Embedded Software Profiling

Do It Yourself

Aleksei Musin

Introduction

- Over 20 years of the embedded development
- Audio, power energy, railway
- Development processes audit and improvement
- Consulting services

Applications

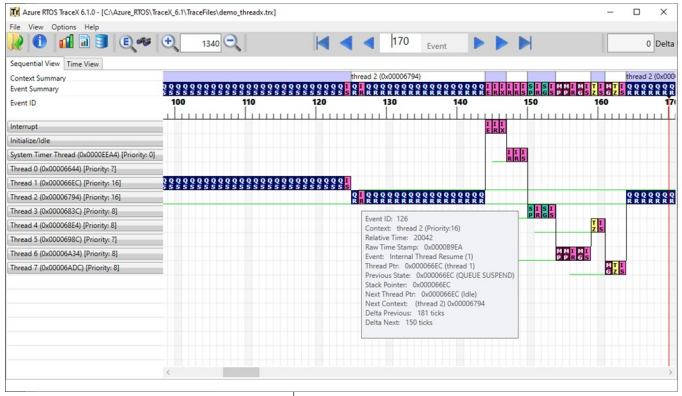
- Real-time low-latency embedded systems
- Digital signal processing
- Power-constrained devices

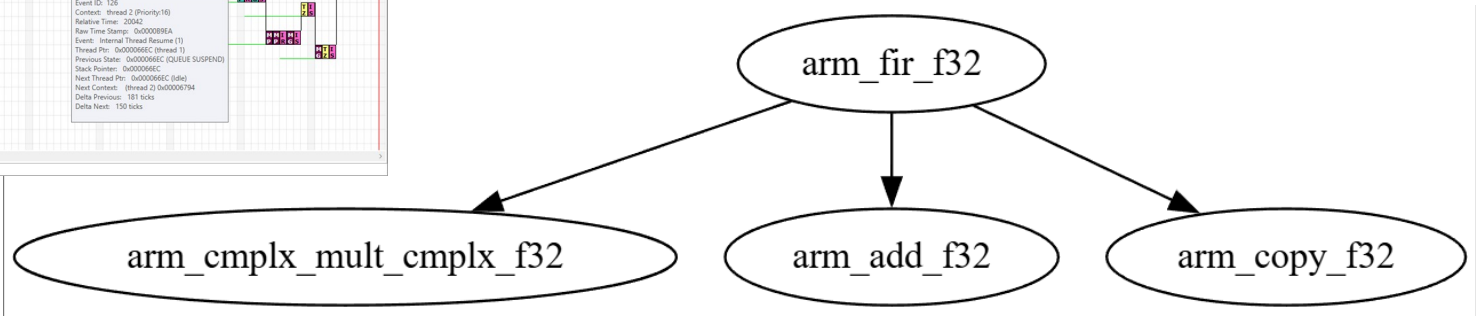
Problems and Issues

- Processing time, latency
- Power consumption
- Source code learning and maintenance

What We Need

- **Call stack tree**
- **Timing diagram with functions execution time**



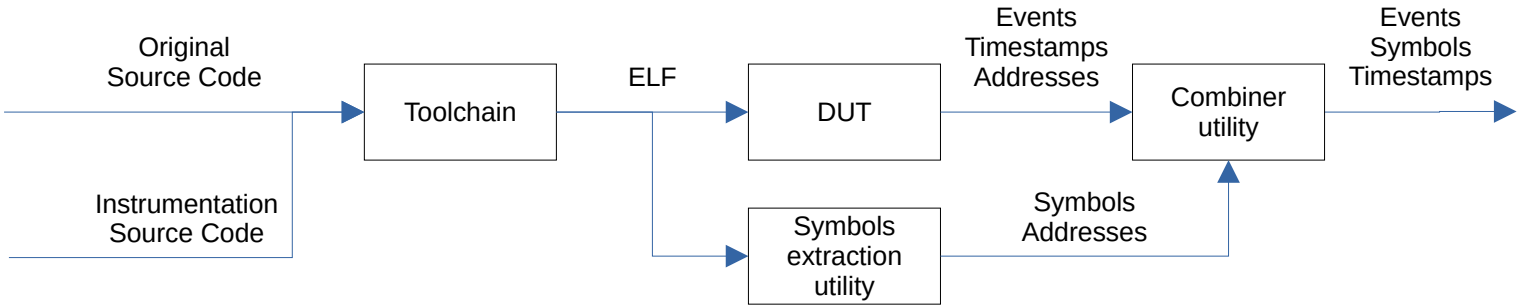


Tools

- GPIO toggling, and a logic analyzer / oscilloscope
- “printf”-like log output
- Segger SystemView (bare-metal and RTOS-based)
- Percepio Tracealyzer (bare-metal and RTOS-based)
- TraceX (for ThreadX RTOS only, open source)
- Keil uVision Event Viewer

Pipeline

- **Instrumentation and the log retrieving**
- **Function addresses replacement with names**
- **Visualization**



Prerequisites

- Source code
- Time service (system timer, CPU clock counter)
- Build system
- Interface to device (UART, USB, display)

Instrumentation

- **Making functions below, and adding events to a log with timestamps**

  ```
  void __cyg_profile_func_enter (void *this_fn, void *call_site);
  void __cyg_profile_func_exit (void *this_fn, void *call_site);
  ```

- **Implementing the log reading (display, USB, UART)**
- **Compilation with the** `-finstrument-functions` **option (valid for GCC and clang)**
- **Source code under test is not changed**

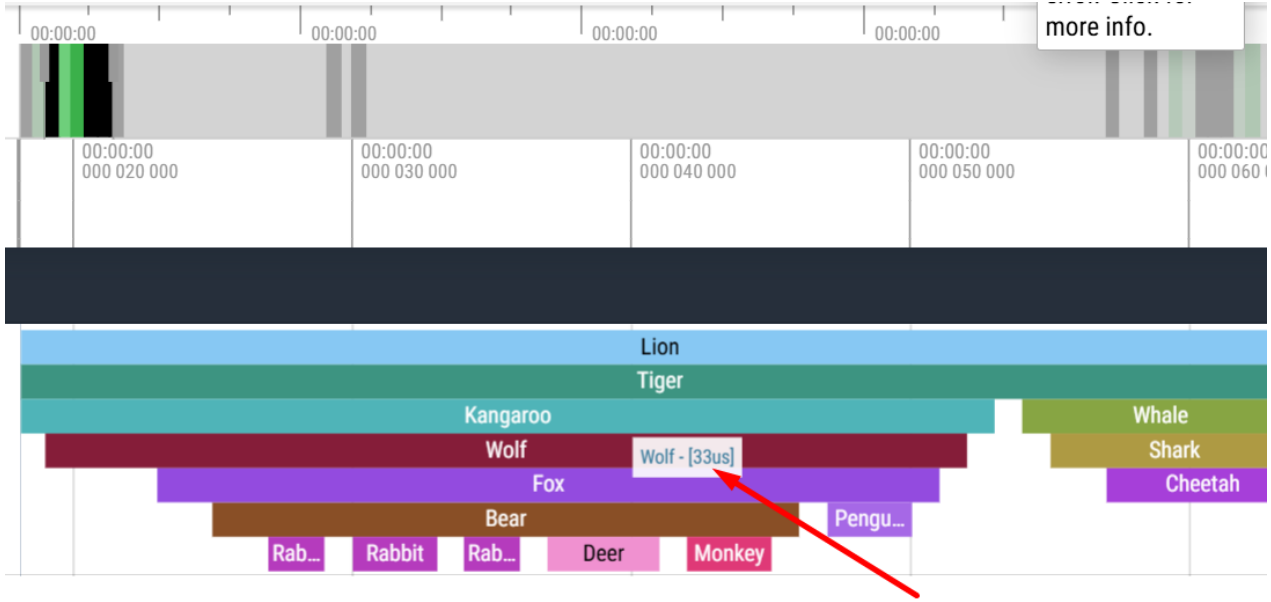
ELF/Listing/Map Parsing

- In the log we retrieved, functions' addresses and entry and exit time are recorded
- Functions' names and addresses are located in ELF, listings and a map-file
- The data can be merged with a script

  Note. Sometimes the script should consider different addresses as equal (in some ARM cores, even and odd addresses)

Visualization

- **Perfetto Trace Viewer (Chrome Event Tracing Format JSON)**



Hints

- Instrumentation is not free, it takes some time in runtime. Thus, small and often called functions have biased measurements. Exclude them

  (see the --finstrument-functions-exclude-file-list option)
- In a multi-threaded application, the instrumentation callbacks should be thread-safe
- Assembly functions are not instrumented, but can be wrapped up with C functions

References

- GCC instrumentation options

https://gcc.gnu.org/onlinedocs/gcc/Instrumentation-Options.html

- Perfetto Trace Viewer - https://perfetto.dev/
- Usage of “-finstrument-functions” for the dynamic memory management in the “release” code, C++ Russia ‘2024 (in Russian)

  https://www.youtube.com/watch?v=0vfyhbTWGcI

Contacts

https://github.com/armusin

https://www.linkedin.com/in/aleksei-musin-b0617441/