly-updated Forth News report, indexes to the Forthwrite magazine and the library as well as specialist contributions such as "Build Your Own Forth" and links to other sites. Don't forget to check out the "FIG UK Hall of Fame".
IRC	Software for accessing Internet Relay Chat is free and easy to use. FIG UK members (and a few others too) get together on the #FIG UK channel every month. Check Forthwrite for details.
Members	The members are our greatest asset. If you have a problem, don't struggle in silence - someone will always be able to help. Do consider joining one of our joint projects. Undertaken by informal groups of members, these are very successful and an excellent way to gain both experience and good friends.
Beyond the UK	FIG UK has links with International FIG, the German Forth-Gesellschaft and the Dutch Forth Users Group. Some of our members have multiple memberships and we report progress and special events. FIG UK has attracted a core of overseas members; please ask if you want an accelerated postal delivery for your Forthwrite.