CamelForth vs 4e4th05a

Why did we make a 4e4th out of CamelForth?

We needed a Forth that was tailored to our experiments with the MSP430 Launchpad made by Texas Instruments. It should be very error-tolerant and easy to use with Dirks programming environment, the 4e4th-IDE, written in his VisualForth, a win32forth projekt. At that time CamelForth, assembled in IAR's "Kickstarter" IDE, was a good starting point for that. But the constant changes to "Kickstarter" was not. This proprietary software became more and more of a hindrance.

But now there is Mike Kohn's assembler. And Brad Rodriguez has ported and improved his CamelForth there. CF now offers tools like MARKER and SAVE and RESTORE, as well as ways to restore the intact Forth kernel even if garbage has been programmed. From the terminal, this is achieved interactively with the SRCUB word. Or via the hardware of the launchpad, the two buttons S1 and S2. So, barely any changes were needed to get our 4e4th look & feel back.

Developing Applications.

CamelForth/MSP430 has the ability to save and autostart user-defined application programs. 4e4th05a works the same way now.[1]

Read the appropriate chapter in **msp430development.pdf** to understand how it works.

This also applies to the following chapters.

SCRUB and RESTORE

In 4e4th there was WIPE instead of SCRUB. Both work identically. In 4e4th05a **both** are included.

CamelForth can perform a RESTORE which equates to a warm start when SAVE has been done.[2]

It is are provided along with COLD for development and debugging purposes. There should never be a need for your application program to call RESTORE or COLD or SCRUB.

Autostart Bypass

4e4th had this, and CamelForth implemented it too. So it works in both of them alike.

Hold down S2 button and press S1 reset button then.

Summary of behaviors

See: **msp430development.pdf, page 4.** 4e4th and CF behave identical.

Creating an autostartable Application

Same as CF. Application programs must reside entirely in ROM! The word which starts your application must be a colon definition, and must be the last word you compile.

MARKER

Same as CF. If you use MARKER, normally you’ll just use one, so this isn’t a major problem. After you’ve finished debugging, and are ready to compile your completed application, you should remove the MARKER from your source file.

---

1 In 4e4th 0.34 you had to ' <word> APP ! to start the auto-launch of an application. Brad eliminated this now. SAVE is enough to start the last defined Forth word automatically after the reset.

2 In 4e4th 0.34 there used to be WARM as a starting option besides COLD. This notation was still from the old AIM65 Forth times.

Set, clear and test bits in a cell.[3]

CamelForth already supports this nice MSP430 instructions.

Clear, set or test a bit in a data **word**.

CLRB	(mask adr – –)
SETB	(mask adr – –)
TSTB	(mask adr – –)

Clear, set or test a bit in a data **byte**.

CLRB	(mask adr – –)
SETB	(mask adr – –)
TSTB	(mask adr – –)

So what is different in 4e4th?

Case sensitivity while typing

CF is case sensitive, 4e4th is not. In the older 4e4th this was switchable, now it is an integral part of 4e4th05a.

The OK-Prompt

After interpreting a line, forth prompts OK if no error occurred. CF does a simple OK. 4e4th gives you some more information.

It displays its number base as a hexadecimal value, format is: $hh

Usually you will see one of these codes:

`$0Aok`	decimal number base.
`$10ok`	hexadecimal number base.
`$02ok`	binary number base.

But you can choose either BASE you like by typing `<value> BASE !`

The prompt pictures the stack by printing a dot for each value on the stack. An empty stack has none.[4]

`$0Aok`	Empty stack.
`$0Aok..`	Two values on stack.
`$10ok...`	Three values on stack.

… and so on.

Stack underflow (SUF) detection

In 4e4th ?STACK aborts interpretation and compilation if a stack underflow occurs, and prints SUF, otherwise does nothing.

Additional features and words

The expression in brackets following a word, is a stack comment. In Forth, as you may know

3 In old 4e4th we named them CLR SET GET. Now we use the CF notation, so this is *not different* any more.

4 Thanks to noForth.

already, words get or put values from a data stack. See the forth word glossary to learn the syntax and how to use them.

(– –)	No stack effect.
(n – –)	Word will consume number n from stack.
(– – n)	Word will leave number n on stack.
(n – – m)	Word will consume number n from stack, leaving number m.

… and so on.

WORDS

It has a stop & go feature. Press the spacebar to toggle listing, other key to abort it.

WORDS	(– –)

\

The backslash character is provided as a word to suspend compilation till end of line. Write comments there after.

\	(– –)

WIPE

Same as SCRUB only for compatibility reasons: Make a clean user flash.

WIPE	(– –)

2CONSTANT GREEN RED S2

CF and 4e4th use the (mask adr --) notation to address a single bit in a memory word or byte, or in a port. So 2CONSTANT is provided to save mask and adr as one word.

GREEN is such a Word. It puts mask and address of the green LED's corresponding port pin onto the stack — the pin on the Launchpad of course.

RED does it for the red LED.

And S2 does it for the S2-button, so you can pull it's state. S2? already does the job, providing a flag on the stack indicating the current status of S2.

These are provided to enable quick first steps in forth on the Launchpad.

2CONSTANT	(n – –) <name>
GREEN	(– – mask adr)
RED	(– – mask adr)
S2	(– – mask adr)

So `GREEN CSETB` will turn the green LED on. And `GREEN CCLRB` will turn it off.

Same with RED.

CTOGB

Toggle a bit in a byte. Let your LED blink.

CTOGB	(mask adr – –)

S2?

Provided to test switch S2 of the Launchpad.

S2?	(– – flag)

P1OUT P1IN

Output and input register address of port1.

P1OUT	(– – adr)
P1IN	(– – adr)

1MS

Delay of about 1 millisecond. No MCU timers used.[5]

1MS   (– –)

MS

Delay of several milliseconds.

MS   (n – –)

`1000 MS` are a second.

So, this is it. Have fun!

Michael

Send erratum to: mik.kalus@gmail.com

5 This is done by nested loops, not a calibrated counter. So do not expect to much precision.