Running the Zephyr RTOS and TensorFlow Lite on RISC-V

RISC-V Summit, Santa Clara, Dec 03, 2018
Michael Gielda, Antmicro, mgielka@antmicro.com
Piotr Zierhoffer, Antmicro, pzierhoffer@antmicro.com
Pete Warden, Google, petewarden@google.com
WHAT IS THE ZEPHYR PROJECT?

“The Zephyr™ Project is a Linux Foundation hosted Collaboration Project, (...) aiming to build a best-in-breed small, scalable, real-time operating system (RTOS) optimized for resource constrained devices, across multiple architectures.”
WHAT IS THE ZEPHYR PROJECT?

“The Zephyr™ Project is a Linux Foundation hosted Collaboration Project, (...) aiming to build a best-in-breed small, scalable, real-time operating system (RTOS) optimized for resource constrained devices, across multiple architectures.”
WHY BOTHER WITH TINY CHIPS?

I’m convinced that machine learning can run on tiny, low-power chips, and that this combination will solve a massive number of problems we have no solutions for right now.

Pete Warden, Google's TensorFlow Mobile Technical Lead
Zephyr Project

- **Open source** real time operating system
- **Vibrant Community** participation
- Built with **safety and security** in mind
- **Cross-architecture** with growing developer tool support
- **Vendor Neutral** governance
- **Permissively** licensed - Apache 2.0
- **Complete**, fully integrated, highly configurable, **modular** for **flexibility**, better than roll-your-own
- **Product** development ready with LTS
- **Certification** ready with Auditable
OTHER REASONS WE NEED ZEPHYR

- targeted at IoT and making it truly vendor-neutral & open source - BlueTooth, OpenThread...
- portability, API standardization
- good scalability perspective between different systems (e.g. heterogeneous multi-core)
- grown-up OS features
- Linux-like look and feel
- modern design, software-driven
- testing, testing, testing
SO, WHO’S IN?

Platinum Members

- Intel
- Linaro
- Nordic Semiconductor
- NXP
AS WELL AS
CROSS-ARCHITECTURE
Zephyr Ecosystem

Zephyr OS
- The kernel and HAL
- OS Services such as IPC, Logging, file systems, crypto

Zephyr Project
- SDK, tools and development environment
- Additional middleware and features
- Device Management and Bootloader

Zephyr Community
- 3rd Party modules and libraries
- Support for Zephyr in 3rd party projects, for example: Jerryscript, Micropython, Iotivity

Kernel / HAL
- Scheduler
- Kernel objects and services
- low-level architecture and board support
- power management hooks and low level interfaces to hardware

OS Services and Low level APIs
- Platform specific drivers
- Generic implementation of I/O APIs
- File systems, Logging, Debugging and IPC
- Cryptography Services
- Networking and Connectivity
- Device Management

Application Services
- High Level APIs
- Access to standardized data models
- High Level networking protocols
# Zephyr Roadmap 2018

## 2018

<table>
<thead>
<tr>
<th></th>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
<th>May</th>
<th>Jun</th>
<th>July</th>
<th>Aug</th>
<th>Sept</th>
<th>Oct</th>
<th>Nov</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Zephyr Releases</strong></td>
<td>♦ 1.11</td>
<td>♦ 1.12</td>
<td>♦ 1.13</td>
<td>♦ 1.14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Zephyr 1.11
- OpenThread support
- Native POSIX Port
- POSIX API Layer (PSE52)
- FOTA Updates (LWM2M, BLE)
- SMP Support
- Lightweight Flash Storage
- Support the kernel (scheduler + objects) as a separate module

## Zephyr 1.12
- AMP Support
- 802.1Q - Virtual LANs
- Persistent Storage for BT
- TAP net device support
- SPI slave support
- CanBUS support
- Source Code modularisation: Support external modules, boards, SoCs
- Command line meta-tool “west”
- Wi-Fi driver

## Zephyr 1.13
- QM level qualification
- MISRA-C 2012: Kernel
- LLVM Support
- Precision Time Protocol (PTP) Support
- Improved Logging Support
- Eco-System: Tracing, Profiling, debugging support through 3rd party tools
- Multiple Git Repos
- Soft real-time tasklets
- Advanced Power Mgmt.

## Future LTS
- Safety and Security Pre-Certification
- Time Sensitive Networking (TSN) Support
- TEE for ARMv8-M
- LoRa Support
- SocketCAN
- Paging Support
- Dynamic Module Loading
- Enhanced Sensor support (support HW FIFOs)
- MIPS

---

*NOTE: Features aligned to releases are subject to change per guidance from the TSC*
RISC-V IN ZEPHYR
RISC-V ZEPHYR PORT

- pretty good documentation on porting and required components
- 4 platforms (including QEMU) supported today, we need more (reach out to us, we can help)!
- our LiteX/VexRiscv port exists but needs to be upstreamed
ADDING YOUR BOARD

• first, you need to read this
• there is some entry work to understand the structure (as with any standardised system), but
• it’s really not so much code (example)
WORKING WITH ZEPHYR
SDK

- comes with an SDK (really a bunch of open source tools, don’t fret) - 0.9.5 currently
- toolchains come bundled, adding a new platform requires providing a toolchain (but you can use your own)
- source a simple script and work in the console
- don’t forget to also do: `pip3 install -r scripts/requirements.txt`
BUILD SYSTEM / CONFIGURATION

- based on CMake && (make || ninja)
- uses Kconfig format with custom extensions
- Python menuconfig implementation
- to be most probably replaced by
  Swiss-Army-knife CLI meta-tool, West
HELLO WORLD EXAMPLE

#include "contiki.h"
#include <stdio.h> /* For printf() */

/*--------------------------------------*/
PROCESS(hello_world_process, "Hello world");
AUTOSTART_PROCESSES(&hello_world_process);
/*--------------------------------------*/
PROCESS_THREAD(hello_world_process, ev, data)
{
    PROCESS_BEGIN();
    printf("Hello, world\n");
    printf("Hello, world\n");
    PROCESS_END();
}

#include <zephyr.h>
#include <misc/printk.h>

void main(void)
{
    printk("Hello World! %s\n", CONFIG_ARCH);
}
TESTING

- currently using SanityCheck, a runner for various simulators (QEMU, Renode, ARC simulator) and real boards
- introducing TCF, new open source framework from Intel for testing on real hardware
- strong focus on testing
- testing working group, meets every Monday
TESTING ZEPHYR IN RENODE

- Open source, permissively licensed framework targeting similar, especially multi-node systems
- Recommended Zephyr tool
- Integration with SanityCheck is being merged, with Mi-V as example platform
- Our Zephyr ports were developed on Renode
- Also working to enhance multi-node testing in Zephyr with Renode
CONTINUOUS INTEGRATION METHODOLOGY

Company Environment

Local PC

Commit code

Interactive test and debug in Renode

Get help from colleagues

Tests pass?

Push to server

CI e.g. with Robot + Renode

Tests with various configurations

Develop with favorite IDE/compiler

Merge changes

Deploy tests / deployment

Renode - a new approach to complex embedded systems development
TensorFlow Lite on RISC-V
TensorFlow Lite

https://www.tensorflow.org/lite/

- Officially supported on Android, iOS, and Raspberry Pi
- Less than 100 kilobytes of binary footprint!
- Few dependencies (for example flatbuffers instead of protobufs)
- Good support for model compression techniques like quantization
TensorFlow Lite for Microcontrollers

- Still very experimental!
- Aimed at running machine learning models on sensor data
- 20KB binary footprint (on Cortex M3), with no memory allocation, floating point, or standard C/C++ library calls
Challenge

- Want to run on RISC-V!
- Already internally running on GreenWaves GAP8
- No external targets available
First Big Question

Which RISC-V?

- Lots of different toolchains and devices
- No ‘apt-get install riscv-gcc’ (yet)!
- Started with the GNU MCU Eclipse toolchain, since it was the easiest to find
- It was hard to figure out how to target something that we could run on a real device (or in Renode)
- A colleague (Marcia Louis) suggested using the SiFive Freedom E toolchain with prebuilt binaries and targeting the SiFive FE310, which has a Renode definition
Getting It Working

https://github.com/antmicro/tensorflow/tree/riscv-mcu

Pre-requisites: Download pre-built RISC-V gnu tools from SiFive


tar xzf riscv64-unknown-elf-gcc-20181030-x86_64-linux-ubuntu14.tar.gz

export PATH=${PATH}:riscv64-unknown-elf-gcc-20181030-x86_64-linux-ubuntu14/bin/

- Download the TensorFlow source with git@github.com:mars20/tensorflow.git
- Enter the source root directory by running cd tensorflow
- Checkout out the "riscv_mcu" branch by running git checkout riscv_mcu
- Download the dependencies by running tensorflow/lite/experimental/micro/tools/make/download_dependencies.sh. This may take a few minutes
- Build and test the library with make -f tensorflow/lite/experimental/micro/tools/make/Makefile TARGET=riscv32_mcu
Work in Progress

- Piotr at Antmicro helped us work through a lot of issues
  - For example link ordering, removing exception handling
- We’re still linking in the standard C library
- RISC-V toolchain seems tricky to use on bare metal
  - memcpy() is used as an optimization under the hood
  - Other C library functions need to be linked in
- We don’t fully understand some of the flags (for example CPU type)
- But it’s alive! And can be run by anyone with a HiFive1 board (or Renode)
- Working on merging this into the mainline, with testing
AS USED IN
Products Running Zephyr

Ellcie-Healthy Smart Connected Eyewear
ProGlove Scanning Gloves
Intellinium Safety Shoes
Grush Gaming Toothbrush
hereO Smartwatch
Blocks Modular Smartwatch
Antmicro Badge
GNARBOX 2.0 SSD
Rigado IoTGateway
Fault-tolerant RISC-V for space

→ Triple-Modular-Redundancy fault-tolerant RISC-V space application demonstrator for Thales
→ SW running in the demonstrator developed in Zephyr RTOS - excellent as standard software stack for POSIX-compliant applications
→ Host platform: Linux on Antmicro’s UltraScale+ devkit
EXAMPLE: RISC-V BADGE

- e-paper, NFC
- runs Zephyr (of course)
- open source, open hardware, including the CPU!
- based on a portable RV32 module
- [https://badge.antmicro.com](https://badge.antmicro.com)
SUMMARY

→ Lots of good progress on both Zephyr and TF Lite
→ we need to get them integrated now!
→ we welcome your input for the Getting Started Guide that is being created
THANK YOU FOR YOUR ATTENTION!