R-Car Series, 3rd Generation

Energy Aware Scheduling

要旨

[Background]

R-Car H3/M3-W/M3-W+はCortex-A57とCortex-A53の2種類で計8個(H3)/6個(M3-W/M3-W+)のCPUを搭載しております。車載情報システムのような、ナビゲーション、音楽再生、動画再生、メーター表示等、複数のアプリケーションを同時に動作させる場合には、CPUの個数が多いマルチコアプロセッシングの環境の方が快適に動作することが一般的であり、本利用シーンではR-Car H3/M3-W/M3-W+の活用が適しています。

一方で、処理性能が異なるCortex-A57とCortex-A53を同時起動する場合は考慮すべき弊害が生じます。Linux OSのスケジューラは各CPUに対して処理負荷が均等になるようにタスクを割り付けるため、本環境下で動作するタスクはCortex-A57とCortex-A53の間を切り替わりながら動作する場合があります。ユーザ視点では、アプリケーションの挙動が急に早くなったり遅くなったり不安定になるため、動画再生でカクツキが発生したり、音楽再生で音飛びが発生する可能性があります。

本問題に対するソリューションは２つあります。１つ目はProcessor Affinity(\*)を用いたタスクの静的割り付けです。これはタスクを指定したCPUに固定的に割り付けるため、意図しないCPUへの切り替えが発生しなくなります。２つ目はEnergy Aware Scheduling(EAS)によるタスクの動的割り付けです。これはLinux OSの標準スケジューラを拡張する対応で、スケジューラがタスクの処理量(アプリケーション側の挙動に依存)を自動で判別して適切なCPUへ割り付けるため、アプリケーションの挙動が不安定になることはありません。本ドキュメントでは、２つ目のソリューションであるEASについて説明します。

(\*) Processor Affinityについては「R-Car Series, 3rd Generation ApplicationNote\_Processor Affinity」をご参照ください。

[Purpose]

現在(2016/10)、EASはLinux Communityで開発途中のプロトタイプ版であり、弊社ではUpstreamに取り込まれるまでは正式サポートしない方針です。そのため、お客様からEASを先行検証するため評価環境の提供をご要求頂く場合を想定して、EASの先行検証を目的としたガイドとして本ドキュメントを作成しました。従って、本ドキュメントの記載範囲はEASの動作概要、パッチの適用方法、EASの適用事例のみとなります。

[Readers]

本ドキュメント読者は、下記の一般的な知識を持っていることを想定しています。

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

[Note]

* 本ドキュメントは弊社が提供するYocto v2.12.0(R-Car H3/M3-W )を対象にしているため、他のYoctoバージョンには適用できません。
* 現在(2016/10)、EASはLinux Communityで開発中のプロトタイプ版であるため潜在問題が含まれております。また、Linux Communityの動向次第では今後変更が発生する場合があります。
* 本ドキュメントで説明するEAS環境は弊社が先行評価を目的に構築したものであり、品質はプロトタイプ相当となります。また、お客様の環境にEASを適用する場合は、弊社のサポート範囲外となるため、お客様の責任にて適用をお願い致します。

ターゲットデバイス

・R-Car H3

・R-Car M3-W/R-Car M3-W+

目次

[1. EAS 4](#_Toc488949404)

[1.1 概要 4](#_Toc488949405)

[1.1.1 EASの概要 4](#_Toc488949406)

[1.1.2 EASの開発状況 5](#_Toc488949407)

[1.1.3 EASの潜在問題 5](#_Toc488949408)

[1.2 動作概要 6](#_Toc488949409)

[1.3 モジュール構成 8](#_Toc488949410)

[1.4 Energy Model 9](#_Toc488949411)

[1.4.1 算出方法 10](#_Toc488949412)

[1.5 検討課題 11](#_Toc488949413)

[1.5.1 任意のアプリケーションを優先して動作 11](#_Toc488949414)

[2. EASの環境構築方法 12](#_Toc488949415)

[2.1 パッチセットの説明 12](#_Toc488949416)

[2.2 ビルド手順 13](#_Toc488949417)

[Appendix 14](#_Toc488949418)

[A1. EASの適用事例 14](#_Toc488949419)

[A1.1 概要 14](#_Toc488949420)

[A1.2 タスクの割り付けの変化 15](#_Toc488949421)

[A1.3 結果まとめ 15](#_Toc488949422)

[A2. Cortex-A57/A53間のタスクスイッチの処理時間 16](#_Toc488949423)

[A2.1 Cortex-A57/A53間のタスクスイッチの処理時間 16](#_Toc488949424)

[A3. EASと電力制御関連機能の同時使用 17](#_Toc488949425)

[A3.1 概要 17](#_Toc488949426)

[A3.2 CPU Idleの使用方法<T.B.D.> 18](#_Toc488949427)

[A3.3 CPU Freqの使用方法<T.B.D.> 19](#_Toc488949428)

[A3.4 IPA/EMSの使用方法 20](#_Toc488949429)

[A3.4.1 EASにおけるIPA/EMSの対応状況 20](#_Toc488949430)

[A3.4.2 Cortex-A53側のEMSの使用方法 20](#_Toc488949431)

# EAS

## 概要

### EASの概要

EASはCFS(\*)を基にCortex-A57とCortex-A53の同時起動環境をサポートした低消費電力志向の拡張スケジューラです。CFSとEASの大きな違いはスケジューリングの方法で、CFSは各CPUの負荷が均等になるようにタスクを割り付けるのに対し、EASは負荷の高いタスクをCortex-A57へ、負荷の小さいタスクをCortex-A53へ割り付けます。つまり、負荷の高いタスクが動作していない場合、 Cortex-A53のみが動作してCortex-A57はIdle状態にするため消費電力の削減が可能です。従って、EASはマルチコアプロセッシングと低消費電力の２つを同時に実現しています。

Cortex-A57/Cortex-A53の同時起動環境下におけるスケジューリングの方法を比較した表を表1-1に示します。

(\*) Completely Fair Schedulerの略でLinux OSの標準スケジューラの名称です。

表 1‑1　スケジューラの種類

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| スケジューリングの方法 | | メリット | デメリット | 弊社の提案 |
| CFS  (従来相当) | スケジューラが各CPUの負荷が均等になるようにタスクを割り付ける | 弊社標準BSP上で利用可能 | CFSはSMP構成を前提としたスケジューラのためCortex-A57/A53同時動作環境での動作は適していません | Cortex-A57/A53同時動作環境での利用は推奨しません |
| Processor Affinity | Processor Affinityによってタスクを固定的にCPUへ割り付ける | 弊社標準BSP上で利用可能 | システム全体の観点からタスク割り付けの検討・設計が必要 | お客様がシステム全体の観点からタスク割り付けの設計・管理が可能な場合、本ソリューションを推奨します |
| EAS | タスクの負荷を基にスケジューラがタスクを自動でCPUへ割り付ける | Cortex-A57とCortex-A53の同時起動環境でタスクの動的割り付けが可能 | 商用利用可能な品質とするためには、最新開発コードの継続的な取得などの継続メンテナンスが必要 | お客様がEASを独自にサポート・メンテナンスが可能な場合、本ソリューションの利用が可能です |

### EASの開発状況

EASはLinux Communityで検討・開発されている機能であり、最新(2016/10時点)のパッチセットはEAS v5.2になります。今後のバージョンアップデートはLinux Communityの動向次第となります。なお、弊社は開発中のEASの正式サポートはせず、Upstreamに取り込まれた後のEASを正式にサポートする方針です。

**2014**

**2015**

**2016**

**2017**

EAS RFC v2.0

July 2014

EAS RFC v1.0

May 2014

EAS RFC v5.0

Jul 2015

EAS RFC v4.0

May 2015

EAS RFC v3.0

Feb 2015

**EAS RFC v5.2**

**Dec 2015**

図 1‑1　EAS開発の経過

### EASの潜在問題

* システム全体が高負荷であっても、一部のCPUにタスクが割り付けられない場合があります。(\*)
* 高負荷のタスクが起床する際に一時的にCortex-A53に割り付けられる場合があり、当該タスクの処理時間が伸びることがあります。(\*)

(\*) 本制限はEAS v5.2の機能制限であり、対策次期はLinux Communityの動向に依存します

## 動作概要

EASによるタスクの割り付けは、EASに付随する機能も併せ、以下の特徴を有します

* CPUの負荷状況を、CPUコア毎にモニタします
* 起床・実行可能となったタスクを、適切なCPUコアに動的に配置します
* 負荷の軽いことが判明しているタスクは、極力低電力のCPUコアに割り付け、高性能・大消費電力型のCPUコアをidle状態化することで、システム全体の消費電力削減を行います

動作原理の概要を示します

EASによるタスクの割り付けについて説明します。EASは処理性能を維持しながら、消費電力が最少となるスケジューリングをコンセプトに設計されており、従来からの変化点として、CPUの処理速度と消費電力を考慮したタスクの割り付けを行います。これにより、CPUの処理速度と消費電力が非対称となるCPUの組み合わせ(Cortex-A57/A53)においても、適切にスケジューリングすることが可能となります。

* タスク割り付けの基本アルゴリズム

STEP1 : タスクの負荷を基にCPUの処理速度内に収まる空きCPUを選定します

STEP2 : 選定されたCPU(複数)から、最も消費電力の低いCPUにタスクを割り付けます

* CPUの処理速度と消費電力

EASではCPUの処理速度と消費電力をEnergy modelと呼びます。Energy modelはEASのスケジューリングに必要なパラメータで、CPUの構成毎に定義する必要があります。詳細は1.4章を参照ください。

図1-2は、EASによるタスクの動的割り付けイメージです。

(\*) 数値は説明する上で用いた値であり特別な意味は持ちません。

Task B

Task A

Task C

Cortex-A57(**1400**)

Cortex-A53(**400**)

Linux Kernel

高負荷

低負荷

中負荷

User

CPU0

CPU1

CPU2

CPU3

Scheduler

**EAS**

CPUの処理速度

1024

800

200

400

1024

512

512

数値 : Taskの負荷

数値(細)：CPUの処理速度

数値(太)：CPUの消費電力

Energy models

図 1‑2　EASのタスクの割り付けイメージ

## モジュール構成

1. SWモジュール構成図

Setting   
operating frequencies

Linux Kernel (4.6)

HW

Cortex-A57s

(big CPU)

Scheduler

: Modified module and process for EAS

CPU topology

(2)Assignment of tasks

(3)Notification

(1)Energy

models

CPU freq

EAS

ARM-Specific framework

Power-management framework

Cortex-A53s

(little CPU)

PMIC

Clock

Setting voltages

図 1‑3　EAS環境のモジュール構成

2. モジュール概要

* CPU topology ：CPUの構成情報を管理します
* EAS ：Cortex-A57/A53構成向けの拡張スケジューラです
* CPU freq ：CPUの動作周波数/電圧を変更します(schedutil govを使用)

3. 処理概要

(1) Energy modelとはCPU毎の処理速度(DMIPS)と消費電力(mW)のテーブル情報で、kernel初期化時に生成されスケジューリングの判定に使用します。

(2) EASはEnergy modelの情報を基に処理性能を保ちながら消費電力が最も小さくなるようにタスクをCPUに割り付けます。

(3) スケジューリング毎にCPU freqにイベントを通知。CPU freqはCPUの負荷情報を基にCPUの動作周波数を動的に変更します。(本処理はLinuxの標準動作)

## Energy Model

Energy modelとはCPU毎の処理速度(DMIPS)と消費電力(mW)を定義したテーブル情報でありSoCの特性に依存します。本テーブル情報を基にタスクをスケジューリングするため適切な値を設定する必要があります。なお、2章で説明するパッチセットにはR-Car H3 Ver.1.1 / R-Car M3-W Ver.1.0に対応したEnergy modelの修正パッチが含まれておりますので、お客様の環境でも本設定値をご使用ください。

参考として、設定値サンプル(R-Car H3 Ver.1.1)を以下に示します。数値は相対値であり実測値ではありません。

1. Cluster毎のidle時のpower定義

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 \*/

};

1. Cluster毎のactive時のcapacity/power定義

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 \*/

};

1. CPU毎のidle時のpower定義

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 \*/

};

1. CPU毎のactive時のcapacity/power定義

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 \*/

};

(\*1) 本テーブルはCPUの周波数テーブルに合わせて設定するため、周波数テーブルの構成を変更する場合は本テーブルも更新が必要です

(\*2) 本テーブルはkernelのarch/arm64/kernel/topology.c内に定義されています。

### 算出方法

本テーブル情報の算出には事前に性能と電流値の測定が必要であり、弊社ではLinaroのガイド(\*)に従い測定を実施しました。

(\*) : <http://www.slideshare.net/linaroorg/bkk16tr08-how-to-generate-power-models-for-eas-and-ipa>

## 検討課題

EASを活用する上で、お客様のユースケースにあわせて追加検討が必要な場合があります。1.5.1章にその例を示します。

### 任意のアプリケーションを優先して動作

1.2章で述べたようにEASではタスクの負荷を基にCPUの処理速度と消費電力を考慮したタスクの割り付けが行なわれ、どのタスクも動作中の負荷状況を基に割り付けられます。しかし、実ユースケースの中には例えば一時的にタスクの負荷が小さくなるような負荷変動が生じるが、一貫して優先動作させたい、常にCortex-A57側で動作させたい場合も出てくることが想定されます。お客様の環境でこのようなケースが存在する場合、EASの利用に追加して、表1-2に示す対策案を参考に、お客様にて対策をご検討ください。

表 1‑2　EAS利用時の追加対応

|  |  |  |
| --- | --- | --- |
| ユースケース | 対策 | 備考 |
| タスクを常にCortex-A57上で動作させる | Cgroup、Taskset、sched\_setaffinity  のいずれかを用いて、タスクをCortex-A57へ関連付けする | Cgroup、Taskset、sched\_setaffinity  の詳細説明は  “R-Car Series 3rd Generation Application Note Processor Affinity”を参照 |
| タスクを常に最優先で動作させる | タスクの属性をリアルタイムプロセス(\*)に変更する。 | システム内にリアルタイムプロセスを乱立させると、他のプロセスが動作できなくなる場合があるためご注意ください。 |

(\*) リアルタイムプロセスは、通常のプロセスより常に優先的にスケジューリングされます。

リアルタイムプロセスの生成/変更方法については、プロセス生成時であればpthread\_create(システムコール)、chrt(コマンド)等が指定可能で、プロセス生成後ならばsched\_setscheduler(システムコール)、chrt(コマンド)等で変更が可能です。

# EASの環境構築方法

## パッチセットの説明

１．EASパッチのダウンロードＵＲＬ

弊社において評価しているEASパッチは以下のgitリポジトリ(\*)で公開しております。

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

(\*) 本EASパッチは下記のgitリポジトリを基にR-Car H3/M3-W向けに移植しています。

<http://www.linux-arm.org/git?p=linux-power.git> (branch : energy\_model\_rfc\_v5.2)

２．R-Car H3/M3-Wを用いた弊社におけるEASパッチの評価対象機能

表 2‑1　EASの機能一覧

**√**：弊社が評価対象としている機能

**―**：弊社が評価対象外としている機能

|  |  |  |  |
| --- | --- | --- | --- |
| Function Name | Function Evaluated or Not? | Description | Remarks |
| EAS core | √ | Controls the assignment of tasks by EAS. |  |
| Capacity awareness | √ | 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). |  |
| CPU topology for aarch32 | **―** | Defines the configuration of the CPUs of the AArch32 platform. | Third-generation R-Car series products do not support AArch32. |
| CPU topology for aarch64 | √ | Defines the configuration of the CPUs of the AArch64 platform. |  |
| Sched governor | **―** | 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. | Sched governor is old and no longer necessary. |
| Sched tune | **―** | Sched tune is for boosting the performance of the Sched governor as CPU freq. | This does not work with the schedutil governor. |
| Dynamic capacity scaling | **―** | Adjusting the processing capacities of CPUs by controlling the operating frequency | This does not work with the schedutil governor. |

## ビルド手順

1．事前準備

STEP1 : RENESAS\_RCH3M3\_YoctoStartupGuideに従い全ビルドを実施してください。

STEP2 :全CPUを起動するため、R-Car Series, 3rd Generation\_ApplicationNote\_Processoraffinityの手順に従い設定変更を行ってください。

2．Building Instructions for EAS

STEP1 :事前準備で作成したYoctoビルドの作業用ディレクトリに移動して

$ cd $WORK

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

$ cd $WORK/meta-renesas-power

EASパッチをダウンロードしてチェックアウトします

STEP2 : source command を実行します

$ cd $WORK

$ source poky/oe-init-build-env

STEP3 : $WORK/build/conf/bblayers.conf にmeta-renesas-powerを追加します

BBLAYERS ?= " \

${TOPDIR}/../poky/meta \

${TOPDIR}/../poky/meta-yocto \

${TOPDIR}/../poky/meta-yocto-bsp \

${TOPDIR}/../meta-renesas/meta-rcar-gen3 \

${TOPDIR}/../meta-linaro/meta-linaro-toolchain \

${TOPDIR}/../meta-openembedded/meta-oe \

${TOPDIR}/../meta-renesas-power \ <--- Add this line.

$ cd $WORK/build

[For Wayland]

bitbake core-image-weston

[For BSP only]

bitbake core-image-minimal

STEP4 : bitbakeでbuildします

イメージ出力パス：$WORK/build/tmp/deploy/images/salvator-x/ directory.

# Appendix

# EASの適用事例

## 概要

本章ではIVI Use-case(3画面デモ)へEASを適用した結果を事例として説明します。本ユースケースの選定理由としては、複数のアプリケーションが同時動作して高負荷であること、R-Car H3のターゲットユースケースとしている統合コックピットに近いこと、これら2点より3画面デモをユースケースとして選定しました。以降に、従来環境とEASの差分を示しながら検証結果を説明します。

1. 構成

Instrument cluster

R-Car H3



Video

Map

Menu

LVDS

HDMI

HDMI

Display for entertainment

Center display

(for the navigation system)

図 A1‑1　3画面デモの接続イメージ

2. 比較環境

|  |  |  |  |
| --- | --- | --- | --- |
| 表 A1‑1　スペック比較 | | Existing Environment | Environment where EAS is Applied |
| SoC | | R-Car H3 (Ver.1.1) | R-Car H3 (Ver.1.1) |
| CPU | | Cortex-A57 x4 1.5GHz | Cortex-A57 x4 1.5GHz  **Cortex-A53 x4 1.2GHz** |
| RAM | | DDR4-2400 2ch split | DDR4-2400 2ch split |
| GPU | | PowerVR GX6650 600MHz | PowerVR GX6650 600MHz |
| Display | Center | LVDS 1024x768 | LVDS 1024x768 |
| Instruments | HDMI 1920x1080 | HDMI 1920x1080 |
| Video | HDMI 1920x1080  MP4(H.264) VCP decode | HDMI 1920x1080  MP4(H.264) VCP decode |

## タスクの割り付けの変化

CPU使用率とタスクの割り付けの変化を表A1-2に示します。

表 A1‑2　CPU使用率とタスク割り付け比較

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Existing Environment | | Environment where EAS is Applied | |
|  | CPU Utilization | Task Name | CPU Utilization | Task Name |
| CPU0(Cortex-A57) | 48.3% | ivi\_infotainment, etc | 10.1% | ivi\_infotainment, etc |
| CPU1(Cortex-A57) | 44.2% | ivi\_instrument\_cluster | 8.9% |  |
| CPU2(Cortex-A57) | 43.2% | ivi\_camera\_daemon | 9.4% |  |
| CPU3(Cortex-A57) | 44.0% | ivi\_passenger,  weston, ivi\_daemon | 9.6% |  |
| CPU4(Cortex-A53) | - | - | 52.2% | weston, ivi\_daemon |
| CPU5(Cortex-A53) | - | - | 40.8% | ivi\_instrument\_cluster |
| CPU6(Cortex-A53) | - | - | 39.9% | ivi\_camera\_daemon |
| CPU7(Cortex-A53) | - | - | 44.7% | ivi\_passenger |

(\*) 各種数値は参考値となります

本結果は、EAS環境において、Cortex-A53上で十分動作できると判定された負荷の軽いタスクがEASによって自動でCortex-A53に割り付けられていることを示しています。

## 結果まとめ

CPU使用率と電流値を比較した結果を表A1-3に示します。

表 A1‑3　CPU使用率と電流値比較

|  |  |  |  |
| --- | --- | --- | --- |
|  | Existing Environment | Environment where EAS is Applied | Difference |
| Zφ (for Cortex-A57) | 1500 MHz | 1500 MHz |  |
| Z2φ (for Cortex-A53) | - | 1200 MHz |  |
| Cortex-A57 usage | 44.9 % | 11.1 % | - 33.8 % |
| Cortex-A53 usage | 0 % | 44.4 % | + 44.4 % |
| GPU usage | 60-70% | 60-70% | ±0 % |
| DVFS 0.8V (Cortex-A57 + GPU) | 3000 mA | 2340 mA | - 660 mA |
| VDD 0.8V (Cortex-A53 + other modules) | 4980 mA | 5380 mA | + 400 mA |
| Current (DVFS 1.0 + VDD 1.0) | 7980 mA | 7720 mA | - 260 mA |

(\*) 各種数値は参考値となります

本適用事例では、従来環境で動作していたアプリケーションが、EASを適用するだけでCortex-A57/A53同時起動環境でも問題なく動作することが確認できております。また、EASにより僅かながら消費電流が削減し、Cortex-A57側の処理余力が増えることが確認できたため、さらなるアプリケーションの追加も可能な見込みです。

# Cortex-A57/A53間のタスクスイッチの処理時間

タスクスイッチに掛かる時間を表A2-1に示します。

表 A2‑1　タスクスイッチの処理時間

|  |  |  |  |
| --- | --- | --- | --- |
|  | Cortex-A57  ->Cortex-A57 | Cortex-A57  ->Cortex-A53 | Cortex-A53  ->Cortex-A57 |
| クラスタ跨ぎ | 無 | 有 | 有 |
| 切り替え時間 | 28us | 33us | 35us |

[環境]

* SW : Yocto v2.12.0 (Linux BSP 3.3.2)
* SoC : R-Car H3 Ver.1.1 (Salvator-X)
* 測定方法：

1. 任意の無限ループする実行プロセス(A)をCPU1で起動

2. tasksetコマンド(\*)でプロセスAをCPU2にスイッチ

(\*) tasksetはプロセスのCPUアサインを変更設定するコマンド

１．CPU1でプロセスAが実行

２．プロセスAを停止

３．プロセスＡをCPU2へスイッチ

４．プロセスAをCPU2上でWakeup

A

CPU1

Run queue

CPU2

Run queue

A

A

Run queue

Run queue

CPU2

Run queue

CPU1

Run queue

CPU2

A

Run queue

CPU1

CPU1

CPU2

Run queue

測定区間

：停止状態

：動作中

図 A2‑1　タスクスイッチの処理フロー

# EASと電力制御関連機能の同時使用

## 概要

EAS環境下でさらに低消費電力の実現を目指すことを目的として、本章で他の電力制御関連機能との同時使用について説明します。まず、EAS環境下における各機能の動作状況を表A3-1に示します。Cortex-A57側は各機能が動作可能であるため、従来環境と同じ使い方が可能です。Cortex-A53側のいくつかの機能は個別拡張により動作可能ですが、Cortex-A53はCortex-A57に比べて低消費電力のCPUチップであるため、システム全体から見たときの電力削減効果は極めて小さくなることが見込まれます。そのため、弊社のEAS評価環境ではCortex-A53側の電力制御関連機能(CPU Idle/CPU Freq/IPA/EMS)は無効にしております。

表 A3‑1　電力制御関連機能のサポート一覧

**√**：動作可能

**∆**：個別拡張により動作可能

|  |  |  |  |
| --- | --- | --- | --- |
| Function name(\*) | Cortex-A57 | Cortex-A53 | Remarks |
| CPU Hotplug | **√** | **√** |  |
| CPU Idle | **√** | **∆** | CPU Idleは本版では機能制限のため未サポートと致しますが、次版以降に動作可能となる予定です。 |
| CPU Freq | **√** | **∆** | Coretex-A53側のCPU Freqは本版では未評価のため未サポートと致しますが、次版以降で個別拡張により動作可能となる予定です。 |
| System Suspend to RAM | **√** | **√** |  |
| IPA | **√** | **∆** | Coretex-A53側のIPAは本版では未評価のため未サポートと致しますが、次版以降で個別拡張により動作可能となる予定です。 |
| EMS | **√** | **∆** |  |

(\*) 各機能の説明はRENESAS\_RCH3M3\_PowerManagement\_UMEを参照

## CPU Idleの使用方法<T.B.D.>

## CPU Freqの使用方法<T.B.D.>

## IPA/EMSの使用方法

### EASにおけるIPA/EMSの対応状況

IPA/EMSはCPUに対する電力制御によりSoCの熱管理を行う機能になります。Cortex-A57については、IPA/EMS共にEAS環境下で動作します。Cortex-A53について、IPAは標準BSPで未評価・未サポートであるため利用できません。EMSは個別拡張を行うことで動作させることが可能です。

### Cortex-A53側のEMSの使用方法

Kernelのdevice treeにEMS発動時にshutdownさせるCPU(Cortex-A53)を定義します。設定例を下記に示します。

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";

};

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

(\*) 本変更は標準Yocto Package外の個別拡張です。弊社ではサポートできませんのでお客様の責任で変更ください。

改訂記録

|  |  |  |  |
| --- | --- | --- | --- |
| Rev. | 発行日 | 改訂内容 | |
| ページ | ポイント |
| 1.00 | 2017.04 | － | 新規作成 |
| 1.01 | 2017.07 | P1,8,13,20 | ・本文の記述を更新 |
| P7,8,14,16 | ・図を更新 (図1-2, 図1-3, 図A1-1, 図A2-1) |
| P4,11,12,14,  15,17 | ・表を更新 (表1-1, 表1-2, 表2-1, 表A1-1, 表A1-2, 表A1-3, 表A3-1) |
| P4 | ・表の誤記訂正 (表1-1) |
| 1.02 | 2019.03 | P1, 2 | ・Background, Target Readers, Target DeviceにR-Car M3-W+を追加 |
| P9,14,16 | ・HWデバイスのバージョン表記をWSからVer.に更新 |
| P13 | ・R-Car Gen3をR-Car Series, 3rd Generationへ表記変更 |



製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策

CMOS製品の取り扱いの際は静電気防止を心がけてください。CMOS製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS製品を実装したボードについても同様の扱いをしてください。

2. 電源投入時の処置

電源投入時は、製品の状態は不定です。電源投入時には、LSIの内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。

3. 電源オフ時における入力信号

当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。

4. 未使用端子の処理

未使用端子は、「未使用端子の処理」に従って処理してください。CMOS製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI周辺のノイズが印加され、LSI内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。

5. クロックについて

リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。

6. 入力端子の印加波形

入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS製品の入力がノイズなどに起因して、VIL（Max.）からVIH（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、VIL（Max.）からVIH（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。

7. リザーブアドレス（予約領域）のアクセス禁止

リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。

8. 製品間の相違について

型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。

2. 当社製品、本資料に記載された製品デ－タ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。

3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。

4. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。

5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

標準水準： コンピュータ、OA機器、通信機器、計測機器、AV機器、家電、工作機械、パーソナル機器、産業用ロボット等

高品質水準： 輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等

当社製品は、データシート等により高信頼性、Harsh environment向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。

6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。

7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。

8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。

9. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。

10. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。

11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。

12. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注2. 本資料において使用されている「当社製品」とは、注1において定義された当社の開発、製造製品をいいます。

(Rev.4.0-1 2017.11)

|  |  |  |
| --- | --- | --- |
| 本社所在地 |  |  |
| 〒135-0061 東京都江東区豊洲3-2-24（豊洲フォレシア）  [www.renesas.com](https://www.renesas.com/) |  |  |
| 商標について |  |  |
| ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。 |  |  |

変更内容〔ルネサス内部向け〕

|  |  |  |  |
| --- | --- | --- | --- |
| Rev. | 発行日 | 改訂内容 | |
| ページ | ポイント |
| 0.5 | Oct. 27th, 2016 |  | ・カラー図をモノクロに変更  - パワポP.10, 11, 20, 24,  ・文言の変更  - パワポP.7「以下」→「表1-1」  - パワポP.10「1-4.章」→「1.4章」  - パワポP.10「右図」→「図1-2」  - パワポP.13「1-2.章」→「1.2章」  - パワポP.13「以下」→「表1-2」  - パワポP.21「以下」→「表A1-2」  - パワポP.22「以下」→「表A1-3」  - パワポP.24「以下」→「表A2-1」  - パワポP.26「下記」→「表A3-1」  ・注記マークを (\*) のスタイルに統一  - パワポP.3, 7, 8, 12, 13, 15, 24 (\*1)  - パワポP.10, 29 ※  - パワポP.12 注1)、注2)  ・全角から半角に変更  - パワポP.16 リスト番号1, 2  ・体裁を整えるための変更  - パワポP.16 1.STEP2 および 2.STEP1 の空白を1個削除 |
| 0.51 | Dec. 26th, 2016 |  | ・文言の統一(スケジューラ)  ・文言の統一(弊社)  ・表1-1のタイトルを追記  ・図1-1のタイトルを追記  ・図1-2のタイトルを追記  ・図1-3のタイトルを追記  ・表1-2のタイトルを追記  ・表2-1のタイトルを追記  ・図A1-1のタイトルを追記  ・表A1-1のタイトルを追記  ・表A1-2のタイトルを追記  ・表A1-3のタイトルを追記  ・表A2-1のタイトルを追記  ・図A2-1のタイトルを追記  ・表A3-1のタイトルを追記  ・1-4 チューニングパラメータの更新 |
| 1.00 | 2017.04 | － | ・Rev0.51をベースとしてRev1.00とする  ・ファイル名変更  ・ターゲットデバイス名を記載  ・ドキュメント管理番号、発行年月を記載  ・ご注意書きのページのCOPYRIGHTを2017年版に変更 |
| 1.01 | 2017.07 | ― | ヘッダーに「CONFIDENTIAL」を付加 |
| P1,8,13,20 | 翻訳フィードバックを反映  ・本文の記述を更新 |
| P7,8,14,16 | ・図を更新 (図1-2, 図1-3, 図A1-1, 図A2-1) |
| P4,11,12,14,  15,17  P4 | ・表を更新 (表1-1, 表1-2, 表2-1, 表A1-1, 表A1-2, 表  A1-3, 表A3-1)  ・表の誤記訂正 (表1-1) |