# myForth – mods for me

- SVFIG 27-Jul-2013
- Glen Worstell
  - Retired engineer, embedded systems programmer,
  - IT worker, applications programmer, pilot, world-cruising sailor, ...
  - HP, Seagate Technology, UCSC, consultant
- Seeking advice from experts on mods to Forth
  - To use for embedded systems.
  - Not for applications programming, better tools available.

## Because we have limited time, and because myForth is for me, Please don't tell me:

- It is not standard.
- It has always been done that way.
- I won't be able to use other's code.
- I won't be able to publish.
- You won't be able to use myForth.
- Please **do give technical reasons** why it is not a good idea (if it isn't).

## I am a sort of a newbie to Forth

- Toyed with it some years ago.
- Never wrote a complete app.
- Did write parts of a Forth system
  - For many different micros.
  - Have created many small embedded systems in assembly language and C for many different micros.

## Ideas to discuss

- 'OK' needs to have proper <crlf> around it.
- 'base' needs to go away.
- 'do' needs to have proper limits.
- words should have stack effects known at compile time.
- core words should not be able to be redefined.
- white space should be <space>, <tab>, <crlf>.

## More ideas...

- <cntl-c> or something needs to break out of a loop.
- printing words could have consistent syntax.
- should implementations be hosted (on a PC)?
- merits of omitting the interpreter?
- others e.g. vocabulary, don't know enuf to have an opinion.

## Why not Forth?

- Outrageous claims
  - Smaller than assembly
  - Faster than C
  - Virtual memory
  - File system
  - Don't need floating point
    - See 'Starting Forth' for a good argument about why we DO need F.P.
    - But – that seems mostly to be in the distant past

# Why not Forth?
# Reputation as “write-only” language.

```
#  include<stdio.h>//  .IOCCC                                          Fluid-  #
#  include <unistd.h> //2012                                          _Sim!_  #
#  include<complex.h>  //||||                     ,____.               IOCCC-  #
#  define              h for(                     x=011;               2012/*  #
#  */-1>x              ++;)b[                     x]//-'               winner  #
#  define              f(p,e)                                          for(/*  #
#  */p=a;              e,p<r;                                         p+=5)//  #
#  define              z(e,i)                                         f(p,p/*  #
## */[i]=e)f(q,w=cabs  (d=*p-  *q)/2-     1)if(0  <(x=1-       w))p[i]+=w*/// ##
   double complex a [  97687]  ,*p,*q     ,*r=a,  w=0,d;     int x,y;char b/* ##
## */[6856]="\x1b[2J"  "\x1b"  "[1;1H     ", *o=  b, *t;    int main   (){/** ##
## */for(              ;0<(x=  getc (     stdin)  );)w=x   >10?32<     x?4[/* ##
## */*r++              =w,r]=  w+1,*r     =r[5]=  x==35,   r+=9:0      ,w-I/* ##
## */:(x=              w+2);;  for(;;     puts(o  ),o=b+   4){z(p      [1]*/* ##
## */9,2)              w;z(G,  3)(d*(     3-p[2]  -q[2])   *P+p[4      ]*V-/* ##
## */q[4]              *V)/p[  2];h=0     ;f(p,(  t=b+10   +(x=*p      *I)+/* ##
## */80*(              y=*p/2  ),*p+=p    [4]+=p  [3]/10   *!p[1])     )x=0/* ##
## */ <=x              &&x<79   &&0<=y&&y<23?1[1  [*t|=8    ,t]|=4,t+=80]=1/* ##
## */, *t              |=2:0;    h=" '`-.|//,\\"  "|\\_"     "\\/\x23\n"[x/** ##
## */%80-              9?x[b]      :16];;usleep(  12321)       ;}return 0;}/* ##
####                                                                        ####
################################################################################
**############################################################################*/
```

**We can do better.**

# Why Forth?

- Arm Cortex M0-M3-M4-M4F
  - Becoming a world standard
  - Lots of vendors
  - Cheap, powerful
  - Slowly becoming hobby-friendly
- BUT …
  - Next slide

## Why Forth?

- Writing and debugging assembly code on MSP430 is pretty easy (my experience)
- Writing and debugging forth (Mecrisp) on MSP430 is even easier! (my experience)
- Compiling and debugging c and/or assembly on Cortex is drudgery.

# 'OK' needs to have proper <crlf> around it.

- Shell output; easy to read; there is a prompt followed by user input, then there is the computer's output, then there is another prompt.

```
glen@dell ~/Documents/svfig $ ls -al
total 1056
drwxr-xr-x  2 glen glen    4096 Jul 26 11:32 .
drwxr-xr-x 14 glen glen    4096 Jul 26 11:08 ..
rw-r--r--  1 glen glen      68 Jul 26 11:32 .~lock.myForth.odp#
-rw-r--r--  1 glen glen 1063766 Jul 26 11:32 myForth.odp
-rw-r--r--  1 glen glen    2497 Jul 26 11:05 svfigTalk.txt
glen@dell ~/Documents/svfig $
```

## 'base' needs to go away.

- Use $nnn (hex)
- Use %nnn (binary)
- Use nnn (decimal) or #nnn if base exists
- Use nrnnn (any base, the first n is the base in decimal)
- Next slide...

# BASE ball

What is the score?

You need to carefully watch the entire game to know.

Or, look at the scoreboard.

27 foobar ( n – )

foobar gets executed n times.

What is n?

Nobody knows. It is not necessarily 27.

decimal base @ .

oh, ya, now I know. I looked at the scoreboard.

(oops – now the score has been changed (maybe) by looking at it.

# Meanings should be clear and normal for humans.

- C: 010 = 8
  - WTF? (This error occurs in many languages)
  - For math and finance, just plain wrong.
  - Humans can get used to anything (but should not need to).
- Forth: 010 = ? (happily does not repeat above)
  - Anything. Depends on base.
  - May also depend on whether 010 or 10 was defined  to mean something else.
  - 42 constant 010
    - Still don't know, depends on base.

## 'do' needs to have proper limits.

- Programmers are not entirely human. :)
- We should try to make our languages human-like.

# Two couples getting married

```
: getMarried  1 do cr
." person " i .
." said I do." loop cr ;
```

```
4 getMarried
person 1 said I do.
person 2 said I do.
person 3 said I do.
 ok.
```

OOPS –
one didn't say “I do”.
Are they married? NOT “ok”

## How many sheep do you have?

Forth, Python, etc: I'll count.  0, 1, 2. OK, I have two sheep.

Lua: I'll count. 1, 2, 3. OK, I have 3 sheep.

Human: Same as Lua.

## Words should have stack effects known at compile time.

- (eg, get rid of '?DUP')
- (n – n n) or (0 – 0)
- Any others?
- Why?
  - Important for optimization.
  - Consistent behavior with other words.
  - Not very difficult to avoid it.
  - I don't like it.

## Core words should not be able to be redefined.

- Forth on modern micros (more flash than ram) should have 3 areas for code to be stored:
  - 'core' flash, must re-compile forth to change.
  - 'user' flash, for new definitions that are pretty solid.
  - RAM, for words under development.
- Core words are supposed to be solid and well-defined. Changing them is simply a bad idea.
- Arbitrary words in 'user' flash are sometimes difficult to erase, so maybe redefining them should be OK. Flash is usually erasable only in blocks. Erasing en masse should be ok.
- Words in RAM are easily forgotten.

# White space should be <space>, <tab>, <crlf>.

Will this compile? (Python)

```
import re
for test_string in ['555-1212', 'ILLEGAL']:
    if re.match(r'^\d{3}-\d{4}$', test_string):
        print test_string, 'is a valid'
    else:
        print test_string, 'rejected'
```

Nobody knows – is the indention spaces, tabs, or a mixture?

Will this compile? (Forth)

```
1    2  +   .   \ add em up
```

Nobody knows...

## More ideas...

- <cntl-c> or something needs to break out of a loop.
- printing words could have consistent syntax.
  - . ." u. u.r d.r .s ud. d. .rs
- should implementations be hosted (on a PC)?
  - Source and docs must be on PC anyway.
- merits of omitting the interpreter?
- others e.g. vocabulary, don't know enuf to have an opinion.

## I forgot what this was supposed to represent. If it walks like a duck and quacks... Forth needs to have words that are easy to remember for my overloaded brain.

# After-Talk Ideas

- Thank you all for the excellent comments.
- I've decided that I do not yet have enough experience and knowledge to modify core forth.
- It is easy to do some of my ideas – define new printing words, for example, without modification.
- If I change anything it will be 'OK' – the only thing that I don't want to live with.
- Un-thanks to the rude person who yelled irrelevant things in my face during my presentation – everyone else was polite.

## Thank you for your ideas.

- I will probably make a modified version of Mecrisp Forth.
  - It will run on Cortex M0.
  - It will have the mods you did not shoot down.
  - M3 is much better and only a bit more expensive.
  - M0 has hobby-friendly packages.
  - M0 has some peripherals not available in M3.
- Cheers, more info at next talk. Bye... gw.