μT-Kernel3.0
C-Frist（RL78 S3コア）　向け
実装仕様書

Version.01.00.00

2021.8.13

目次

1. 概要

1.1 目的

本書はC-First（RL78 S3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RL78用に改変したソースコードのうち、C-First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	C-First	マルツエレック製
搭載マイコン	RL78/G14 R5F104LE	ルネサスエレクトロニクス製

1.3 ターゲット名

C-First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	_CFirst_RL78_	
対象システム	C-First	
対象CPU	RL78/G14	R5F104LE
対象CPU アーキテクチャ	CPU_CORE_RL78S3	RL78 S3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
実機	絵解き マイコンCプログラミング教科書	CQ出版社
搭載マイコン	RL78/G14　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
    ├ cfirst_rl78                        C-Frist RL78 依存部
    ├ :
    ├ <ターゲットn>                      ターゲットn 依存部
    └ cpu                                CPU 依存部
        ├ <r5f104le>                         R5F104LE 依存部
        ├ :
        └ <CPUn>                             CPUn 依存部
            └ core                           コア依存部
                ├ <rl78s3>                       RL78/S3 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	RL78/S3
ROM	64KB（内蔵フラッシュROM）
RAM	5.5KB（内蔵RAM）

2.2　実行モード（メモリ・モデル）と保護レベル

RL78/S3コアはプログラムの動作モードを持っていない。また、RL78/S3コア特有のメモリ・モデルは、スモール・モデル（変数、関数共にnear領域の64KB以内）が対象である。関数が64KBを超えたミディアム・モデルには対応していない。

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（HL、DE、BC、AX）、SP、PSW、PC、ES、CPを有するが、ESとCPはタスクコンテキストの対象外である（スモール・モデルのみ対象、ミディアム・モデルは対象外）。

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 {
        UH      rp[4];          /* 汎用レジスタ RP0-RP3(HL/DE/BC/AX) */
} T_REGS;
```

(2) 例外時に保存されるレジスタ T_EIT

```
typedef struct t_eit {
        UH      pc;             /* プログラム・カウンタ */
        UB      psw;            /* プログラム・ステータス・ワード */
} T_EIT;
```

(3) 制御レジスタ T_CREGS

```
typedef struct t_cregs {
        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/cfirst_rl78/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

本マイコンにはコプロセッサは存在しない。

よって、プロファイル TK_SUPPORT_COPn はすべて FALSE である。コプロセッサに関するAPI（tk_set_cpr, tk_get_cpr）は提供されない。

3.　メモリ

3.1　メモリモデル

RL78/S3は1M（20bit） のアドレス空間を有するが、本実装では変数、関数が共にnear領域の64KB（16bit）のアドレス空間を対象とする。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、関数呼び出しが可能とする。

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RL78/G14（R5F104LE）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
0x00000 ～ 0x0FFFF	内蔵ROM	512	プログラム領域
0xF0000 ～ 0xF07FF	拡張特殊機能レジスタ	2	
0xF1000 ～ 0xF1FFF	データ・フラッシュ・メモリ	4	
0xF2000 ～ 0xFE8FF	Mirror	50.25	ミラー領域
0xFE900 ～ 0xFFEDF	内蔵RAM	5.5	データ領域
0xFFEE0 ～ 0xFFEFF	汎用レジスタ	32Byte	
0xFFF00 ～ 0xFFFFF	特殊機能レジスタ	256Byte	

3.3　OS のメモリマップ

本実装ではマイコンの内蔵ROM（ミラー領域を含む）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
0x00000 ～ 0x0007F	ベクタテーブル	
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

(2) 内蔵RAMのメモリマップ

アドレス （開始 ～ 終了）	種別	内容
0xFE900 ～ 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 ～ 0xFFE1F	例外スタック領域	割込みハンドラなどのスタック領域
0xFFE20 ～ 0xFFEDF	SADDR領域	OS管理外のRAM領域（※）

SADDR領域はユーザシステムで自由に利用することが可能

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) テンポラリスタック

現在のコンテキストを破棄して、別のタスクコンテキストに移行するまでの間に使用するスタックであり、以下に示す要因の際に使用する。

(a) 実行中のタスクを終了し、次に実行すべきタスクにディスパッチするまでの間

(b) 実行すべきタスクが存在しない状態でディスパッチャが起動された場合

(c) 実行すべきタスクが存在しない状態でハンドラが起動された場合

テンポラリスタックのサイズは/config/config.h(inc)の CFN_TMP_STACK_SIZE で指定する。

(3) 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_C-First構築手順書を参照のこと。

3.5 OS 内の動的メモリ管理

OS内の動的メモリ管理に使用するOS管理メモリ領域は、以下のように定められる。なお、本実装ではコンフィギュレーション CNF_SYSTEMAREA_TOP と CNF_SYSTEMAREA_END はサポートしていない。

(1) OS管理メモリ領域の開始アドレス

内蔵RAM上のプログラムのデータ領域（BSS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

内蔵RAM上の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・テーブル・アドレス）	割込みの種別	備考
0x0000	リセット	OSで使用
0x0002 ～ 0x0036	内部・外部割込み	OSの割込み管理機能で管理
0x0038	インターバル信号検出	OSで使用
0x003A ～ 0x0062	内部・外部割込み	OSの割込み管理機能で管理
0x007E	BRK命令の実行	OSで使用

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/rl78s3/cstart.asm、リセット以外のベクタテーブルは/kernel/sysdepend/cpu/r5f104le/vector.asm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

RL78のS3コアは、割込み優先度を2bit(0～3)の４段階に設定できる（優先度の数字の小さい方が優先度は高い）。ただし、優先度0は割込み優先度によるマスクが不可能であるため、カーネル管理外の割込みとなる。結果、カーネル管理内の割込みとして扱える優先度は1～3の範囲（変更可能）である。

4.3.2　多重割込み対応

本マイコンの優先順位指定フラグ・レジスタ（PR00L、PR00H、PR01L、PR01H、PR02L、PR02H、PR10L、PR10H、PR11L、PR11H、PR12L、PR12H）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/r5f104le/sysdef.h(inc)にて以下のように定義される。

```
#define MAX_INT_PRI             (1)             // sysdef.h
MAX_INT_PRI             .EQU    (1)             ;  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インターバル・タイマを使用するが、インターバル・タイマの割込みは 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/r5f104le/sysdef.h(inc)にて以下のように定義される。

```
#define TIM_INT_PRI             (1)             // sysdef.h
TIM_INT_PRI             .EQU    (1)             ;  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

以下、本実装で設定可能な割込み優先度の一覧を示す。

OS管理外の割込み優先度	0	0	0	0,1	0,1	0,1,2
MAX_INT_PRI（ｸﾘﾃｨｶﾙｾｸｼｮﾝ）	1	1	1	2	2	3
TIM_INT_PRI（ｼｽﾃﾑﾀｲﾏ）	1	2	3	2	3	3
利用可能なOS管理内の割込み優先度	1,2,3	1,2,3	1,2,3	2,3	2,3	3

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・テーブル・アドレス）	割込みの種別	備考
0x0000	リセット	
0x0038	インターバル信号検出	システムタイマとして使用
0x007E	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、割込みハンドラの管理を行う。

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マイコンのベクタ・テーブル・アドレスと同一とする。

例えば、INTP0は割込み番号0x0008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以下の処理が実行される。

(1) システム変数knl_taskindpのインクリメント

システム変数knl_taskindpが 0 以上の値のとき、タスク独立部であることを示す。

(2) 割込みハンドラの実行

割込み番号の値を参照し、テーブルknl_inthdr_tblに登録されている割込みハンドラを実行する。

(3) システム変数knl_taskindpのデクリメント

割込みハンドラから戻ったら(1)でインクリメントした値を戻す。

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/rl78s3/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（プログラム・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ISP1とISP0に最高外部割込み優先度MAX_INT_PRI-1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ISP1とISP0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ISP1とISP0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 3 から 0 の値（値の小さい方が高い優先度）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 3 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ISP1とISP0の設定値を参照する。

4.6.2 割込みコントローラ制御

マイコン内蔵の割込み優先順位フラグ・レジスタ（PRmnH/L）、割り込みマスク・フラグ・レジスタ（MK*H/L）、割り込み要求フラグ・レジスタ（IFmH/L）の制御を行う。

① 割込みコントローラの割込み許可（EnableInt）

割り込みマスク・フラグ・レジスタ（MKmH/L）を設定し、指定された割込みを許可する。同時に割込み優先順位フラグ・レジスタ（PRmnH/L）に指定された割込み優先度を設定する。割込み優先度は 3 から 0 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）

割り込みマスク・フラグ・レジスタ（MKmH/L）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）

割り込み要求フラグ・レジスタ（IFmH/L）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、割り込み要求フラグ・レジスタ（IF*H/L）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。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 1 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 2 や 3 に変更可能である。よって、優先度 0 以外にも優先度 1 や 2 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・テーブル・アドレス 0x0004（割込み名称：INTWDTI）のウォッチドッグ・タイマのインターバル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/r5f104le/vector.asm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】

```
knl_inthdr_entry2       .VECTOR 0x04            ;  INTWDTI
knl_inthdr_entry3       .VECTOR 0x06            ;  INTLVI
```

【改変後のソースコード】

```
;knl_inthdr_entry2      .VECTOR 0x04            ;  INTWDTI
knl_inthdr_entry3       .VECTOR 0x06            ;  INTLVI
```

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-RL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=アドレス）とレジスタ・バンク指定（bank={rb0|rb1|rb2|rb3}）と多重割込み許可指定（enable={true|false}）を行う。

(1) ベクタテーブル指定（vect=アドレス）

アドレスに当該割込みのベクタ・テーブル・アドレスを指定することでベクタテーブルが生成できる。アドレスは定数、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割込み名称を使うことが可能である。

(2) レジスタ・バンク指定（bank={RB0|RB1|RB2|RB3}）

レジスタ・バンクの切り替えを指定できる。同一優先度の割込みであれば、同じレジスタ・バンクを指定することが可能である。レジスタ・バンクの切り替えを指定すると、割込み発生時の汎用レジスタの退避・復旧が不要となるため、割込み処理の高速化、コード効率の向上が行える。ただし、OSがバンク0（RB0）のレジスタ・バンクを使用しているため、OS管理外割込みで使用できるレジスタ・バンクはRB1～RB3となる（割込み優先度とレジスタ・バンクの番号は無関係である）。

なお、異なる優先度の割込みで同じレジスタ・バンクを使用したり、バンク0（RB0）のレジスタ・バンクを使用した場合の動作は保証しない。

(3) 多重割込み許可指定（enable={true|false}）

マスク不可能な割込み優先度が 0 以外のOS管理外割込みでは、必要に応じて多重割込みの許可が可能である。trueを指定すれば多重割込み許可、falseまたは未指定ならば多重割込み禁止となる。

なお、割込み優先度が 0 のOS管理外割込みで多重割込みを許可した場合の動作は保証しない（割込み優先度によるマスクが不可能であるため）。

【OS管理外割込み関数の例】

```
#pragma interrupt intwdti(vect=INTWDTI,bank=RB3,enable=true)
void intwdti(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RL78/S3コアに固有の処理であるため/kernel/sysdepend/cpu/core/rl78s3/cstart.asmのスタートアップ・ルーチン経由で/kernel/sysdepend/cpu/core/rl78s3/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cstart.asm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/ap_rx63n/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/ap_rx63n/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。

	本実装では処理は未実装である(※)。
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_COPn	×	本マイコンはFPUを持たない。
TA_FPU	×	

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	0x86	割込み許可
AX	第一引数	stacd タスク起動コード
DE	第二引数	*exinf タスク拡張情報
SP	タスクスタックの先頭アドレス	

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273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UART Channel 0

ボーレート	115,200bps
データ形式	data 8bit, stop 1bit, no parity

7. 問い合わせ先

本実装に関する問い合わせ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