Express Logic’s ThreadX RTOS for RISC-V
Background

- ThreadX® is a commercial RTOS from Express Logic, Inc., San Diego, CA.
- In production since 1997, ThreadX is used by major IoT product manufacturers in the areas of Consumer Electronics, Industrial Controls, Medical Devices, and Transportation.
- ThreadX has been used in the development of, and is currently deployed in over 6.2 billion products.

© 2017, Express Logic, Inc.
ThreadX Overview

• ThreadX is a small footprint, priority-based, fully preemptive RTOS, with a single linear address space.
• ThreadX offers services for creation and use of Threads, Message Queues, Semaphores, Mutexes, Timers, Memory Byte and Block Pools, and Event Flags.
• ThreadX has been safety certified to IEC-61508/62304/26262, etc., to SIL-4, Class C, ASIL D.
• In Addition, ThreadX offers advanced technology, including:
  – Preemption-Threshold Scheduling
  – Real-time Event Trace
  – Memory-Protected Modules
Preemption-Threshold™ Scheduling

- A technique to reduce context switches
- Preemption-Threshold establishes a priority ceiling for disabling preemption – preemption requires a priority higher (lower number) than the ceiling

- For example, assume a thread’s priority is 20, and its Preemption-Threshold is set to 15
- Threads with priority lower than (larger number) 14, even if higher than (smaller number) the running thread’s priority (20), will not preempt the running thread.

<table>
<thead>
<tr>
<th>Priority</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Preemption allowed for threads with priorities from 0 to 14 (inclusive)</td>
</tr>
<tr>
<td>14</td>
<td>Thread is assigned Preemption-Threshold = 15 [this has the effect of disabling preemption for threads with priority values from 15 to 19 (inclusive)]</td>
</tr>
<tr>
<td>15</td>
<td>Thread is assigned Priority = 20</td>
</tr>
</tbody>
</table>
ThreadX Event Trace

Trace Buffer, Macro, and TraceX® in Operation

Target Memory

- ThreadX RTOS
  - Event Trace Code

- Events (32-bytes)
- Circular Trace Buffer

Host PC

- MACRO: save_trace_buffer()
- trace.trx

expresslogic

© 2017, Express Logic, Inc.

5
ThreadX Application Modules

Module 1
- Application Thread 1
- Application Thread 2
- Application Thread 3
- Application Threads …
- Application Thread “n”
  Interface

Module 2
- Application Thread 1
- Application Thread 2
- Application Thread 3
- Application Threads …
- Application Thread “n”
  Interface

Module n
- Application Thread 1
- Application Thread 2
- Application Thread 3
- Application Threads …
- Application Thread “n”
  Interface

Interface

Application Code
Module Manager

ThreadX Kernel

Kernel Executable
Testing The Port

• The ThreadX / RISC-V port was developed and first tested in a software environment (SPIKE) that simulates the RISC-V.

• We then ran the port on a Microsemi Smartfusion2 Creative Development Board (FUTUREM2SF-EVB).
  – The RISC-V FPGA bit file image (https://github.com/RISCV-on-Microsemi-FPGA/M2S025-Creative-Board.git) was used for testing.
  – Hardware timers and a debug UART are provided as part of the kit.
Finally, we verified that the standard 8-thread ThreadX demo (demo_threadx.c) operates normally on RISC-V.
ThreadX Footprint

• **Instruction Area (ROM, Flash) Sizes**
  - Core Services: 2,082 Bytes
  - Queue Services: 858
  - Event Flag Services: 756
  - Semaphore Services: 384
  - Mutex Services: 1,254
  - Block Memory Services: 472
  - Byte Memory Services: 820

• **RAM Sizes**
  - Global Data Area: 500 – 2,000 Bytes
  - Thread Control Block: 176
  - Timer Control Block: 44
  - Queue Control Block: 56
  - Semaphore Control Block: 28
  - Mutex Control Block: 52
  - Event Flag Control Block: 36
  - Block Memory Control Block: 48
  - Byte Memory Control Block: 52

Note: Measurements for RISC-V were not available at press time. These measurements are typical for a 32-bit RISC processor.
# ThreadX Service Times

## Fast Execution (120MHz Processor)

<table>
<thead>
<tr>
<th>Function</th>
<th>Immediate Response (IR)</th>
<th>Thread Suspend (TS)</th>
<th>Thread Resumed (TR)</th>
<th>Thread Resumed and Context Switched (TRCS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>tx_thread_suspend</td>
<td>1.2μs</td>
<td>1.4μs</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>tx_thread_resume</td>
<td>-</td>
<td>-</td>
<td>1.0μs</td>
<td>1.9μs</td>
</tr>
<tr>
<td>tx_thread_relinquish</td>
<td>0.4μs</td>
<td>-</td>
<td>-</td>
<td>1.3μs</td>
</tr>
<tr>
<td>tx_queue_send</td>
<td>0.6μs</td>
<td>1.6μs</td>
<td>1.4μs</td>
<td>2.3μs</td>
</tr>
<tr>
<td>tx_queue_receive</td>
<td>0.5μs</td>
<td>1.6μs</td>
<td>1.7μs</td>
<td>2.6μs</td>
</tr>
<tr>
<td>tx_semaphore_get</td>
<td>0.3μs</td>
<td>1.6μs</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>tx_semaphore_put</td>
<td>0.3μs</td>
<td>-</td>
<td>1.2μs</td>
<td>2.0μs</td>
</tr>
<tr>
<td>tx_mutex_get</td>
<td>0.4μs</td>
<td>1.6μs</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>tx_mutex_put</td>
<td>0.5μs</td>
<td>-</td>
<td>1.8μs</td>
<td>2.6μs</td>
</tr>
<tr>
<td>tx_event_flags_set</td>
<td>0.6μs</td>
<td>-</td>
<td>1.5μs</td>
<td>2.4μs</td>
</tr>
<tr>
<td>tx_event_flags_get</td>
<td>0.6μs</td>
<td>1.8μs</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>tx_block_allocate</td>
<td>0.4μs</td>
<td>1.6μs</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>tx_block_release</td>
<td>0.4μs</td>
<td>-</td>
<td>1.1μs</td>
<td>2.0μs</td>
</tr>
<tr>
<td>tx_byte_allocate</td>
<td>1.2μs</td>
<td>1.6μs</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>tx_byte_release</td>
<td>0.9μs</td>
<td>-</td>
<td>2.7μs</td>
<td>3.4μs</td>
</tr>
</tbody>
</table>

**Context Switch (CS):** 1.0μs  
**Boot Time (BT):** 3.3μs  
**Interrupt Latency Range (ILR):** 0.0μs - 1.1μs

- **Immediate Response (IR):** Time required to process the request immediately, i.e., no thread suspension or thread resumption.
- **Thread Suspend (TS):** Time required to process the request when the calling thread is suspended due to unavailability of the resource.
- **Thread Resumed (TR):** Time required to process the request when a previously suspended thread (of the same or lower priority) is resumed as a result of the request.
- **Thread Resumed and Context Switched (TRCS):** Time required to process the request when a previously suspended higher-priority thread is resumed as a result of the request. Since the resumed thread is higher-priority, a context switch to the resumed thread is also performed from within the request.
- **Context Switch (CS):** Time required to save current thread’s context, find highest priority ready thread, and restore its context.
- **Boot Time (BT):** Time required from `tx_kernel_enter` to the dispatch of the first thread.
- **Interrupt Latency Range (ILR):** Amount of time interrupts are disabled.

*Note: Measurements for RISC-V were not available at press time. These measurements are typical for a 120MHz processor.*
We anticipate commercial availability of ThreadX for RISC-V before the end of 2017.

Please contact Express Logic for further information:

- info@expresslogic.com
- 1-888-THREADX
- 1-858-613-6640
- www.rtos.com

Thank you for your interest.