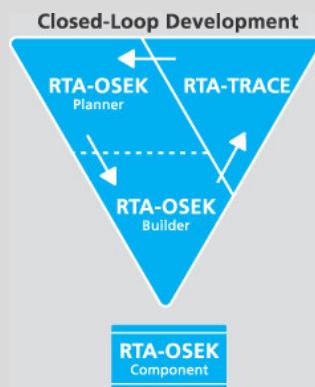


RTA-OSEK

Freescale HC08 with the Cosmic Compiler



Features at a Glance

- OSEK/VDX OS version 2.2 certified OS
- RTOS overhead: 12 bytes RAM, 101 bytes ROM
- Category 2 interrupt latency: 75 CPU cycles
- Applications include: Automotive Body Electronics, Mechatronics, Mechatronics and Industrial Process Control

RTA-OSEK

RTA-OSEK provides an application design environment that combines the smallest and fastest OSEK RTOS with an unique timing analysis tool.

This port data sheet discusses the Freescale HC08 family port of the RTA-OSEK kernel alone and should be read in conjunction with the Technical Product Overview “*Developing Embedded Real-Time Applications with RTA-OSEK*” available from LiveDevices.

The kernel element of RTA-OSEK is a fixed priority, pre-emptive real-time operating system that is compliant to the OSEK/VDX OS standard version 2.2 for all four conformance classes (BCC1, BCC2, ECC1 and ECC2) and intra processor communication using OSEK COM Conformance Classes A and B (CCCA and CCCB).

All CPU overheads of the kernel have low worst case bounds and little variability in execution time. The kernel is particularly suited to systems with very tight constraints on hardware costs and where run-time performance must be guaranteed.

The kernel is configured using an offline tool provided with RTA-OSEK. Determining in advance which features are used allows memory requirements to be minimized and API calls to be optimized for greatest efficiency.

All tasks and ISRs in RTA-OSEK run on a single stack – even extended tasks. This allows dramatic reductions in application stack space requirements.

The RTA-OSEK kernel is designed to be scalable. When a task uses queued activation or waits on events, the additional RTOS overhead required to support these features is paid by the task rather than by the system. This means that a basic single activation task uses the same resources in a BCC1 system as it does in an ECC2 system.

Compiler/Assembler/Linker

The libraries containing the code for the RTA-OSEK kernel have been built using the following tools:

- Cosmic cx6808 Version 4.4b
- Cosmic ca6808 Version 4.2r
- Cosmic clnk Version 4.30

Memory Model

The RTA-OSEK port to the Freescale HC08 with the Cosmic compiler can access variables more efficiently if they are located in 8-bit addressable memory (i.e. in the range 0x0000 to 0x00FF). The RTA-OSEK kernel takes advantage of this to maximize access efficiency for a small number of frequently referenced variables.

This both reduces the size and increases the speed of the kernel, while using only a few bytes of the available zero page RAM.

ORTI Debugger Support

ORTI is the OSEK Run-Time Interface. Currently there are no ORTI compatible debuggers supported by RTA-OSEK for this target.

Hardware Environment

RTA-OSEK supports all variants of the Freescale HC08 family.

Interrupt Model

One level of interrupts is supported.

Floating Point Support

The Freescale 68HC08 uses software floating point only and therefore needs no specific support from the kernel. Any software floating point libraries must be reentrant.

Evaluation Board Support

RTA-OSEK for the Freescale HC08 can be used with any evaluation board. An example application is provided to run on a Hitex AX6811/EVAHC08A evaluation board. This application can be adapted for other target boards by adjusting the linker command file (to alter the RAM locations) and one source file (if alternative output pins are required).

Functionality

The below table outlines the restrictions on the maximum number of operating system objects allowed by RTA-OSEK.

| | BCC1 | BCC2 | ECC1 | ECC2 |
|------------------------|-------------------------|------|------|------|
| Max no of tasks | 16 plus an idle task | | | |
| Max tasks per priority | 1 | 16 | 1 | 16 |
| Max queued activations | 1 | 255 | 1 | 255 |
| Max events per task | n/a | n/a | 8 | 8 |
| Max nested resources | 255 | | | |
| Max alarms | not limited by RTA-OSEK | | | |
| Max standard resources | 255 | | | |
| Max internal resources | not limited by RTA-OSEK | | | |
| Max application modes | 255 | | | |

Note that OSEK specifies that queued activations in an ECC2 system are only possible for basic tasks. Where tasks share a priority level, the maximum number of queued activations per priority level is 255.

The number of alarms, tasksets, schedules and schedule arrival-points is only limited by available hardware resources.

Memory Usage

The memory overhead of RTA-OSEK is:

| Memory type | Overhead (bytes) |
|-------------|------------------|
| RAM | 12 |
| ROM/Flash | 101 |

In addition to the RTOS overhead, each object used by an application has the following memory requirements:

| Object | RAM Bytes | ROM Bytes |
|-------------------|-----------|-----------|
| BCC1 task | 0 | 17 |
| BCC2 task | 5 | 24 |
| ECC1 task | 8 | 28 |
| ECC2 task | 10 | 32 |
| Category 1 ISR | 0 | 0 |
| Category 2 ISR | 0 | 64 |
| Resource | 0 | 10 |
| Internal Resource | 0 | 0 |
| Event | 0 | 1 |
| Alarm | 5 | 24 |
| Counter | 2 | 56 |
| Taskset (RW) | 2 | 2 |
| Taskset (RO) | 0 | 2 |
| Schedule | 7 | 16 |
| Arrivalpoint (RW) | 6 | 6 |
| Arrivalpoint (RO) | 0 | 6 |

In addition to these static memory requirements each task priority and Category 2 interrupt has a stack overhead (in addition to application stack usage). The single stack model means that this overhead applies to each priority level rather than to each task. Similarly, for Category 2 interrupts this overhead applies for each unique interrupt priority. The below table shows stack usage for these objects.

| Object | Stack Bytes |
|----------------------|-------------|
| Task priority level | 19 |
| Category 2 interrupt | 13 |

RTA-OSEK provides an optimization for task termination if the user can guarantee that tasks only terminate from their entry function. Tasks that terminate from elsewhere are not eligible for this optimization and duly require 44 more stack bytes per priority level than indicated in the table above.

Performance

The following table gives the key kernel timings for operating system behavior in CPU cycles.

| Task Type | Basic | Extended | Ref |
|--------------------------------|-------|----------|-----|
| Category 1 ISR Latency | 19 | 19 | K |
| Category 2 ISR Latency | 75 | 75 | A |
| Normal Termination | 106 | 398 | D |
| ChainTask | 294 | 748 | J |
| Pre-emption | 283 | 638 | C |
| Triggered by alarm | 471 | 826 | F |
| Schedule | 222 | 547 | Q |
| ReleaseResource | 263 | 588 | M |
| SetEvent | n/a | 1095 | S |
| Category 2 exit switch latency | 173 | 498 | E |

All performance figures are for the non-optimized interface to RTA-OSEK. Using the optimized interface will result in shorter execution times for some operations. All tasks use lightweight termination and no pre or post task hooks were specified.

The execution time for every kernel API call is available on request from LiveDevices.

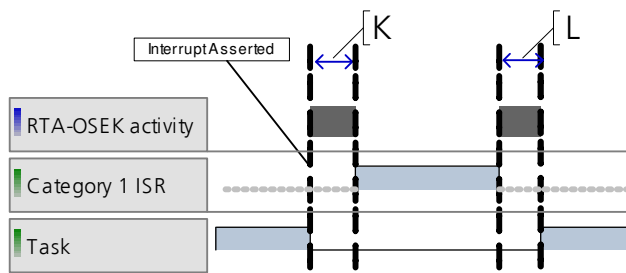


Figure 1 - Category 1 interrupt with return to interrupted task

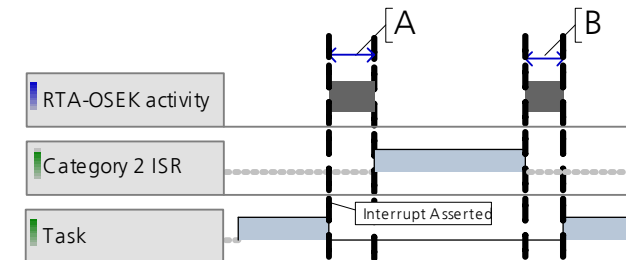


Figure 2 - Category 2 interrupt with return to interrupted task

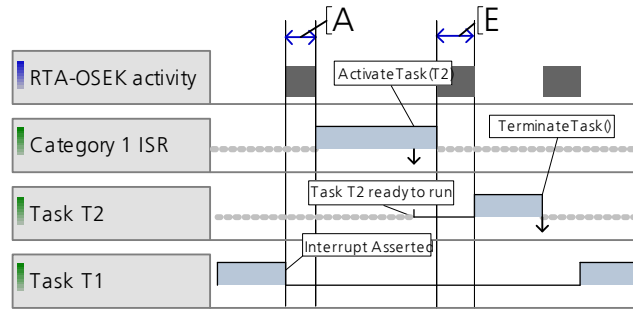


Figure 3 - Category 2 interrupt activates a higher priority task

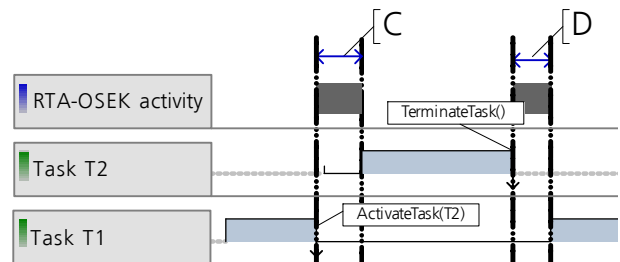


Figure 4 - Task activates a higher priority task

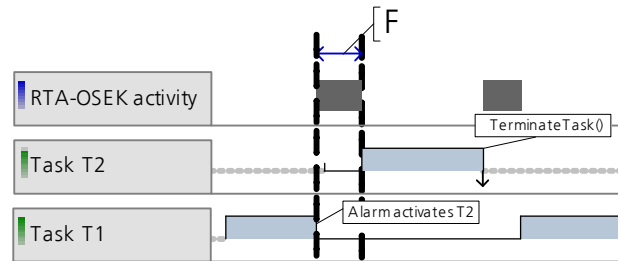


Figure 5 - Alarm activates task

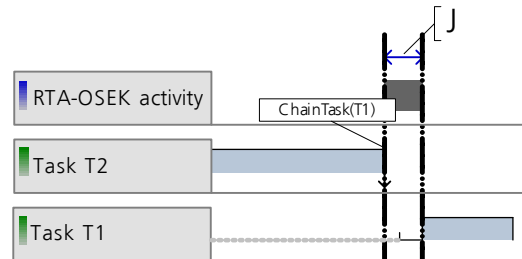


Figure 6 - Task chaining

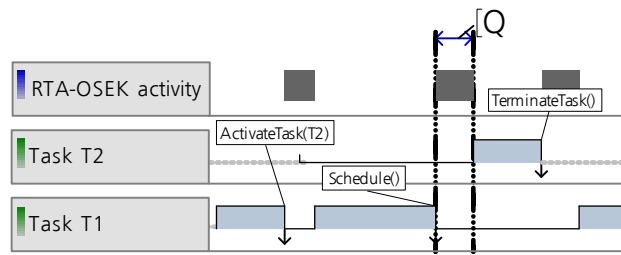


Figure 7 - Schedule() call

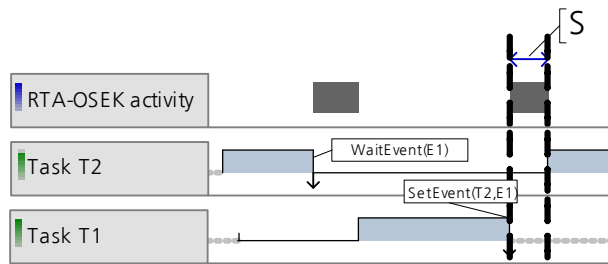


Figure 8 - Activation by SetEvent()

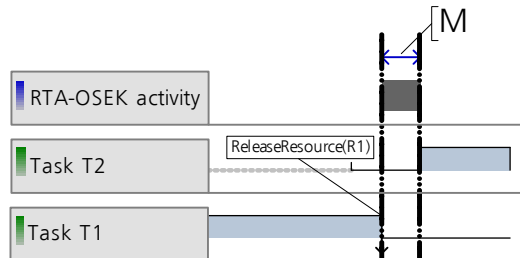


Figure 9 - ReleaseResource()

Benchmarks

The following sections shows benchmarks for RTA-OSEK memory usage for BCC1, BCC2, ECC1 and ECC2 conformant applications. The applications have the following framework:

- 8 tasks plus the idle task
- All basic tasks are lightweight tasks
- 1 Category 2 ISR with a 10ms minimum inter-arrival time
- 1 Counter
- 7 or 8 alarms, all attached to the same counter
- No resources or internal resources
- No hooks
- No schedules
- No tasksets
- Built using standard status

The following table shows the task priority configuration for each benchmark application:

| Task/ISR | Stack (bytes) | Period (ms) | BCC1 | BCC2 | ECC1 | ECC2 |
|----------|---------------|-------------|------|------|------|------|
| ISR1 | 10 | 10 | IPL1 | IPL1 | IPL1 | IPL1 |
| A | 10 | 10 | 8 | 8 | 8 | 8 |
| B | 20 | 20 | 7 | 7 | 7 | 7 |
| C | 30 | 20 | 6 | 6 | 6 | 6 |
| D | 40 | 30 | 5 | 5 | 5 | 5 |
| E | 50 | 50 | 4 | 4 | 4 | 4 |
| F | 60 | 80 | 3 | 3 | 3 | 3 |
| G | 70 | 100 | 2 | 2 | 2 | 2 |
| H | 80 | 150 | 1 | 1 | 1 | 2 |
| Idle | 10 | - | idle | idle | idle | idle |

The overhead figures give the ROM and RAM required for RTA-OSEK in addition to that required by the application. The RAM figure is shown split into RAM data and RAM stack.

BCC1

The BCC1 application uses 8 basic tasks with unique priorities. This application has the following overheads:

| Memory usage | Bytes |
|----------------------|-------------|
| OS ROM | 1506 |
| OS RAM | 210 |
| comprising RAM data | 54 |
| comprising RAM stack | 165 |

BCC2

The BCC2 application uses 8 basic tasks with unique priorities. Tasks A-G are attached to 7 alarms. Task H is activated multiple times from Task A and has maximum queued activation count of 255.

This application has the following overheads:

| Memory usage | Bytes |
|----------------------|-------------|
| OS ROM | 1878 |
| OS RAM | 219 |
| comprising RAM data | 52 |
| comprising RAM stack | 167 |

ECC1

The ECC1 application uses 7 basic tasks and 1 extended task with unique priorities. Task H is the extended task and it waits on a single event that is set by basic tasks A-G.

This application has the following overheads:

| Memory usage | Bytes |
|----------------------|-------------|
| OS ROM | 2481 |
| OS RAM | 236 |
| comprising RAM data | 62 |
| comprising RAM stack | 174 |

ECC2

The ECC2 application uses 6 basic tasks and 2 extended tasks. Tasks G and H are the extended tasks and share a priority. The extended tasks wait on a single event that is set by tasks A-F.

This application has the following overheads:

| Memory usage | Bytes |
|----------------------|-------------|
| OS ROM | 3495 |
| OS RAM | 263 |
| comprising RAM data | 80 |
| comprising RAM stack | 183 |

Stack Optimization

Using stack optimization with the benchmark example identifies that the following tasks can share internal resources:

- Tasks A, B and C
- Tasks D, E and F
- Tasks G and H

The benefit of this optimization is shown in the following table:

| Total Stack Space (bytes) | BCC 1 | BCC 2 | ECC 1 | ECC 2 |
|---------------------------|------------|------------|------------|------------|
| Non-optimized | 524 | 526 | 525 | 548 |
| OS Overhead | 144 | 146 | 145 | 168 |
| Application Overhead | 380 | 380 | 380 | 380 |
| Optimized | 269 | 269 | 270 | 270 |
| OS Overhead | 89 | 89 | 90 | 90 |
| Application Overhead | 180 | 180 | 180 | 180 |

Contact addresses:

LiveDevices Ltd.
 Atlas House
 Link Business Park
 Osbaldwick Link Road
 Osbaldwick
 York YO10 3JB, Great Britain
 Phone +44 (1904) 56 25 80
 Fax +44 (1904) 56 25 81
 info@livedevices.com
 www.livedevices.com

ETAS GmbH
 Borsigstraße 14
 70469 Stuttgart, Germany
 Phone +49 (711) 8 96 61-102
 Fax +49 (711) 8 96 61-106
 sales@etas.de
 www.etas.de

ETAS Inc.
 3021 Miller Road
 Ann Arbor, MI 48103, USA
 Phone +1 (888) ETAS INC
 Fax +1 (734) 997-9449
 sales@etas.us
 www.etas.us

ETAS K.K.
 Queen's Tower C-17F
 2-3-5, Minatomirai
 Nishi-ku
 Yokohama 220-6217, Japan
 Phone +81 (45) 222-0900
 Fax +81 (45) 222-0956
 sales@etas.co.jp
 www.etas.co.jp

ETAS S.A.S.
 1, place des Etats-Unis
 SILIC 307
 94588 Rungis Cedex, France
 Phone +33 (1) 56 70 00 50
 Fax +33 (1) 56 70 00 51
 sales@etas.fr
 www.etas.fr

ETAS Korea Co., Ltd.
 4F, 705 Bldg. 70-5
 Yangjae-dong, Seocho-gu
 Seoul 137-889, Korea
 Phone +82 (2) 57 47-016
 Fax +82 (2) 57 47-120
 sales@etas.co.kr
 www.etas.co.kr

www.etasgroup.com

Subject to changes (07/05)