RTOS for IoT (Scorpius)

Multi-task Real-Time OS for small embedded devices

Overview

Scorpius is an ITRON-like real-time operating system.
High-speed task scheduling operation is provided with small-code size, making this operating system perfect for the needs below.

- To run multi-tasking on small embedded devices such as IOT devices where code size and memory limitations are critical
- To run high-speed task switching even with a powerless CPU
- To implement a multi-task environment without high-level knowledge about OS
- To implement power saving functions that are essential for IoT devices
- To implement a security function to IoT device
- To monitor and check a status of internal operations of the system such as task transitions.
- To consider some technical support for future expansion of software and execution analysis

ScoRPiuS

Features

- Task priority based scheduler
- 0.95 KB kernel size in the smallest configuration
- Approximately 1 us(Note 1) task switching
- Supports all service functions required by applications like small embedded devices
- Easy migration to various processors
- Supports power saving and security functions in the near future
- Trace-log function to visualize internal operations of system
- Basically provided for free (royalty not required)

Specifications

Specifications	
Kernel Code Size(Note 2)	0.95 KBytes (task functions only) to 2.5 KBytes (all functions)
Memory size for Each Task management	Approximately 80 Bytes(Note 3)
Task Dispatch Time	4.2 us(Note 4) with ARM7/40 MHz, 1 us with Cortex-M3/80 MHz
Supported Service Calls(Note 5)	Task, Semaphore, Event Flag, Mailbox, Trace Log, Energy Saving(Note 6), Memory Pool(Note 6), Security(Note 6)
Trace Log Function	Compatible with TOPPERS Project trace log viewer TraceLogVisualizer(Note 7)

Supported Devices

- ARM7
- Cortex-M3/M4
- Nios II (Standard)

Deliverables

- Source code
- User’s manual
- Reference environment

Evaluation Environment

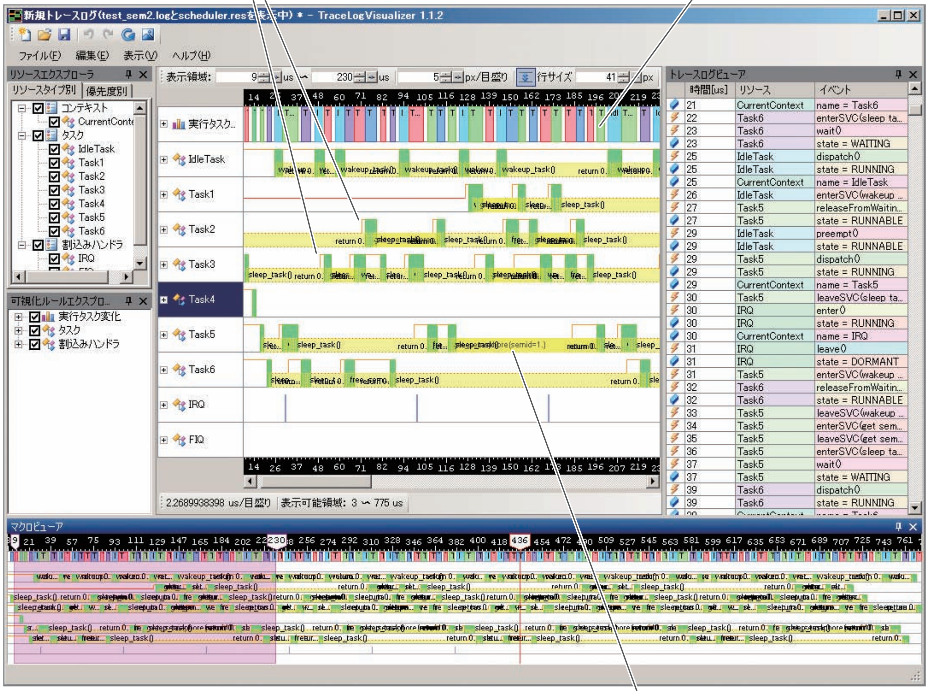
- Silicon Labs EFM32TM Pearl Gecko Starter Kit
- Silicon Labs EFM32TM Giant Gecko Starter Kit
- Analog Devices ADuCM320 Evaluation Board
- Terasic MAX® 10 Nios II Embedded Evaluation Kit

ScoRPiuS

Example Task Transition Display Using TraceLogVisualizer

Transition status of each task at a glance!(Note 8)

Processing time at a glance!(Notes 9, 10)



時間[us]	リソース	イベント
21	CurrentContext	name = Task6
22	Task6	enterSVC(sleep ta...
23	Task6	wait()
23	Task6	state = WAITING
25	IdleTask	dispatch()
25	IdleTask	state = RUNNING
25	CurrentContext	name = IdleTask
26	IdleTask	enterSVC(wakeup ...
27	Task5	releaseFromWaitin...
27	Task5	state = RUNNABLE
29	IdleTask	preempt()
29	IdleTask	state = RUNNABLE
29	Task5	dispatch()
29	Task5	state = RUNNING
29	CurrentContext	name = Task5
30	Task5	leaveSVC(sleep ta...
30	IRQ	enter()
30	IRQ	state = RUNNING
30	CurrentContext	name = IRQ
31	IRQ	leave()
31	IRQ	state = DORMANT
31	Task5	enterSVC(wakeup ...
32	Task6	releaseFromWaitin...
32	Task6	state = RUNNABLE
33	Task5	leaveSVC(wakeup ...
34	Task5	enterSVC(get sem...
35	Task5	leaveSVC(get sem...
36	Task5	enterSVC(sleep ta...
37	Task5	wait()
37	Task5	state = WAITING
39	Task6	dispatch()
39	Task6	state = RUNNING

You can also check the status of each service call!

(Note 1) Measured time required for dispatch processing (task context switching) at Cortex-M3/80 MHz operation.

(Note 2) Compilation result for ARM7 core using Macnica environment.

(Note 3) Memory area size required for every addition of a task.

(Note 4) In the case of typical commercially available RTOS supporting similar CPUs, task dispatch time is around 5 us at 60 MHz

(Note 5) Kernel can be created with minimum required functions only.

(Note 6) Support planned in near future.

(Note 7) Refer to http://www.toppers.jp/en/index.html

(Note 8) Trace log function may change overall system processing order.

(Note 9) Requires implementation of a timer function.

(Note 10) Displays execution time including the trace log function code.