Forth on microcontrollers

From the minicomputer to the microcontroller

Jan Bramkamp, 2020-05-08

What defines Forth

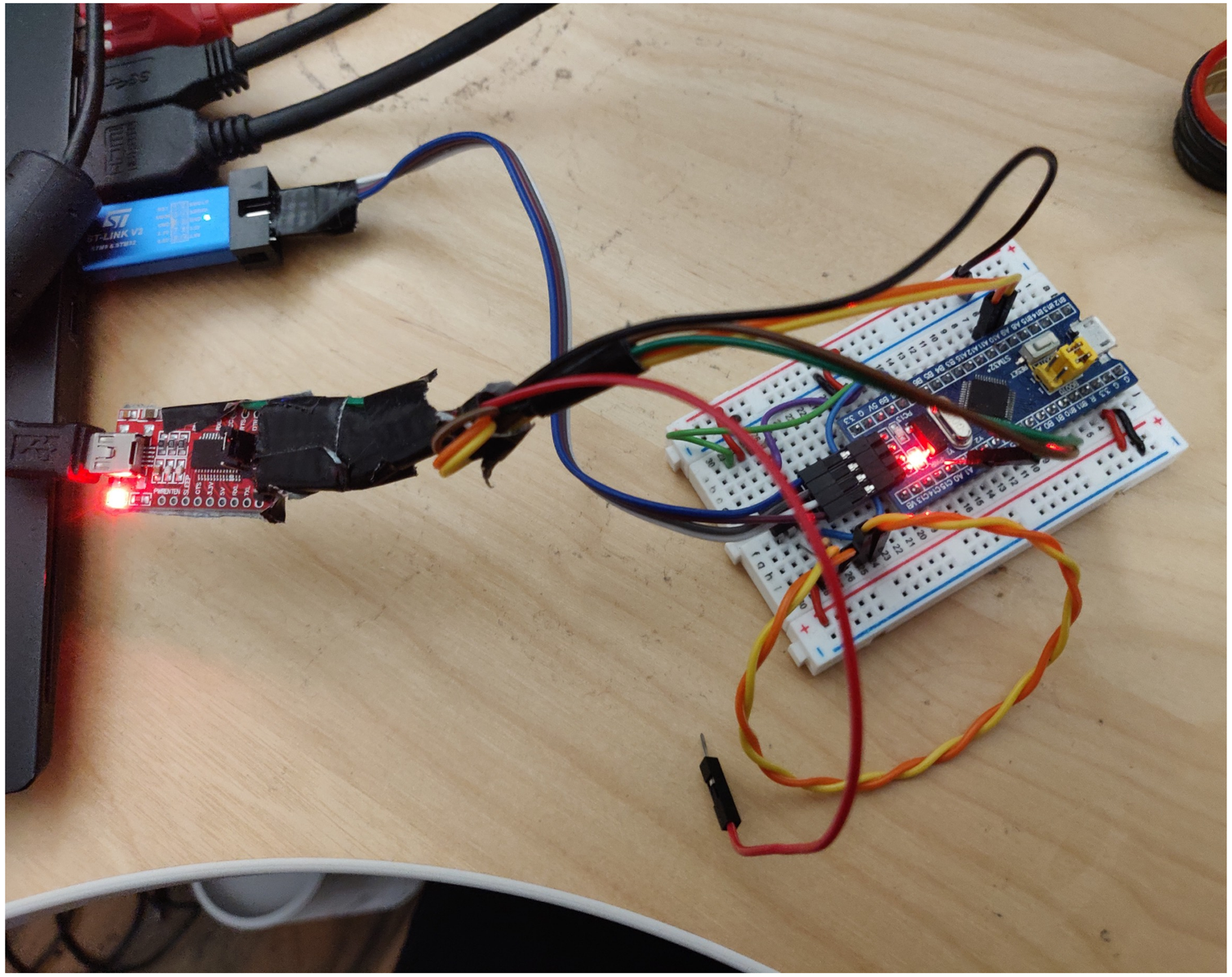
Extending the compiler

- interactive development
- bottom up programming
- lots of refactoring
- grow the language to fit your problem
  - shorter programs
  - avoid overgeneralisation

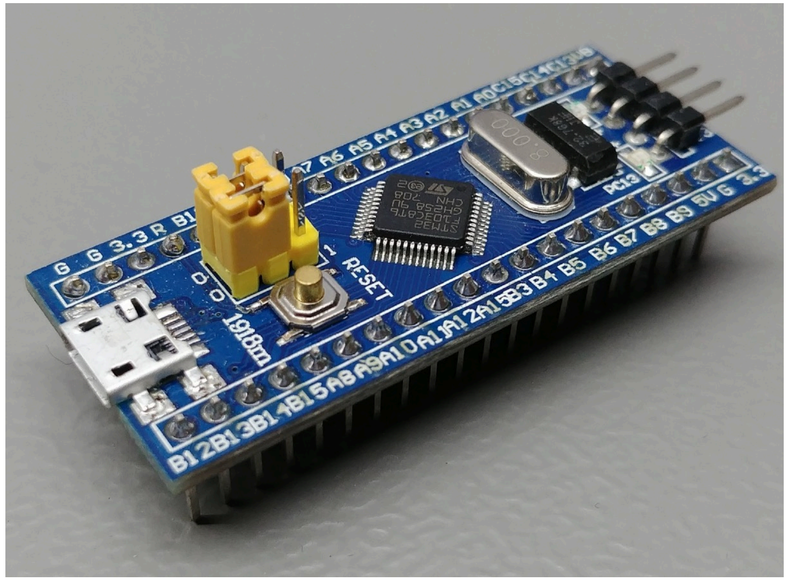
Hardware requirements

The Agony of Choice

- required:
  - a MCU board
  - a compatible Programmer/Debugger
  - a compatible USB<->RS232 converter
  - Jump wires
- really nice to have: logic analyzer, multimeter




ST-LINK V2
STM8 & STM32
RST
GND
SWIM
3.3V
5.0V
SWDIO
GND
SWCLK
3.3V
5.0V



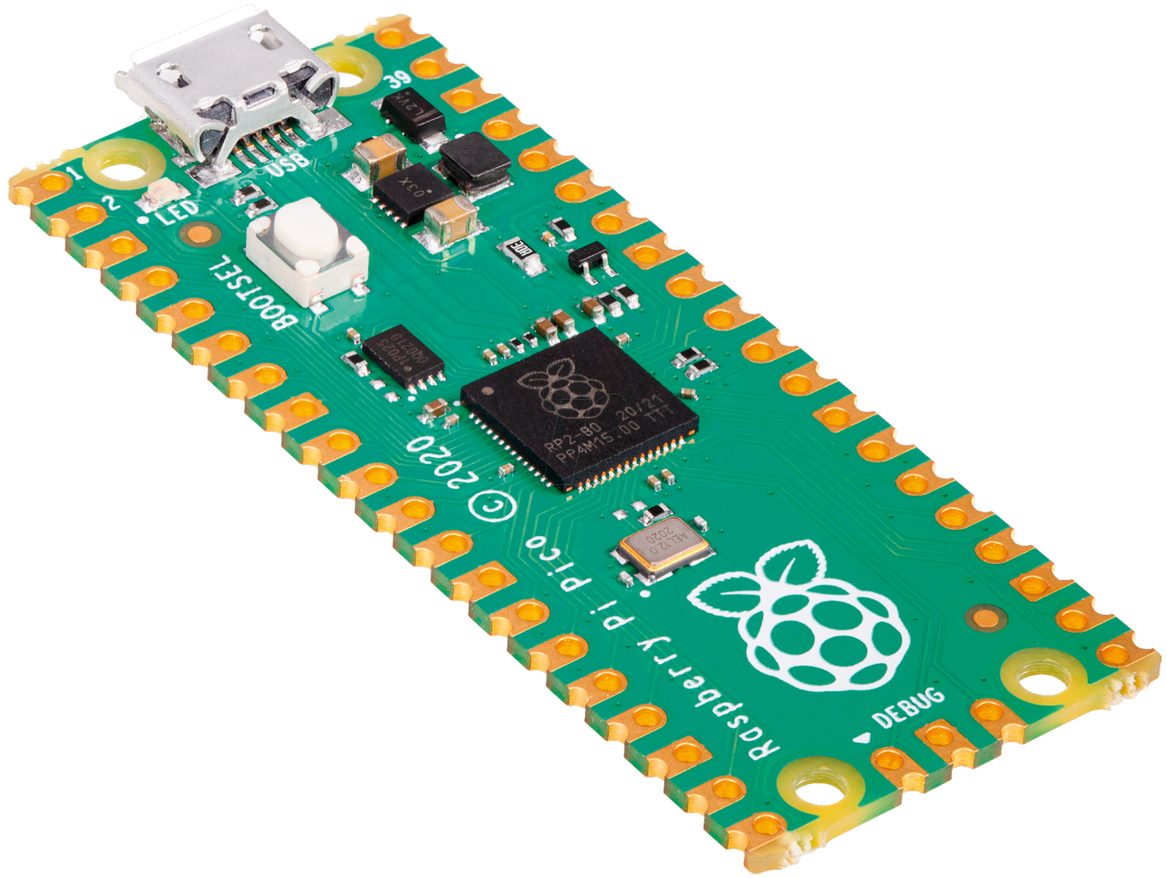
As cheap as possible

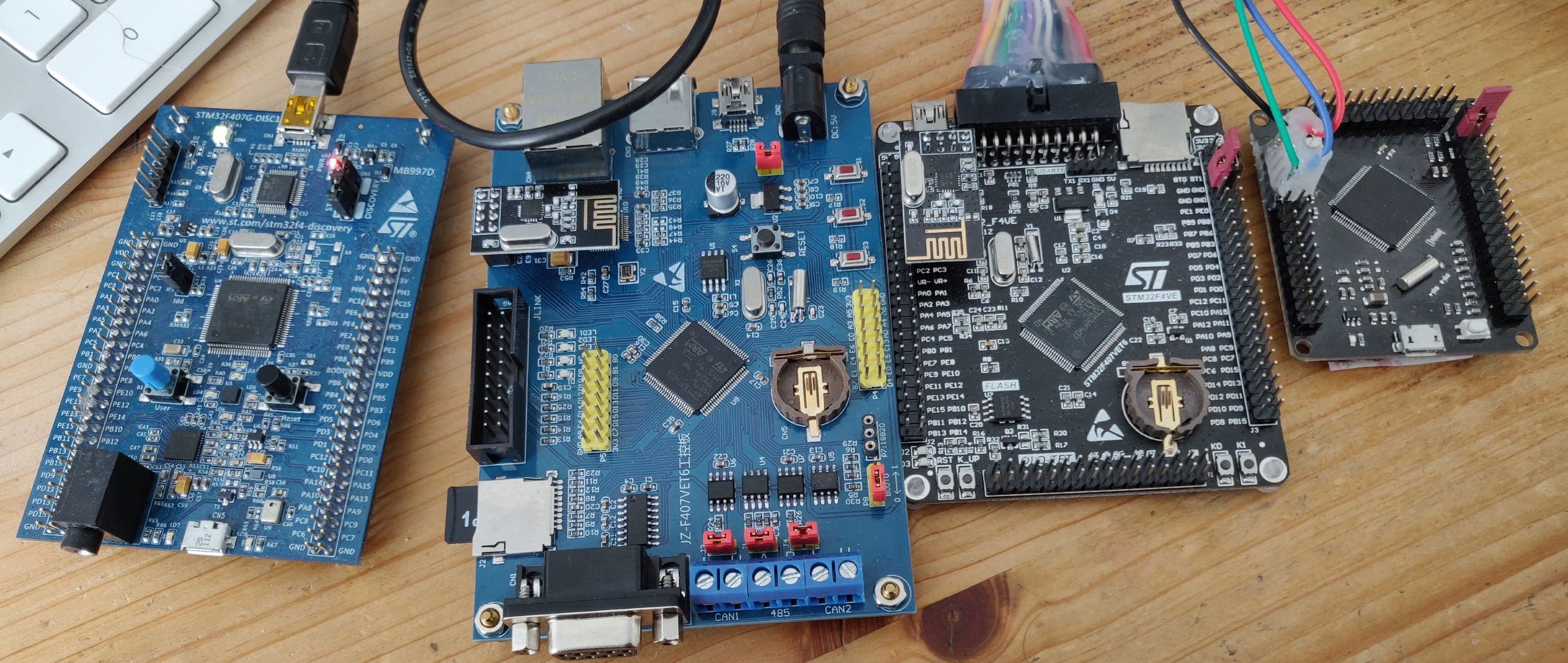
Some corners were cut

- Board: "STM32F103" blue pill board
  - 72 MHz CPU, 20 kiB SRAM, 64 kiB flash (often 128 kiB)
- Programmer/Debugger: STLINK v2 clone
- Serial converter: 3.3V FT232 module (optional with swd2)
- Logic analyser: "the 10€ 8ch 24MHz logic analyzer"® (Cypress FX2 based)
- Cheap "single use" jump wires
- A multimeter for anything analog

Raspberry Pico

- 133 MHz (dual core)
- 256kiB SRAM
- 2MiB flash
- USB bootloader
- interesting peripherals
- useable as debugger/programmer





STM32F407

ST Discovery, AliExpress COM board, black board, diymore

Nicer hardware

you get what you pay for (unless you're getting ripped off)

- easy to get: ST Nucleo boards, ST Discovery boards
- specialised boards:
  - CAN bus, Ethernet, (multiple) USB ports, level shifters
  - motor drivers, displays
- Different vendors
  - Nordic Semi: low power, wireless communication
  - TI: up to 1GHz ARM "micro"controller

Your time is valuable!

DEMO 1 (first steps)

Installation

On most STM32 boards

- st-flash erase
- st-flash write $KERNEL.bin 0x08000000
- Connect to UART with 115200 8n1 (at 3.3V, but often 5V tolerant)

Development Environment

- $EDITOR
- make
  - build, erase, flash
  - console upload
    - line delay
    - fast serial
    - tmux + swd2

DEMO 2 (LEDs)

Mecrisp Stellaris

How much compiler can you fit in 20kiB

- Written in ARM Thumb Assembler by Matthias Koch
- Runs on the target MCU
- Optimising native code compiler
  - Constant folding
  - Opcoding
  - Inlining
  - Register Allocation

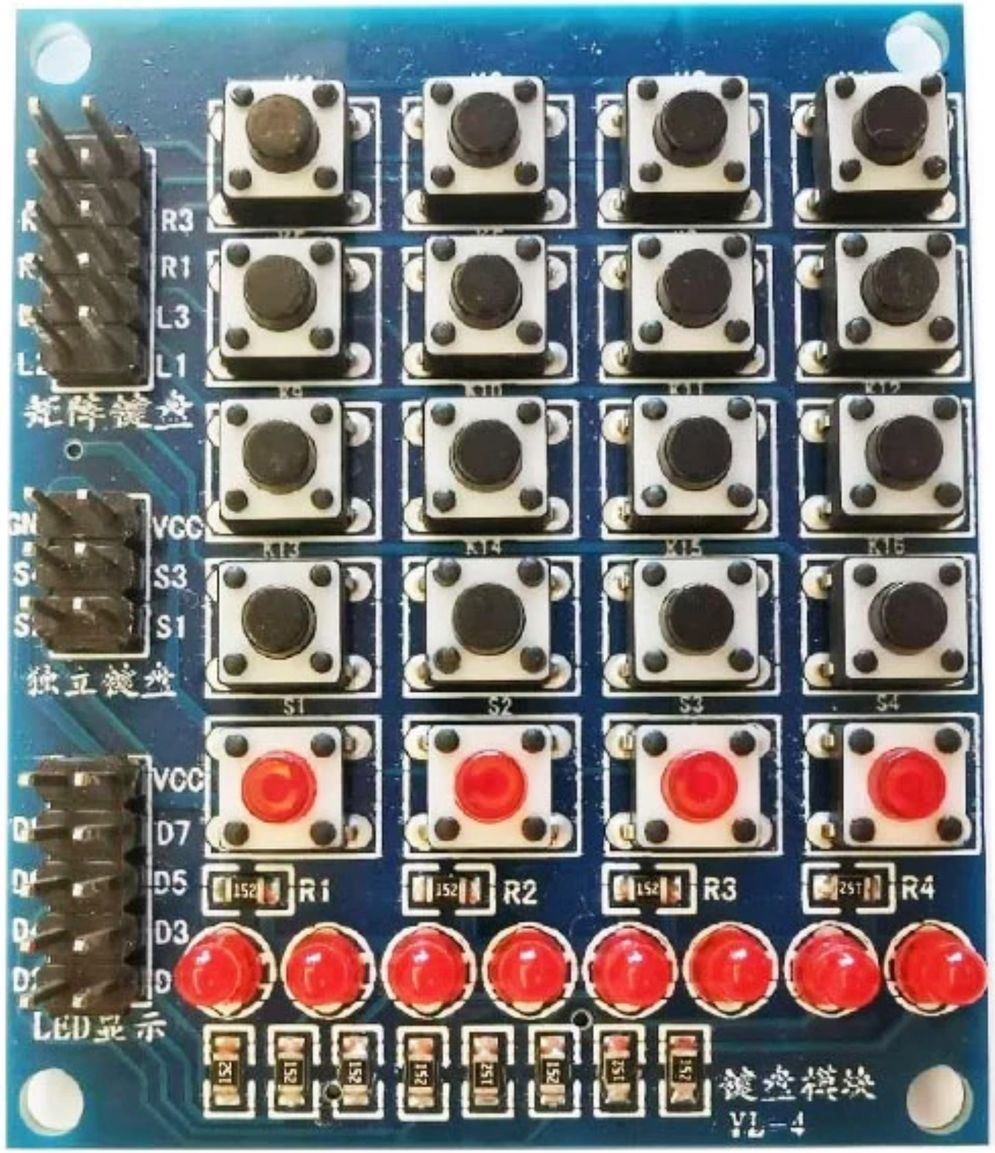
Interfacing with peripherals

Avoid metal arithmetic and typing from data-sheets

- svd2forth
  - register addresses
  - bitfields
  - .equ

DEMO 3

- The hardware
  - 8 red LEDs
  - 4 red switches
  - 4 x 4 matrix keypad
- The software
  - debounce buttons
  - control LEDs



Other Forth implementations

If you know one Forth, you know ONE Forth

- Gforth: fully featured Forth system for *nix
- zeptoforth: an embedded Forth system just for STM32F4 and L4
- FIG Forth: old, but portable
- AmForth: 16 bit Forth for AVR
- muforth: very portable, uses a meta compiler to bootstrap
- If it exists someone has probably written a Forth for it

Forth on a minicomputer (1/2)

How it all started

- Assembler: painful and low productivity
- FORTRAN, COBOL:
  - requires access to a large system
  - generated large code
  - unsuitable for direct hardware access
- BASIC: slow, not realtime capable

Forth on a minicomputer (2/2)

How it all started

- Forth:
  - fast enough
  - realtime capable
  - easy enough to use
  - self hosted
  - multitasking and multiuser support
  - can interface with specialised peripherals

Forth on earlier microcontrollers

How to conquer hostile systems

- Multiple address spaces
- Read only code memory
- Different data and address width
- Very small memories

Forth implementation strategies

Looking back where we came from

- Native code
  - Optimising
  - Subroutine threading
- Threading
  - Direct Threading
  - Indirect Threading
  - Token Threading

Contact information

- Join us on #mecrisp in freenode
- Visit the Mecrisp Stellaris Unofficial UserDoc
- My code is on GitHub