Forth for Education - 4E4th and 4E4th IDE

Dirk Bruehl
BRUEHLCONSULT
www.BruehlConsult.com

Abstract

This paper describes the necessity to offer special easygoing tools for the next generations of enthusiastic and engaged Forth programmers education, frankly reporting ways and means, showing a trailblazing example on the path to get there.

Preface

This is rather more an experience report than a technological review, but it has to be that way, to later make the conclusions better understandable for the reader.

Recently I received a TED Talk by Kevin Kelly [1]: How technology evolves.

I heard him saying "*today, there are millions of young children being born whose technology of self-expression has not yet been invented. We have a moral obligation to invent technology so that every person on the globe has the potential to realize their true difference*."

Forth has the ability to do it's part of this moral obligation for millions of young children who are already born - the Forth technology of self-expression has been invented and has to be fed into practice for this purpose. That's why I am proposing an Education Initiative giving the opportunity to teach and learn Forth in a new, up to now unknown, and easygoing fashion.

We call it 4E4th - Forth 4 Education.

How it began

The starting point – not long ago - was Texas Instruments offering the MSP-EXP430FR5739 Experimenter Board [2], having TI's first commercially available FRAM [3] microcontroller [4] on board. When in 1990 I first read about FRAM [5], I immediately recognized that this is the future of microprocessor memory - like in the seventies when I recognized CMOS, microprocessors, LEDs and LCDs to be the future of electronics.

Now there it was, the Forth Dream Machine, a microcontroller where you didn't have to worry about special programming and saving procedures, the first time that features which make PC programming easy could be transferred to microcontroller programming, especially useful in Forth. Two decades I had been waiting for this to happen.

I am a Forth user, not a Forth system writer. So I asked my friend Michael Kalus to help get a Forth running on this MSP-EXP430FR5739 Experimenter Board, and Michael ported Brad Rodriguez's MSP430 CamelForth [6] - removed the FLASH part and changed everything what was needed to make a pleasant and easy to use Forth out of it, a Forth where you could do incremental programming, no worries needed in case of power outage. When power is back, all data are still there because of the nonvolatile FRAM storing mechanism. Works great!

At that time I tried to transfer an audio project which I had done with TI's MSP430F2012 [7] - for price and power reasons, a consumer gadget - to TI's MSP430G2553 [8], the new "Value Line" [9] microcontroller, which should allow my customer to program some audio features by himself.

The MSP430F2012 I had to program using C - which I didn't like. I had to do so because of memory constraints. Not enough memory for Forth. Now I tried to get my program running on the new MSP430G2553, but to no avail. TI had changed a lot of I/O functions, and even that I only needed basic functions like ADC, SPI, and PWM, I couldn't get it running. And the user programmability demanded to have a Forth kernel. I asked Michael again, this time to port Brad Rodriguez's MSP430 CamelForth to the MSP430G2553.

When done, we recognized that this was the right deal: TI's LaunchPad [10], a small inexpensive board, flashing Forth on it, very useful to teach basic microprocessor training in Forth - a dream Michael and others have had since a while: "Recently we had a meeting in the Lower Rhine region dreaming about a small nice affordable board with a modern MCU and a compact Forth inside" Michael later remembered.

That was just a few weeks before the 2012 annual Forth meeting of the German Forth Chapter, the "Forth-Gesellschaft e.V." Since $4.30 at that time have been 3.40€, we could offer a 4€-Forth selling the LaunchPad, despite the price of $1.20 we had to pay for the MSP430G2553, which was not part of the LaunchPad then.

That's how the idea started: a 4€-Forth.
When ordering our website it was immediately clear that a 4€4th website was not available because of character constraints. So www.4e4th.eu was born, and with it the idea to start an European Education Initiative; 4E4th standing for Forth 4 Education. The Forth meeting at Beukenhof, the Netherlands [11], was a first true European start and a great success. I managed to get fifty LaunchPads from three different sources to be delivered just in time and everybody at this meeting bought a 4€4th board - TI's LaunchPad with 4E4th inside.
To my surprise most of these LaunchPads have been delivered loaded with the MSP430G2553, which we needed. TI had changed the LaunchPad design meanwhile to fit our demands. The luck of the courageous, I would say.
4E4th was born.

4E4th for the LaunchPad - Forth for Education under the Hood

This version of 4E4th for the TI Launchpad with MSP430G2553 is based on CamelForth MSP430 V0.3 [12], created by Brad Rodriguez for TI 's MSP430F1611. He wrote: "This is an ALPHA TEST version of CamelForth/430, an ANSI Standard Forth for the Texas Instruments MSP430 family of microprocessors. This means that I have tested the bulk of this code for correct functioning, but you may still discover bugs. I'd appreciate hearing of any such by email at bj@camelforth.com.... CamelForth should be usable with any MSP430 device having at least 512 bytes of RAM, 8K of ROM, and one USART." [13]

This is the Memory map now:

```
  0000-01FFh: Peripherals
  0200-03FFh: 4E4th RAM, 512 bytes (stacks, buffers, variables)
  1000-10FFh: "Information" Flash ROM (used for special variables)
  C000-DFFFh: Program Flash ROM for new 4E4th definitions
  E000-FFFFh: 4E4th kernel
```

The MSP430 uses a von-Neumann architecture — all program, data memory and peripherals share a common bus structure, consistent CPU instructions and addressing modes are used [14], too, ideally suited for Forth.

New 4E4th definitions are compiled into the Program (Instruction) space. New data structures (e.g., Variables) are allocated in the Data space [15]. 4E4th compiles source code directly into the MSP430's Flash memory.

The project has been created using TI's offer of IAR Kickstart [16].

4E4th is equipped with Autostart. The reset procedure executes BOOT. If there is a valid application it will start with this application. Otherwise WIPE is executed which cleans the FLASH to have a fresh start.

After executing SAVE the Forth state will survive a reset. Pressing and holding button S2 during reset will force a WIPE. This way 4E4th can be resurrected from a deadlock [17].

First Experience with starting 4E4th

The first steps to get 4E4th onto TI's LaunchPad have been using IAR's Embedded Workbench Kickstart, a task to meticulously follow Brad's eleven steps [15] from starting the Workbench to downloading 4E4th onto the target board.

Everybody who is following this way has to install IAR's Embedded Workbench Kickstart, a huge monster, occupying nearly one Gigabyte of disk space altogether. People who don't know 4E4th have to do this anyway [18].

Looking for less space and time consuming opportunities Michael discovered Elpotronics free FET-Pro430 Lite Software [19] and furnished the 4e4th.a43 file which reduced the LaunchPad flashing to less than a minute. To be able to do so, a special TI software from another TI source has to be started to prepare the USB to recognize the LaunchPad. Still a to do list.

We have been looking for ways to streamline the whole process, from flashing 4E4th to TI's MSP430G2553 LaunchPad, over getting interactively connected to 4E4th on the LaunchPad hardware; typing words, downloading 4th-files, and even starting and maintaining 4th-Projects. That's where our 4E4th Terminal-IDE comes in.

Four Steps towards the 4E4th Terminal-IDE

The 4E4th Terminal-IDE is the result of nearly thirty years of programming experience, using Forth in different projects and writing tools over this time to make life easier.

I have to share a short summary of circumstances which led to the idea for the 4E4th Terminal-IDE.

Step 1: Writing a virtual microprocessor engine

In 1979, using a KIM in Euro PCB format, I started a project with video output and database features [20].

In those times I wrote my programs with paper and pencil, reading each command from an Instruction Set Summary Card and using a hex key pad to get these commands into the microprocessors memory. To make complex programming easier, I always wrote little subroutines that could be tested easily and complete (years later I learned this is called structured programming [21]), so the video output and database routines have been a collection of subroutines. Typing each subroutine with six keystrokes soon was boring and I looked for better and faster ways to get this done.

Nearly two hundred subroutines had been accumulated, and easily I could write a table with all these starting addresses - reducing the number of keystrokes in half - because now I only had to type the number of the subroutine without adding any command. A little virtual engine took care of taking the subroutine and executing. That worked really great and totally fast. This was the start to write my own programming environment.

But then the August 1979 BYTE magazine featuring Forth [22] came in and I recognized: there it is, already done, and even better than mine, using words instead of numbers.

It took five years until I got my first Forth.

Step2: Starting with RSC-Forth

In 1984 I started with RSC-Forth [23], and it worked from start as expected.

Randy Dumse [24] had done a great job.

With RSC-Forth I was still able to work in a system I was used to, but in a much better way. RSC-Forth made it possible to add a 3” floppy while developing. I wrote my first own tool, a screen editor, not to depend on the command line input RSC-Forth provided with ending up typing lines again and again after typos.

There is only one thing with Forth which always bothered me:

There is a direct input mode, typing words and definitions directly to test these words, and then I had to type these definitions into the editor again to be saved and accumulated for the final download finishing the application. Doubling the workload again.

Having my first laptop some years later made the in system 3” floppy obsolete and made a safe download possible. Checking each byte echoed by RSC-Forth and stopping download when an error occurred, gave the possibility to jump automatically into the editor just at the line and character where the download error came up. This special communication program I had written in F-PC.

Step3: Fulfilling a dream: visualFORTH

When I worked for Lucent Technologies in 2000, I had to use Visual Basic [25] to write a programming tool for their production. To me VB was fascinating, different to plain text programming which I was used to. But it was not Forth, and therefore after a while it was boring.

Nearly ten years I waited for a visual Forth, not finding anything like this. There was a tool, associated with Win32Forth [26] - named ForthForm [27]. It was designed to make GUIs – which was a huge task, I am sure of that – but I didn't see how to get something working with this. I didn't recognize how to make a button to do some work when clicking on it.

So I started to upgrade ForthForm to make it usable for me and others [28], finally naming the result “visualFORTH” [29] - but that's another story.

VisualFORTH was the last step needed to make the 4E4th Terminal-IDE possible.

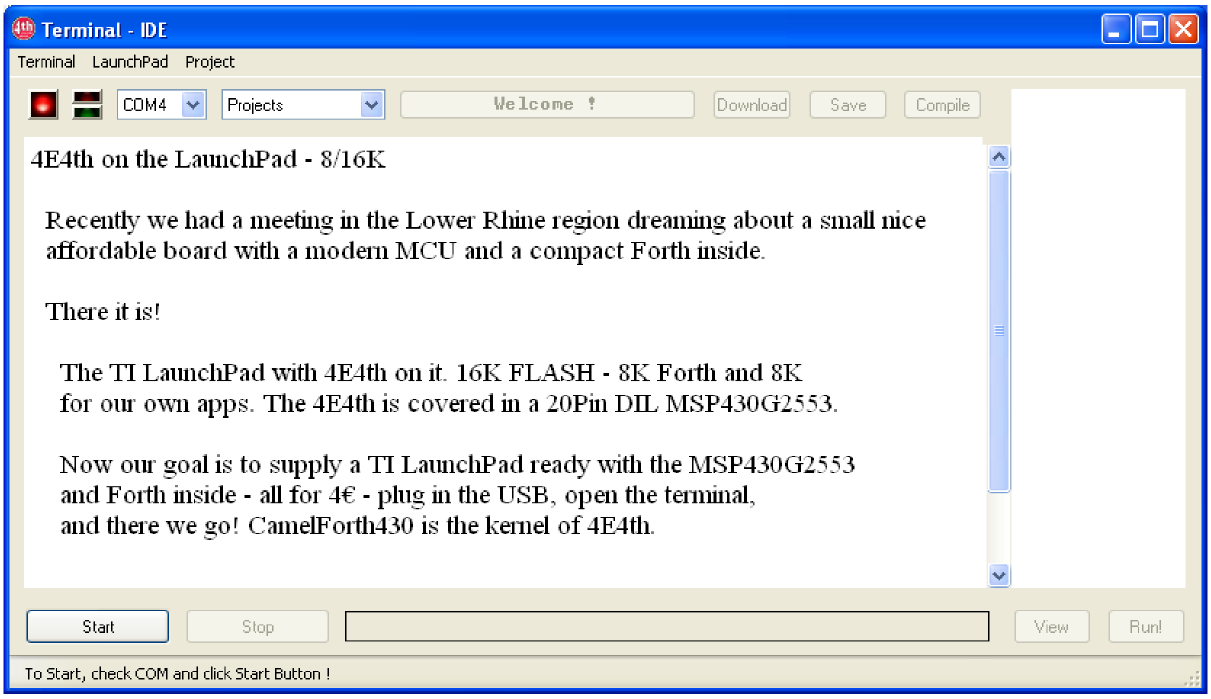
Step4: The 4E4th Terminal-IDE

At this point I will switch to a series of pictures, because “A picture is worth a thousand words” [30], they say.

There is one simple reason to do so: to show the simplicity of an IDE tool specially developed for Education in mind, reducing learning time to a minimum. The focus of the user should be on gaining experience in programming, avoiding all unnecessary distractions.

Starting the Terminal-IDE

The 4E4th Terminal-IDE needs no installation. Unpack and start right away!



4E4th on the LaunchPad - 8/16K

Recently we had a meeting in the Lower Rhine region dreaming about a small nice affordable board with a modern MCU and a compact Forth inside.

There it is!

The TI LaunchPad with 4E4th on it. 16K FLASH - 8K Forth and 8K for our own apps. The 4E4th is covered in a 20Pin DIL MSP430G2553.

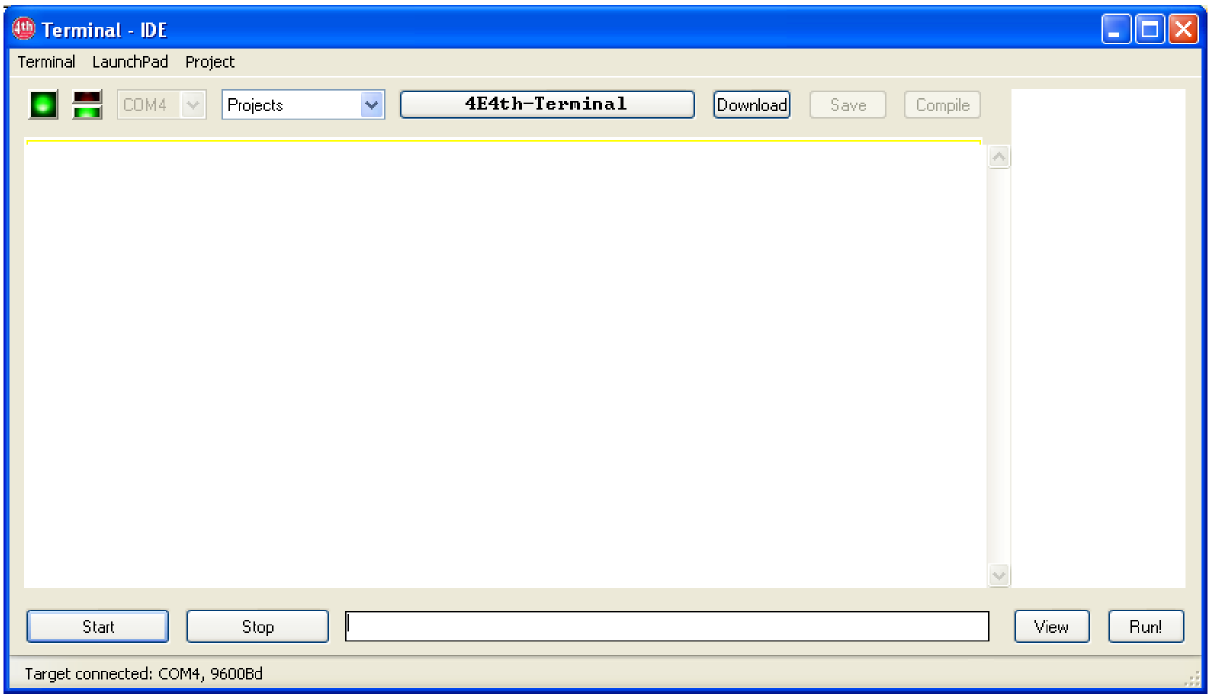
Now our goal is to supply a TI LaunchPad ready with the MSP430G2553 and Forth inside - all for 4€ - plug in the USB, open the terminal, and there we go! CamelForth430 is the kernel of 4E4th.

To Start, check COM and click Start Button !

The first appearance of the 4E4th Terminal-IDE is mainly a screen and a few buttons, following Dick Pountain's philosophy of “Simpleware” [31]: “Buttons … but no more than six of them.” And: “Things that are not active should go grey: live things go black again.”

At start a red LED and only one button to start with:

Now the target is connected:

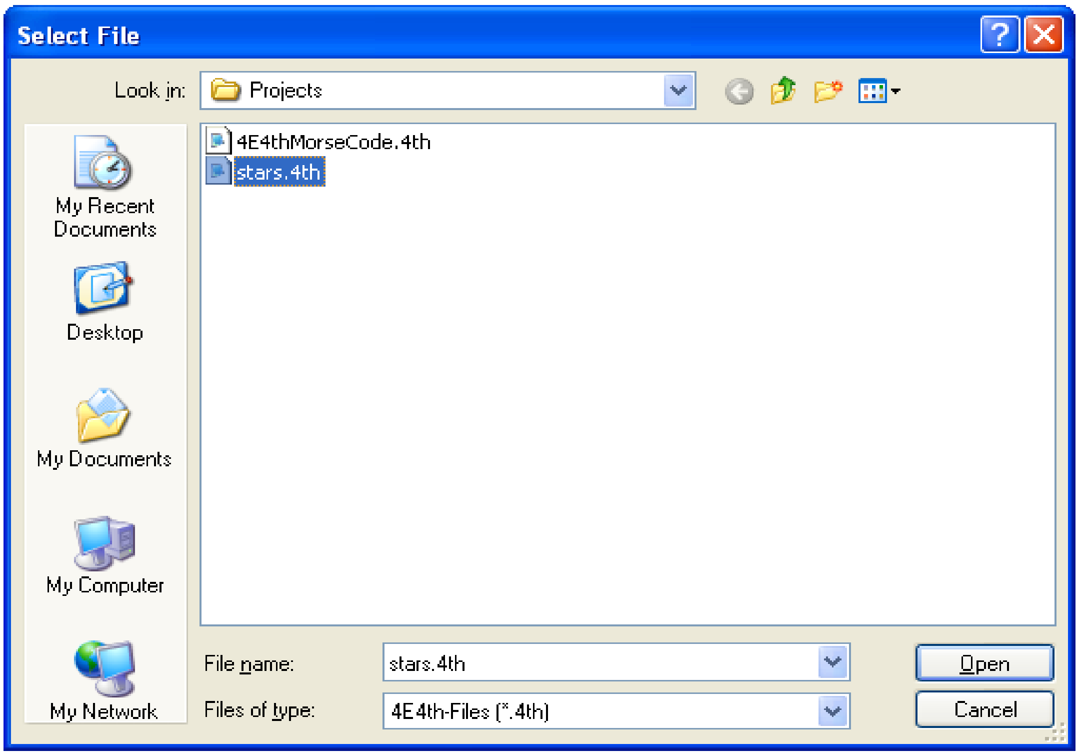


Target connected: COM4, 9600Bd

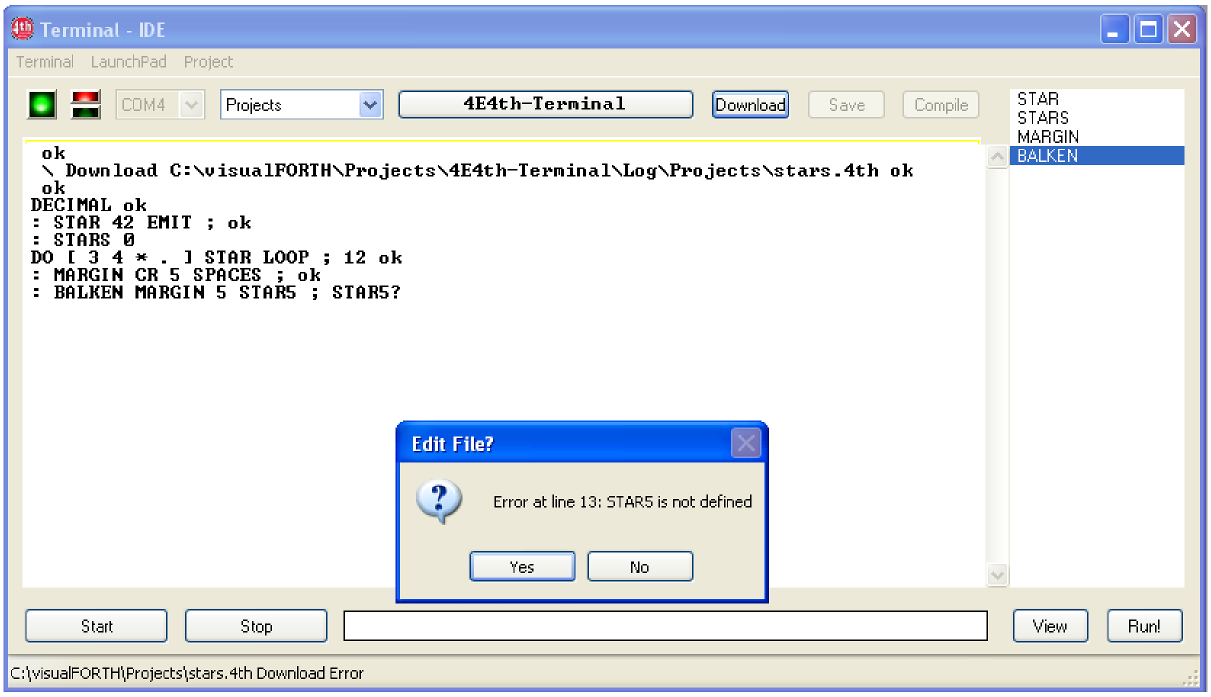
The LaunchPad USB Interface is always set to 9600Bd, 8N1. 4E4th Terminal-IDE checks this connection and connects to the COM port used by this USB.

Downloading a File

Click on “Download” opens a file-selection window, downloads with click on “Open”:



Download started. An error occurred while downloading:

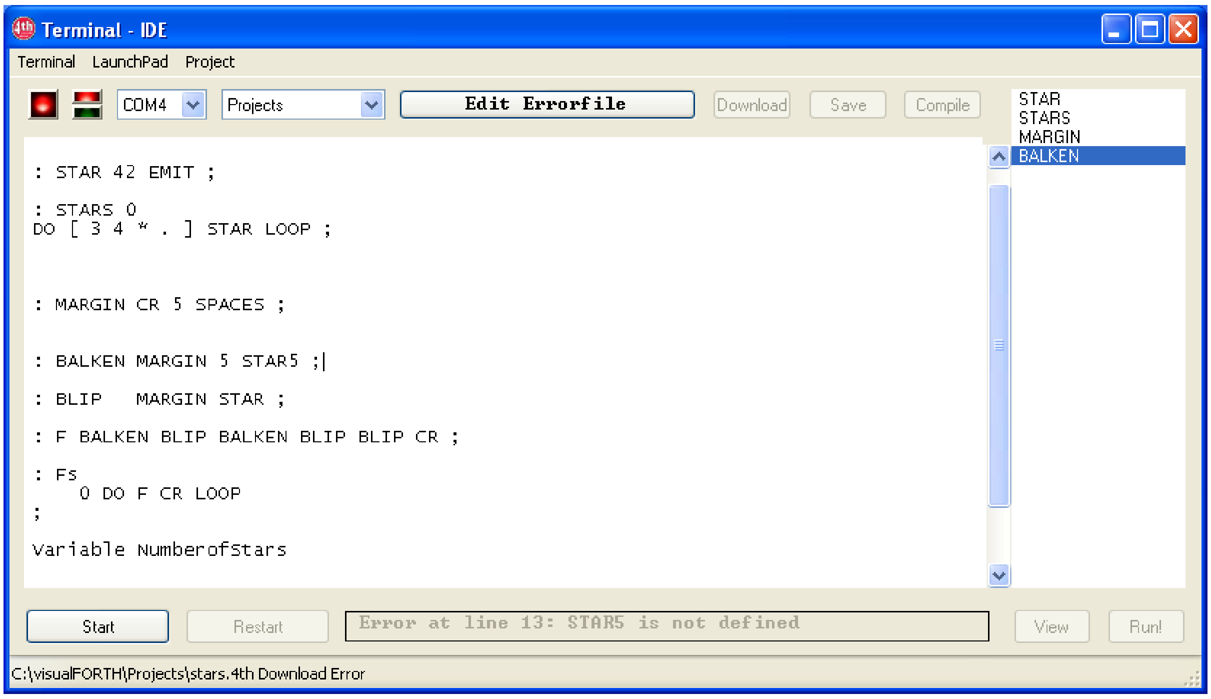


```
 ok
 \ Download C:\visualFORTH\Projects\4E4th-Terminal\Log\Projects\stars.4th ok
 ok
DECIMAL ok
: STAR 42 EMIT ; ok
: STARS 0
DO [ 3 4 * . ] STAR LOOP ; 12 ok
: MARGIN CR 5 SPACES ; ok
: BALKEN MARGIN 5 STAR5 ; STAR5?
```

Error at line 13: STAR5 is not defined

C:\visualFORTH\Projects\stars.4th Download Error

A click on “Yes” switches to “Edit Errorfile” mode, the cursor is at the error line:

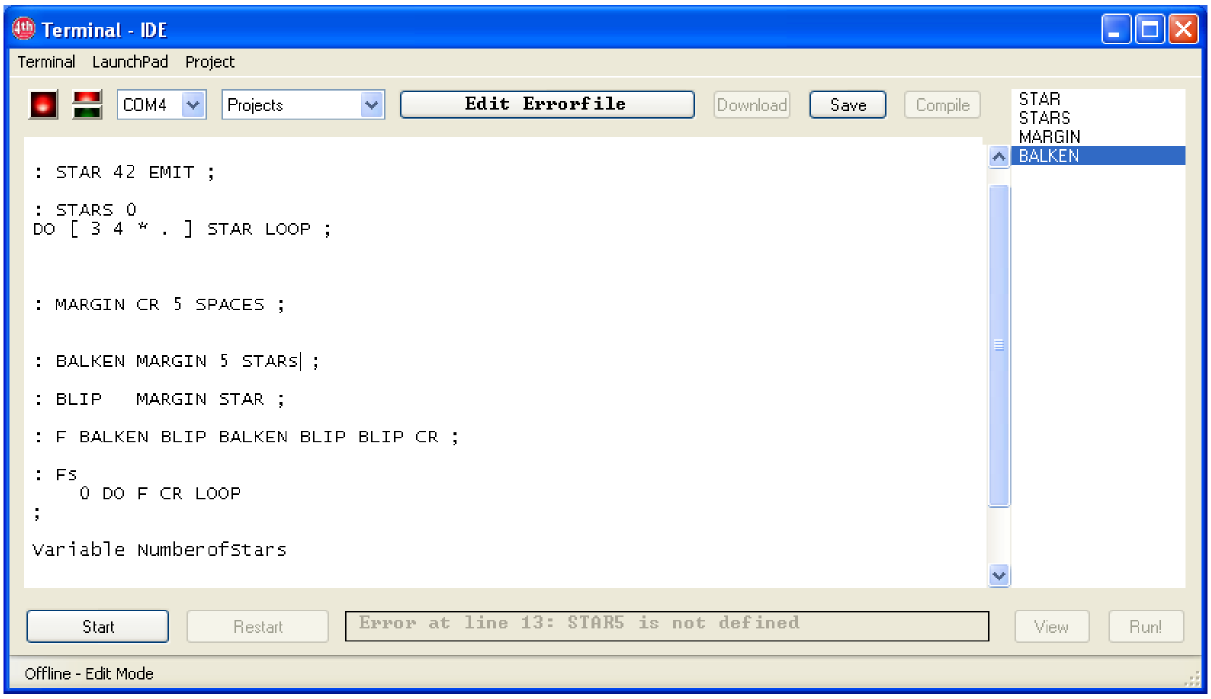


```
: STAR 42 EMIT ;

: STARS 0
DO [ 3 4 * . ] STAR LOOP ;


: MARGIN CR 5 SPACES ;


: BALKEN MARGIN 5 STAR5 ;|

: BLIP   MARGIN STAR ;

: F BALKEN BLIP BALKEN BLIP BLIP CR ;

: Fs
    0 DO F CR LOOP
;

Variable NumberofStars
```

Error at line 13: STAR5 is not defined

C:\visualFORTH\Projects\stars.4th Download Error

“Download”, “Save”, and “Compile” button are disabled. The 4E4th Terminal-IDE waits for error correction, so something has to be typed in to make the “Save” button work. The best thing of course is to correct this line:



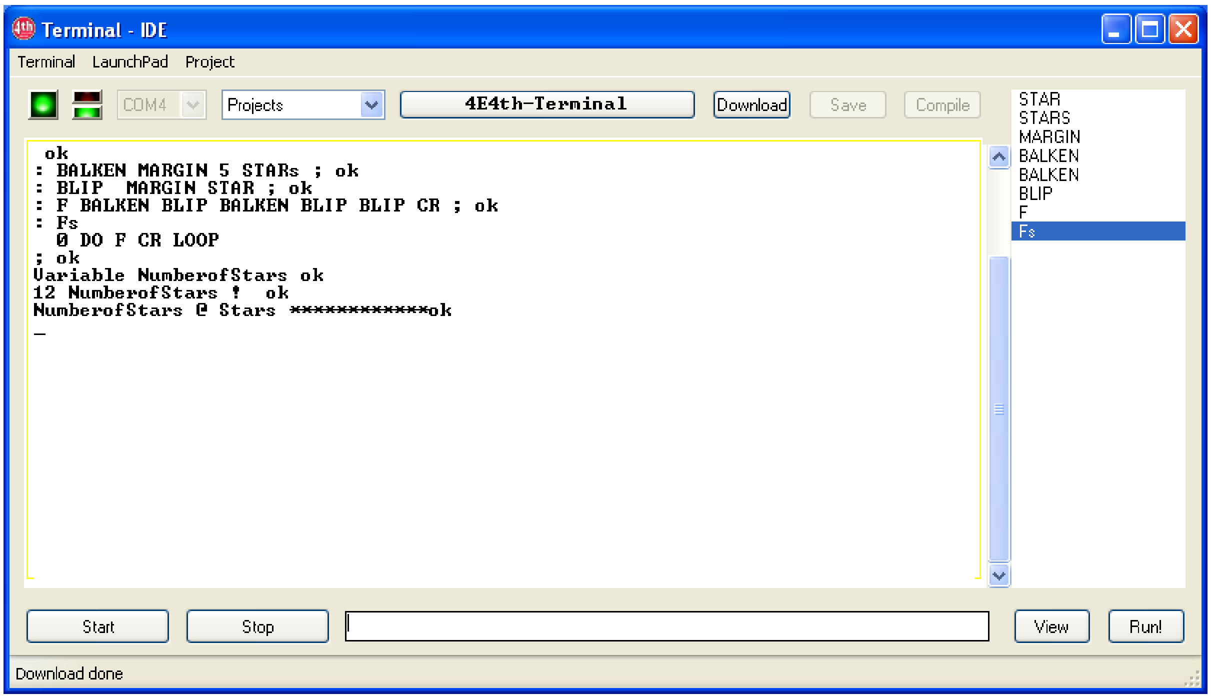
```
: STAR 42 EMIT ;

: STARS 0
DO [ 3 4 * . ] STAR LOOP ;


: MARGIN CR 5 SPACES ;


: BALKEN MARGIN 5 STARS| ;

: BLIP   MARGIN STAR ;

: F BALKEN BLIP BALKEN BLIP BLIP CR ;

: Fs
    0 DO F CR LOOP
;

Variable NumberofStars
```

Error at line 13: STAR5 is not defined

Offline - Edit Mode

When the correction is done, a click on “Save” saves this file, and a click on “Compile” downloads the needed parts of this file to the target to be compiled.

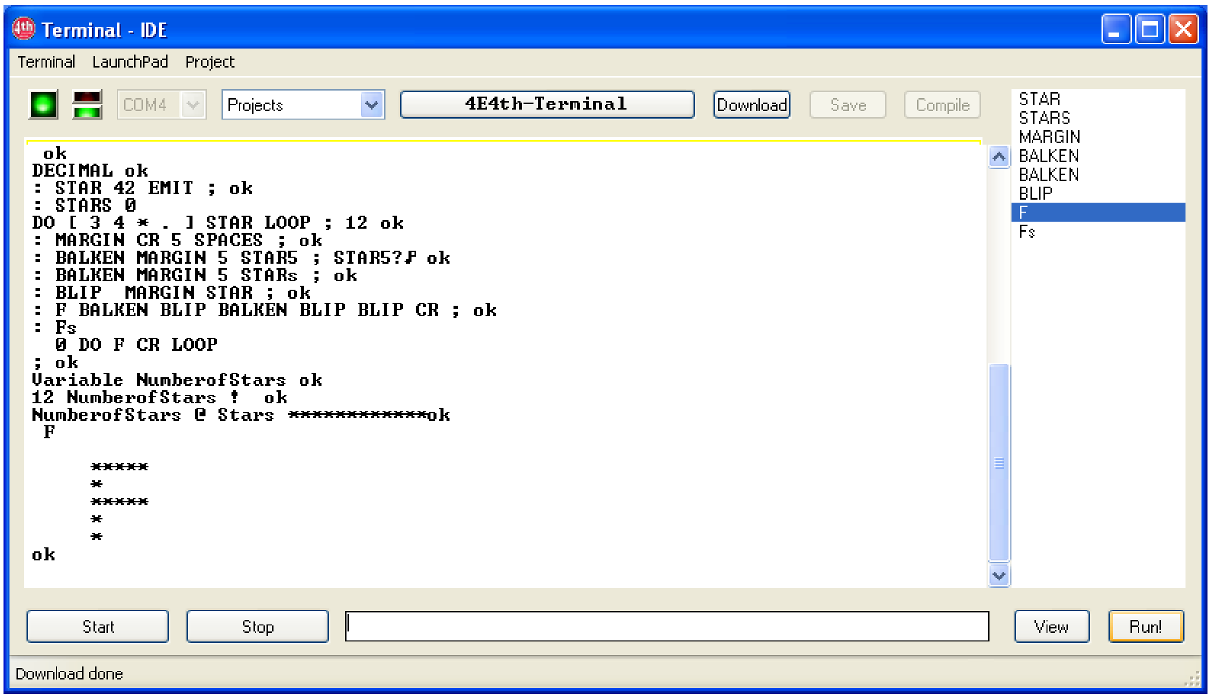
The 4E4th Terminal-IDE checks the last valid download line, and the download proceeds with the corrected definition, time saving real incremental compiling:



```
 ok
: BALKEN MARGIN 5 STARs ; ok
: BLIP  MARGIN STAR ; ok
: F BALKEN BLIP BALKEN BLIP BLIP CR ; ok
: Fs
  0 DO F CR LOOP
; ok
Variable NumberofStars ok
12 NumberofStars !  ok
NumberofStars @ Stars ************ok

_
```

Running a Program

Now we can test the program by highlighting the function to be tested and click on "Run!"

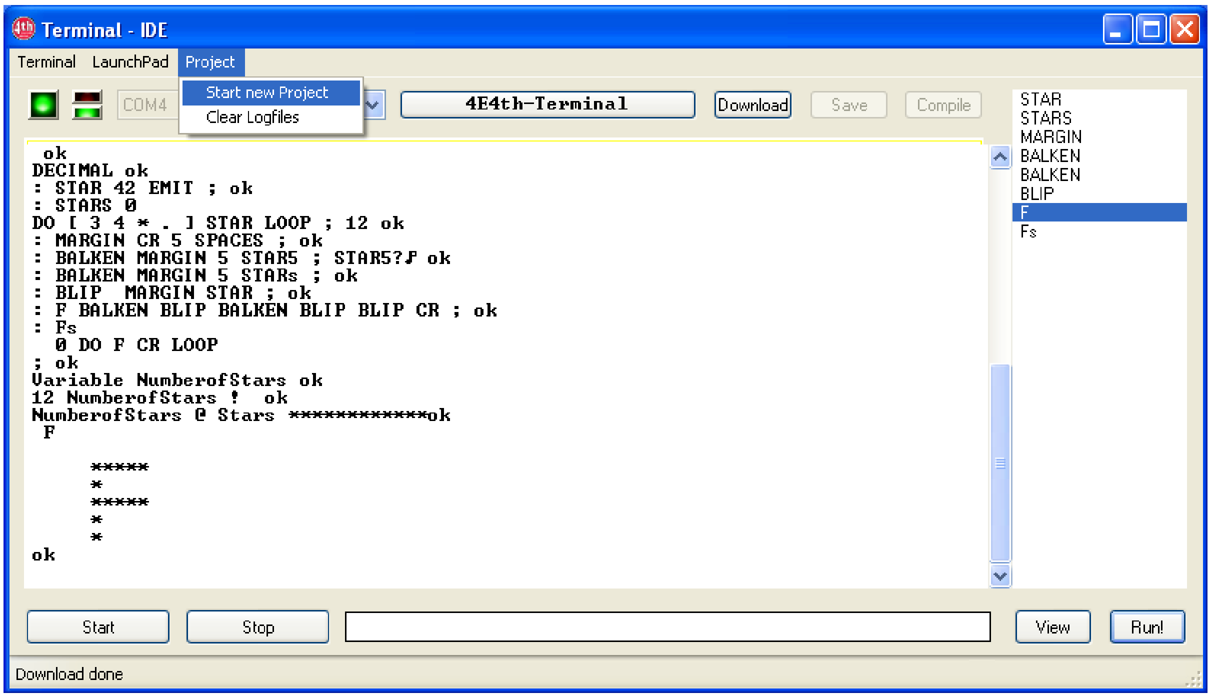


```
 ok
DECIMAL ok
: STAR 42 EMIT ; ok
: STARS 0
DO [ 3 4 * . ] STAR LOOP ; 12 ok
: MARGIN CR 5 SPACES ; ok
: BALKEN MARGIN 5 STAR5 ; STAR5?♪ ok
: BALKEN MARGIN 5 STARs ; ok
: BLIP  MARGIN STAR ; ok
: F BALKEN BLIP BALKEN BLIP BLIP CR ; ok
: Fs
  0 DO F CR LOOP
; ok
Variable NumberofStars ok
12 NumberofStars !  ok
NumberofStars @ Stars ************ok
 F

     *****
     *
     *****
     *
     *
ok
```

Voilà!

Starting a new Project

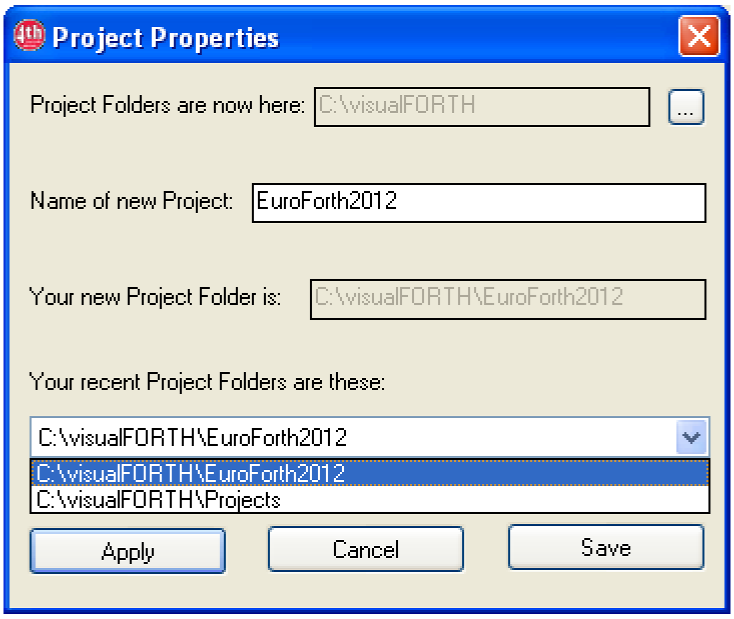
Like to start another project? Nothing simpler than that!
Dick Pountain wrote “Menus are poison”, but sometimes they are really useful:



```
 ok
DECIMAL ok
: STAR 42 EMIT ; ok
: STARS 0
DO [ 3 4 * . ] STAR LOOP ; 12 ok
: MARGIN CR 5 SPACES ; ok
: BALKEN MARGIN 5 STAR5 ; STAR5?♪ ok
: BALKEN MARGIN 5 STARs ; ok
: BLIP  MARGIN STAR ; ok
: F BALKEN BLIP BALKEN BLIP BLIP CR ; ok
: Fs
  0 DO F CR LOOP
; ok
Variable NumberofStars ok
12 NumberofStars !  ok
NumberofStars @ Stars *************ok
 F

     *****
     *
     *****
     *
     *
ok
```

Download done

To start a new project, simply use the “Project” menu, click on “Start new Project” and type the name of your new project into the blank line of the “Project Properties” pop up:



Project Folders are now here: C:\visualFORTH

Name of new Project: EuroForth2012

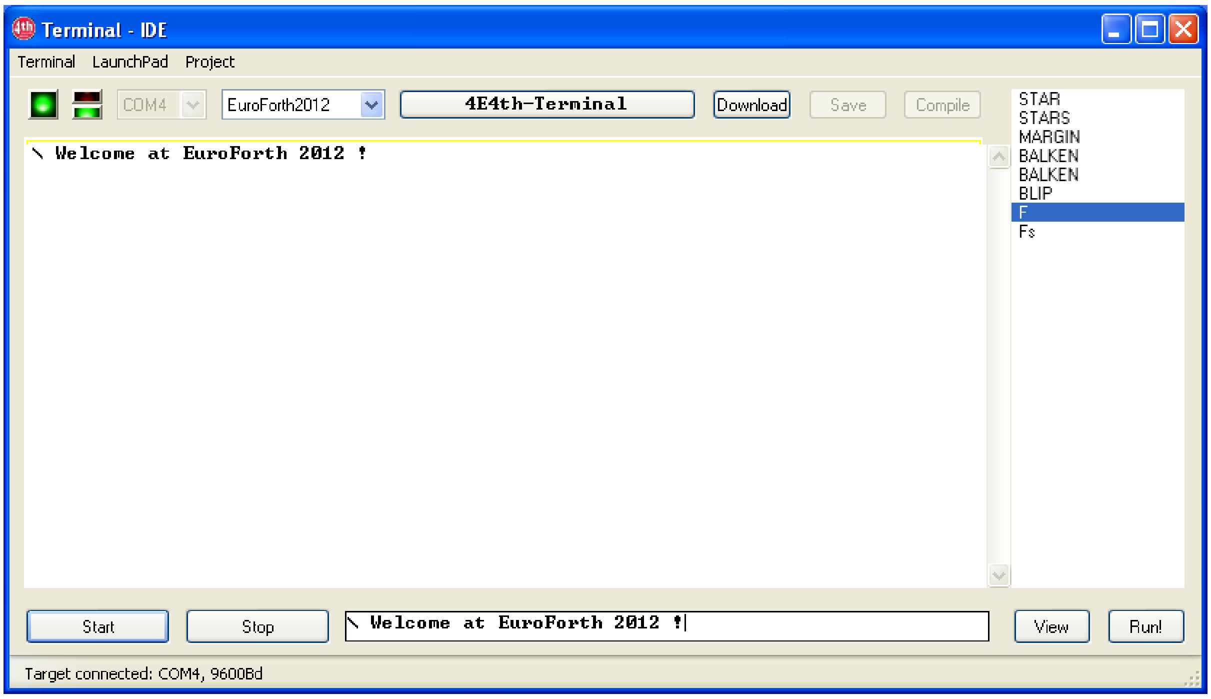
Your new Project Folder is: C:\visualFORTH\EuroForth2012

Your recent Project Folders are these:

C:\visualFORTH\EuroForth2012
C:\visualFORTH\EuroForth2012
C:\visualFORTH\Projects

Click on “Apply” and “Save”, and 4E4th Terminal-IDE starts your new project with a blank screen, ready to type in commands for the microprocessor.

Only a comment is shown here: “Welcome at EuroForth 2012”



```
\ Welcome at EuroForth 2012 !
```

```
\ Welcome at EuroForth 2012 !
```

Target connected: COM4, 9600Bd

The definitions are still there, as long as they are not wiped off.

To me the fascinating thing is that the user doesn't need to care about files.
The 4E4th Terminal-IDE takes care of all project files.
Nothing gets lost, and all previous works and changes are stored at a safe place.
A session may be interrupted and started at the same point again another day.

And imagine, the 4E4th Terminal-IDE occupies only 1MB disk space, including all necessary software to flash a new LaunchPad – the IAR-IDE, which is recommended to use with the MSP430, needs 1000 times as much!

This is only the beginning. Forth gives the ability to add other features over time. Learning with the user the toolmaker is able to sharpen his tools.

Conclusions

This experience report, including some technological review, has been done to make aware of some very important problems:

Knowledge accumulated over the centuries is diminishing from the conscience of people and accumulating nearly only on computer memories.

More and more programmers are coming directly from school or university, not having much experience of real life. They are used to producing wonderful pictures on computer screens but not caring about the experience of users who like to get a task done on a computer instead of waiting for animations to be downloaded.
That's my observation.

But the main thing is that these programmers have to be elite programmers because of the complexity of the products they are working on. My observation is that the number of elite programmers doesn't grow – the crew of elite programmers moves forward with the advances of technologies - leaving the others behind.

No one should be left behind. My goal is to give ordinary people the chance to do wonderful works of creation by themselves by offering a tool and a programming language which is reduced to a minimum of effort to use it.

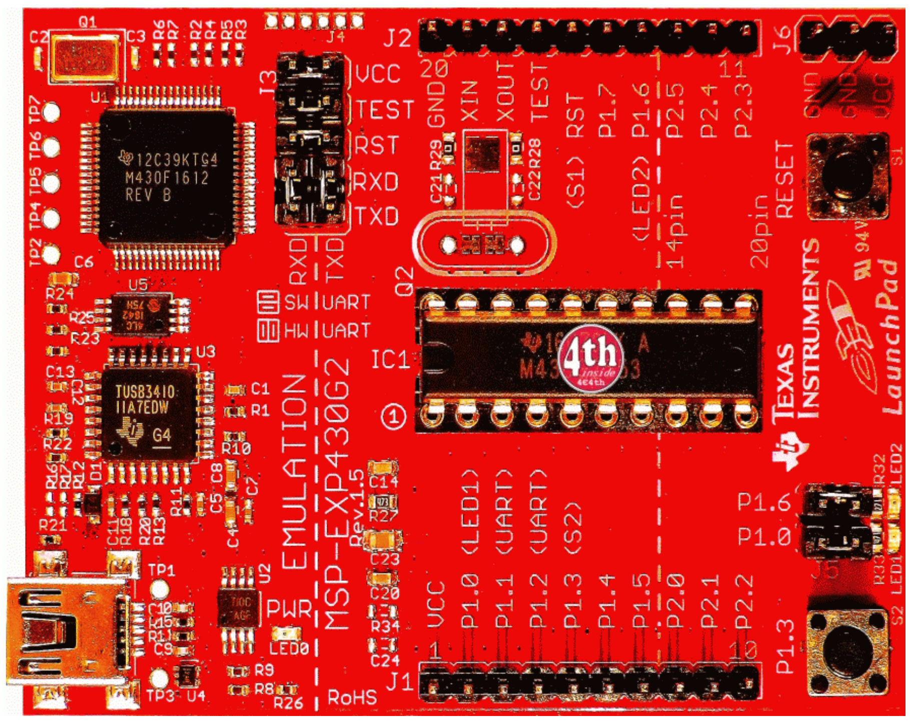
Forth is the programming language which stripped off all unnecessary attachment and distraction leaving pure logic to work with. Our 4E4th Terminal-IDE replicates that tradition, making real programming fun and easy.

I encourage everybody to think about that. Our approach is not patented, it is free to be copied. The intended goal to help young people to achieve self realization by doing great things should always be in mind. I don't have any idea what they will do with it - I trust our new generations to do the best for mankind. For mankind to survive, for example. Only Forth programs have the inherent ability to be readable and understandable even after centuries.

With 4€ they may start – here in the heart of Europe.

A bunch of experiments with 4E4th are offered at http://www.forth-ev.de/wiki/doku.php/projects:4e4th:4e4th:start:msp430g2553_experimente (waiting for translation).

It was never more affordable and easier to start a programming education – self education or school education. This situation should be taken advantage of by delivering the right tools to enable programming education, discarding all unnecessary obstacles, to make it a Newbees [32] delight.

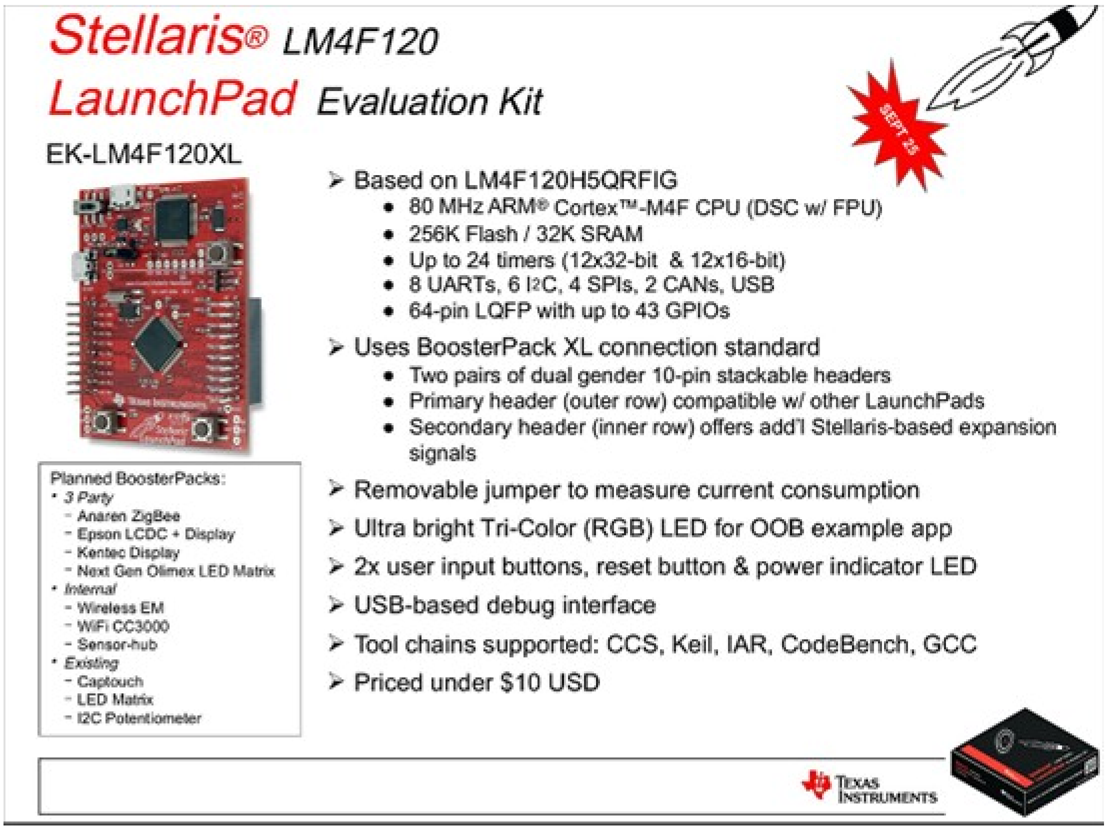


LaunchPad
with 4E4th

A next Step

Here a small glimpse into the future:

TI's Stellaris LaunchPad at the promotional price of $4.99 [33] – another 4€-Forth promotional opportunity - is coming soon, announced for taking regular orders on September 25th – ten days from today:



Stellaris® LM4F120
LaunchPad Evaluation Kit

EK-LM4F120XL

SEPT 25

- Based on LM4F120H5QRFIG
  - 80 MHz ARM® Cortex™-M4F CPU (DSC w/ FPU)
  - 256K Flash / 32K SRAM
  - Up to 24 timers (12x32-bit & 12x16-bit)
  - 8 UARTs, 6 I2C, 4 SPIs, 2 CANs, USB
  - 64-pin LQFP with up to 43 GPIOs
- Uses BoosterPack XL connection standard
  - Two pairs of dual gender 10-pin stackable headers
  - Primary header (outer row) compatible w/ other LaunchPads
  - Secondary header (inner row) offers add'l Stellaris-based expansion signals
- Removable jumper to measure current consumption
- Ultra bright Tri-Color (RGB) LED for OOB example app
- 2x user input buttons, reset button & power indicator LED
- USB-based debug interface
- Tool chains supported: CCS, Keil, IAR, CodeBench, GCC
- Priced under $10 USD

Planned BoosterPacks:
- *3 Party*
  - Anaren ZigBee
  - Epson LCDC + Display
  - Kentec Display
  - Next Gen Olimex LED Matrix
- *Internal*
  - Wireless EM
  - WiFi CC3000
  - Sensor-hub
- *Existing*
  - Captouch
  - LED Matrix
  - I2C Potentiometer

Texas Instruments

Let's do the right steps now. Let's start a real **Forth Programming Education Initiative** to reach out to young people, even kids, giving them the opportunity to learn how easy they will be able to express themselves and unfold their abilities. To quote Andrew Reid [34]: “With your 4E4th we can show people how clever they can become.”

One last word:

Kevin Kelly [1] stated another remarkable truth: "Technologies don't die."

So relax, Forth will live forever.

Acknowledgements

My special thanks go to Michael Kalus, whose efforts brought 4E4th into life, and of course to Brad Rodriguez, who created CamelForth, upon which 4E4th was build.

A praise to Texas Instruments, especially to the people in Freising [35], Germany, who created these extraordinary power saving MSP430 microprocessors.

A great thank you of course to Chuck Moore, the genius behind all Forth inventions, and many thanks to Andrew Reid, who encouraged me to present this paper.

Many thanks to the Win32Forth crew, especially to Tom Zimmer, Ezra Boyce, Jos van de Ven, Camille Doiteau, Dirk Busch, Rod Oakford for preparing the ground to build 4E4th Terminal-IDE on.

And last, but not least, many thanks to Barbara Bruehl, who supported my endeavor, especially with proof reading.

References

[1] Kevin Kelly: How technology evolves, TED TALKS, 2006
http://www.ted.com/talks/kevin_kelly_on_how_technology_evolves.html

[2] MSP-EXP430FR5739 Experimenter Board, Texas Instruments Inc., 2012
http://www.ti.com/tool/msp-exp430fr5739

[3] Ferroelectric RAM, WIKIPEDIA, 2012
http://en.wikipedia.org/wiki/Ferroelectric_RAM

[4] F2MC-8FX Family FRAM Microcontrollers, Fujitsu, 2009
http://www.fujitsu.com/downloads/EDG/binary/pdf/find/27-1e/1.pdf

[5] Ramtron International, WIKIPEDIA, 2012
http://en.wikipedia.org/wiki/Ramtron

[6] CamelForth, Bradford J. Rodriguez, 2009
http://www.camelforth.com/page.php?8

[7] MSP430F2012 MIXED SIGNAL MICROCONTROLLER, Texas Instr. Inc., 2011
http://www.ti.com/lit/ds/symlink/msp430f2012.pdf

[8] MSP430G2553, Texas Instruments Inc., 2012
http://www.ti.com/product/msp430g2553

[9] MSP430 LaunchPad Value Line Development kit, Texas Instruments Inc., 2012
http://www.ti.com/tool/msp-exp430g2

[10] MSP430 LaunchPad (MSP-EXP430G2), Texas Instruments Inc., 2012
http://processors.wiki.ti.com/index.php?title=MSP430_LaunchPad_(MSP-EXP430G2)

[11] Verslag Jaarvergadering Duitse Forth Vereniging, HCC!Forth, 2012
http://www.forth.hcc.nl/w/Nieuws/Nieuws

[12] MSP430 CamelForth version 0.3, B. J. Rodriguez, 2009
http://www.camelforth.com/download.php?view.12

[13] CamelForth/MSP430, B. J. Rodriguez, 2009
http://www.camelforth.com/page.php?8

[14] MSP430 Ultra-Low-Power MCUs, Texas Instruments Inc., 2002
http://www.ti.com/sc/docs/products/micro/msp430/msp430bulletin2.pdf

[15] CAMELFORTH FOR THE MSP430, Bradford J. Rodriguez, 2009
http://www.forth-ev.de/repos/4e4th/readme.430

[16] IAR Embedded Workbench Kickstart - Free 4KB IDE, Texas Instr. Inc., 2012
http://www.ti.com/tool/iar-kickstart&DCMP=MSP430&HQS=Other+OT+iarkickstart

[17] 4E4TH FOR THE MSP430G2553 on LaunchPad, Michael Kalus, 2012
http://www.forth-ev.de/repos/4e4th/README_4e4th.txt

[18] IAR Embedded Workbench for TI MSP430, Texas Instruments Inc., 2012
http://processors.wiki.ti.com/index.php/IAR_Embedded_Workbench_for_TI_MSP430

[19] Flash Programmers for TI's MSP430 MCU, Elpotronic Inc., 2012
http://www.elprotronic.com/download.html

[20] Rechnergesteuerte Funkalarmierung mit dem neuen FAS, Erich Bumiller, 1979
http://www.somersetweb.com/BruehlConsult/Projekte/FAS.html

[21] Notes on Structured Programming, Prof.dr. Edsger W. Dijkstra, 1970
http://www.cs.utexas.edu/users/EWD/ewd02xx/EWD249.PDF

[22] Threads of a FORTH Tapestry, Gregg Williams, BYTE, August 1980
http://www.colorforth.com/byte.htm

[23] RSC-FORTH User's Manual, Rockwell International, October 1983
http://www.smallestplcoftheworld.org/RSC-FORTH_User's_Manual.pdf

[24] The R65F11 and F68K Single-Chip Forth Computers, Randy M. Dumse,
The Journal of Forth Application and Research Volume2, Number 1, 1984
http://soton.mpeforth.com/flag/jfar/vol2/no1/article1.pdf

[25] Visual Basic, WIKIPEDIA 2012
http://en.wikipedia.org/wiki/Visual_Basic

[26] Win32Forth, Dirk Busch, 2007
http://win32forth.sourceforge.net/doc/p-index.htm

[27] Developmental History for ForthForm, Ezra Boyce, 2007
http://win32forth.sourceforge.net/doc/ForthForm/FF-History.htm

[28] visualFORTH, Dirk Bruehl, 2010
http://www.visualforth.blogspot.com/

[29] visualFORTH, Dirk Bruehl, 2010
http://www.visualforth.org/

[30] A picture is worth a thousand words, The Phrase Finder, 2012
http://www.phrases.org.uk/meanings/a-picture-is-worth-a-thousand-words.html

[31] Simpleware Principles, Dick Pountain, 2009
https://sites.google.com/site/dickpountainspages/home/simpleware

[32] Newbie Variants, WIKIPEDIA,2012
http://en.wikipedia.org/wiki/Newbie#Variants

[33] The Stellaris® ARM® Cortex™-M4F LaunchPad, Texas Instruments Inc., 2012
http://www.ti.com/ww/en/launchpad_site/stellaris.html?DCMP=stellaris-launchpad&HQS=stellaris-launchpad

[34] ThermoSense MK1, Andrew Reid, 2012
http://www.sustainabilitymeasurement.com

[35] TI Germany - Freising, Texas Instruments Inc., 2008
http://www.ti.com/europe/docs/sites/germany.htm