

Cortex-M 対応 MULCOS-MK

(MULTi Core Operating System – Multilayer Kernel)

μ ITRON4.0 仕様版

RA デバイス依存部ユーザーズガイド



第1版 2025年4月

改訂履歴

版数	発行日	改訂内容
第 1 版	2025 年 4 月	Cortex-M 対応 MULCOS-MK μ ITRON4.0 仕様版 RA デバイス依存部ユーザーズガイドの初版を発行

はじめに

MULCOS-MK (MULTi Core Operating System - Multilayer Kernel) とは、セキュリティレベルの異なる階層化したリアルタイム・オペレーティング・システム (Real Time Operating System、以降 RTOS と省略) の実行環境を管理する多層化カーネル (以降では、本製品を単に本カーネルと呼ぶ) です。

μ ITRON4.0 仕様版の RTOS (以降では、単に本 RTOS と呼ぶ) は、最も多く使われている仕様の 1 つで、当時の社団法人トロン協会が策定し公開した「 μ ITRON4.0 仕様」(現在は、トロンフォーラムが公開) に準拠しています。

本書は、Cortex-M 対応 MULCOS-MK の μ ITRON4.0 仕様版の RA デバイス依存部ユーザーズガイドで、ルネサス エレクトロニクス社のデバイスに依存した機能とその使い方とを説明しています。その他に共通部ユーザーズガイドが用意されています。

(μ ITRON とは Micro Industrial TRON、TRON とは The Realtime Operating system Nucleus の略称)

参考資料

- ・「ARM®v8-M Architecture Reference Manual」
- ・「ARM® Cortex®-M33 Processor Technical Reference Manual」
- ・「ARM® Cortex®-M85 Processor Technical Reference Manual」
- ・社団法人トロン協会発行「 μ ITRON4.0 仕様 Ver.4.03.00」
- ・「RA8M1 Group User's Manual: Hardware」
- ・「RA6M5 Group User's Manual: Hardware」

目次

改訂履歴	0
はじめに	1
目次	2
第 1 章 MULCOS-MK の概要	3
1-1. 対応デバイスの種類	3
第 2 章 製品構成	4
2-1. ディレクトリ構成とファイルの種類	4
2-2. カーネルライブラリの種類	6
2-3. サンプルプロジェクト	6
2-4. プロセッサ依存部ファイル	6
2-4-1. 周辺モジュールのレジスタ定義ファイル	6
2-4-2. サンプルドライバのソースファイル	8
第 3 章 RTOS 使用時の特記事項	9
3-1. 割込み番号	9
3-2. 浮動小数点ユニット対応	10
3-3. 割込み優先度	10
3-4. スタックオーバフローの検出	11
3-5. CPU ロック状態と割込みマスク	11
3-6. セキュリティ属性の分離と定義	13
3-6-1. 割込みの定義	13
3-6-1. 周辺モジュールの定義	13
3-7. メモリ領域の構成	14
第 4 章 標準サンプルドライバ	16
4-1. 例外番号割付けドライバ	16
4-1-1. 例外番号の割付け	18
4-1-2. イベント番号の参照	18
4-1-3. 例外番号の解放	18
4-2. シリアルドライバ	19
4-2-1. シリアルドライバのコンフィグレーション	21
4-2-2. シリアルチャネルの初期化	22
4-2-3. シリアルチャネルの制御	23
4-2-4. シリアルチャネルへの一文字送信	24
4-2-5. シリアルチャネルへの文字列送信	26
4-2-6. シリアルチャネルからの一文字受信	27
4-2-7. シリアルチャネルからの文字列受信	28
4-2-8. シリアルチャネルの状態参照	29

第 1 章 MULCOS-MK の概要

Cortex-M 対応 MULCOS-MK では、ARM®v8-M アーキテクチャの TrustZone®に対応し、上位層であるセキュア層と、下位層である非セキュア層とで、2 つの μ ITRON4.0 仕様の RTOS 実行環境を管理します。

ARMv7-M アーキテクチャを含め TrustZone を実装していないプロセッサの場合では、単一の RTOS 実行環境で構築されます。

1-1. 対応デバイスの種類

ルネサスエレクトロニクス社製のデバイスとして、Cortex-M85、Cortex-M33、或いは Cortex-M4 コアを搭載した RA シリーズのプロセッサに対応しています。ただし、リトルエンディアンのみの対応です。

第 2 章 製品構成

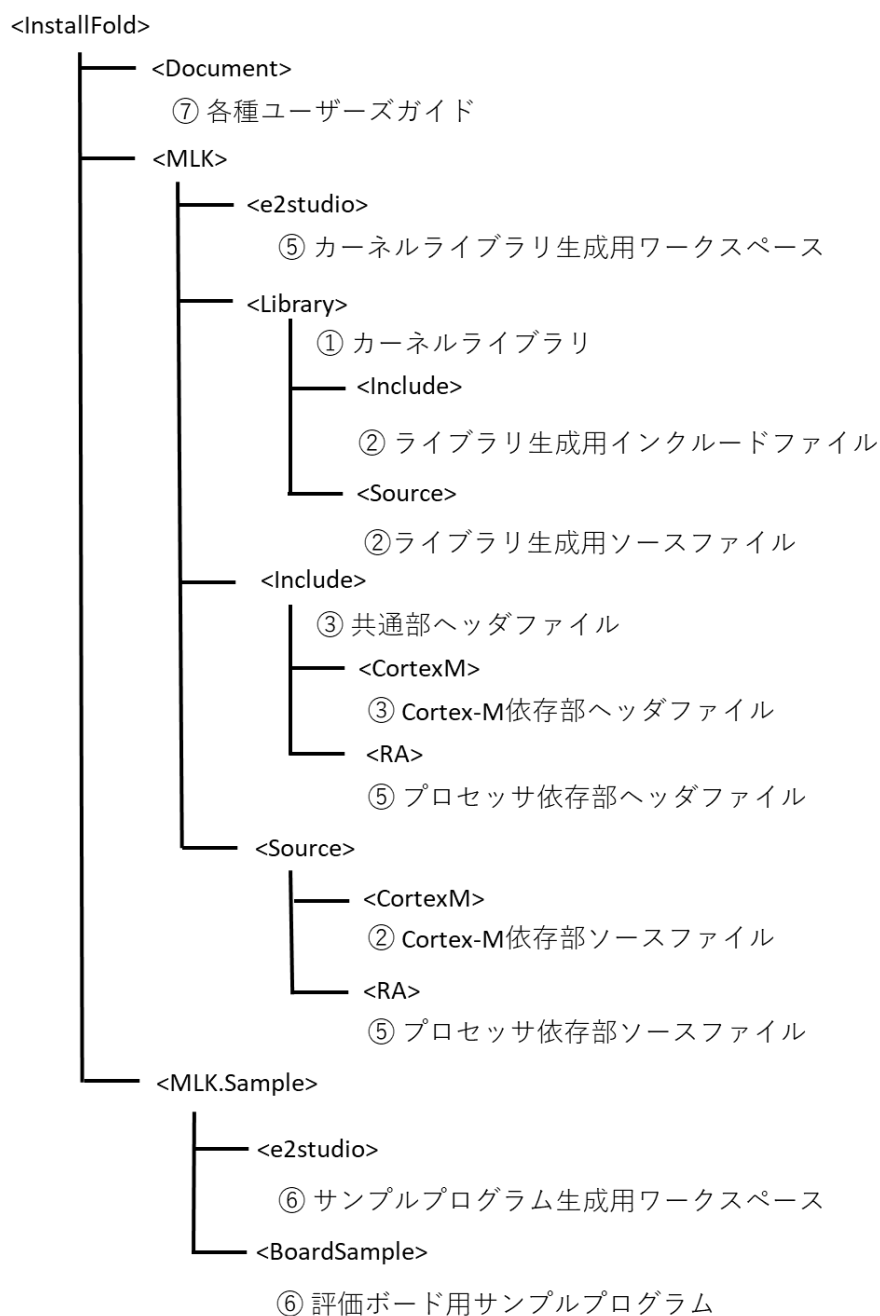
Cortex-M 対応 MULCOS-MK μ ITRON4.0 仕様版では、以下の製品構成になっています。

- ユーザーズガイド
- カーネルライブラリ
- カーネルライブラリの生成に必要なソースコード
- サンプルドライバのソースコード
- サンプルプロジェクト

2-1. ディレクトリ構成とファイルの種類

ディレクトリ構造として、フォルダは以下のように分かれています。

- ① カーネルライブラリは、本 RTOS を使用するアプリケーションプログラムにリンクするライブラリファイルです。
- ② カーネルライブラリの生成に必要なファイルです。
- ③ カーネルライブラリの生成に必要なファイルで、アプリケーションプログラムの生成にも必要なファイルです。
- ④ カーネルライブラリの生成用のワークスペースです。
- ⑤ サンプルドライバとアプリケーションプログラムのビルドに必要なファイルです。
- ⑥ 評価ボードに対応したアプリケーション用サンプルプログラムです。
- ⑦ 本書を含めたユーザーズガイドです。



<BoardSample>とは、EK_RA8M1 や EK_RA6M5 などの評価ボードの名称です。

2-2. カーネルライブラリの種類

Cortex-M85、Cortex-M33、Cortex-M4 コアのプロセッサに対応した以下のカーネルライブラリを用意しています。

コア アーキテクチャ	浮動小数点 ユニット	カーネルライブラリ
		e2studio
Cortex-M85	FPv5-SP or FPv5-DP	libmulcosxe_cm85s.a
Cortex-M33	FPv5-SP or FPv5-DP	libmulcosxe_cm33s.a
Cortex-M4	-	libmulcosxe_cm4.a

タイプ2 評価版に収録されたカーネルライブラリは、それぞれの生成オブジェクト数が5つに制限され、ライブラリの名称に”_evel”が付加されています。

2-3. サンプルプロジェクト

市販の評価ボード上で動作するサンプルプロジェクトとして、以下を用意しています。

評価ボード名	コアアーキテクチャ	概要
EK-RA8M1	Cortex-M85	LED 点滅、ユーザスイッチ割込み入力、 シリアルドライバによる調歩同期通信のサンプル
EK-RA6M5	Cortex-M33	LED 点滅、ユーザスイッチ割込み入力、 シリアルドライバによる調歩同期通信のサンプル
EK-RA6M3	Cortex-M4	LED 点滅、ユーザスイッチ割込み入力のサンプル

2-4. プロセッサ依存部ファイル

プロセッサに依存するファイルとして、プロセッサ内臓デバイスのレジスタ定義を記述したヘッダファイルと、プロセッサ内臓デバイスを使用するサンプルドライバのヘッダファイルとソースコードファイルとがあります。

2-4-1. 周辺モジュールのレジスタ定義ファイル

本 RTOS が用意するプロセッサ内臓デバイスのレジスタ定義には、スタートアップに必要なデバイスと、シリアルデバイスとがあります。用意されたレジスタ定義に不足がある場合には、必要なレジスタ定義を追加します。

これらには”dv_”から始まるファイル名を用い、デバイスのレジスタ定義を示しています。また、ユーザプログラムがインクルードするメインのファイルと、個々のデバイス定義とをファイルを分離し、容易に追加できるようになっています。

【RA8M1 シリーズの例】

RA8M1 のレジスタ定義ファイル dv_ra8m1.h のコードは以下のようになっています。

```
#ifndef _DV_RA8M1_H_
#define _DV_RA8M1_H_

/* System Controller (SYSC) */
#include "dv_ra8m1_sysc.h"

/* CPU System Security Control Unit (CPSCU) */
#include "dv_ra8m1_cpscu.h"

/* Peripheral Security Control Unit (PSCU) */
#include "dv_ra8m1_pscu.h"

/* Interrupt Controller Unit (ICU) */
#include "dv_ra8m1_icu.h"

/* Pin Function Controller and PORT (PFS, PORT) */
#include "dv_ra8m1_pfs_port.h"

/* Flash memory controller (Flash) */
#include "dv_ra8m1_flash.h"

/* Sram memory controller (SRAM) */
#include "dv_ra8m1_sram.h"

/* Serial Communications Interface (SCI) */
#include "dv_ra8m1_sci.h"

/* System Module Stop control (MSTP) */
#include "dv_ra8m1_mstp.h"

/* Asynchronous General Purpose Timer (AGT) */
#include "dv_ra8m1_agt.h"

/* 必要なヘッダファイルをインクルード */

#endif /* _DV_RA8M1_H_ */
```

このように、いくつかのデバイス定義がインクルードしていますが、必要に応じて新たなデバイス定義を指定箇所に追加します。

2-4-2. サンプルドライバのソースファイル

サンプルドライバを構成するインクルードファイルとソースコードファイルがあります。これらの使用方法は、「第 4 章 標準サンプルドライバ」を参照してください。

【RA8M1 シリーズの例】

<code>drv_ra8ma1_icu.c</code>	例外番号割付けドライバのソースコードファイル
<code>drv_ra8ma1_uart.c</code>	シリアルドライバのソースコードファイル
<code>drv_ra8ma1_uart.h</code>	シリアルドライバのインクルードファイル

第 3 章 RTOS 使用時の特記事項

本カーネルでは、TrustZone を使用して複数の RTOS 実行環境を管理しますが、さまざまなプロセッサのアーキテクチャ共通の仕様にするのは困難なため、アーキテクチャ毎に依存する仕様を最小限にし、それらを明確化しています。

3-1. 割込み番号

本 RTOS の Cortex-M アーキテクチャ版では、割込み番号として例外番号を使用します。そのため、割込みハンドラや割込みサービスルーチンで使用する割込み番号は 15 以上の値になります。

例外番号	例外種別	割込み使用可否
0	StackPointer	使用禁止
1	Reset	使用禁止
2	NMI	使用禁止
3	HardFault	使用禁止
4	MemManage	使用禁止
5	BusFault	使用禁止
6	BusFault	使用禁止
7-9	システム予約	使用禁止
10	不正例外ハンドラ	使用禁止
11	SVCall	使用禁止
12	DebugMonitor	使用禁止
13	システム予約	使用禁止
14	PendSV	使用禁止
15	SysTick	使用可
16	外部割込み番号 0	使用可
17	外部割込み番号 1	使用可
18	外部割込み番号 2	使用可
:	:	:
110	外部割込み番号 94	使用可
111	外部割込み番号 95	使用可

N : プロセッサにより異なる

ルネサス エレクトロニクス社の RA シリーズの外部割り込み番号は、割込みコントローラユニット (ICU) のイベントリンク機能により割付けられます。つまり、外部割込み番号 n は ICU.IELSR n レジスタに関連付けられ、このレジスタにイベント番号を設定することで

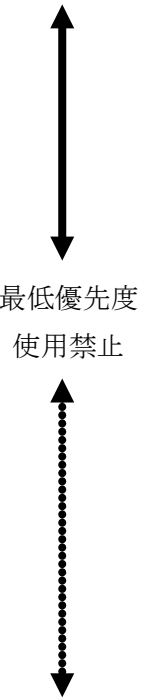

割付けられます。また、この処理関数はサンプルドライバとして提供されています。
例外番号 10 は、システム予約のため例外として使用しませんが、本 RTOS では不正例外ハンドラのアドレスを設定します。

3-2. 浮動小数点ユニット対応

Cortex-M アーキテクチャでは、浮動小数点ユニットの有効化は対応プロセッサでは自動で判別されます。浮動小数点レジスタを初めてアクセスすると、制御レジスタを自動的に有効として設定します。浮動小数点レジスタは、コンテキストの一部とされ、タスクの起動時、ハンドラの起動時の浮動小数点ユニットは無効状態になっています。つまり、タスクが起動した直後や、割込みハンドラ、割込みサービスルーチン、周期ハンドラが起動した直後の浮動小数点ユニットは無効状態です。

3-3. 割込み優先度

Cortex-M アーキテクチャの割込み優先度は 8 ビットで管理されますが、その上位 4 ビットのみ実装されて、下位 4 ビットは RAZ/WI (読み出しは 0、書込みは無視) になっています。そのため、以下のように 16 レベルの割込み優先度を持ち、割込みハンドラ及び割込みサービスルーチンの割込み優先度には、この値を用います。

優先順位		割込み優先度値
セキュア層	非セキュア層	
最高優先度	最高優先度	0x00
		0x10
		0x20
		0x30
		0x40
		0x50
		0x60
		0x70
		0x80
		0x90
		0xA0
		0xB0
		0xC0
		0xD0
		0xE0
	最低優先度	0xF0

非セキュアと見なされる ARMv7-M アーキテクチャを含め TrustZone を実装していないプロセッサの場合と、TrustZone を使用した非セキュア層の場合とでは、全割込み優先度を使用できます。しかし、TrustZone を使用したセキュア層の場合では、下位半分の割込み優先度（0x80～0xF0）を使用することができません。

3-4. スタックオーバーフローの検出

TrustZone を実装した Cortex-M でのみ、スタックオーバーフローを検出でき、違反した場合には HradFault 例外が発生します。このスタックオーバーフローとは、指定されたスタック領域外をアクセスすることではなく、スタックポインタの値が範囲外を示した場合を言います。

3-5. CPU ロック状態と割込みマスク

Cortex-M アーキテクチャでは、CPU ロック状態を実現するため、全割込みの禁止許可をシステムレジスタの PRIMASK を使用します。これに対して、割込みマスク優先度の変更には、システムレジスタの BASEPRI を操作します。

CPU ロック状態と割込みマスクの変更により、任意の割込み優先度までの割込みを抑止することができます。この関係は次のようになります。

【非セキュア層】

CPU ロック	割込みマスク 優先度	割込みの優先度								
		0x00	0x10	0x20	0x30	...	0xC0	0xD0	0xE0	0xF0
ロック	0x00-0xF0	×	×	×	×	...	×	×	×	×
解除	0x10	○	×	×	×	...	×	×	×	×
解除	0x20	○	○	×	×	...	×	×	×	×
解除	0x30	○	○	○	×	...	×	×	×	×
.	.	.								
.	.	.								
解除	0x70	○	○	○	○	...	×	×	×	×
解除	0x80	○	○	○	○	...	×	×	×	×
.	.	.								
.	.	.								
解除	0xE0	○	○	○	○	...	○	○	×	×
解除	0xF0	○	○	○	○	...	○	○	○	×
解除	0x00	○	○	○	○	...	○	○	○	○

○：割込みを受け付ける

×

【セキュア層】

CPU ロック	割込みマスク 優先度	割込みの優先度								
		0x00	0x10	0x20	0x30	0x40	0x50	0x60	0x70	0x80-0xF0
ロック	0x00-0xF0	×	×	×	×	×	×	×	×	×
解除	0x10	○	×	×	×	×	×	×	×	×
解除	0x20	○	○	×	×	×	×	×	×	×
解除	0x30	○	○	○	×	×	×	×	×	×
解除	0x40	○	○	○	○	×	×	×	×	×
解除	0x50	○	○	○	○	○	×	×	×	×
解除	0x60	○	○	○	○	○	○	×	×	×
解除	0x70	○	○	○	○	○	○	○	×	×
解除	0x80-0xF0	—	—	—	—	—	—	—	—	—

○：割込みを受け付ける

×

—：設定不可

このように、CPU ロック状態の場合は、割込みマスク優先度の値に関わらず全割込みがマスクされます。CPU ロック状態を解除すると割込みマスク優先度が有効になり、値に応じた割込み優先度の割込みがマスクされます。

セキュア層では割込み優先度の 0x80-0xF0 を使用できません。セキュア層の割込み優先度の 0x80-0xF0 には、非セキュア層割込み優先度の 0x00-0xF0 が割当てられているからです。そのため、非セキュアが CPU ロック状態になった場合でも、割込みマスク優先度が 0x80 と同等となり、セキュア層の割込みに影響を与えません。

TrustZone 未実装のプロセッサでは、非セキュア層のみと見なされます。

Copyright© 2025 SPIRAL-TECH Co., Ltd

12

3-6. セキュリティ属性の分離と定義

TrustZone を実装したプロセッサでは、内蔵の各種周辺モジュールと割込み要因を、セキュア層で使用するのか、或いは非セキュア層で使用するのか、セキュア層の初期化時に決定します。

3-6-1. 割込みの定義

外部割込み要因をセキュアか非セキュアかのどちらで使用するのかセキュリティ属性を定義します。NVIC の NVIC_ITNSs と割込みコントローラユニット (ICU) のセキュリティ属性レジスタと同じ値を設定します。本 RTOS では、割込みハンドラや割込みサービスルーチンの生成時に正当性を判断するために、この値を参照します。

```
REG_SYSC.PRCR = 0xA510;
NVIC_ITNS[0] = 0x0000FFFF;
NVIC_ITNS[1] = 0x00000000;
NVIC_ITNS[2] = 0x00000000;
REG_CPSCU.ICUSARG = 0x0000FFFF;
REG_CPSCU.ICUSARH = 0x00000000;
REG_CPSCU.ICUSARI = 0x00000000;
REG_SYSC.PRCR = 0xA500;
```

※上記例は、外部割込み番号 0～15 を非セキュア層で使用する設定です。

3-6-1. 周辺モジュールの定義

周辺モジュールをセキュアか非セキュアかのどちらで使用するのかセキュリティ属性を定義します。周辺モジュールセキュリティ属性レジスタを周辺モジュール毎にビットを設定し、同時に非特権モードのタスクから使用するか否かも設定します。

非セキュアが I/O ポートを使用する場合は、ポートセキュリティ属性レジスタも設定します。

```
REG_SYSC.PRCR = 0xA510;
REG_PSCU.PSARB = 0x80000000;
REG_PSCU.PSARC = 0x00000000;
REG_PSCU.PSARD = 0x00000000;
REG_PSCU.PSARE = 0x00000000;
REG_PSCU.PPARB = 0xFFFFFFFF;
REG_PSCU.PPARC = 0xFFFFFFFF;
REG_PSCU.PPARD = 0xFFFFFFFF;
REG_PSCU.PPARE = 0xFFFFFFFF;
REG_SYSC.PRCR = 0xA500;
```

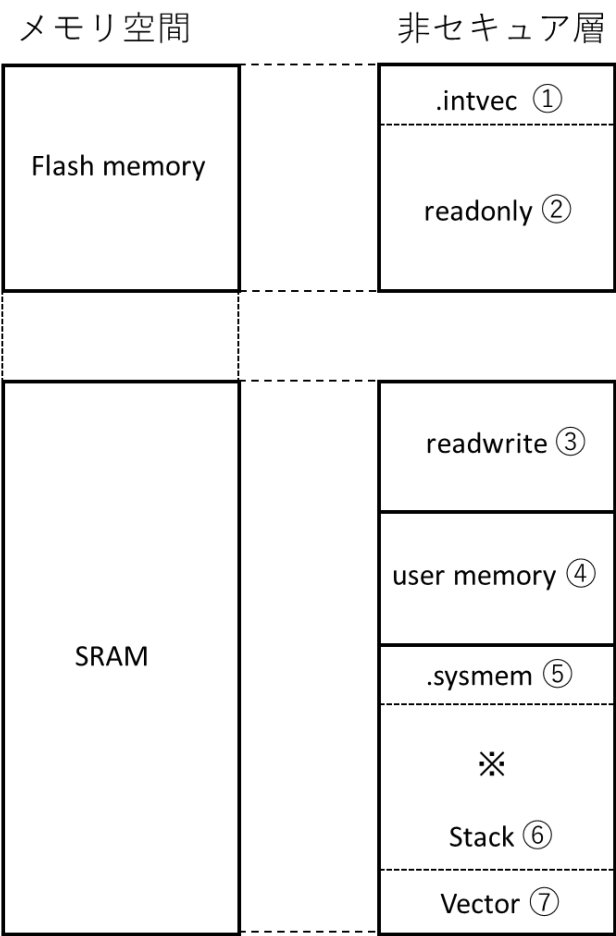
※上記例は、SCIO のみを非セキュアで、すべて非特権タスクで使用する設定です。

3-7. メモリ領域の構成

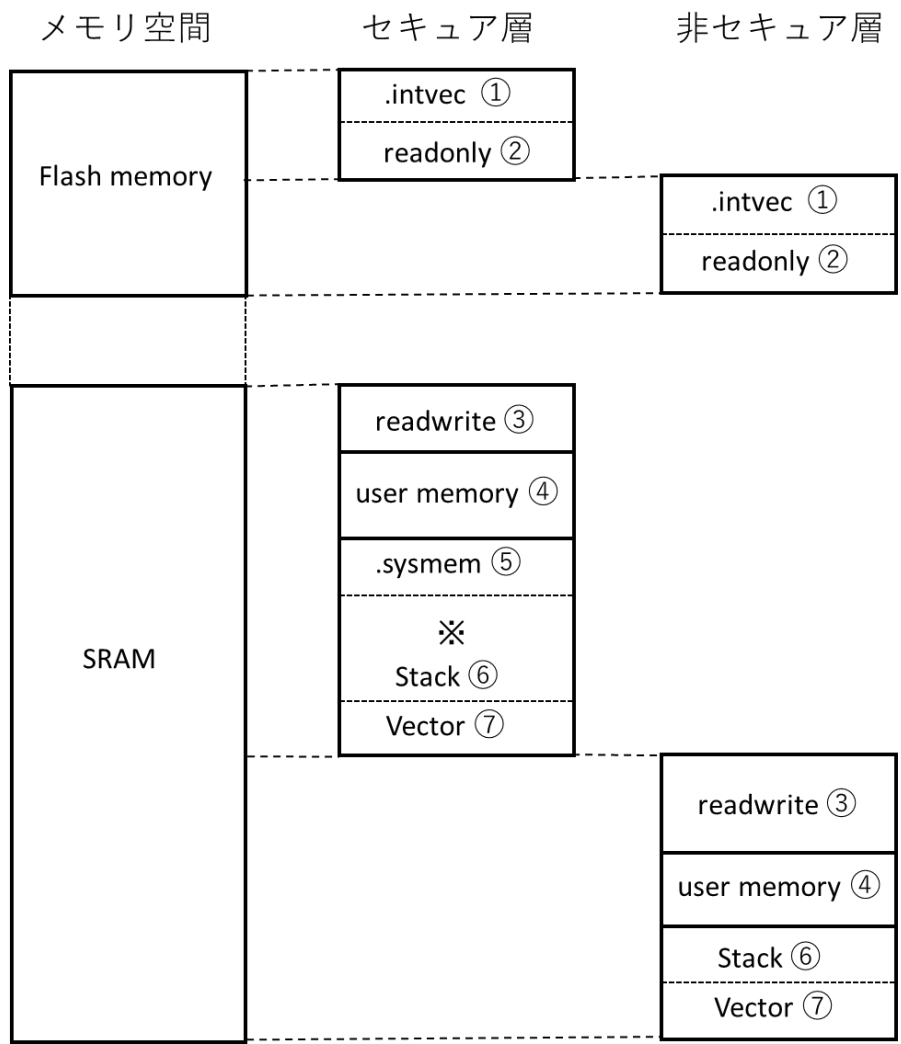
本 RTOS では、一般的に使用されるメモリ領域のセクション以外に、専用のメモリ領域を必要とします。

- ① ベクタテーブルに使用するメモリ領域（ブート時のみ使用）
- ② コード領域
- ③ アプリケーションプログラムがアクセスする RTOS の影響下でないメモリ領域
- ④ アプリケーションプログラムがアクセスする RTOS が割振るユーザーメモリ領域
（この領域はリンカスクリプトで指定するのではなく、コンフィグレーションで番地と容量と直接指定します。「Corex-M 対応 MULCOS-MK μ ITRON4.0 仕様版共通部ユーザーズガイド」を参照）
- ⑤ 各種の制御ブロックなどに使用する RTOS だけがアクセスするシステムメモリ領域
- ⑥ ハンドラモードで使用するシステムスタックメモリ領域
- ⑦ RTOS 起動後にベクタテーブルとして使用するメモリ領域

【TrustZone 未実装の場合】



【TrustZone 実装の場合】



※.sysmem セクションから、システムスタック Stack までの領域がシステムメモリ領域

第4章 標準サンプルドライバ

プロセッサ内蔵のデバイスに対応したドライバを用意しており、移植性を考慮し、サービスコール（本書では、RTOS のシステムコールに対して、ドライバではサービスコールと呼ぶ）の標準 API を規定しています。ただし、これらのドライバは、本 RTOS を使用する際に必須とはなりません。

4-1. 例外番号割付けドライバ

RA シリーズのプロセッサで割込みを使用するには、割込みの要因となるイベントと割込み入力とを関連付ける必要があります。本ドライバでは、割込みコントローラユニット (ICU) を設定し、これらの割付けを行います。

イベント番号と例外番号を静的に割付ける場合では、スタートアップ時に ICU.IELSRn レジスタに例外番号を設定することで実現できます。

例外番号割付けドライバのサービスコールには、以下の機能があります。

- 例外番号の割付け
- イベント番号の参照
- 例外番号の解放

【コーディング例】

```

#include "kernel.h"
#include "dr_ra8m1.h"
#include "drv_ra8m1_icu.h"

#define EVENT_IRQ12 13

void gpio_s1_int(VP_INT exinf);
void gpio_s2_int(void);

void func(void)
{
    T_CISR cisir_s1;
    ER_ID idno;

    idno = allocate_excp_id(EVENT_IRQ12, 0);
    if (idno > 0) {
        cisir_s1.isratr = TA_HLNG;
        cisir_s1.exinf = (VP_INT)0;
        cisir_s1.intno = (INTNO)idno;
        cisir_s1.isr = gpio_s1_int;
        cisir_s1.imask = 0x70;
        cisir_s1.name = "USER S1";
        acre_isr((T_CISR *)&cisir_s1);
        if (idno == get_excp_id(EVENT_IRQ12))
            release_excp_id(EVENT_IRQ12);
        idno = allocate_excp_id(EVENT_IRQ12, 0);
        if (idno > 0) {
            def_inh(idno, (T_DINH *)&dinh_s2);
        }
    }
}

```

4-1-1. 例外番号の割付け

書式	ER_ID excp_id = allocate_excpc_id(ID event_no, UW attr)		
引数	ID	evento_no	イベント番号
	UW	attr	属性
戻り値	ER_ID	excp_id	正常終了（0 以上）またはエラーコード
エラーコード	E_NOID	ID 番号不足（割付け可能な例外番号がない）	
	E_PAR	パラメータエラー（イベント番号が不正）	
	E_OBJ	オブジェクト状態エラー（割付け済みイベント番号）	

未使用の ICU.IELSRn レジスタを検索し、イベント番号 event_no のイベントを設定し、その例外番号を返します。いずれかの ICU.IELSRn レジスタに当該 event_no が既に設定されていた場合は、E_OBJ エラーを返します。未使用の例外番号に空きが無い場合は、E_NOID エラーを返します。

4-1-2. イベント番号の参照

書式	ER_ID excp_id = get_excpc_id (ID event_no)		
引数	ID	evento_no	イベント番号
戻り値	ER_ID	excp_id	正常終了（0 以上）またはエラーコード
エラーコード	E_PAR	パラメータエラー（イベント番号が不正）	
	E_OBJ	オブジェクト状態エラー（割付けられていないイベント番号）	

イベント番号 event_no のイベントがいずれかの ICU.IELSRn レジスタに設定されテイル場合は、その例外番号を返します。いずれの ICU.IELSRn レジスタにも設定されていない場合は、E_OBJ エラーを返します。

4-1-3. 例外番号の解放

書式	ER ercd = release_excpc_id (ID event_no)		
引数	ID	evento_no	イベント番号
戻り値	ER_ID	excp_id	正常終了（0 以上）またはエラーコード
エラーコード	E_PAR	パラメータエラー（イベント番号の不正）	
	E_OBJ	オブジェクト状態エラー（未割付けのイベント番号）	

イベント番号 event_no のイベントを設定された ICU.IELSRn レジスタを検索し、設定を解除します。いずれの ICU.IELSRn レジスタにも設定されていない場合は、E_OBJ エラーを返します。

4-2. シリアルドライバ

サンプルドライバとしてプロセッサ内蔵のシリアルデバイス (SCI) に対応した調歩同期方式のドライバを収録しています。このシリアルドライバは、移植性を考慮し、サービスコールの標準 API を規定しています。また、コンソールなどの簡素な使用を前提としており、標準の実装ではソフトウェアフロー制御には対応していません。

本シリアルドライバの実装では、チャンネル毎に 1 つのイベントフラグを用います。その生成には `acre_flg` を用いるため、RTOS のコンフィグレーション時にイベントフラグの個数と ID 自動割付けが可能になるよう定義します。また、本シリアルドライバでは、例外番号割付けドライバと一緒に使用します。

シリアルドライバのサービスコールには、以下の機能があります。

- シリアルチャンネルの初期化
- シリアルチャンネルの状態参照
- シリアルチャンネルの制御
- シリアルチャンネルの全送信データの送出待ち
- シリアルチャンネルへの一文字送信
- シリアルチャンネルへの文字列送信
- シリアルチャンネルからの一文字受信
- シリアルチャンネルからの文字列受信

【コーディング例】

```
#include "kernel.h"
#include "drv_uart.h"

T_UART_DEF uart_def = {115200, PAR_NONE, BLEN8, SBIT1};
VB Message[] = "Serial driver¥n";
#define COM_CH 2

void MainTask(VP_INT stacode)
{
    ER ret;
    VB chr;
    UB sts;
    UINT cnt;

    ini_uart(COM_CH, &uart_def);
    ctr_uart(COM_CH, ENA_TX|ENA_RX);

    cnt = sizeof(Message) -1;
    puts_uart(COM_CH, Message, &cnt, TMO_FEVR);
    for(;;) {
        ret = getc_uart(COM_CH, &chr, &sts, 1000);
        if (ret == E_OK) {
            if (chr == 0x1B) {
                break;
            } else {
                putc_uart(COM_CH, chr, TMO_FEVR);
            }
        }
    }
}
```

4-2-1. シリアルドライバのコンフィグレーション

シリアルドライバのコンフィグレーションは、ヘッダファイル `drv_uart_cfg.h` に記述します。また、この中で記述する内容はプロセッサに依存します。

これらのコンフィグレーションを反映するには、シリアルドライバの初期化関数を RTOS の初期化以降に呼び出します。初期化関数は、ドライバのファイル名に `_initialize` が続いた名称になっています。

【RA8M1 シリーズの例】

チャンネル数はプロセッサに依存しますが、チャンネル毎に定義する内容と、共通の内容とがあります。また、UART としていますが、デバイスとしては `SCIIn` を使用します。

チャンネル毎に定義する内容には、以下があります。(UARTx は UART0～)

```
#define CFG_UARTx                /* RA8M1 UARTx を有効にする */
#define CFG_UARTx_SECLIB         /* セキュアライブラリ化する */
#define CFG_UARTx_CTS_RTS      3 /* 0:RTS のみ, 1:CTS のみ, 3:RTS&CTS */
#define CFG_UARTx_TXBUFSZ    1024 /* 送信バッファサイズ */
#define CFG_UARTx_RXBUFSZ    1024 /* 受信バッファサイズ */
#define CFG_UARTx_TWCNT       4 /* 送信割込み発生データ数の閾値 */
#define CFG_UARTx_RWCNT       12 /* 受信割込み発生データ数の閾値 */
#define CFG_UARTx_IPL         0x40 /* 割込み優先度 */
```

送受信 FIFO を無効にする場合は、送受信の FIFO 閾値に 0 を指定します。セキュアライブラリ化が必要ない場合は、`CFG_UARTx_SECLIB` を未定義にします。

共通の内容には、以下があります。

```
#define CFG_UART_PCLKA          12000000UL /* 120MHz */
```

シリアルドライバの初期化は、以下のようにコーディングします。

```
drv_ra8m1_uart_initialize();
```

4-2-2. シリアルチャネルの初期化

書式	ER ercd = ini_uart(UINT ch, T_UART_DEF *pk_uart_def);		
引数	UINT	ch	チャンネル番号
	T_UART_DEF	*pk_uart_def	シリアルチャネル初期化情報を格納したパケットへのポインタ
pk_uart_def の内容 (T_UART_DEF 型)			
	UINT	baoudrate	ボーレート (BPS)
	UINT	parity	パリティビット
	UINT	len	データビット
	UINT	sbit	ストップビット
戻り値	ER	ercd	正常終了 (E_OK) またはエラーコード
エラーコード	E_PAR	パラメータエラー (チャンネル番号、pk_uart_def が不正)	
	E_OBJ	初期化済みシリアルチャネル	
	E_NOSPT	未定義シリアルチャネル	

ch で指定されるチャンネル番号のシリアルチャネルを、pk_uart_def で指定される初期化情報を基にして生成します。初期化情報では、baoudrate でボーレート (BPS) を、parity でパリティビットを、len ではデータビット数を、sbit ではストップビット数を指定します。

パリティビット	
PAR_NONE	パリティビットなし
PAR_EVEN	偶数パリティビット
PAR_ODD	奇数パリティビット
データビット数	
BLLEN8	8 ビットデータ
BLLEN7	7 ビットデータ
BLLEN6	6 ビットデータ
BLLEN5	5 ビットデータ
ストップビット数	
SBIT1	1 ストップビット
SBIT2	2 ストップビット

4-2-3. シリアルチャネルの制御

書式	ER ercd = ctr_uart(UINT ch, UINT cdata);		
引数	UINT	ch	チャネル番号
	UINT	cdata	制御データ
戻り値	ER	ercd	正常終了 (E_OK) またはエラーコード
エラーコード	E_PAR	パラメータエラー (チャネル番号)	
	E_OBJ	未初期化シリアルチャネル	
	E_NOSPT	未定義シリアルチャネル	

ch で指定されるチャネル番号のシリアルチャネルを、cdata で指定される制御データを基にして制御します。制御データでは、送信待ちデータを格納した送信バッファのリセット、デバイスからの受信済みデータを格納した受信バッファのリセット、送受信の許可／禁止を指定します。

制御データ	
RST_TXBUF	送信バッファのリセット (送信待ちデータをクリアする)
RST_RXBUF	受信バッファのリセット (受信データをクリアする)
DIS_TX	送信禁止
DIS_RX	受信禁止
ENA_TX	送信許可
ENA_RX	受信許可
DIS_CTS	CTS 制御禁止
DIS_RTS	RTS 制御禁止
ENA_CTS	CTS 制御許可
ENA_RTS	RTS 制御許可

4-2-4. シリアルチャネルの全送信データの送出待ち

書式	ER ercd = fls_uart(VB data, TMO timeout);		
引数	UINT	ch	チャンネル番号
	TMO	timeout	タイムアウト指定
戻り値	ER	ercd	正常終了 (E_OK) またはエラーコード
エラーコード	E_PAR	パラメータエラー (チャンネル番号)	
	E_OBJ	未初期化シリアルチャネル、送信待ちタスク有り	
	E_NOSPT	未定義シリアルチャネル	
	E_TMOUT	ポーリング失敗またはタイムアウト	

ch で指定されるチャンネル番号のシリアルチャネルの送信バッファと送信 FIFO に格納されたデータが、すべて送信されるまで待ち状態に遷移します。初期化していないシリアルチャネルか、送信許可にされていない場合は、またはディスパッチ保留状態で呼び出された場合は、E_OBJ エラーを返します。

タイムアウト指定 timeout に TMO_POL が指定された場合は待ち状態には遷移しないポーリング動作を、TMO_FEVR が指定された場合は送信できるまで永久に待ち状態になる動作を、それら以外の場合は送信できるまでかタイムアウト時刻になるまで待ち状態になる動作をします。

4-2-5. シリアルチャネルへの一文字送信

書式	ER ercd = putc_uart(UINT ch, VB data, TMO timeout);		
引数	UINT	ch	チャンネル番号
	VB	data	送信文字
	TMO	timeout	タイムアウト指定
戻り値	ER	ercd	正常終了 (E_OK) またはエラーコード
エラーコード	E_PAR	パラメータエラー (チャンネル番号)	
	E_OBJ	未初期化シリアルチャネル、送信待ちタスク有り	
	E_NOSPT	未定義シリアルチャネル	
	E_TMOUT	ポーリング失敗またはタイムアウト	

ch で指定されるチャンネル番号のシリアルチャネルから、data で指定される文字を送信します。初期化していないシリアルチャネルか、送信許可にされていない場合は、またはディスパッチ保留状態で呼び出された場合は、E_OBJ エラーを返します。

タイムアウト指定 timeout に TMO_POL が指定された場合は待ち状態には遷移しないポーリング動作を、TMO_FEVR が指定された場合は送信できるまで永久に待ち状態になる動作を、それら以外の場合は送信できるまでかタイムアウト時刻になるまで待ち状態になる動作をします。

4-2-6. シリアルチャネルへの文字列送信

書式	ER ercd = puts_uart(UINT ch, VB *p_data, UINT *p_count, TMO timeout);		
引数	UINT	ch	チャネル番号
	VB	p_data	送信文字列を格納した領域の番地
	UINT	p_count	送信文字数を格納した領域の番地
	TMO	timeout	タイムアウト指定
戻り値	ID	ercd	正常終了 (E_OK) またはエラーコード
エラーコード	E_PAR	パラメータエラー (チャネル番号)	
	E_OBJ	未初期化シリアルチャネル、送信待ちタスク有り	
	E_NOSPT	未定義シリアルチャネル	
	E_TMOUT	ポーリング失敗またはタイムアウト	

ch で指定されるチャネル番号のシリアルチャネルから、p_data で指定される文字列の p_data で指定された文字数だけ送信します。初期化していないシリアルチャネルか、送信許可にされていない場合は、またはディスパッチ保留状態で呼び出された場合は、E_OBJ エラーを返します。

タイムアウト指定 timeout に TMO_POL が指定された場合は待ち状態には遷移しないポーリング動作を、TMO_FEVR が指定された場合は送信できるまで永久に待ち状態になる動作を、それら以外の場合は送信できるまでかタイムアウト時刻になるまで待ち状態になる動作をします。

4-2-7. シリアルチャネルからの一文字受信

書式	ER ercd = getc_uart(UINT ch, VB *p_data, UB *p_sts, TMO timeout);		
引数	UINT	ch	チャネル番号
	VB	p_data	受信文字の格納領域の番地
	UB	p_sts	エラーステータス
	TMO	timeout	タイムアウト指定
戻り値	ER	ercd	正常終了 (E_OK) またはエラーコード
エラーコード	E_PAR	パラメータエラー (チャネル番号)	
	E_OBJ	未初期化シリアルチャネル、送信待ちタスク有り	
	E_NOSPT	未定義シリアルチャネル	
	E_TMOUT	ポーリング失敗またはタイムアウト	

ch で指定されるチャネル番号のシリアルチャネルから受信し、p_data で指定される領域に文字を、p_sts で指定される領域にエラーステータスを格納します。p_sts に 0 を指定した場合は、エラーステータスを返しません。初期化していないシリアルチャネルか、送信許可にされていない場合は、またはディスパッチ保留状態で呼び出された場合は、E_OBJ エラーを返します。

タイムアウト指定 timeout に TMO_POL が指定された場合は待ち状態には遷移しないポーリング動作を、TMO_FEVR が指定された場合は受信できるまで永久に待ち状態になる動作を、それら以外の場合は送信できるまでかタイムアウト時刻になるまで待ち状態になる動作をします。

エラーステータス	
UART_ERP	パリティエラー
UART_BRK	ブレーク検出
UART_ERF	受信フレームエラー
UART_EROR	受信バッファのオーバフローエラー

4-2-8. シリアルチャネルからの文字列受信

書式	ER ercd = gets_uart(UINT ch, VB *p_data, UB *p_sts, UINT *p_count, TMO timeout);		
引数	UINT	ch	チャネル番号
	VB	p_data	受信文字の格納領域の番地
	UB	p_sts	エラーステータスの格納領域の番地
	UINT	p_count	受信文字数の格納領域の番地
	TMO	timeout	タイムアウト指定
戻り値	ER	ercd	正常終了 (E_OK) またはエラーコード
エラーコード	E_PAR	パラメータエラー (チャネル番号)	
	E_OBJ	未初期化シリアルチャネル、送信待ちタスク有り	
	E_NOSPT	未定義シリアルチャネル	
	E_TMOUT	ポーリング失敗またはタイムアウト	

ch で指定されるチャネル番号のシリアルチャネルから受信し、p_data で指定される領域に文字を、p_sts で指定される領域にエラーステータスを、p_count で指定した領域に受信した文字数を格納します。p_sts に 0 を指定した場合は、エラーステータスを返しません。初期化していないシリアルチャネルか、送信許可にされていない場合は、またはディスパッチ保留状態で呼び出された場合は、E_OBJ エラーを返します。

タイムアウト指定 timeout に TMO_POL が指定された場合は待ち状態には遷移しないポーリング動作を、TMO_FEVR が指定された場合は受信できるまで永久に待ち状態になる動作を、それら以外の場合は送信できるまでかタイムアウト時刻になるまで待ち状態になる動作をします。

エラーステータスは、「4-2-7. シリアルチャネルからの一文字受信」と同様です。

4-2-9. シリアルチャネルの状態参照

書式	ER ercd = ref_uart(UINT ch, T_UART_DEF *pk_uart_ref);		
引数	UINT	ch	チャンネル番号
	T_UART_REF	*pk_uart_ref	シリアルチャネル状態を格納した領域へのポインタ
pk_uart_ref の内容 (T_UART_REF 型)			
	UINT	status	ステータス
	UINT	rbuffer	受信バッファ内のデータ数
	UINT	sbuffer	送信バッファ内のデータ数
戻り値	ER	ercd	正常終了 (E_OK) またはエラーコード
エラーコード	E_PAR	パラメータエラー (チャンネル番号が不正)	
	E_NOSPT	未定義シリアルチャネル	

ch で指定されるチャンネル番号のシリアルチャネルの状態を、pk_uart_ref で指定される領域に格納します。ステータス status にはシリアルドライバの状態を、rbuffer には受信済みで getc_uart 或いは gets_uart により読み出していない受信データの数を、sbuffer には puc_uart 或いは puts_uart により送信を試みたが未だに送信できていない送信データの数が格納されます。

ステータス	
UART_INIT	初期化済み状態
UART_ENTX	送信許可状態
UART_ENRX	受信許可状態

(余白)

(余白)

Cortex-M 対応 MULCOS-MK μ ITRON4.0 仕様版

RA デバイス依存部ユーザーズガイド

2025 年 4 月 第 1 版発行

株式会社スパイラルテック

Mail tech-info@spiral-tech.co.jp