μT-Kernel3.0
EK-RX72N（RXv3コア）向け
実装仕様書

Version.01.00.00

2023.10.29

目次

1. 概要

1.1 目的

本書はEK-RX72N（RXv3コア）向けのμT-Kernel3.0の実装仕様を記載した実装仕様書である。対象は、TRONフォーラムから公開されているμT-Kernel3.0(V3.00.00)をルネサスエレクトロニクス社のRX用に改変したソースコードのうち、EK-RX72N（RX72N ENVISION KIT）向けの実装部分である。

ハードウェアに依存しない共通の実装仕様はμT-Kernel3.0共通実装仕様書を参照のこと。以降、単にOSと称する場合はμT-Kenrel3.0を示し、本実装と称する場合、前述の実装を示す。

1.2 対象ハードウェア

実装対象のハードウェアは以下の通りである。

分類	名称	備考
実機	EK-RX72N-	ルネサスエレクトロニクス製
搭載マイコン	RX72N R5F572NNHxFB	ルネサスエレクトロニクス製

1.3 ターゲット名

EK-RX72Nのターゲット名は以下とする。

分類	名称	対象
ターゲット名	_EK_RX72N_	
対象システム	EK-RX72N	
対象CPU	RX72N	R5F572NNHxFB
対象CPU アーキテクチャ	CPU_CORE_RXV3	RXv3コア

1.4 関連ドキュメント

OSの標準的な仕様は「μT-Kernel3.0仕様書」に記載される。

ハードウェアに依存しない共通の実装仕様は「μT-Kernel3.0共通実装仕様書」に記載される。

また、対象とするマイコンを含むハードウェアの仕様は、それぞれの仕様書などのドキュメントに記載される。以下に関連するドキュメントを記す。

分類	名称	発行
OS	μT-Kernel3.0仕様書 (Ver.3.00.00)	TRONフォーラム TEF020-S004-3.00.00
	μT-Kernel3.0共通実装仕様書	TRONフォーラム TEF033-W002-191211
T-Monitor	T-Monitor仕様書	TRON フォーラム TEF-020-S002-01.00.01
搭載マイコン	RX72Nグループ ユーザーズマニュアル　ハードウェア編	ルネサスエレクトロニクス

1.5　ソースコード構成

機種依存定義sysdepedディレクトリ下の本実装のディレクトリ構成を以下に示す。太文字で書かれた箇所が、本実装のディレクトリである。

```
― sysdepend                          実装依存定義
    ├ ek_rx72n                             EK-RX72N RX 依存部
    ├ :
    ├ <ターゲットn>                        ターゲットn 依存部
    └ cpu                                  CPU 依存部
        ├ <r5f572n>                            R5F572N 依存部
        ├ :
        └ <CPUn>                               CPUn 依存部
            └ core                             コア依存部
                ├ <rxv3>                           RXv3 依存部
                ├ :
                └ <core n>                         コアn 依存部
```

2.　基本実装仕様

2.1　対象マイコン

実装対象のマイコンの基本的な仕様を以下に記す。

項目	内容
CPUコア	RXv3
ROM	4MB（内蔵フラッシュROM）
RAM	1MB（内蔵RAM）

2.2　プロセッサモードと保護レベル

RXv3コアはプロセッサモードとしてスーパバイザモードとユーザモードの２種類を持っているが、本実装ではスーパバイザモードで動作し、ユーザモードで動作することはない。

OSが提供する保護レベルは、すべて保護レベル0とみなす。カーネルオブジェクトに対して保護レベル1～3を指定しても保護レベル0を指定されたものとして扱う。

プロファイル TK_MEM_RING0 ～ TK_MEM_RING3 はすべて 0 が返される。

2.3　CPU レジスタ

本マイコンは内部レジスタとして、汎用レジスタ（R1-R15）、PC、PSW、ACC0、ACC1、FPSW、SPを有する。ただし、ACC0、ACC1とFPSWはコンフィグレーション（USE_DSPマクロとUSE_FPUマクロ）によってはタスクコンテキストの対象外となる。また、SPにはISPとUSPが存在するが、タスクコンテキストとして使用されるのはUSPである。

OSのAPI（tk_set_reg、tk_get_reg）を用いて実行中のタスクのコンテキストのレジスタ値を操作できる。APIで使用するマイコンのレジスタのセットは以下のように定義する。

(1) 汎用レジスタ T_REGS

```
typedef struct t_regs {
        UW      r[15];          /* 汎用レジスタ R1-R15 */
} T_REGS;
```

(2) 例外時に保存されるレジスタ T_EIT

```
typedef struct t_eit {
        UW      pc;             /* プログラムカウンタ */
        UW      psw;            /* プロセッサステータスワード */
} T_EIT;
```

(3) 制御レジスタ T_CREGS

```
typedef struct t_cregs {
        UW      acc0[3];        /* アキュムレータ */
        UW      acc1[3];        /* アキュムレータ */
        UW      fpsw;           /* 浮動小数点ステータスワード */
        void    *sp;            /* スタックポインタ */
} T_CREGS;
```

OSのAPIによって操作されるのは、実際にはスタック上に退避されたレジスタの値である。よって、実行中のタスクに操作することはできない（OS仕様上、自タスクへの操作、またはタスク独立部からのAPI呼出しは禁止されている）。

2.4 低消費電力モードと省電力機能

省電力機能はサポートしていない。プロファイル TK_SUPPORT_LOWPOWER は FALSE である。

よって、マイコンの低消費電力モードに対応する機能は持たない。

省電力機能のAPI（low_pow、off_pow）は、/kernel/sysdepend/ek_rx72n/power_func.cに空関数として記述されている。

なお、本関数に適切な省電力処理を記述し、プロファイル TK_SUPPORT_LOWPOWER を TRUE に変更すれば、OSの省電力機能（tk_set_pow）は使用可能となる。

2.5 コプロセッサ対応

本マイコンはコプロセッサ（倍精度浮動小数点レジスタ）として、DR0-DR15、DPSW、DCMR、DECNT、DEPCを有する。

よって、プロファイル TK_SUPPORT_FPU と TK_SUPPORT_COP0 は TRUE 、TK_SUPPORT_COPn（nは1～3）はすべて FALSE である。ただし、CC-RXのオプション設定によって、コプロセッサ（倍精度浮動小数点レジスタ）を未サポートにすることができる（詳細はuTK3.0_EK-RX72N構築手順書を参照のこと）。その場合、プロファイル TK_SUPPORT_FPU と TK_SUPPORT_COP0 は FALSE となる。

コプロセッサに関するAPI（tk_set_cpr, tk_get_cpr）で使用するコプロセッサのセットは以下のように定義する。

倍精度浮動小数点レジスタ T_COPREGS

```
typedef struct t_copregs {
        UD      dr[16];         /* 倍精度浮動小数点データレジスタ DR0-DR15 */
        UW      dpsw;           /* 倍精度浮動小数点ステータスワード DPSW */
        UW      dcmr;           /* 倍精度浮動小数点比較結果レジスタ DCMR */
        UW      decnt;          /* 倍精度浮動小数点例外処理動作制御レジスタ DECNT */
        UW      depc;           /* 倍精度浮動小数点例外プログラムカウンタ DEPC */
} T_COPREGS;
```

3. メモリ

3.1 メモリモデル

RXv3は4G（32bit） のアドレス空間を有するが、MMU（Memory Management Unit：メモリ管理ユニット)は有さず、単一の物理アドレス空間のみである。

本実装では、プログラムは一つの実行オブジェクトに静的にリンクされていることを前提とする。OSとユーザプログラム（アプリケーションなど）は静的にリンクされており、関数呼出しが可能とする。

3.2 マイコンのアドレス・マップ

アドレス・マップは、実装対象のマイコンのアドレス・マップに従う。

以下にRX72N（R5F572NNHxFB）のアドレス・マップを記す。（詳細はマイコンの仕様書を参照のこと）。

アドレス （開始 ～ 終了）	種別	サイズ （KByte）	備考
0x00000000 ～ 0x0007FFFF	内蔵RAM	512	データ領域
0x00100000 ～ 0x00107FFF	データフラッシュメモリ	32	
0x00800000 ～ 0x0087FFFF	内蔵RAM	512	RAMディスク
0xFFC00000 ～ 0xFFFFFFFF	内蔵ROM	4096	プログラム領域

3.3 OS のメモリマップ

本実装ではマイコンの内蔵ROMおよび内蔵RAMを使用する。

OSを含む全てのプログラムのコードは内蔵ROMに配置され、実行される。以下に内蔵ROMおよび内蔵RAMのメモリマップを示す。表中でアドレスに「ー」が記載された箇所はデータのサイズによりC言語の処理系にてアドレスが決定され、OS内ではアドレスの指定は行っていない。

(1) 内蔵ROMのメモリマップ

アドレス （開始 ～ 終了）	種別	内容
0xFFC00000 ～ 0xFFC003FF	割込みベクタテーブル	
0xFFC00400 ～ ー	プログラムコード	C言語のプログラムコードが配置される領域
ー	定数データ	C言語の定数データなどが配置される領域
0xFFFFFF80 ～ 0xFFFFFFFF	例外ベクタテーブル	

(2) 内蔵RAMのメモリマップ

アドレス （開始 ～ 終了）	種別	内容
0x00000000 ～ ー	プログラムデータ	C言語の変数等が配置される領域
ー	OS管理領域	OS内部の動的メモリ管理の領域

ー　～ 0x0007FFFF	例外スタック領域	割込みハンドラなどのスタック領域

3.4　スタック

本実装で使用するスタックには共通仕様に従い以下の種類がある。

(1) タスクスタック

割込みハンドラ以外で使用するスタックであり、タスク毎に１本ずつ存在する。

tk_cre_tsk発行時のスタックサイズ（T_CSTK.stksz）で指定する。

(2) 例外スタック

割込みハンドラで使用するスタックであり、システムスタックとは独立したスタック領域が割り当てられる。割込みスタックのサイズは/config/config.h(inc)の CFN_EXC_STACK_SIZE で指定する。

【備考】

各種スタックのサイズ計算方法に関しては、uTK3.0_EK-RX72N構築手順書を参照のこと。

3.5　OS 内の動的メモリ管理

OSのAPIの処理において以下のメモリが動的に確保される。

- メモリプールのデータ領域
- メッセージバッファのデータ領域
- タスクのスタック

ただし、コンフィギュレーション USE_IMALLOC が指定されていない場合は、動的メモリ管理は行われない（初期値は動的メモリ管理を行う）。OS内の動的メモリ管理に使用するOS管理メモリ領域（システムメモリ領域）は、以下のように定められる。

(1) OS管理メモリ領域の開始アドレス

コンフィギュレーション CFN_SYSTEMAREA_TOP の値が 0 以外の場合は、その値が開始アドレスとなる。コンフィギュレーション CFN_SYSTEMAREA_TOP の値が 0 の場合、プログラムが使用しているデータ領域の最終アドレスの次のアドレスが開始アドレスとなる。

(2) OS管理メモリ領域の終了アドレス

コンフィギュレーション CFN_SYSTEMAREA_END の値が 0 以外の場合は、その値が終了アドレスとなる。コンフィギュレーション CFN_SYSTEMAREA_END の値が 0 の場合、例外スタックの開始アドレスの前のアドレスが終了アドレスとなる。

4.　割込みおよび例外

4.1　マイコンの割込みおよび例外

本マイコンには以下の例外が存在する。なお、OS仕様上は例外、割込みをまとめて、割込みと称している。

割込み番号 （ベクタ番号）	割込みの種別	備考
例外ベクタテーブル（0xFFFFFFFE）	リセット	OSで使用
例外ベクタテーブル（上記以外）	内部・外部割込み	OSで使用（無限ループ）
割込みベクタテーブル（0）	BRK命令の実行	OSで使用
割込みベクタテーブル（28）	CMT0のCMI0割込み	OSで使用
割込みベクタテーブル（上記以外）	内部・外部割込み	OSの割込み管理機能で管理

4.2　ベクタテーブル

本マイコンでは前述の各種例外に対応する例外ハンドラのアドレスを設定したベクタテーブルを有する。本実装では、リセット時のベクタテーブルは/kernel/sysdepend/cpu/r5f572n/fixed_vector.src、リセット以外のベクタテーブルは/kernel/sysdepend/cpu/r5f572n/vector.srcに定義される。

4.3　割込み優先度とクリティカルセクション

4.3.1　割込み優先度

RXv3コアは、割込み優先度を4bit(0～15)の15段階（0は除く）に設定できる（優先度の数字の大きい方が優先度は高い）。

4.3.2　多重割込み対応

本マイコンの割り込み要因プライオリティレジスタ（IPRn）の設定により，多重割込みに対応している。割込みハンドラの実行中に、より優先度の高い割込みが発生した場合は実行中の割込みハンドラに割り込んで優先度の高い割込みハンドラが実行される。

4.3.3　クリティカルセクション

本実装では、クリティカルセクションはCPU内部レジスタのPSWに最高外部割込み優先度 MAX_INT_PRI を設定することにより実現する。MAX_INT_PRI は、本OSが管理する割込みの最高の割込み優先度であり、/include/sys/sysdepend/cpu/r5f572n/sysdef.h(inc)にて以下のように定義される。

```
#define MAX_INT_PRI             (12)            // sysdef.h
MAX_INT_PRI             .EQU    (12)            ;  sysdef.inc
```

クリティカルセクション中は MAX_INT_PRI 以下の優先度の割込みはマスクされる。

4.3.4　システムタイマの割込み優先度

本実装では、システムタイマとしてCMT0（コンペアマッチタイマのチャネル０）を使用するが、CMT0の割込みは TIM_INT_PRI の割込み優先度までマスクされた状態で実行される。TIM_INT_PRI はシステムタイマの割込み優先度であり、/include/sys/sysdepend/cpu/r5f572n/sysdef.h(inc)にて以下のように定義される。

```
#define TIM_INT_PRI             (10)            // sysdef.h
TIM_INT_PRI             .EQU    (10)            ;  sysdef.inc
```

4.3.5　各割込み優先度の関係

OS管理外の割込み優先度、クリティカルセクションの割込み優先度、システムタイマの割込み優先度は以下の条件が満足されれば変更可能である。

OS管理外の割込み ＞ クリティカルセクション ≧ システムタイマ

4.4　OS 内部で使用する割込み

本OSの内部で使用する割込みには、以下のように本マイコンの割込みまたは例外が割り当てられる。該当する割込みまたは例外はOS以外で使用してはならない。

割込み番号 （ベクタ番号）	割込みの種別	備考
例外ベクタテーブル（0xFFFFFFFE）	リセット	
割込みベクタテーブル（0）	CMT0のCMI0割込み	システムタイマとして使用
割込みベクタテーブル（28）	BRK命令の実行	割込みハンドリングに使用

4.5　μT-Kenrel/OS の割込み管理機能

本実装の割込み管理機能は、マイコンの内部・外部割込み(OS内部で使用する割込み以外)を対象とし、割込みベクタテーブルの割込みのみ割込みハンドラの管理を行う。例外ベクタテーブルの内部割込みは管理しない。

4.5.1　割込み番号

OSの割込み管理機能が使用する割込み番号は割込みベクタテーブルのベクタ番号と同一とする。例えば、IRQ0は割込み番号64となる。

4.5.2　割込みハンドラの優先度

割込みハンドラの優先度（当該割込みの割込み優先順位、4.3.2項参照）は、「4.3.5　各割込み優先度の関係」に記載した「使用可能なOS管理内の割込み優先度」が使用可能である。逆に同項に記載した「OS管理外の割込み優先度」は使用してはならない。

4.5.3　割込みハンドラ属性

本実装では、TA_ASM属性の割込みハンドラはサポートしていない。割込みハンドラはTA_HLNG属性を指定したC言語の関数としてのみ記述可能である。TA_HLNG属性の割込みハンドラは、割込みの発生後、OSの割込み処理ルーチン（BRK命令を使用）を経由して呼び出される。OSの割込み処理ルーチンでは割込みハンドラの実行が行われる。

なお、割込みハンドラでFPSWやACC0、ACC1を利用することは許されない。例え USE_FPU や USE_DSP のコンフィグレーションを有効にしたとしても、割込みハンドラ内での使用は禁止である。同様に割込みハンドラ内で倍精度浮動小数点レジスタを利用こともできない。

4.5.4　デフォルトハンドラ

割込みハンドラが未登録の状態で割込みが発生した場合はデフォルトハンドラが実行される。デフォルトハンドラは/kernel/sysdepend/cpu/core/rxv3/interrupt.cのDefault_Handler関数として実装されている。

デフォルトハンドラはプロファイル USE_EXCEPTION_DBG_MSG を有効にすることにより、デバッグ用の情報を出力する（初期設定は有効である）。

必要に応じてユーザがデフォルトハンドラを記述することにより、未定義割込み発生時の処理を行うことができる。

4.6　μT-Kernel/SM の割込み管理機能

μT-Kernel/SMの割込み管理機能は、CPUの割込み管理機能および割込み優先順位フラグ・レジスタの制御を行う。各APIの実装方法について以降に記す。

4.6.1　CPU 割込み制御

CPU割込み制御はマイコンのPSW（プロセッサステータスワード）を制御して実現する。

① CPU割込みの禁止（DI）

CPU割込みの禁止（DI）は、PSWのIPLに最高外部割込み優先度 MAX_INT_PRI を設定し、それ以下の優先度の割込みを禁止する。

② CPU割込みの許可（EI）

割込みの許可（EI）は、PSWのIPLの値をDI実行前に戻す。

③ CPU内割込みマスクレベルの設定（SetCpuIntLevel）

CPU内割込みマスクレベルの設定（SetCpuIntLevel）は、PSWのIPLの値を指定した割込みマスクレベルに設定する。割込みマスクレベルは 0 から 15 の値（値の大きい方が高い優先度）が指定可能である。指定したマスクレベル以下の優先度の割込みはマスクされる。また、割込みマスクレベルに 0 が指定された場合は、すべての割り込みはマスクされない。

④ CPU内割込みマスクレベルの参照（GetCpuIntLevel）

CPU内割込みマスクレベルの取得（GetCpuIntLevel）は、PSWのIPLの設定値を参照する。

4.6.2　割込みコントローラ制御

マイコン内蔵の割り込み要因プライオリティレジスタ（IPRn）、割り込み要求許可レジスタ（IERm）、割り込み要求レジスタ（IRn）の制御を行う。

① 割込みコントローラの割込み許可（EnableInt）

割り込み要求許可レジスタ（IERm）を設定し、指定された割込みを許可する。同時に割り込み要因プライオリティレジスタ（IPRn）に指定された割込み優先度を設定する。割込み優先度は 0 から 15 の値が使用可能である。

② 割込みコントローラの割込み禁止（DisableInt）

割り込み要求許可レジスタ（IERm）を設定し、指定された割込みを禁止する。

③ 割込み発生のクリア（ClearInt）

割り込み要求レジスタ（IRn）を設定し、指定された割込みが保留されていればクリアする。

④ 割込みコントローラのEOI 発行（EndOfInt）

本マイコンではEOIの発行は不要である。よって、EOI発行（EndOfInt）は何も実行しないマクロとして定義される。

⑤ 割込み発生の検査（CheckInt）

割込み発生の検査（CheckInt）は、割り込み要求レジスタ（IRn）を参照することにより実現する。

⑥ 割込みモード設定（SetIntMode）

未実装である（プロファイル TK_SUPPORT_INTMODE は FLASE である）。

⑦ 割込みコントローラの割込みマスクレベル設定（SetCtrlIntLevel）

本機能はないため、未実装である（プロファイル TK_SUPPORT_CTRLINTLEVEL は FLASE である）。

⑧ 割込みコントローラの割込みマスクレベル取得（GetCtrlIntLevel）

本機能はないため、未実装である（プロファイル TK_SUPPORT_CTRLINTLEVEL は FLASE である）。

4.7　OS 管理外割込み

最高外部割込み優先度 MAX_INT_PRI より優先度の高い割込みは、OS管理外割込みとなる。OS管理外割込みはOS自体の動作よりも優先して実行される。よって、OSから制御することはできない。また、管理外割込みの中でOSのAPIなどの機能を使用することも原則できない。

本実装のデフォルト設定では MAX_INT_PRI は優先度 12 と定義されている。ただし、「4.3.5　各割込み優先度の関係」の内容に従って優先度 1 から 15 に変更可能である。よって、優先度 13 から 15 以外にも優先度 2 から 15 までをOS管理外割込みとすることが可能ある。

4.8　OS 管理外割込みの記述例

OS管理外割込みは、OSの割込み処理ルーチンを介さず、割込み発生時に直接起動されなければならない。このため、以下に記載する２つの手順が必要となる。なお、以下はベクタ番号31、コンペアマッチタイマWのチャネル１の割込み（CMTW1のCMWI1割込み）を例に紹介する。

4.8.1　ベクタテーブルの改変

ベクタテーブル/ kernel/sysdepend/cpu/r5f572n/vector.srcに登録されている当該ベクタのOS処理ルーチンのベクタアドレスを削除またはコメントアウトする。

【改変前のソースコード】

```
        .RVECTOR  30, knl_inthdr_entry30        ; CMTW0   CMWI0
        .RVECTOR  31, knl_inthdr_entry31        ; CMTW1   CMWI1
;       .RVECTOR  32, knl_inthdr_entry32        ;
```

【改変後のソースコード】

```
        .RVECTOR  30, knl_inthdr_entry30        ; CMTW0   CMWI0
;       .RVECTOR  31, knl_inthdr_entry31        ; CMTW1   CMWI1
;       .RVECTOR  32, knl_inthdr_entry32        ;
```

目的の行の先頭カラムに「;」を入れれば、その行の終わりまでがコメントとなり、フォント色が緑色になる。

4.8.2　OS 管理外割込み関数の記述

OS管理外の割込み関数は、CC-RXコンパイラが持つ#pragma interruptの拡張機能を使って記述する。また、割込み仕様でベクタテーブル指定（vect=ベクタ番号）と、必要であれば多重割込み許可指定（enable）やレジスター括退避機能の利用（bank=バンク番号）を行う。

(1) ベクタテーブル指定（vect=ベクタ番号）

ベクタ番号に当該割込みのベクタ番号を指定することでベクタテーブルが生成できる。ベクタ番号は定数、またはiodefine.hのヘッダファイルをインクルードすることでVECTマクロを使うことが可能である。

(2) 多重割込み許可指定（enable）

必要に応じて多重割込みの許可が可能である。enableを指定すれば多重割込み許可、未指定ならば多重割込み禁止となる。

(3) レジスター括退避機能の利用（bank=バンク番号）

必要に応じてレジスター括退避機能の利用が可能である。bankを指定すれば汎用レジスタ等の退避・回復命令が不要となり、例外スタック領域の削減も可能となります。ただし、バンク番号は割込みレベルと同一の値を指定すること。それ以外の値を指定した際の動作は保証しない。

【OS管理外割込み関数の例】

```
#pragma interrupt cmtw1_cmwi1(vect=VECT( CMTW1, CMWI1 ),bank=14,enable)
void cmtw1_cmwi1(void)
{

}
```

5.　起動および終了処理

5.1　リセット処理

リセット処理は、マイコンのリセットベクタに登録され、マイコンのリセット時に実行される。リセット処理はRXv3コアに固有の処理であるため/kernel/sysdepend/cpu/core/rxv3/resetprg.srcのスタートアップ・ルーチン経由で/kernel/sysdepend/cpu/core/rxv3/reset_hdr.cのReset_Handler関数として実装される。

Reset_Handler関数の処理手順を以下に示す。なお、　なお、C言語のグローバル変数領域の初期化はresetprg.srcのスタートアップ・ルーチンで行われている。

(1) ハードウェア初期化（knl_startup_device）

リセット時の必要最小限のハードウェアの初期化を行う。

knl_startup_device は「5.2　デバイスの初期化および終了処理」を参照のこと。

(2) OS初期化処理（main）の実行

リセット処理を終了するOSの初期化処理（main）を実行し、リセット処理は終了する。

5.2　デバイスの初期化および終了処理

デバイスの初期化および終了処理は、以下の関数として実装されている。ユーザのアプリケーションに応じて、関数の処理内容の変更は可能である。ただし、これらの関数はOSの共通部からも呼ばれるため、形式を変更してはならない。

ファイル：/kernel/sysdepend/ek_rx72n/start_dev.c

関数名	内容
knl_startup_device	**デバイスのリセット** リセット時の必要最小限のハードウェアの初期化を行う。 本実装では処理は未実装である。
knl_shutdown_device	**デバイスの停止** 周辺デバイスをすべて終了し、マイコンを終了状態とする。 本実装では、割込み禁止状態で無限ループとしている。
knl_restart_device	**デバイスの再起動** 周辺デバイスおよびマイコンの再起動を行う。 本実装ではデバイスの再起動には対応していない。処理のひな型のみを記述している。

ファイル：/kernel/sysdepend/ek_rx72n/devinit.c

関数名	内容
knl_init_device	**デバイスの初期化** 周辺デバイスの初期化を行う。

	本実装では簡易I2Cのデバイスドライバが実装済みである。
knl_start_device	**デバイスの実行** デバイスドライバの登録、実行を行う。 本実装では処理は未実装である(※)。
knl_finish_device	**デバイスの終了** デバイスドライバを終了する。 本実装では処理は未実装である(※)。

※ 本実装では、デバイスドライバを登録していないため、関数は何の処理も行っていない。
デバイスドライバを登録する場合は、上記の関数に必要な処理を記述する。記述内容は、基本実装仕様書を参照のこと。

6.　その他の実装仕様

6.1　タスク

6.1.1　タスク属性

タスク属性のハードウェア依存仕様を以下に示す。

属性	可否	説明
TA_COP0	○	本マイコンは倍精度浮動小数点コプロセッサを有する。
TA_FPU	○	
TA_COPn（nは1～3）	×	本マイコンは上記以外のコプロセッサは持っていない。

6.1.2　タスクの処理ルーチン

タスクの処理ルーチンの実行開始時の各レジスタの状態は以下である。これ以外のレジスタの値は不定である。

レジスタ	値	補足
PSW	0x00030000	割込み許可
FPSW	0x00000100	USE_FPU が 1 の場合
R1	第一引数	stacd タスク起動コード
R2	第二引数	*exinf タスク拡張情報
USP	タスクスタックの先頭アドレス	

また、タスク属性として TA_FPU または TA_COP0 を指定したタスクの処理ルーチンの実行開始時の倍精度浮動小数点レジスタの状態は以下である。これ以外の倍精度浮動小数点レジスタの値は不定である。

レジスタ	値	補足
DPSW	0x00000100	
DCMR	0x00000000	
DECNT	0x00000001	

6.2　時間管理機能

6.2.1　システムタイマ

本実装では、マイコン内蔵のインターバル・タイマをシステムタイマとして使用する。

システムタイマのティック時間は1ミリ秒から50ミリ秒の間で設定できる。

ティック時間の標準の設定値は1ミリ秒である。

6.2.2　タイムイベントハンドラ

タイムイベントハンドラの実行中の割込み優先度は、システムタイマの割込み優先度 TIM_INT_PRI と同じであり、タイムイベントハンドラの実行中は TIM_INT_PRI 以下の優先度の割込みはマスクされる。TIM_INT_PRI の設定値に関しては「4.3.4　システムタイマの割込み優先度」を参照。

6.3 T-Monitor 互換ライブラリ

6.3.1 ライブラリ初期化

T-Monitor互換ライブラリを使用するにあたって、ライブラリの初期化関数を実行する必要がある。
本初期化関数はOSの起動処理の中で実行される。

6.3.2 コンソール入出力

APIによるコンソール入出力の仕様を以下に示す。

項目	内容
デバイス	内蔵SCI Channel 2
ボーレート	115,200bps
データ形式	data 8bit, stop 1bit, no parity

7.　簡易 I2C ドライバ

7.1　簡易 I2C ドライバの概要

簡易I2Cドライバは、マイコン内蔵のSCIチャネルを簡易I2Cモードで利用するためのドライバです。また、簡易I2CドライバをμT-Kernelに登録するためのsiicDriverEntry関数をサービス関数として提供します。ユーザシステムではsiicDriverEntry関数を呼び出すことにより、以降の処理で簡易I2CドライバをμT-Kernel/SMのデバイス管理機能を使って利用可能となります。

7.2　簡易 I2C ドライバの仕様

7.2.1　対象デバイス

内蔵SCIチャネルを簡易I2Cモードで利用するデバイスです。

7.2.2　デバイス名

デバイス名は "siicl" です。

7.2.3　固有機能

なし

7.2.4　属性データ

なし

7.2.5　固有データ

以下の固有データをサポートします。

開始番号（start）：デバイスアドレス
サイズ（size）　：デバイスと送受信するデータのサイズ
データ（buf）　　：送受信するデータの先頭アドレス

7.2.6　事象通知

未サポート

7.2.7　エラーコード

μT-Kernel3.0仕様書のデバイス管理機能の項を参照ください。簡易I2C固有の特殊なエラーコードは存在しません。

7.3　簡易 I2C ドライバが使用する資源

7.3.1　使用する資源の概要

簡易I2Cドライバは処理を実施するに当たり、１つのイベントフラグ、１つのタスク、２つの割込みハンドラを利用します。これらの資源は7.5節で紹介するsiicDrvEntry関数を呼び出すと生成または定義されます。

使用する資源	名称	優先度
タスク	siic_tsk	タスクの優先度は SIIC_CFG_TASK_PRIORITY
割込みハンドラ	sci11_txi11_hdr	割込みハンドラの割込み優先度は SIIC_CFG_INT_PRIORITY
	sci11_rxi11_hdr	

注：SIIC_CFG_TASK_PRIORITY と SIIC_CFG_INT_PRIORITY は次節を参照

7.3.2　イベントフラグ

簡易I2Cドライバ内で利用するイベントフラグです。ユーザアプリケーションからのAPIの受け付けキュー、割込みハンドラとタスクの同期処理に利用します。

項目	値
拡張情報	不定
イベントフラグ属性	TA_TPRI | TA_WMUL | TA_DSNAME
DSオブジェクト名称	"siic_f"

備考：

TA_DSNAMEのイベンツフラグ属性とDSオブジェクト名称はカーネルのコンフィグレーションによって未サポートとなる場合があります。

7.3.3　タスク（siic_tsk）

簡易I2Cの制御を行うタスクです。ユーザアプリケーションからのAPIの受付け、簡易I2Cの制御を行います。

項目	値
拡張情報	不定
タスク属性	TA_HLNG | TA_DSNAME | TA_USERBUF
タスク起動アドレス	siic_tsk
タスク起動時優先度	SIIC_CFG_TASK_PRIORITY
スタックサイズ	320バイト
DSオブジェクト名称	"sdc_t"

備考：

TA_DSNAME、TA_USERBUFのタスク属性とDSオブジェクト名称はカーネルのコンフィグレーションによ

って未サポートとなる場合があります。

7.3.4　割込みハンドラ（sci11_rxi11_hdr、sci11_txi11_hdr）

簡易I2Cモードで使用するSCI11の割込みハンドラです。割込みの発生によりイベントフラグ経由でsiic_tskを動作させます。

項目	値
割込み番号	114（受信完了割込み）
	115（送信準備完了割込み）
割込み優先度	SIIC_CFG_INT_PRIORITY
割込みハンドラ起動アドレス	sci11_rxi11_hdr：RXI（受信完了割込み）
	sci11_txi11_hdr：TXI（送信準備完了割込み）

7.4　簡易 I2C ドライバのコンフィグレーション

7.4.1　簡易 I2C ドライバのコンフィグレーション・ファイル

簡易I2Cドライバのコンフィグレーション・ファイルは、mtkernel_3¥device¥siic¥sysdepend¥ターゲット¥siic_config.h です。

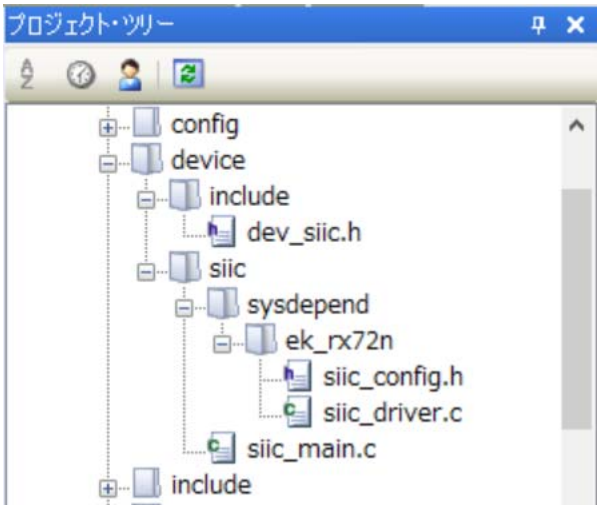

図7.1　簡易I2Cドライバのコンフィグレーション・ファイル（EK-RX72Nの例）

7.4.2　タスクの優先度と割込みハンドラの割込み優先度

● SIIC_CFG_TASK_PRIORITY

簡易I2Cの制御を行うタスク（siic_tsk）のタスク優先度です。

システムの都合に応じて、1 ～ CFN_MAX_TSKPRI（タスク優先度の最大値）の範囲で変更できます。最低でも簡易I2Cドライバを利用するタスクの優先度よりも高く設定することを推奨します。デフォル

トは 4 に設定されています。

● SIIC_CFG_INT_PRIORITY

内蔵周辺機能のSCIの割込み優先度です。

システムの都合に応じて、1 ～ MAX_INT_PRI（最高外部割込み優先度）の範囲で変更できます。TIM_INT_PRI（システムタイマの割込み優先度）よりも低く設定することを推奨します。デフォルトでは 9 に設定されています。

```
/* SIIC control task priority. */
#define SIIC_CFG_TASK_PRIORITY              (4)

/* SIIC interrupt priority level. */
#define SIIC_CFG_INT_PRIORITY               (9)
```

7.5　サンプルプログラム

RX72N ENVISION KIT搭載のISL29034センサを照度センサモードに設定し、照度センサ値をターミナルに500msの間隔で表示するサンプルです。

```
EXPORT void sensor_tsk(INT stacd, void *exinf)
{
ID dd;
UB buf[2];
SZ asize;
INT data;
        dd = tk_opn_dev( "siicl", TD_UPDATE );                  // Open Simple IIC Driver
        tk_swri_dev( dd, 0x88, "¥x0F¥x28", 2, &asize );         // Initialize ISL29034
        tk_swri_dev( dd, 0x88, "¥x00¥xC0", 2, &asize );         // Set Measures IR Mode
        while( 1 )  {
                tk_swri_dev( dd, 0x88, "¥x02",  1, &asize );   // Select Data(LSB)
                tk_srea_dev( dd, 0x88, &buf[0], 1, &asize );   // Read D7-D0
                tk_swri_dev( dd, 0x88, "¥x03",  1, &asize );   // Select Data(MSB)
                tk_srea_dev( dd, 0x88, &buf[1], 1, &asize );   // Read D15-D8
                data = ( buf[1] << 8 ) + buf[0];               // Make Sensor Value
                tm_printf("%d¥n",data);                        // Output Console
                tk_dly_tsk( 500 );                             // Wait 500ms
        }
}
```

7.6　スタックサイズ

7.6.1　スタック見積もりツール

本実装においてもスタック見積もりツールを使用することが可能です。スタック見積もりツールの起動方法やリアルタイムOSオプションの設定等はターゲット毎の構築仕様書の第５章を参照ください。

以降はスタック見積もりツールを使用することを前提に各スタックサイズを紹介します。以下にスタック見積もりツールの表示例を示します。

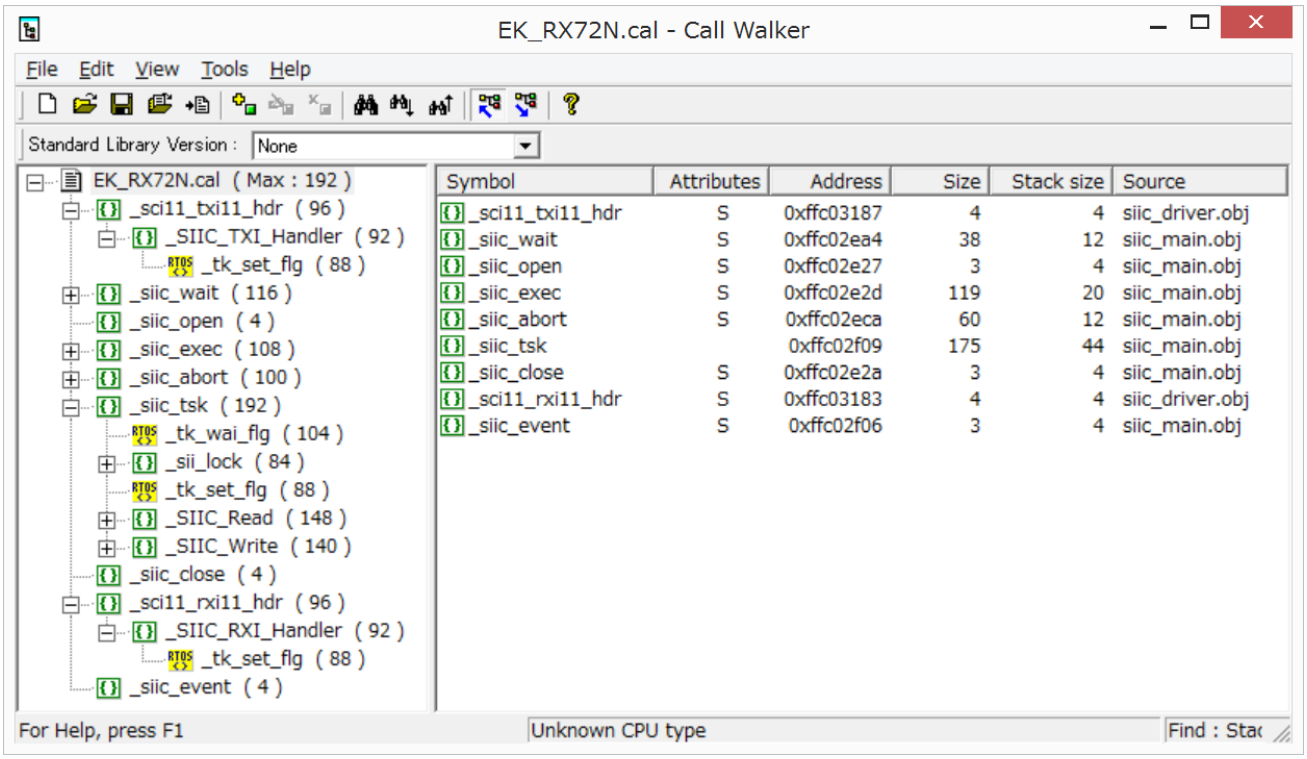

図7.2　簡易I2Cドライバのスタック見積もり結果

7.6.2　siic_tsk のスタックサイズ

siic_tskタスクが独自に使用するスタックサイズは192バイトです。コンパイラのバージョン等が変化しても、それほど大きな違いはないはずです。このサイズにタスクコンテキストのサイズを加えても、現状のサンプルで確保されている300バイトは超えないと考えて構いません。

ただし、カーネル管理外割込みを複数レベル使用すると300バイトを超える可能性があります。もし、カーネル管理外割込みを複数レベル使用する場合はターゲット毎の構築仕様書の第５章の内容に従ってスタックサイズを算出し、mtkernel_3¥device¥siic¥siic_maini.cで生成・確保されているスタックサイズを変更してください。

```
u.t_ctsk.stksz = 300;                    // Set Task StackSize
u.t_ctsk.itskpri = SIIC_CFG_TASK_PRIORITY;    // Set Task Priority
```

7.6.3　割込みハンドラのスタックサイズ

sci11_rxi11_hdrとsci11_txi11_hdrの割込みハンドラが独自に使用するスタックサイズは共に96バイトです。このサイズが簡易I2Cドライバのコンフィグレーションで指定した SIIC_CFG_INT_PRIORITY 割込み優先度で実行される割込みハンドラのスタックサイズとなります。この数値を使って例外スタックのサイズを算出してください。

7.6.4　簡易 I2C ドライバの処理関数

簡易I2Cドライバの処理関数の中で各構築仕様書の5.3.7項に記載された加算条件を超えるものは存在

しません。結果、siicDrvEntry関数を含め、Call Walkerに表示された値はそのまま各スタックサイズの計算式に利用可能です。

8.　問い合わせ先

本実装に関する問い合わせや他のRXファミリへのポーティングに関する相談は以下のメールアドレス宛にお願い致します。

yuji_katori@yahoo.co.jp　　トロンフォーラム学術・教育WGメンバ
鹿取　祐二（かとり　ゆうじ）

なお、上記のメールアドレスは余儀なく変更される場合がありますが、その際はご了承ください。

以上