**Introduction**

[Background]

Third-generation R-Car series products have two types of CPU: Cortex-A57 and Cortex-A53 cores (eight in total for the R-Car H3 and six in total for the R-Car M3-W). In-vehicle information systems require several applications, such as for navigation, the playback of music or video, and the display of information on meters, to run at the same time. A multi-core environment, which includes multiple CPUs, is generally more suitable for such cases. Accordingly, R-Car H3 and R-Car M3-W products are suitable in such cases.

On the other hand, booting up the Cortex-A57 and Cortex-A53 cores with their different performances at the same time creates problems that we must take into account. Namely, since the Linux scheduler assigns tasks to CPUs such that the load on each of the CPUs is equal across the environment, the CPU by which a task is executed may switch between a Cortex-A57 and a Cortex-A53 during its execution. This leads to a problem of applications behaving unstably, including changes of operating speed on the user’s side. For example, this may lead to skipping of music during playback or video playback being interrupted.

Two methods can serve as solutions to this problem. One is applying processor affinity* in the static assignment of tasks. This allows binding a task to a specified type of CPU to prevent another type of CPU from unexpectedly having to handle the task. The other is applying energy aware scheduling (EAS) in the dynamic assignment of tasks. This is provided as an enhanced function of the standard scheduler of the Linux OS and allows the scheduler to automatically judge the processing loads given tasks impose (which depend on the behavior of applications) and assign each task to an optimal CPU. This prevents unstable behavior by applications. This document describes EAS, the latter solution.

Note:  * For details on processor affinity, see the separate application note “R-Car Series, 3rd Generation ApplicationNote Processor Affinity”.

[Purpose]

At the time of writing (as of October in 2016), EAS is a work in progress by the Linux community. Renesas will not formally support EAS until it’s officially posted by the upstream parties. Therefore, on the assumption that users will request an environment to evaluate EAS in advance, we provide this document as a guide to the advance evaluation of EAS. For this reason, the document only covers the EAS operations in outline, how to apply the patch set, and examples of applying EAS.

[Target Readers]

Readers of this document are assumed to have general knowledge in the fields and specific technologies listed below.

- Engineering, logic circuits, microcontrollers, and Linux.
- The functionality of the multiple processor cores of R-Car H3 and R-Car M3-W products.
- The electrical specifications of the multiple processor cores of R-Car H3 and R-Car M3-W products.
- The functions of the BSP drivers for R-Car H3 and R-Car M3-W products.
[Notes]

- Statements in relation to operating systems in this document only apply to Yocto v2.12.0 from Renesas (for the R-Car H3 and R-Car M3-W) and not to any other versions of Yocto.
- As EAS is a prototype which the Linux community is developing at the time of writing (as of October of 2016), it may have problems. It may also be subject to change, depending on the thinking of the Linux community.
- The environment in which EAS is applied in the descriptions of this document was built in order to evaluate EAS in advance and its quality is thus equivalent to that of a prototype. In addition, Renesas does not support the application of EAS in your environment. If you do so, it is at your own responsibility.

**Target Device**

- R-Car H3
- R-Car M3-W
Contents

1. Energy Aware Scheduling (EAS) ................................................................. 4
   1.1 Overview ......................................................................................... 4
      1.1.1 Overview of EAS .................................................................. 4
      1.1.2 State of EAS Development .................................................. 5
      1.1.3 Potential Problems with EAS ............................................... 5
   1.2 Operation in Outline ....................................................................... 6
   1.3 Configuration of Modules ............................................................... 8
   1.4 Energy Model ................................................................................ 9
      1.4.1 Calculating Values for Use in Energy Models .................... 10
   1.5 Further Items to Consider .......................................................... 11
      1.5.1 Giving Higher Priority to the Execution of Desired Applications .... 11

2. Building an Environment in which EAS is Applied .................................. 12
   2.1 Patch Set ....................................................................................... 12
   2.2 Procedure for Building .................................................................. 13

Appendix .................................................................................................. 14

A1. Examples of Applying EAS ................................................................. 14
   A1.1 Overview ..................................................................................... 14
   A1.2 Effects of EAS in Changing the Assignment of Tasks .................. 15
   A1.3 Summary of Results ................................................................... 15

A2. Times for Task Switching between the Cortex-A57 and Cortex-A53 Cores ...... 16

A3. Using EAS with Other Functions Related to Power Control ..................... 17
   A3.1 Overview ..................................................................................... 17
   A3.2 Using the “CPU Idle” Function (T.B.D.) ....................................... 18
   A3.3 Using the “CPU Freq” Function (T.B.D.) ...................................... 19
   A3.4 Using Intelligent Power Allocation (IPA) and Emergency Shutdown (EMS) .......... 20
      A3.4.1 State of Support for IPA and EMS with EAS ..................... 20
      A3.4.2 Applying EMS on the Cortex-A53s ..................................... 20
1. Energy Aware Scheduling (EAS)

1.1 Overview

1.1.1 Overview of EAS

EAS is an enhanced scheduler based on the completely fair scheduler (CFS)* and is for obtaining lower power consumption. It supports environments in which Cortex-A57 and Cortex-A53 cores are booted up at the same time. The major difference between CFS and EAS is the method of scheduling. The CFS assigns tasks to CPUs such that the load on each CPU is equal, while EAS assigns heavy-load tasks to the Cortex-A57s and low-load tasks to the Cortex-A53s. Furthermore, EAS applies control so that only the Cortex-A53 cores operate and places the Cortex-A57 cores in the idle state while no heavy-load tasks are running. This reduces power consumption. Namely, EAS handles a kind of low power consumption mode for multi-core processing.

A comparison of scheduling methods applicable in environments where the Cortex-A57 and Cortex-A53 cores are booted up at the same time is given in table 1-1.

Note: * The completely fair scheduler (CFS) is the standard scheduler for the Linux OS.

<table>
<thead>
<tr>
<th>Method</th>
<th>Advantages</th>
<th>Disadvantages</th>
<th>Recommendation from Renesas</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFS (the standard Linux scheduler)</td>
<td>The scheduler assigns tasks to CPUs such that the load on each CPU is equal.</td>
<td>Operation of the CFS is on the assumption of a symmetric multiprocessing (SMP) architecture, so its operation is not suitable in an environment where the Cortex-A57 and Cortex-A53 cores are operating at the same time.</td>
<td>We do not recommend its use in an environment in which Cortex-A57 and Cortex-A53 cores are operating at the same time.</td>
</tr>
<tr>
<td>Processor affinity</td>
<td>Processor affinity binds tasks to specified CPUs.</td>
<td>The assignment of tasks requires consideration and design from the viewpoint of the system as a whole.</td>
<td>We recommend applying this as a solution if you are able to design and control the assignment of tasks from the viewpoint of the system as a whole.</td>
</tr>
<tr>
<td>EAS</td>
<td>EAS allows the scheduler to automatically assign tasks to CPUs that are suitable for the loads of the respective tasks.</td>
<td>Continuous maintenance, such as continuously obtaining the latest code under development, is required to secure quality approaching suitability for commercial use.</td>
<td>You can only apply EAS as a solution with its support and maintenance as your own responsibility.</td>
</tr>
</tbody>
</table>
1.1.2 State of EAS Development
EAS was investigated and is under development by the Linux community. The latest patch set (as of October in 2016) is EAS v5.2. Forthcoming version updates and their timing depend on the Linux community. Note that Renesas does not formally support EAS during its development, but plan to formally support it after a mature version is posted upstream.

![Progress of EAS Development](image)

1.1.3 Potential Problems with EAS
- Even if the system as a whole imposes a heavy load, tasks may not be assigned to some of the CPUs.*
- When a heavy-load task is awakened, the task may be temporarily assigned to a Cortex-A53 core. This is likely to result in overly long times for processing.*

Note: * These restrictions apply to EAS v5.2. Resolution of these potential problems depends on the Linux community.
1.2 Operation in Outline

EAS when applied to assigning tasks handles the following functions, including some that are incidental to EAS.

- Monitoring how much load is being imposed on each of the CPU cores
- Dynamically assigning tasks that have been awakened and can be executed to suitable CPU cores
- Assigning tasks that have been recognized as low-load to low-power-consumption CPU cores as far as possible so that high-performance and high-power-consumption CPU cores can be placed in the idle state. This reduces power consumption of the system as a whole.

An outline of the principle of operation is given below.

EAS assigns tasks as follows. The design of EAS is based on a concept that scheduling proceeds so as to minimize power consumption while securing the specified processing performance of the system. Assigning tasks with the processing speed and power consumption of CPUs taken into account is a different approach from previous concepts. Accordingly, this enables scheduling that is appropriate in a system that runs with a combination of two types of CPU (the Cortex-A57 and Cortex-A53 cores) with their asymmetrical processing speeds and levels of power consumption.

- Basic algorithm used to assign tasks
  - Step 1: Select the CPUs that are currently idle and have the processing speed to suit the load that a given task imposes.
  - Step 2: Assign the task to the CPU with the lowest power consumption from among the CPUs selected in step 1.

- Processing speed and power consumption of CPUs
  - In EAS, a description of the processing speed and power consumption of the CPUs is referred to as an energy model. An energy model consists of the parameters required in scheduling and must be defined for each CPU configuration. For details, see section 1.4.

Figure 1-2 is a schematic view of the application of EAS to the dynamic assignment of tasks.

Note: The values in the figure are for reference, so do not have particular meanings.
Figure 1-2  Schematic View of Applying EAS to the Assignment of Tasks
1.3 Configuration of Modules

1. Block Diagram of Configuration of Software Modules

![Diagram of Configuration of Software Modules](image)

**Figure 1-3** Configuration of Modules in an Environment where EAS is Applied

2. Overview of Software Modules
   - CPU topology: Controls information on the CPU configuration.
   - EAS: An enhanced scheduler for configurations with combinations of Cortex-A57 and Cortex-A53 cores
   - CPU freq: Changes the operating frequency and voltage of the CPUs (through the schedutil gov).

3. Processing in Outline
   1. An energy model is a table of information on the processing speeds (in DMIPS) and levels of power consumption (in mW) of the individual CPUs. It is created at initialization of the kernel and used to make judgements on scheduling.
   2. EAS uses the energy model to assign tasks to CPUs as to minimize power consumption while still securing the specified processing performance of the system.
   3. EAS notifies the “CPU freq” module of events each time scheduling proceeds. The “CPU freq” module uses information on the load on the given CPU in dynamically changing its operating frequency. This processing is part of the standard way the Linux OS operates.
1.4 Energy Model

An energy model is an information table including definitions of the processing speeds (in DMIPS) and levels of power consumption (in mW) of the individual CPUs in an SoC and thus depend on the characteristics of the SoC. This information table is used to schedule tasks. Accordingly, appropriate values must be specified in the table. Note that the patch set described in section 2 includes correction patch files for the energy models for use in WS 1.1 of the R-Car H3 and WS 1.0 of the R-Car M3-W. Use the settings in these files in your environment.

For reference, sample settings (for WS1.1 of the R-Car H3) are given below. Values here are not measured values, but are relative values derived from measurements.

<1> Definitions of power consumption in the idle state for each cluster

```c
static struct idle_state idle_states_cluster_h3_a53[] = {
    { .power = 17 }, /* arch_cpu_idle() = WFI */
    { .power = 17 }, /* WFI */
    { .power = 0 }, /* cpu-sleep-0 */
    { .power = 0 }, /* cluster-sleep-0 */
};

static struct idle_state idle_states_cluster_h3_a57[] = {
    { .power = 98 }, /* arch_cpu_idle() = WFI */
    { .power = 98 }, /* WFI */
    { .power = 0 }, /* cpu-sleep-0 */
    { .power = 0 }, /* cluster-sleep-0 */
};
```

<2> Definitions of processing capacity usage and levels of power consumption in the active state for each cluster

```c
static struct capacity_state cap_states_cluster_h3_a53[] = {
    /* Power per cluster */
    { .cap = 383, .power = 17, }, /* 1200 MHz */
};

static struct capacity_state cap_states_cluster_h3_a57[] = {
    /* Power per cluster */
    { .cap = 310, .power = 66, }, /* 500 MHz */
    { .cap = 560, .power = 82, }, /* 1000 MHz */
    { .cap = 903, .power = 98, }, /* 1500 MHz */
    { .cap = 964, .power = 126, }, /* 1600 MHz */
    { .cap = 1024, .power = 154, }, /* 1700 MHz */
};
```
<3> Definitions of power consumption in the idle state for each CPU

```c
static struct idle_state idle_states_core_h3_a53[] = {
    { .power = 17 }, /* arch_cpu_idle() = WFI */
    { .power = 17 }, /* WFI */
    { .power = 0 }, /* cpu-sleep-0 */
    { .power = 0 }, /* cluster-sleep-0 */
};

static struct idle_state idle_states_core_h3_a57[] = {
    { .power = 148 }, /* arch_cpu_idle() = WFI */
    { .power = 148 }, /* WFI */
    { .power = 0 }, /* cpu-sleep-0 */
    { .power = 0 }, /* cluster-sleep-0 */
};
```

<4> Definitions of processing capacity usage and levels of power consumption in the active state for each CPU

```c
static struct capacity_state cap_states_core_h3_a53[] = {
    /* Power per cpu */
    { .cap = 383, .power = 85, }, /* 1200 MHz */
};

static struct capacity_state cap_states_core_h3_a57[] = {
    /* Power per cpu */
    { .cap = 310, .power = 361, }, /* 500 MHz */
    { .cap = 560, .power = 590, }, /* 1000 MHz */
    { .cap = 903, .power = 820, }, /* 1500 MHz */
    { .cap = 964, .power = 1116, }, /* 1600 MHz */
    { .cap = 1024, .power = 1344, }, /* 1700 MHz */
};
```

Notes:
1. This table is set according to the frequency table of the CPUs. Accordingly, changes to parts of the frequency table require updating of this table.
2. This table is defined in a statement in the arch/arm64/kernel/topology.c file for the kernel.

1.4.1 Calculating Values for Use in Energy Models
Calculating values for these information tables requires the advance measurement of values for performance and current. Renesas has measured these in accord with guidelines from Linaro (refer to the Web page at the URL below).

http://www.slideshare.net/linaroorg/bkk16tr08-how-to-generate-power-models-for-eas-and-ipa
1.5 Further Items to Consider

Applying EAS may require further considerations to be taken into account to suit the use case in which you are to apply it. Examples of such items are given in section 1.5.1.

1.5.1 Giving Higher Priority to the Execution of Desired Applications

As described in section 1.2, EAS involves examining the processing speeds and levels of power consumption of the CPUs to determine which type of CPU is suitable for the load that a given task imposes. EAS then assigns the task to the selected type of CPU. It does this with reference to the state of the load of the task to be executed. Note that the states of loads change, such as due to temporarily being lowered. In some actual use cases, however, users may wish to give a task higher priority throughout its execution, or to have it run solely on Cortex-A57 cores independently of the state of the load. In such cases, consider additional countermeasures for this in the environment where you are applying EAS. Table 1-2 lists some use cases and measures in response for reference.

Table 1-2 Additional Measures when EAS is being Applied

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Countermeasure</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Executing a given task solely on Cortex-A57</td>
<td>Bind the task to the Cortex-A57 cores through any of the following methods:</td>
<td>For more details on cgroup, the taskset command, and the sched_setaffinity function, see the separate application note with the filename “R-Car series 3rd Generation Application Note Processor Affinity”.</td>
</tr>
<tr>
<td>cores</td>
<td>setting up cgroup, issuing the taskset command, or calling the sched_setaffinity function</td>
<td></td>
</tr>
<tr>
<td>Always giving the highest priority to a</td>
<td>Changing the attribute of the task to a realtime process*</td>
<td>Note that creating too many realtime processes in the system may lead to the other processes not being executed.</td>
</tr>
<tr>
<td>particular task for execution</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: * Realtime processes are always given higher priority in scheduling than normal processes.

The pthread_create system call or the chrt command, etc. can be used to make a process realtime at the time of its generation, and the sched_setscheduler system call or the chrt command, etc. can be used to give a process the realtime attribute after its creation.
2. Building an Environment in which EAS is Applied

2.1 Patch Set

1. URL for Downloading the EAS Patch Set
   The EAS patch set being evaluated by Renesas is released on the git repository* below.

   git://github.com/renesas-rcar/meta-renesas-power.git

Note: * This EAS patch set was created by porting the git repository at the URL below for use on R-Car H3 and R-Car M3-W products.
   http://www.linux-arm.org/git?p=linux-power.git (branch: energy_model_rfc_v5.2)

2. Functions in the EAS Patch Set that Have been Evaluated by Renesas for Use on R-Car H3 and R-Car M3-W Products

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Function Evaluated or Not?</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAS core</td>
<td>✓</td>
<td>Controls the assignment of tasks by EAS.</td>
<td></td>
</tr>
<tr>
<td>Capacity awareness</td>
<td>✓</td>
<td>When a task is awakened, if it imposed a heavy load while it was previously running, it will be assigned to a CPU with higher processing capacity (a Cortex-A57).</td>
<td></td>
</tr>
<tr>
<td>CPU topology for aarch32</td>
<td>—</td>
<td>Defines the configuration of the CPUs of the AArch32 platform.</td>
<td>Third-generation R-Car series products do not support AArch32.</td>
</tr>
<tr>
<td>CPU topology for aarch64</td>
<td>✓</td>
<td>Defines the configuration of the CPUs of the AArch64 platform.</td>
<td></td>
</tr>
<tr>
<td>Sched governor</td>
<td>—</td>
<td>The Sched governor is the CPU freq governor for v5.2 of EAS. This is different from the new CPU freq governor, Schedutil, which has already been merged with the BSP for third-generation R-Car series products.</td>
<td>Sched governor is old and no longer necessary.</td>
</tr>
<tr>
<td>Sched tune</td>
<td>—</td>
<td>Sched tune is for boosting the performance of the Sched governor as CPU freq.</td>
<td>This does not work with the schedutil governor.</td>
</tr>
<tr>
<td>Dynamic capacity scaling</td>
<td>—</td>
<td>Adjusting the processing capacities of CPUs by controlling the operating frequency</td>
<td>This does not work with the schedutil governor.</td>
</tr>
</tbody>
</table>
2.2 Procedure for Building

1. Preparation in Advance
   
   Step 1: Proceed with all build processes according to the procedure in the separate guidelines with the filename "RENESAS_RCH3M3_YoctoStartupGuide".
   
   Step 2: Change the settings according to the procedure in the separate application note with the filename "R-CarGen3_ApplicationNote_Processoraffinity" to boot all CPUs up.

2. Instructions for Building EAS
   
   Step 1: Download the EAS patch set and load it into the working directory for building the Yocto environment you have created in the advance preparations above, and check it out.

   ```
   $ cd $WORK
   $ git clone git://github.com/renesas-rca/meta-renesas-power.git
   $ cd $WORK/meta-renesas-power
   
   $ cd $WORK
   $ source poky/oe-init-build-env
   
   $ cd $WORK/build
   
   [For Wayland]
   bitbake core-image-weston
   
   [For BSP only]
   bitbake core-image-minimal
   
   Path for image output: $WORK/build/tmp/deploys/images/salvator-x/ directory.
   ```
Appendix

A1. Examples of Applying EAS

A1.1 Overview

The result of applying EAS in an in-vehicle infotainment (IVI) use case in a demonstration (in a demonstration with three monitors) is described here as an example. In this use case, several applications are running at the same, so the load is heavy. In addition, the use case is close in configuration to an integrated cockpit, which is the target use case for the R-Car H3. These are the two reasons why we selected the triple monitor demo as a sample use case. The results of evaluation, including the differences between the existing environment (i.e., one where EAS was not applied) and an environment where EAS is applied, are given on the following pages.

1. Configuration

![Schematic View of the Connection of Three Monitors for the Demonstration](image)

Figure A1-1  Schematic View of the Connection of Three Monitors for the Demonstration

2. Comparison of Environments

| Table A1-1 Comparison of Specifications |
|-------------------------------|-------------------------------|
| **Existing Environment**     | **Environment where EAS is Applied** |
| SoC   | R-Car H3 (WS1.1) | R-Car H3 (WS1.1) |
| CPU   | Cortex-A57 × 4, 1.5 GHz | Cortex-A57 × 4, 1.5 GHz  
Cortex-A53 × 4, 1.2 GHz | Cortex-A57 × 4, 1.5 GHz  
Cortex-A53 × 4, 1.2 GHz |
| RAM   | DDR4-2400, 2-ch. split | DDR4-2400, 2-ch. split |
| GPU   | PowerVR GX6650, 600 MHz | PowerVR GX6650, 600 MHz |
| Display Center | LVDS, 1024 × 768 | LVDS, 1024 × 768 |
| Instruments | HDMI, 1920 × 1080 | HDMI, 1920 × 1080 |
| Video | HDMI, 1920 × 1080  
MP4 (H.264) VCP decoding | HDMI, 1920 × 1080  
MP4 (H.264) VCP decoding |
### A1.2 Effects of EAS in Changing the Assignment of Tasks

The changes to the CPU utilization and task assignment after applying EAS are given in table A1-2.

<table>
<thead>
<tr>
<th>CPU Utilization</th>
<th>Task Assignment</th>
<th>CPU Utilization</th>
<th>Task Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU0 (Cortex-A57)</td>
<td>48.3%</td>
<td>ivi_infotainment, etc</td>
<td>10.1%</td>
</tr>
<tr>
<td>CPU1 (Cortex-A57)</td>
<td>44.2%</td>
<td>ivi_instrument_cluster</td>
<td>8.9%</td>
</tr>
<tr>
<td>CPU2 (Cortex-A57)</td>
<td>43.2%</td>
<td>ivi_camera_daemon</td>
<td>9.4%</td>
</tr>
<tr>
<td>CPU3 (Cortex-A57)</td>
<td>44.0%</td>
<td>ivi_passenger, weston, ivi_daemon</td>
<td>9.6%</td>
</tr>
<tr>
<td>CPU4 (Cortex-A53)</td>
<td>—</td>
<td>—</td>
<td>52.2%</td>
</tr>
<tr>
<td>CPU5 (Cortex-A53)</td>
<td>—</td>
<td>—</td>
<td>40.8%</td>
</tr>
<tr>
<td>CPU6 (Cortex-A53)</td>
<td>—</td>
<td>—</td>
<td>39.9%</td>
</tr>
<tr>
<td>CPU7 (Cortex-A53)</td>
<td>—</td>
<td>—</td>
<td>44.7%</td>
</tr>
</tbody>
</table>

Note: The values given here are for reference.

This result indicates that low-load tasks, which were judged to be able to run sufficiently well on Cortex-A53 cores, have automatically been assigned to Cortex-A53 cores in an environment with task assignment controlled by EAS.

### A1.3 Summary of Results

The results in terms of the levels of CPU utilization and values for current drawn are given in table A1-3.

<table>
<thead>
<tr>
<th>CPU Utilization</th>
<th>Task Assignment</th>
<th>CPU Utilization</th>
<th>Task Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zϕ (for Cortex-A57s)</td>
<td>1500 MHz</td>
<td>1500 MHz</td>
<td>—</td>
</tr>
<tr>
<td>Z2ϕ (for Cortex-A53s)</td>
<td>—</td>
<td>1200 MHz</td>
<td>—</td>
</tr>
<tr>
<td>Cortex-A57 utilization</td>
<td>44.9%</td>
<td>11.1%</td>
<td>- 33.8%</td>
</tr>
<tr>
<td>Cortex-A53 utilization</td>
<td>0%</td>
<td>44.4%</td>
<td>+ 44.4%</td>
</tr>
<tr>
<td>GPU utilization</td>
<td>60% to 70%</td>
<td>60% to 70%</td>
<td>±0%</td>
</tr>
<tr>
<td>DVFS 0.8 V (Cortex-A57s + GPU)</td>
<td>3000 mA</td>
<td>2340 mA</td>
<td>- 660 mA</td>
</tr>
<tr>
<td>VDD 0.8 V (Cortex-A53s + other modules)</td>
<td>4980 mA</td>
<td>5380 mA</td>
<td>+ 400 mA</td>
</tr>
<tr>
<td>Current (DVFS 1.0 + VDD 1.0)</td>
<td>7980 mA</td>
<td>7720 mA</td>
<td>- 260 mA</td>
</tr>
</tbody>
</table>

Note: The values given here are for reference.

The result of this example indicates that applying EAS in an existing environment allows applications from the existing environment to run normally in an environment where the Cortex-A57 and Cortex-A53 cores are booted up at the same time. It also indicates that applying EAS only produced a small reduction in current drawn, but increased the spare processing capacities of the Cortex-A57 cores. This may allow additional applications to run.
A2. Times for Task Switching between the Cortex-A57 and Cortex-A53 Cores

Table A2-1 shows the times for task switching.

<table>
<thead>
<tr>
<th>Straddling of clusters</th>
<th>Cortex-A57 -&gt; Cortex-A57</th>
<th>Cortex-A57 -&gt; Cortex-A53</th>
<th>Cortex-A53 -&gt; Cortex-A57</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same cluster, so none</td>
<td>28 µs</td>
<td>33 µs</td>
<td>35 µs</td>
</tr>
</tbody>
</table>

<Measurement environment>
- Software: Yocto v2.12.0 (Linux BSP 3.3.2)
- SoC: R-Car H3 WS1.1 (on a Salvator-X board)
- Measurement method
  1. Awaken any executable process (process A) which endlessly loops on CPU1.
  2. Use the taskset command* to switch the CPU on which process A runs to CPU2.

Note: * taskset is a command used to change the settings for assigning processes to CPUs.

Figure A2-1 Flow of Task Switching
A3. Using EAS with Other Functions Related to Power Control

A3.1 Overview

This subsection describes how to use EAS with other functions related to power control at the same time for further lowering the power consumption in environments where EAS is being applied. Table A3-1 describes how each of the functions behaves in such an environment. All functions are able to operate with the Cortex-A57s, and can be used in the same way in the existing environment. On the other hand, some functions will only be made able to operate with the Cortex-A53s through individual extensions. In the first place, however, the Cortex-A53 is a low-power-consumption core in comparison with the Cortex-A57. Accordingly, even if we attempt to use any of the functions to save power on the Cortex-A53s, their effects may easily be assumed to be extremely small from the viewpoint of the system as a whole. For this reason, the functions related to power control, including “CPU idle”, “CPU freq”, intelligent power allocation (IPA), and emergency shutdown (EMS), are disabled for the Cortex-A53s in the EAS evaluation environment from Renesas.

Table A3-1 Support for Functions Related to Power Control

<table>
<thead>
<tr>
<th>Function</th>
<th>Cortex-A57</th>
<th>Cortex-A53</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU hotplug</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>CPU idle</td>
<td>√</td>
<td>Δ</td>
<td>This version does not support the “CPU idle” function because of a functional limitation. The function will be able to operate with later versions.</td>
</tr>
<tr>
<td>CPU freq</td>
<td>√</td>
<td>Δ</td>
<td>This version does not support the “CPU freq” function on the Cortex-A53 because it has not been evaluated. The function will be able to operate with later versions following individual extensions.</td>
</tr>
<tr>
<td>System Suspend to RAM</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>IPA</td>
<td>√</td>
<td>Δ</td>
<td>This version does not support the IPA function on the Cortex-A53 because it has not been evaluated. The function will be able to operate with later versions following individual extensions.</td>
</tr>
<tr>
<td>EMS</td>
<td>√</td>
<td>Δ</td>
<td></td>
</tr>
</tbody>
</table>

Note: * For details on each of the functions above, see the separate document with the filename “RENASAS_RCH3M3_PowerManagement_UME”.
A3.2 Using the “CPU Idle” Function (T.B.D.)
A3.3 Using the “CPU Freq” Function (T.B.D.)
A3.4 Using Intelligent Power Allocation (IPA) and Emergency Shutdown (EMS)

A3.4.1 State of Support for IPA and EMS with EAS

The IPA and EMS functions change power to the CPUs to control the generation of heat in the SoC. For the Cortex-A57, both IPA and EMS are able to operate with EAS. For the Cortex-A53, IPA is not usable because it has not been evaluated and is not supported by the standard BSP. EMS may be made to operate by extending its functionality.

A3.4.2 Applying EMS on the Cortex-A53s

Define CPUs (Cortex-A53 cores) to be shut down when EMS starts in a statement in the device tree file of the kernel. An example of the settings is shown below.

```plaintext
arch/arm64/boot/dts/renesas/r8a7795.dtsi

emergency {
    polling-delay = <1000>;
    on-temperature = <110000>;
    off-temperature = <95000>;
    target_cpus = <&a57_1>,
                  <&a57_2>,
                  <&a57_3>,
                  <&a53_0>,  <---- CPU4 will be shut down when EMS starts.
                  <&a53_1>,  <---- CPU5 will be shut down when EMS starts.
                  <&a53_2>,  <---- CPU6 will be shut down when EMS starts.
                  <&a53_3>;  <---- CPU7 will be shut down when EMS starts.
    status = "disabled";
};
```

Note: An individually extended function other than those of the standard Yocto package is used to change the settings in this case. Renesas does not support this extended function. Use it at your own responsibility.
## Revision History

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Page</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>July, 2017</td>
<td>—</td>
<td>First edition issued</td>
</tr>
</tbody>
</table>
General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Handling of Unused Pins
   Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.
   - The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on
   The state of the product is undefined at the moment when power is supplied.
   - The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
     In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
     In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   - The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   - When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products
   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.
   - The characteristics of Microprocessing unit or Microcontroller unit products in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.
Notice

1. Descriptions of circuits, software and other related information in this document is provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for your incorporation or, other use of the circuits, software and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.

2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other disputes involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawing, program, algorithm, application notes, or other related documents.

3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.

4. You shall not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics products.

5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product quality grade, as indicated below:
   - "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots, etc.
   - "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial derivatives; aviation safety control equipment; etc. Renesas Electronics products are intended for use in environments where the failure of parts may pose a direct threat to human life or bodily injury (life-threatening) or cause serious damage or disorders (surgical complications) or may cause severe property damages (severe and widespread repetitive; nuclear power control systems; strong system controls; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.

6. When using the Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.) and ensure that usage conditions are within the ranges specified by Renesas Electronics without making any modifications, operating under power supply voltage range, heat radiation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions or failure occurring out of the use of Renesas Electronics products beyond such specified ranges.

7. Although Renesas Electronics endeavours to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions, under certain usage conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please ensure to implement safety measures to guard against the possibility of bodily injury, injury, or damage caused by fire, and social damage in the event of failure or malfunction of Renesas Electronics products, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other essential measures by your own means as warranty for your products/system. Because the evaluation of microcomputer software alone is very difficult and not practical, please evaluate the safety of the final products or system manufactured by you.

8. Please contact a Renesas Electronics sales office for details as to environmental stresses such as the environmental compatibility of each Renesas Electronics product. Please investigate applicable laws and regulations that regulate the restriction or control of hazardous substances, including without limitation, the EU RoHS Directive carefully and sufficiently and use Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.

9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems where a malfunction, use, or sale is prohibited under any applicable domestic or foreign laws, or any other laws. Renesas Electronics products or technologies for (1) any purpose relating to the development, design, manufacture, use, stocking, etc. of weapons of mass destruction, such as nuclear weapons, chemical weapons, biological weapons, or missiles (including conventional aerial vehicles (CAVs) for carrying such weapons), (2) any purpose relating to the development, design, manufacture, or use of conventional weapons, or (3) any other purpose of causing international peace and security, and you shall not sell, export, lease, transfer, or reissue Renesas Electronics products or technologies to any third party without obtaining prior written consent from Renesas Electronics. The third party must agree not to use or disclose the Renesas Electronics products or technologies to any third party unless the third party agrees in writing not to use or disclose the Renesas Electronics products or technologies to any third party.

10. Please acknowledge and agree that you shall bear all the losses and damages which are incurred from the misuse or violation of the terms and conditions described in this document, including the notice, and hold Renesas Electronics harmless, if such induce or result from your resale or making Renesas Electronics products available to any third party.

11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.

12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

Note 1: "Renesas Electronics" as used in this document means Renesas Electronics Corporation and all its affiliates, majority-owned subsidiaries.

Note 2: "Renesas Electronics products" means any product developed or manufactured by or for Renesas Electronics.