PERFORMANCE ANALYSIS USING NXP’S I.MX RT1050 CROSSOVER PROCESSOR AND THE ZEPHYR™ OS

MAUREEN HELM
LEOTESCU FLORIN
MARIUS CRISTIAN VLAD

BENCHMARKING TEAM
AGENDA

Zephyr Project introduction
Analysis scope
Hardware configuration
Benchmark study methodology
Run to run performance variation
Performance comparison with i.MX 6UL
Conclusions
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

Open Source, RTOS, Connected, Embedded Fits where Linux is too big

Zephyr OS

- 3rd Party Libraries
- Application Services
- OS Services
- Kernel
- HAL
Why Zephyr?

The Zephyr OS addresses broad set of embedded use cases across a broad set of platforms and architectures using a modular and configurable infrastructure. It addresses the need for RTOS consolidation.

**Address Fragmentation**
- No single RTOS addresses broad set of embedded use cases across a broad set of platforms and architectures
- Disjoint use cases have led to fragmentation in RTOS space
- Existing commercial solutions force roll your own solutions and duplication of software components

**Modular Infrastructure**
- Modular and configurable infrastructure allows creation of highly compact and optimal solutions for different products from a common origin
- Reuse allows NRE costs to be amortized across multiple products and solutions
- Multi-architecture support reduces platform switching costs and vendor lock-in concerns

**Open-Source**
- Roll your own is expensive & difficult to develop & maintain
- Permissively licensed corresponds to ease of adoption
- Corporate sponsorship assures long term commitment and longevity
- Community innovation has proven faster for progression and project development is a collaboration of industry experts

**Feature Richness**
- Need for a solution or semi-complete solution rather than just an ingredient.
- Lowers entry level barrier for new products and speeds up software delivery using existing feature and hardware support
- Encourages adherence to standards and promotes collaboration on complex features inside the organization
- Developers focus on the end-user facing interfaces instead of re-inventing low level interfaces

Reduce costs and improve efficiency through reuse
### Architecture

- **Highly Configurable, Highly Modular**
- **Cooperative and Pre-emptive Threading**
- **Memory and Resources are typically statically allocated**
- **Integrated device driver interface**
- **Memory Protection: Stack overflow protection, Kernel object and device driver permission tracking, Thread isolation**
- **Bluetooth® Low Energy (BLE 4.2, 5.0) with both controller and host, BLE Mesh**
- **Native, fully featured and optimized networking stack**

**Fully featured OS allows developers to focus on the application**
Analysis scope

Asses the real-time efficiency of an embedded system (i.MX RT1050 crossover processor)

Determine the performance gap between the MIMXRT1050-EVK board equipped with an embedded ARM SoC and a similar board equipped with an application processor.
Hardware configuration

**i.MX RT1050 configuration**
- Development board: MIMXRT1050-EVK
- Processor: MIMXRT1052DVL6A Arm® Cortex®-M7 core
- Number of cores: 1
- Core Frequency: 600 MHz
- Board schematic: SCH-29538 REV A1
- OS name: Zephyr OS 1.11.99
- OS type: Real Time OS

**i.MX 6UL configuration**
- Development board: i.MX 6UL EVK
- Processor: i.MX 6UltraLite Processor based on Arm Cortex-A7 core
- Number of cores: 1
- Core Frequency: 528 MHz
- Board schematic: SCH-29163 REV A2
- OS name: Linux BSP - kernel 4.9.88-imx_4.9.88_2.0.0_ga
- OS Type: Non- Real Time OS
Benchmarking methodology

THREAD_BENCH
- Thread creation time
- Thread join time

HEAP_BENCH
- Heap allocation time
- Heap deallocation time

MUTEX_BENCH
- Mutex lock time
- Mutex unlock time

CTX_BENCH
- Context switch time

RUNTIME C / POSIX LIBRARY

OPERATING SYSTEM

HARDWARE DEVICE
Benchmarking methodology

- **THREAD_BENCH**
  - Thread creation time
  - Thread join time

- **HEAP_BENCH**
  - Heap allocation time
  - Heap deallocation time

- **MUTEX_BENCH**
  - Mutex lock time
  - Mutex unlock time

- **CTX_BENCH**
  - Context switch time

**RUNTIME C / POSIX LIBRARY**

- `pthread_create()`
- `pthread_join()`

**OPERATING SYSTEM**

**HARDWARE DEVICE**
Benchmarking methodology

**THREAD_BENCH**
- Thread creation time
- Thread join time

**HEAP_BENCH**
- Heap allocation time
- Heap deallocation time

**MUTEX_BENCH**
- Mutex lock time
- Mutex unlock time

**CTX_BENCH**
- Context switch time

---

**RUNTIME C / POSIX LIBRARY**

```
malloc()
k_malloc() - Zephyr
free()
k_free() - Zephyr
```

**OPERATING SYSTEM**

**HARDWARE DEVICE**
Benchmarking methodology

**THREAD_BENCH**
- Thread creation time
- Thread join time

**HEAP_BENCH**
- Heap allocation time
- Heap deallocation time

**MUTEX_BENCH**
- Mutex lock time
- Mutex unlock time

**CTX_BENCH**
- Context switch time

**RUNTIME C / POSIX LIBRARY**
- `pthread_mutex_lock()`
- `pthread_mutex_unlock()`

**OPERATING SYSTEM**

**HARDWARE DEVICE**
Benchmarks methodology

- **THREAD_BENCH**
  - Thread creation time
  - Thread join time

- **HEAP_BENCH**
  - Heap allocation time
  - Heap deallocation time

- **MUTEX_BENCH**
  - Mutex lock time
  - Mutex unlock time

- **CTX_BENCH**
  - Context switch time

**RUNTIME C / POSIX LIBRARY**

**OPERATING SYSTEM**

**HARDWARE DEVICE**

```
sched_yield();
FUTEX / MUTEX
```
Benchmarking methodology

- **THREAD_BENCH** - Thread creation time
  - Thread join time

- **HEAP_BENCH** - Heap allocation time
  - Heap deallocation time

- **MUTEX_BENCH** - Mutex lock time
  - Mutex unlock time

- **CTX_BENCH** - Context switch time

**TIME MEASUREMENT**

Linux

- `clock_gettime()`

- `TIMING_INFO_PRE_READ()`

- `TIMING_INFO_OS_GET_TIME()`

Zephyr™
Predictable results observed. No variations from run to run.

**TIMING_INFO_PRE_READ()** used to record elapsed time.
Results varies from run to run, depending on what the OS is doing “behind the scenes”.

Used `clock_get_time()` to record the elapsed time.
## Performance comparison

<table>
<thead>
<tr>
<th>OS</th>
<th>Zephyr OS 1.11.99</th>
<th>Linux BSP 4.9.88-imx 4.9.88-2.0.0_ga</th>
<th>Difference (x times)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Board Name</td>
<td>i.MX RT1050 EVK</td>
<td>i.MX 6UL EVK</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Arm® Cortex® M7</td>
<td>Arm® Cortex® A7</td>
<td></td>
</tr>
<tr>
<td>CPU Cores</td>
<td>1</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>Core Frequency (MHz)</td>
<td>600</td>
<td>528</td>
<td>-</td>
</tr>
<tr>
<td>Average heap malloc time (cycles)</td>
<td>1001</td>
<td>11499</td>
<td>11x</td>
</tr>
<tr>
<td>Average heap free time (cycles)</td>
<td>1126</td>
<td>4870</td>
<td>4x</td>
</tr>
<tr>
<td>Average pthread_mutex_lock time (cycles)</td>
<td>53</td>
<td>799</td>
<td>15x</td>
</tr>
<tr>
<td>Average pthread_mutex_unlock time (cycles)</td>
<td>83</td>
<td>818</td>
<td>10x</td>
</tr>
<tr>
<td>Average pthread_create time (cycles)</td>
<td>719</td>
<td>85478</td>
<td>118x</td>
</tr>
<tr>
<td>Average pthread_join time (cycles)</td>
<td>1702</td>
<td>89219</td>
<td>52x</td>
</tr>
<tr>
<td>Average Context switch time (cycles)</td>
<td>47</td>
<td>1284</td>
<td>27x</td>
</tr>
</tbody>
</table>

The average time is calculated in cycles (lower is better).

Zephyr OS running on the i.MX RT1050 presents a significant improvement in all time cycles compared to the Linux BSP + i.MX 6UL EVK.
Conclusions

- The performance analysis was done by running custom microbenchmarks on two different hardware & software platforms.
- Benchmarks were developed around a common API to have comparable results.

**i.MX RT1050 EVK with Zephyr OS performs significantly better than i.MX 6UL with Linux BSP when doing the same tasks.**
NXP Platforms supported by Zephyr OS

NXP Boards
- MIMXRT1050-EVK
- MIMXRT1060-EVK
- FRDM-K64F
- FRDM-KL25Z
- FRDM-KW41Z
- Hexiwear
- LPC54114 (M0 Core)
- LPC54114 (M4 Core)

• Partner boards:
  - UDOO Neo Full (with i.MX 6SoloX - Arm Cortex-M4 Core only)
  - Colibri iMX7 (i.MX7 SoC - Arm Cortex-M4 Core only)

www.nxp.com/zephyr
Thank You!