Zephyr RTOS - embedded software leaving the stone age

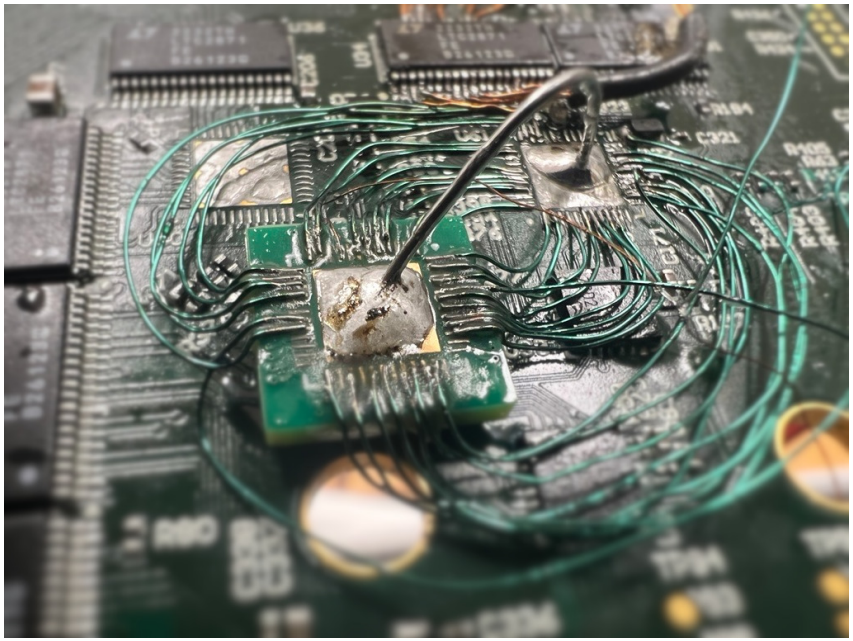
Roland Lezuo
<roland.lezuo@embedded-solutions.at>

SILA embedded solutions GmbH

2025

SILA
EMBEDDED SOLUTIONS

Introduction



your are an embedded software engineer when your patches start to look like this

- implemented Java VM JITs (assembler codegeneration)
- thesis about compiler verification (lots of logic)
- R&D at a small company, freelancer, employees
- merged with SILA St.Pölten, approx. 20 employees
- we design, implement and manufacture electronics solutions for industrial applications

SILA
EMBEDDED SOLUTIONS

Some pictures says more than a thousand words ...

SILA
EMBEDDED SOLUTIONS

It's true

Commercial embedded software development moves slowly

- IAR (a major compiler vendor) announced C99 support in 2011 (12 years delay)[1]
- Arm Keil compiler still only supports C90 and we just started a project using it[2]
- STM32[3] and TI[4] picked up eclipse and finally support Linux work-flows

Expect limitations or additional pain on Linux hosts though ...
STM32's support is very good though

[1] https://www.iar.com/dev-dynamic-custom-objects/iar-systems-provides-c99-compliance-to-8051-software-tools-ce3b6a15

[2] https://developer.arm.com/documentation/ka004425/latest

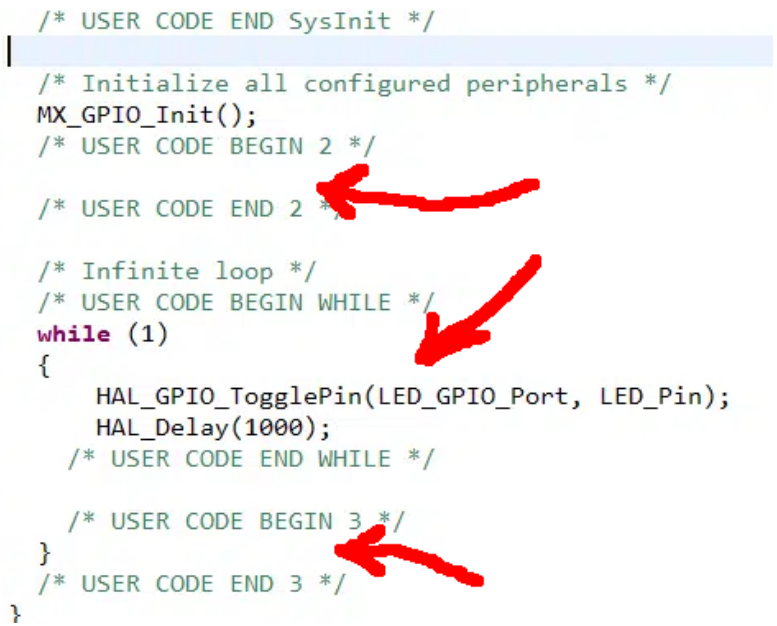
[3] https://www.st.com/en/development-tools/stm32cubeide.html

[4] https://www.ti.com/tool/CCSTUDIO

But at least commercial tool are well maintained...

If you accept the following behavior:

- auto code-generation simply overwrites source files in your tree
- but you are expected to modify those files
- user code regions for the win



SILA EMBEDDED SOLUTIONS

But at least commercial tool are well maintained cont'd...

Or don't have an issue with things like

- auto update of IDE and frameworks on startup, preventing your project from compilation, with customer in line crying for a hot-fix
- saving last-opened-timestamps in your project setting, making your git history more lively
- using pre-build hook: every build is dirty unless you manually git-reset those changes in parallel

But at least commercial tool are well maintained cont'd (2)...

Lets do some CI or at least have some build-machines

- you can generate a make based build-system for you project running on some build-slave, or can't you?

```
GUIApp.elf: $(OBJS) $(USER_OBJS)
	arm-non-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16
-O2 -fmessage-length=0 -fsigned-char -ffunction-sections -fdata-sections -Wunused -W
uninitialized -Wall -Wextra -Wmissing-declarations -Wconversion -Wpointer-arith -Wsh
adow -Wlogical-op -Waggregate-return -Wfloat-equal -g3 -T "C:\\Users\\User\\e2_studi
o\\workspace\\GUIApp\\script\\S7G2.ld" -Xlinker --gc-sections -L "C:\\Users\\User\\e
2_studio\\workspace\\GUIApp\\synergy\\ssp\\src\\bsp\\cmsis\\DSP_Lib\\cm4_gcc" -L "C:
\\Users\\User\\e2_studio\\workspace\\GUIApp\\synergy\\ssp\\src\\framework\\el\\gx\\c
m4_gcc" -L "C:\\Users\\User\\e2_studio\\workspace\\GUIApp\\synergy\\ssp\\src\\framew
ork\\el\\tx\\cm4_gcc" -Wl,-Map,"GUIApp.map" --specs=nano.specs --specs=rdimon.specs
-o "GUIApp.elf" -Wl,--start-group $(OBJS) $(USER_OBJS) $(LIBS) -Wl,--end-group
~
Makefile                                                    2,304-311      All
/"C[^"]*"
```

- and btw, post- and pre-build event are not mapped to Makefile, mapping too hard I guess ...
- additional fun fact, path-separator is "\" even on Linux hosts

SILA EMBEDDED SOLUTIONS

Leading to only one conclusion ...

SILA
EMBEDDED SOLUTIONS

Getting some things straight...

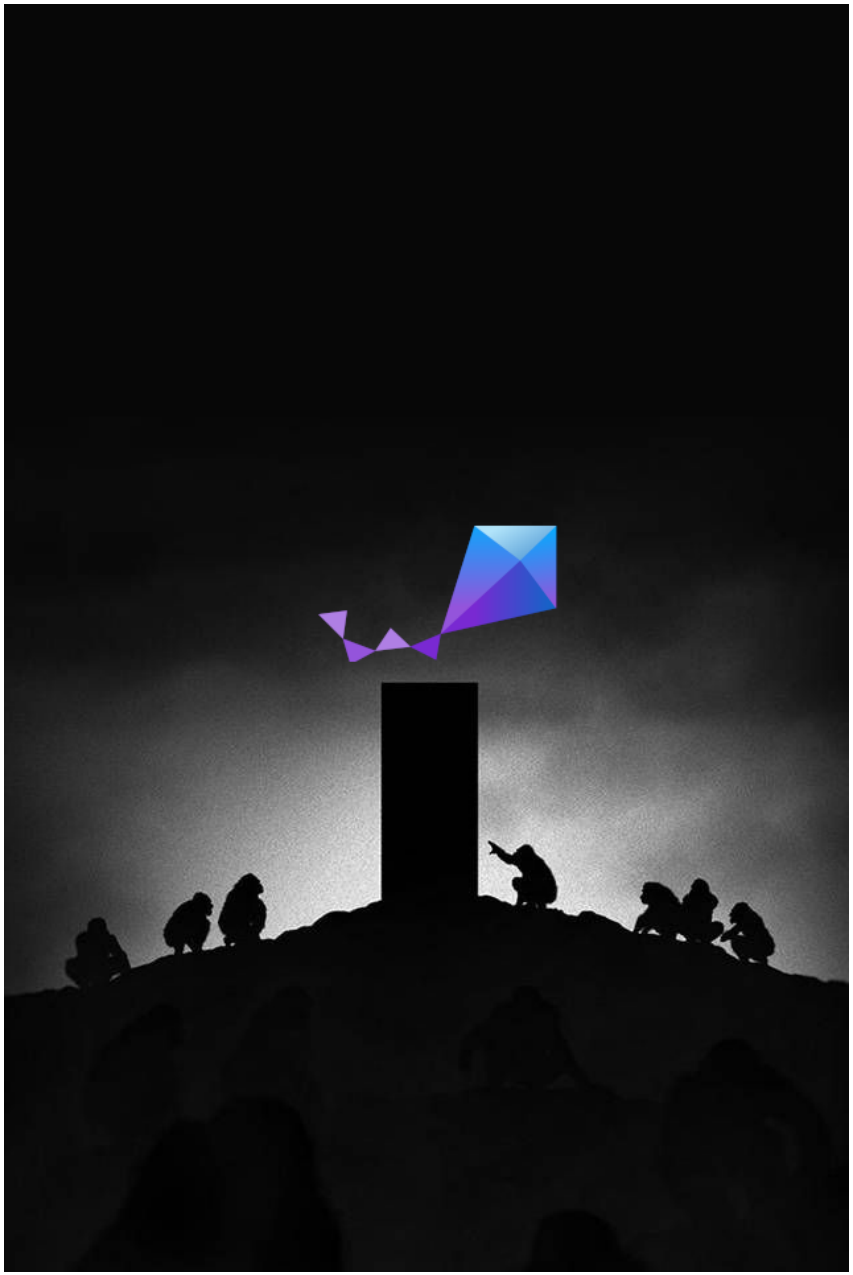
- sometimes you just need assembly language
- C is the correct abstraction for many embedded projects
- you need a linker language (looking at you mold[5])
- you often want to use an RTOS as well but they are either
  - good quality, but proprietary (QNX, vxworks)
  - open source and extremely ugly (freeRTOS)
  - proprietary and extremely ugly (embOS)

(n.b. they all do their job, somehow)

[5] https://github.com/rui314/mold

SILA EMBEDDED SOLUTIONS

Introducing Zephyr RTOS



- 2001 as RTOS for DSPs
- 2015 open-sourced by Wind River Systems (VxWorks)
- 2016 renamed Zepyhr under Linux Foundation's roof
- 2025 most contributed to open-source RTOS
- support for >8 architectures, dozens of µC vendors, 750+ boards

Why? Because of its overall excellence

SILA EMBEDDED SOLUTIONS

west for repositories and build-chrome

- software dependencies described by manifest file
- manifest part of your application
- given a SDK and installed host dependencies build as easy as
- `$ west update && west build`
- reproducible builds, build-slaves here we come!

```
manifest:
  remotes:
    - name: [redacted]
      url-base: git@bitbucket.org:[redacted]

  projects:
    - name: zephyr
      remote: [redacted]
      revision: [redacted]640afc6e57cc0f0df5cda6f972
      import:
        - path-blocklist:
            - bootloader/*
            - modules/hal/*
        - path-allowlist:
            - modules/hal/stm32
            - modules/hal/cmsis
    - name: wxlua
      path: tools/wxlua
      remote: [redacted]
      revision: [redacted]9acda22ec3dc2d2bd454a6fb
```

SILA EMBEDDED SOLUTIONS

CMake and Ninja

application and zephyr module (think libraries + meta-information) use CMake

- fast
- parallel
- extensible
- well-known
- portable
- re-usable software modules

=> same fun debugging build-systems as real developers 

SILA
EMBEDDED SOLUTIONS

A real hardware-abstraction-layer

- based on Linux kernel's well-known device-tree (dts), but with deviations
- dts resolved and consumed in build-phase (no dtb file!)
- defines µC platform and fully specifies peripherals
- hooks into zephyr's driver model
- results in real portable (across µC vendors!) embedded applications
- no more vendor-specific kindergarden-level code-generation

```
&i2c1 {
        pinctrl-0 = <&i2c1_scl_pb8 &i2c1_sda_pb9>;
        pinctrl-names = "default";

        pexp: pcal6416@20 {
                compatible = "nxp,pcal6416a";
                reg = <0x20>;   // 8-bit format
                ngpios = <16>;

                gpio-controller;
                #gpio-cells = <2>;

                int-gpios = <&gpioa 12 GPIO_ACTIVE_LOW>;
                reset-gpios = <&gpioe 0 GPIO_ACTIVE_LOW>;

                status = "okay";
        };
```

SILA EMBEDDED SOLUTIONS

A real configuration system

- based on Linux kernel's well-known Kconfig
- almost everything needs to be enabled (saving precious flash)
- tons of options configurable
- no more vendor specific config tool running on windows95 only

need to implement a USB device supporting HID class drivers?

```
&usb {
        pinctrl-0 = <&usb_dm_pa11 &usb_dp_pa12>;
        pinctrl-names = "default";
        status = "okay";
};
```

```
../boards/arm/e[redacted].dts

CONFIG_USB_DEVICE_STACK=y
CONFIG_USB_DEVICE_HID=y
CONFIG_HID_INTERRUPT_EP_MPS=64
CONFIG_USB_HID_BOOT_PROTOCOL=n
CONFIG_USB_DEVICE_MANUFACTURER="SILA Embedded Solutions GmbH"
CONFIG_USB_DEVICE_PRODUCT="[redacted] Demonstator [redacted]"
CONFIG_USB_DEVICE_VID=0x[redacted]
CONFIG_USB_DEVICE_PID=0x[redacted]
```

SILA EMBEDDED SOLUTIONS

unified device-handling

- based on openOCD
- part of zephyr-sdk
- unified interface to program a device (`$ west flash`)
- unified debugger interface (`$ west debug`) using gdb

as of today: µC family specific usage experience

SILA
EMBEDDED SOLUTIONS

its enterprisy too

- mostly Apache 2.0 licensed 
- worry not, we contribute
- pulls in vendor HAL libraries and framework (no need to re-invent wheels)
- generates SBOM (`$ west spdx`)

SILA
EMBEDDED SOLUTIONS

I can haz cake and eat it too?

- host setup much more complex than clicking setup.exe
- lot of complex technologies included
- steep learning curves ahead on multiple occasions
- embedded software can not (fully) abstract away from µC
- developer very much needs to understand Zephyr's concepts

SILA EMBEDDED SOLUTIONS

Conclusion

- it's our default RTOS for all embedded projects, there must be reasons against Zephyr RTOS
- experience in 15+ projects, some quite large, proven stability
- definitely worth your time, needs some upfront investment though

Questions?

:wq

SILA EMBEDDED SOLUTIONS

kernel features

- threads: cooperative, priority-based, non-preemptive, preemptive
- IRQ services
- various memory allocation schemes
- synchronization: semaphores, mutex, queues, signals
- power-management: different operating modes, needs driver support
- multiple scheduling algorithms
- memory-protection, application crash-reports
- USB, BLE, Ethernet, IP, TCP, UDP, ...
- very good logging framework
- unit-tests (can be executed on host zephyr environment)
- interactive shell (!)
- POSIX APIs

SILA
EMBEDDED SOLUTIONS