μT-Kernel 3.0
デバイスドライバ説明書

Version 01.00.00.B0

2020.07.13

目次

概要 4
1.1 目的と位置づけ 4
1.2 特徴 4
1.3 対象とするハードウェア 4
1.4 制限事項 5
1.5 ソースコードの構成 5
2. 使用方法 7
2.1 OS への組み込み方法 7
2.2 コンフィギュレーション 7
2.3 ユーザプログラムからの使用 8
3. デバイスドライバ 9
3.1 共通仕様 9
3.1.1 デバイスの初期化 9
3.1.2 デバイスのオープンモード 9
3.1.3 多重オープンと排他制御 9
3.1.4 同期アクセスと割込み制御 9
3.2 シリアル通信ドライバ 10
3.2.1 概要 10
3.2.2 基本仕様 10
3.2.3 使用方法 10
3.2.4 属性データ 14
3.2.5 コンフィギュレーション 16
3.2.6 ハードウェア依存仕様 17
3.2.7 その他 18
3.3 A/D 変換デバイスドライバ 19
3.3.1 概要 19
3.3.2 基本仕様 19
3.3.3 使用方法 19
3.3.4 属性データ 22
3.3.5 コンフィギュレーション 22
3.3.6 ハードウェア依存仕様 23
3.3.7 その他 24
3.4 I2C 通信ドライバ 25
3.4.1 概要 25

3.4.2 基本仕様.....25
3.4.3 使用方法.....25
3.4.4 属性データ.....30
3.4.5 コンフィギュレーション.....30
3.4.6 ハードウェア依存仕様.....31
3.4.7 その他.....31

概要

1.1 目的と位置づけ

μT-Kernel 3.0 デバイスドライバ(以下、本ソフトと称する)は、リアルタイム OS μT-Kernel 3.0 向けにトロンフォーラムにて開発されたデバイスドライバであり、以下を目的としてソースコードをサンプルとして、T-License に基づき公開する。

- μT-Kernel 3.0 のデバイス管理機能から使用可能な基本的なデバイスドライバの仕様を規定する
- μT-Kernel 3.0 のデバイスドライバを開発するためのサンプルコードを提供する。

μT-Kernel 3.0 仕様には個々のデバイスドライバの仕様は含まれない。これは、デバイスドライバの仕様がハードウェアに大きく依存し、また用途に応じて仕様も変わるため、OS 自体の仕様から分離する意図である。
よって、本ソフトに含まれる個々のデバイスドライバの仕様は μT-Kernel 3.0 の仕様に含まれない。

1.2 特徴

前述の目的より、本ソフトは以下の特徴を持つ。

- μT-Kernel 3.0 のデバイス管理機能の API から使用可能である。
- 特定のハードウェアへの依存性を抑え、OS の API からはハードウェアの差異を大きく意識することなく使用できる。
- ソースコード上は、ハードウェアに依存する部分はハードウェア依存部として独立し、異なったハードウェアへの対応が用意である。

1.3 対象とするハードウェア

本ソフトは以下に示すマイコンの内蔵デバイスに対応する。また、動作確認を行った実機(ボード)を記す。

マイコン	実機(ボード)	対応デバイス
TX03 シリーズ M360 グループ TMPM367FDFG	TX03 M367 IoT-Engine	シリアル (UART) A/D 変換 (ADC) I2C 通信 (SBI)
RX200シリーズ RX231 グループ R5F52318ADFL (ルネサス エレクトロニクス製)	RX231 IoT-Engine	シリアル (SCI) A/D 変換 (S12ADE)

1.4 制限事項

本ソフトのデバイスドライバは、対象とするデバイス(ハードウェア)のすべての機能、性能に対応するものではない。同種のデバイスに共通する基本的な機能を、同じ API によりハードウェアの差異を大きく意識することなく操作することを重視している。
また、本ソフトはサンプルプログラムの扱いであり、製品レベルの品質を保証するものではない。

1.5 ソースコードの構成

本ソフトのソースコードのディレクトリ構成を以下に示す。

```
device
 ├ include              定義ファイル
 ├ common               デバイス共通ファイル
 ├ config               コンフィギュレーション
 ├ ser                  シリアル通信デバイス
 │   └ sysdepend              ハードウェア依存部
 ├ adc                  A/D 変換ドライバ
 │   └ sysdepend              ハードウェア依存部
 └ i2c                  I2C 通信デバイス
     └ sysdepend              ハードウェア依存部
```

各ディレクトリの説明を記す。

(1) 定義ファイル include
各デバイスドライバを使用するための定義ファイルを格納する。デバイスドライバを使用する際には、該当するデバイスの定義ファイルをインクルードする。詳細は「2.3 ユーザプログラムからの使用」を参照。

(2) デバイス共通ファイル common
デバイスドライバが共通で使用するプログラムコードを格納する。本ディレクトリの内容は、デバイスドライバのユーザが直接使用することはない。

(3) コンフィギュレーション config
デバイスドライバ全般のコンフィギュレーション設定を記述したコンフィギュレーション定義フィル(devconf.h)を格納する。詳細は「2.2 コンフィギュレーション」を参照。

(4) シリアル通信デバイス
シリアル通信デバイスのソースコードを格納する。

(5) A/D 変換ドライバ
シリアル通信デバイスのソースコードを格納する。

(6) I2C 通信デバイス
I2C 通信デバイスのソースコードを格納する。

各デバイスの機種依存部のコードは、sysdepend ディレクトリ下に格納される。

2. 使用方法

2.1 OS への組み込み方法

本ソフトは、μT-Kernel 3.0 の OS 本体とビルド(コンパイル、リンク)する。本ソフトのソースコードのディレクトリ(¥device)を、μT-Kernel 3.0 のソースコードのディレクトリ(¥mtkernel_3)内に配置し、一括でビルドすればよい。
本ソフト中の必要なドライバ、ライブラリのみを選択してビルドするには、次項のコンフィグレーションにて指定を行う。

2.2 コンフィギュレーション

本ソフトの各種コンフィギュレーションを行うには、コンフィギュレーション・ファイル(¥device¥config¥devconf.h)を変更する。
devconf.h は C 言語のソースコードであり、各設定項目の値を#define にて定義している。

```
/* ------------------------------------------------------------------------ */
/* Device usage settings
 *          1: Use      0: Do not use
 */
#define DEVCNF_DEV_SER              1           // Serial communication device
#define DEVCNF_DEV_ADC              1           // A/D conversion device
#define DEVCNF_DEV_IIC              1           // I2C communication device
```

以下に設定項目も説明を記す。

名称	説明
DEVCNF_DEV_SER	シリアル通信デバイスの使用 1: 使用する 0: 使用しない
DEVCNF_DEV_ADC	A/D 変換デバイスの使用 1: 使用する 0: 使用しない
DEVCNF_DEV_IIC	I2C 通信デバイスの使用 1: 使用する 0: 使用しない
DEVCNF_LIB_IOT	IoT-Engine 用デバイス制御ライブラリの使用 1: 使用する 0: 使用しない

2.3 ユーザプログラムからの使用

ユーザプログラムから本ソフトを使用するには、使用するデバイスドライバの API 定義ファイルをインクルードする。

API 定義ファイルは本ソフトの¥device¥include ディレクトリに置かれている。デバイスドライバ毎に以下がある。

名称	説明
dev_ser.h	シリアル通信デバイスの API 定義ファイル
dev_adc.h	A/D 変換デバイスの API 定義ファイル
dev_iic.h	I2C 通信デバイスの API 定義ファイル

3. デバイスドライバ

3.1 共通仕様

3.1.1 デバイスの初期化

各デバイスドライバは API として、初期化関数を提供する。
初期化関数によりデバイスドライバは初期化され、以降は、OS のデバイス管理 API を通じて操作することができる。初期化関数は、ハードウェアの初期化、OS へのデバイスの登録、割込みハンドラの登録、OS 資源の確保などを行う。

3.1.2 デバイスのオープンモード

デバイスのオープン時に指定するオープンモードにより、読込み専用 TD_READ、書込み専用 TD_WRITE、読み書き可能 TD_UPDATE のいずれかが選択できる。
オープンモードが影響するのは、デバイスの固有データのみである。読込み専用の場合は、固有データの書込みが不可となり、書込み専用の場合は固有データの読込みが不可となる。不可のアクセスを行った場合は API が E_OACV エラーとなる。
属性データのアクセスは、オープンモードに関係なく、個別に決められる。

3.1.3 多重オープンと排他制御

各デバイスは多重オープンが可能である。多重オープンした場合、同一デバイスへのアクセスは排他制御される。よって、複数のタスクが同一デバイスをオープンし、同時に使用することは可能である。なお、同一デバイスへのアクセスが衝突した場合、排他制御によりいずれかのタスクが待ち状態となることがある。

3.1.4 同期アクセスと割込み制御

各デバイスは同期アクセスのみに対応し、非同期アクセスは行わない。
よってデバイスの読み書きは、同期 API **tk_srea_dev** および **tk_swri_dev** を使用する。非同期 API **tk_rea_dev** および **tk_wri_dev** を使用することも可能であるが、内部の処理は同じとなるため、同期 API の使用を推奨する。
同期アクセスは API の処理内において読み書きの実行が完了する。つまり、API から呼び出し元に戻った時点で処理は完了している。
ハードウェアへのアクセスは、ビジーウェイトを避けるため、原則として割込みを使用する。よって、API の処理内において割込みの完了待ちが生じる場合がある。この時、API を呼び出したタスクは待ち状態となる。

3.2 シリアル通信ドライバ

3.2.1 概要

シリアル通信ドライバは、マイコン内蔵の調歩同期(非同期)シリアル通信デバイスの制御を行うデバイスドライバである。
調歩同期(非同期)シリアル通信を持ちいて、データの送信および受信が可能である。

3.2.2 基本仕様

シリアル通信ドライバの名称は"ser"とし、非同期シリアル通信デバイスのデバイス(ハードウェア)毎にユニットをアサインする。
シリアル通信ドライバとハードウェアの対応を「表 3-1 シリアル通信ドライバとハードウェアの対応」に記す。

表 3-1 シリアル通信ドライバとハードウェアの対応

マイコン	デバイス(ハードウェア)	ユニット番号	デバイス名
TX03-M367	UART4 (UART Channel4)	0	“sera”
	UART5 (UART Channel4)	1	“serb”
RX231	SCI0	0	“sera”
	SCI1	1	“serb”
	SCI5	2	“serc”
	SCI6	3	“serd”
	SCI8	4	“sere”
	SCI9	5	“serf”
	SCI12	6	“serg”

3.2.3 使用方法

(1) デバイスドライバの初期化

デバイスドライバを使用するにあたって、初期化関数を呼び出す。初期化関数の呼び出しは、各デバイス(ハードウェア)について一回きりとする。
初期化関数により、デバイスドライバに必要な初期化処理が実行され、OS にデバイスが登録される。以降はOSのデバイス管理機能のAPIを用いて、デバイスの制御を行うことができる。

以下のデバイス初期化関数の仕様を記す。

書式	ER　dev_init_ser (UW unit) ;
引数	UW unit　　ユニット番号(0 から始まる)
戻り値	エラーコード
説明	引数 unit で指定したデバイスの初期化、登録を行う。 ユニット番号は「表 3-1 シリアル通信ドライバとハードウェアの対応」を参照。

(2) デバイスのオープン

デバイスの使用開始にあたり、μT-Kernel3.0 の API **tk_opn_dev** により対象デバイスをオープンする。以降、対象デバイスへのアクセスが可能となる。

以下に API の概要を記す。詳細は「μT-Kernel3.0 仕様書」を参照のこと。

書式	ER　tk_opn_dev (CONST UB *devnm, UINT omode) ;
引数	CONST UB *devnm　　デバイス名
	UINT omode　　オープンモード
戻り値	エラーコード
説明	devnm で指定したデバイスを omode のモードでオープンする。 デバイス名 devnm は「表 3-1 シリアル通信ドライバとハードウェアの対応」を参照。

同一デバイスの多重オープンは許されるが、実際にオープン処理が実行されるのは最初のオープンのみである。

オープンモードは任意に指定できる。読込み専用とした場合はデバイスの固有データの書込み(データ送信)はできない。また、書込み専用とした場合はデバイスの固有データの読込み(データ受信)はできない。

デバイスの属性データは、オープンモードに関係なく、読み書き可能である。

オープン時の通信デバイスの設定(通信フォーマット、速度など)は、コンフィギュレーションで指定された初期値に設定される。

また、通信デバイスの設定は、デバイスの属性データを書き込むことにより初期値から変更

が可能である。詳細は「3.2.6 ハードウェア依存仕様」を参照。

(3) シリアル通信データの受信（固有データの読込み）

シリアル通信デバイスは同期アクセスのみに対応する。よって、データの読込みには API **tk_srea_dev** を使用する。

API(tk_srea_dev)を用いて対象デバイスから固有データを読み込むことにより、データ受信を行うことができる。

以下に API の概要を記す。詳細は「μT-Kernel3.0 仕様書」を参照のこと。

書式	ER ercd = tk_srea_dev_d(ID dd, D start_d, void *buf, SZ size, SZ *asize);	
引数	ID dd	対象デバイスのデバイスディスクリプタ （tk_opn_deb の戻り値）
	W start	0 以上：固有データ（値は任意） 0 未満：属性データ
	void *buf	受信データを格納する領域の先頭アドレス
	SZ size	要求する受信データ数（バイト単位）
	SZ &asize	実際に受信したデータ数を返す領域のアドレス
戻り値	ER	エラーコード
説明	dd で指定したシリアル通信デバイスから、size バイトのデータを受信する。実際に受信されたデータは*buf に格納され、受信データ数は*asize に返される。	

引数 start で指定する固有データ番号は 0 以上の任意の数であり、値による動作の差異はない。

API（tk_srea_dev）の処理は、実際にはデバイスドライバ内の受信バッファからのデータの読込みである。API 発行時に既に受信バッファに要求数のデータがある場合は、API は速やかに終了する。

受信バッファのデータが足りない場合は、API を呼び出したタスクは待ち状態となる。データ受信の待ちのタイムアウト時間は、デバイスの属性データで指定できる。また、初期値はコンフィギュレーションにて指定される。

引数 asize で指定した領域に実際に受信したデータのサイズが返される。API が正常終了した場合、size と asize の値は等しい。

要求する受信データ数(size)に 0 を指定した場合は、読込み可能のデータ数が asize に返される。この値はその時点での受信バッファ内のデータ数である。

(4) シリアル通信のデータ送信(固有データの書込み)

シリアル通信デバイスは同期アクセスのみに対応する。よって、データの書込みには API **tk_swri_dev** を使用する。

API(tk_swri_dev)を用いて対象デバイスへ固有データを書き込むことにより、データ送信を行うことができる。

以下に API の概要を記す。詳細は「μT-Kernel3.0 仕様書」を参照のこと。

書式	ER ercd = tk_swri_dev(ID dd, W start, CONST void *buf, SZ size, SZ *asize);	
引数	ID dd	対象デバイスのデバイスディスクリプタ (tk_opn_deb の戻り値)
	W start	0 以上:固有データ(値は任意) 0 未満:属性データ
	void *buf	送信データを格納する領域の先頭アドレス
	SZ size	要求する送信データ数(バイト単位)
	SZ &asize	実際に送信したデータ数を返す領域のアドレス
戻り値	ER	エラーコード
説明	dd で指定したシリアル通信デバイスから、*buf に格納されたデータを size バイト送信する。実際に送信されたデータ数は*asize に返される。	

引数 start で指定する固有データ番号は 0 以上の任意の数であり、値による動作の差異はない。

API(tk_swri_dev)の処理は、実際にはデバイスドライバ内の送信バッファへのデータの書込みである。API 発行時に送信バッファに空きがある場合は、API は速やかに終了する。送信バッファ内のデータは実際に送信されるのは API が終了した後であることに注意が必要である。

送信バッファに空きがない場合は、API を呼び出したタスクは待ち状態となる。データ送信の待ちのタイムアウト時間は、デバイスの属性データで指定できる。また、初期値はコンフィギュレーションにて指定される。

引数 asize で指定した領域に実際に送信したデータのサイズが返される。API が正常終了した場合、size と asize の値は等しい。

要求する送信データ数(size)に 0 を指定した場合は、書込み可能のデータ数が asize に返される。この値はその時点での送信バッファの空きデータ数である。

(5) デバイスのクローズ

デバイスの使用終了にあたり、μT-Kernel3.0 の API **tk_cls_dev** により対象デバイスをクローズ

する。以降、対象デバイスへのアクセスが不可となる。

以下に API の概要を記す。詳細は「μT-Kernel3.0 仕様書」を参照のこと。

書式	ER ercd = tk_cls_dev(ID dd, UINT option);
引数	ID dd　　対象デバイスのデバイスディスクリプタ （tk_opn_deb の戻り値）
	UINT option　　クローズオプション
戻り値	エラーコード
説明	dd で指定したデバイスをクローズする。 本デバイスドライバは option を無視する。

多重オープンされている場合、オープンに対応したクローズが必要である。すべてクローズされるとデバイスドライバは処理を終える。
クローズされたデバイスは、再びオープンされることにより使用可能となる。

3.2.4　属性データ

デバイスの属性データは、API **tk_srea_dev** および **tk_seri_dev** でデータ番号を指定することにより読み書き可能である。

本デバイスの属性データを以下に示す。

種別	名称	データ番号	型	意味
共通属性	TDN_EVENT	-1	ID	事象通知用メッセージバッファ ID ※1
デバイス種別属性	TDN_SER_MODE	-100	UW	シリアル通信モード ※2
	TDN_SER_SPEED	-101	UW	シリアル通信速度 ※2
	TDN_SER_SNDTMO	-102	UW	送信タイムアウト時間
	TDN_SER_RCVTMO	-103	UW	受信タイムアウト時間
	TDN_SER_COMERR	-104	TMO	通信エラー
	TDN_SER_BREAK	-105	TMO	ブレーク信号送出

※1 本デバイスはデバイス事象通知をサポートしない。この属性データは将来の拡張のためにある。
※2 シリアル通信モードおよび速度は、属性データを変更した時点では反映されず、デバイスをオープンした際に適用される。つまり、属性データを変更した場合はデバイスをクローズしたのち、再度オープンしなければならない。

各デバイス種別属性データについて以下に説明する。

(1) シリアル通信モード

シリアル通信の通信モードを以下のように指定する。

mode := (DEV_SER_MODE_7BIT || DEV_SER_MODE_8BIT)
| (DEV_SER_MODE_1STOP || DEV_SER_MODE_2STOP)
| (DEV_SER_MODE_PODD || DEV_SER_MODE_PEVEN || DEV_SER_MODE_PNON)
| [DEV_SER_MODE_CTSEN] | [DEV_SER_MODE_RTSEN]

DEV_SER_MODE_7BIT	データ長 7bit
DEV_SER_MODE_8BIT	データ長 8bit
DEV_SER_MODE_1STOP	1 ストップビット
DEV_SER_MODE_2STOP	2 ストップビット
DEV_SER_MODE_PODD	奇数パリティ
DEV_SER_MODE_PEVEN	偶数パリティ
DEV_SER_MODE_PNON	パリティビットなし
DEV_SER_MODE_CTSEN	CTS ハードウェアフロー制御
DEV_SER_MODE_RTSEN	RTS ハードウェアフロー制御

なお、それぞれのモードの実際の値は、ハードウェア(マイコン)毎に異なる。よって、モードの指定は定義名で行わなくてはならない。
また、ハードウェアによって指定できる値の制限がある場合がある。

(2) シリアル通信速度

シリアル通信の速度(ボーレート)を指定する。指定可能な範囲はハードウェアにより異なる。

(3) 送信タイムアウト時間

データ送信処理において、送信バッファの空き待ち時間の上限を指定する。タイムアウトが発生すると、API はタイムアウトエラーで終了する。
このタイムアウト時間は、API 処理内の待ちの合計時間ではない。API 処理中に生じる個々のタスクの待ち状態のタイムアウト時間である。

(4) 受信タイムアウト時間

データ受信処理において、データ受信の待ち時間の上限を指定する。タイムアウトが発生すると、API はタイムアウトエラーで終了する。
このタイムアウト時間は、API 処理内の待ちの合計時間ではない。API 処理中に生じる個々のタスクの待ち状態のタイムアウト時間である。

(5) 通信エラー（読込みのみ可能）

通信中に発生したエラーが記録される。データの内容はハードウェア(マイコン)毎に規定される。本属性データは読込みのみ可能である。読み込むとデータは 0 にクリアされる。

(6) ブレーク信号送出（書込みのみ可能）

本属性データに 1 を書き込むとブレーク信号の送出を開始する。0 を書き込むとブレーク信号の送出を停止する。

ただし、ブレーク信号が送出可能か否かはハードウェア(マイコン)毎に異なる。

本属性データは書込みのみであり、読込みは出来ない。

3.2.5 コンフィギュレーション

シリアル通信デバイスの各初期設定をコンフィギュレーションにより指定できる。

コンフィギュレーションはハードウェアに依存しない共通コンフィグレーションと、依存するデバイス固有コンフィギュレーションがある。

デバイス固有コンフィギュレーションは「3.2.6 ハードウェア依存仕様」に記す。

共通コンフィギュレーションは、コンフィギュレーション定義ファイル ser_cnf.h に記述される。内容を以下に記す。

名称	初期値	意味
DEVCNF_SER_DEVNAME	“ser”	デバイス名
DEVCONF_SER_BUFFSIZE	50	通信バッファサイズ(バイト単位)
DEVCNF_SER_SPEED	115200	通信速度(ボーレート)※1
DEVCNF_SER_MODE	DEV_SER_MODE_CTSEN | DEV_SER_MODE_RTSEN | DEV_SER_MODE_8BIT | DEV_SER_MODE_1STOP | DEV_SER_MODE_PNON	通信モード※1
DEVCNF_SER_SND_TMO	TMO_FEVR(無限待ち)	送信タイムアウト時間※1
DEVCNF_SER_RCV_TMO	TMO_FEVR(無限待ち)	受信タイムアウト時間※1

※1 これらの値はデバイスの属性データに初期値として設定される。

3.2.6 ハードウェア依存仕様

(1) TX03-M367 固有仕様

- マイコン内蔵の非同期シリアル通信チャネル（UART）を使用する。シリアルチャネル(SIO/UART)には非対応である。
- コンフィギュレーション定義ファイル ser_cnf_m367.h にて以下のハードウェアに依存するデバイス固有コンフィギュレーションを定義する。

名称	初期値	意味
DEVCNF_SER_INTPRI	5	シリアル通信の割込み優先順位

- UART が使用するマイコンの端子には複数の機能が割り当てられているので、UART 用に初期化が必要である。本デバイスドライでは端子の初期化は行わないので、マイコンの初期化時などに、本デバイスを使用する前に使用する端子の初期化を行う必要がある。

  ただし、μT-Kernel 3.0 の IoT-Engine 向けの標準の実装では、シリアル通信はデバッグ用シリアル通信に使用するため、UART5 の端子 RDX5、TXD5、RTS5.、CTS5 は初期化されている。他の UART を使用する場合は、マイコンの初期化処理などで、使用する端子の設定が必要である。

- UART 内蔵の FIFO の初期設定は ser_m367.h に以下のように記述 される。この値はデバイス初期化時に UARTxIFLS レジスタに設定される。

```
#define UARTxIFLS_RXINI    UARTxIFLS_1_2     // Receive FIFO 1/2
#define UARTxIFLS_TXINI    UARTxIFLS_1_4     // Send FIFO 1/4
```

(2) RX231 固有仕様

- マイコン内蔵のシリアルコミュニケーションインタフェース(SCI)を使用する。SCI の調歩同期式シリアル通信のみに対応する。他の通信方式には非対応である。
- コンフィギュレーション定義ファイル ser_cnf_rx231.h にて以下のハードウェアに依存するデバイス固有コンフィギュレーションを定義する。

名称	初期値	意味
DEVCONF_SER_INIT_MSTP	FALSE ※2	初期化時にモジュールストップの解除を行う ※1
DEVCNF_SER_INTPRI	5	割込み優先順位

※1 マイコンの初期化処理で初期化済みの場合は FALSE を指定することにより、処理およびコードを節約できる。
※2 μT-Kernel 3.0 の標準の実装では、シリアル通信はデバッグ用シリアル通信に使用するため、SCI6 が初期化済みである。他の SCI を使用する場合は、TRUE とするか、マイコンの初期化処理などで、本デバイスを使用する前にモジュールストップを解除する必要がある。

- SCI が使用するマイコンの端子には複数の機能が割り当てられている。また、SCI の信号は複数の端子にアサイン可能である。本デバイスドラでは端子の初期化は行わないので、マイコンの初期化時などに、本デバイスを使用する前に使用する端子の初期化を行う必要がある。

  ただし、μT-Kernel 3.0 の IoT-Engine 向けの標準の実装では、シリアル通信はデバッグ用シリアル通信に使用するため、SCI6 の端子 RDX6 および TXD6 は初期化され、PB0 および PB1 にアサインされている。他の SCI を使用する場合は、マイコンの初期化処理などで、使用する端子の設定が必要である。

- ブレーク信号の送出には対応しない（SCI にはブレーク信号送出の機能はない）。

- SCI のハードウェアフロー制御は、CTS、RTS のいずれか一方のみを有効にできる（同時に両方を有効にはできない）。よって、属性データによるハードウェアフロー制御には対応しない。
  ハードウェアフロー制御の設定は SPMR レジスタに行う。SPMR レジスタの初期設定を ser_rx231.h に以下のように記述する。デバイス初期化時に SPRM レジスタに設定される。

```
#define    SCI_SPMR_INI      (0x00)              // SPMR initial value
```

3.2.7 その他

シリアル通信デバイスは、以下の機能に対応しない。

- ドライバ要求イベント
- デバイス事象通知

3.3 A/D 変換デバイスドライバ

3.3.1 概要

A/D 変換ドライバは、マイコン内蔵の A/D コンバータの制御を行うデバイスドライバである。
A/D コンバータを介して外部からのアナログ入力の値を取得することができる。

3.3.2 基本仕様

A/D 変換ドライバの名称は"adc"とし、A/D コンバータのデバイス（ハードウェア）毎にユニットをアサインする。
A/D 変換ドライバは、各ユニットの特定のチャンネルからの A/D 変換値の取得、および複数のチェンネルからの連続した A/D 変換値の取得を行うことができる。
A/D 変換ドライバとハードウェアの対応を「表 3-2 A/D 変換ドライバとハードウェアの対応」に記す。

表 3-2 A/D 変換ドライバとハードウェアの対応

マイコン	デバイス(ハードウェア)	ユニット番号	チャンネル数	デバイス名
TX03-M367	ADC ユニット A	0	4	“adca”
	ADC ユニット B	1	4	“adcb”
RX231	S1ADE	0	24	“adca”

3.3.3 使用方法

(1) デバイスの初期化

デバイスドライバを使用するにあたって、初期化関数を呼び出す。初期化関数の呼び出しは、各デバイス（ハードウェア）について一回きりとする。
初期化関数により、デバイスドライバに必要な初期化処理が実行され、OS にデバイスが登録される。以降は OS のデバイス管理機能の API を用いて、デバイスの制御を行うことができる。

以下のデバイス初期化関数の仕様を記す。

書式	ER dev_init_adc (UW unit) ;
引数	UW unit ユニット番号（0 から始まる）
戻り値	エラーコード
説明	引数 unit で指定したデバイスの初期化、登録を行う。 ユニット番号は「表 3-2 A/D 変換ドライバとハードウェアの対応」を参照。

(2) デバイスのオープン

デバイスの使用開始にあたり、μT-Kernel3.0 の API **tk_opn_dev** により対象デバイスをオープ

ンする。以降、対象デバイスへのアクセスが可能となる。

以下に API の概要を記す。詳細は「μT-Kernel3.0 仕様書」を参照のこと。

書式	ER　tk_opn_dev (CONST UB *devnm, UINT omode) ;	
引数	CONST UB *devnm	デバイス名
	UINT omode	オープンモード
戻り値	エラーコード	
説明	devnm で指定したデバイスを omode のモードでオープンする。 デバイス名 devnm は「表 3-2 A/D 変換ドライバとハードウェアの対応」を参照。	

同一デバイスの多重オープンは許されるが、実際にオープン処理が実行されるのは最初のオープンのみである。
オープンモードは任意に指定できる。ただし、A/D 変換ドライバの固有データは読込みのみ可能である。よって、TD_READ または TD_UPDATE の属性を指定しなければならない。
デバイスの属性データは、オープンモードに関係なく、読み書き可能である。

オープン時の A/D 変換デバイスの設定は、コンフィギュレーションで指定された初期値に設定される。

(3) A/D 変換値の取得(固有データの読込み)

A/D 変換デバイスは同期アクセスのみに対応する。よって、データの読込みには API **tk_srea_dev** を使用する。
API(tk_srea_dev)を用いて対象デバイスデバイスの固有データを読み込むことにより、A/D コンバータから変換値を取得することができる。

以下に API の概要を記す。詳細は「μT-Kernel3.0 仕様書」を参照のこと。

書式	ER ercd = tk_srea_dev_d(ID dd, D start_d, void *buf, SZ size, SZ *asize);	
引数	ID dd	対象デバイスのデバイスディスクリプタ (tk_opn_deb の戻り値)
	W start	0 以上:固有データ A/D 変換値を取得するチャンネル番号 0 未満:属性データ
	void *buf	A/D 変換値を格納する領域の先頭アドレス A/D 変換値は 32bit のデータ(UW 型)

	SZ size	A/D 変換値を取得するチャンネル数
	SZ &asize	A/D 変換値を取得したチャンネル数を返す領域のアドレス
戻り値	ER	エラーコード
説明	dd で指定した A/D 変換ドライバについて、引数 start で指定したチャンネルから、size 個の連続したチャンネルの A/D 変換値を取得する。実際に受信されたデータは*buf に格納され、受信データ数は*asize に返される。	

A/D 変換値を取得するチャンネルの番号を固有データ番号として引数 start に設定する。
複数のチャンネルから A/D 変換値を取得する場合は、引数 start に先頭のチャンネルの番号を設定し、引数 size にチャンネル数を設定する。チャンネルの番号は連続している必要がある。

固有データは 32bit(UW 型)である。ただし、有効な A/D 変換値はハードウェアの仕様に依存する。
取得したデータを格納する領域は、UW 型の配列とし、配列数は size で指定したチャンネル数とする。この領域の先頭アドレスを引数 buf に設定する。

A/D 変換は API が発行されたタイミングで実行される。A/D 変換の完了待ちには割込みを使用する。よって、API 実行中は、API を発行したタスクは待ち状態となる場合がある。その際のタイムアウト時間はコンフィギュレーションで設定される。タイムアウトした場合、API はタイムアウトエラーで終了する。

引数 asize で指定した領域に実際に取得した A/D 変換値の数(チャンネル数)が返される。
API が正常終了した場合、size と asize の値は等しい。

(4) 固有データの書込み
本デバイスは書込み可能な固有データを持たない。書込みの APIを呼び出した場合はE_PAR エラーが返る。

(5) デバイスのクローズ
デバイスの使用終了にあたり、μT-Kernel3.0 の API **tk_cls_dev** により対象デバイスをクローズする。以降、対象デバイスへのアクセスが不可となる。

以下に API の概要を記す。詳細は「μT-Kernel3.0 仕様書」を参照のこと。

書式	ER ercd = tk_cls_dev(ID dd, UINT option);
引数	ID dd　　対象デバイスのデバイスディスクリプタ （tk_opn_deb の戻り値）
	UINT option　　クローズオプション
戻り値	エラーコード
説明	dd で指定したデバイスをクローズする。 本デバイスドライバは option を無視する。

多重オープンされている場合、オープンに対応したクローズが必要である。すべてクローズされるとデバイスドライバは処理を終える。
クローズされたデバイスは、再びオープンされることにより使用可能となる。

3.3.4　属性データ

デバイスの属性データは、API **tk_srea_dev** および **tk_seri_dev** でデータ番号を指定することにより読み書き可能である。

本デバイスの属性データを以下に示す。

種別	名称	データ番号	型	意味
共通属性	TDN_EVENT	-1	ID	事象通知用メッセージバッファ ID ※1

※1 本デバイスはデバイス事象通知をサポートしない。この属性データは将来の拡張のためにある。

3.3.5　コンフィギュレーション

A/D 変換ドライバの各初期設定をコンフィギュレーションにより指定できる。
コンフィギュレーションはハードウェアに依存しない共通コンフィグレーションと、依存する固有コンフィギュレーションがある。
デバイス固有コンフィギュレーションは「3.3.6 ハードウェア依存仕様」に記す。
共通コンフィギュレーションは、コンフィギュレーション定義ファイル adc_cnf.h に以下のように記述される。

名称	初期値	意味
DEVCNF_ADC_DEVNAME	“adc”	デバイス名

3.3.6 ハードウェア依存仕様

(1) TX03-M367 固有仕様

- マイコン内蔵の 12 ビット逐次変換方式 A/D コンバータ(ADC)に対応する。
- コンフィギュレーション定義ファイル adc_cnf_m367.h にて以下のハードウェアに依存する固有コンフィギュレーションを定義する。

ユニット	名称	初期値	意味
A	DEVCNF_ADCA_CLK	0x00000001	クロック設定 ※1
	DEVCNF_ADCA_INTPRI	5	割込み優先順位
	DEVCNF_ADCA_TMOSCAN	1000	A/D 変換タイムアウト時間 ※2
B	DEVCNF_ADCB_CLK	0x00000001	クロック設定 ※1
	DEVCNF_ADCB_INTPRI	5	割込み優先順位
	DEVCNF_ADCB_TMOSCAN	1000	A/D 変換タイムアウト時間 ※2

※1 この値はデバイスのオープン時に ADxCLK レジスタに設定される。

※2 この値は A/Dj 変換の完了待ちのタイムアウト時間に使用される。この時間を超えると API はタイムアウトエラーとなる。このタイムアウト時間は、API 処理内の待ちの合計時間ではない。API 処理中に生じる個々のタスクの待ち状態のタイムアウト時間である。

- 本デバイスドライでは端子の初期化は行わないので、必要に応じて、本デバイスを使用する前に使用する端子の初期化を行う。
  ただし、本マイコンは初期状態で ADC が使用可能である。

(2) RX231 固有仕様

- マイコン内蔵の 12 ビット逐次変換方式 A/D コンバータ(S12ADE)に対応する。
- コンフィギュレーション定義ファイル adc_cnf_rx231.h にて以下のハードウェアに依存する固有コンフィギュレーションを定義する。

名称	初期値	意味
DEVCONF_ADC_INIT_MSTP	TRUE	初期化時にモジュールストップの解除を行う ※1
DEVCNF_ADSSTR0_INI	6	クロック設定 ※2
DEVCNF_ADSSTR1_INI	6	クロック設定 ※2
DEVCNF_ADSSTR2_INI	6	クロック設定 ※2
DEVCNF_ADSSTR3_INI	6	クロック設定 ※2

DEVCNF_ADSSTR4_INI	6	クロック設定 ※2
DEVCNF_ADSSTR5_INI	6	クロック設定 ※2
DEVCNF_ADSSTR6_INI	6	クロック設定 ※2
DEVCNF_ADSSTR7_INI	6	クロック設定 ※2
DEVCNF_ADSSTRL_INI	6	クロック設定 ※2
DEVCNF_ADC_INTPRI	5	割込み優先順位
DEVCNF_ADC_TMOSCAN	1000	A/D 変換タイムアウト時間 ※3

※1 マイコンの初期化処理で初期化済みの場合は FALSE を指定することにより、処理およびコードを節約できる。

※2 この値はデバイスの初期化時に ADSSTRn レジスタに設定される。

※3 この値は A/Dj 変換の完了待ちのタイムアウト時間に使用される。この時間を超えると API はタイムアウトエラーとなる。このタイムアウト時間は、API 処理内の待ちの合計時間ではない。API 処理中に生じる個々のタスクの待ち状態のタイムアウト時間である。

- 本デバイスドライでは端子の初期化は行わないので、必要に応じて、本デバイスを使用する前に使用する端子の初期化を行う。

3.3.7 その他

本デバイスは、以下の機能に対応しない。

- ドライバ要求イベント
- デバイス事象通知

3.4 I2C 通信ドライバ

3.4.1 概要

I2C 通信ドライバは、マイコン内蔵の I2C 通信デバイスの制御を行うデバイスドライバである。
以下の I2C 通信に対応する。

- マスターモード(シングルマスタ)
- 7 ビットスレーブアドレス
- 任意のデータの送信、受信、および連続した送信・受信

3.4.2 基本仕様

シリアル通信ドライバの名称は"iic"とし、I2C 通信デバイスのデバイス(ハードウェア)毎にユニットをアサインする。
I2C 通信ドライバとハードウェアの対応を「表 3-3 I2C 通信デバイスとハードウェアの対応」に記す。

表 3-3 I2C 通信デバイスとハードウェアの対応

マイコン	デバイス(ハードウェア)	ユニット番号	デバイス名
TX03-M367	SBI0	0	“iica”
	SBI1	1	“iicb”
	SBI2	2	“iicc”

3.4.3 使用方法

(1) デバイスドライバの初期化

デバイスドライバを使用するにあたって、初期化関数を呼び出す。初期化関数の呼び出しは、各デバイス(ハードウェア)について一回きりとする。
初期化関数により、デバイスドライバに必要な初期化処理が実行され、OS にデバイスが登録される。以降は OS のデバイス管理機能の API を用いて、デバイスの制御を行うことができる。

以下のデバイス初期化関数の仕様を記す。

書式	ER dev_init_i2c (UW unit) ;
引数	UW unit ユニット番号(0 から始まる)
戻り値	エラーコード
説明	引数 unit で指定したデバイスの初期化、登録を行う。 ユニット番号は「表 3-3 I2C 通信デバイスとハードウェアの対応」を参照。

(2) デバイスのオープン

デバイスの使用開始にあたり、μT-Kernel3.0 の API **tk_opn_dev** により対象デバイスをオープンする。以降、対象デバイスへのアクセスが可能となる。

以下に API の概要を記す。詳細は「μT-Kernel3.0 仕様書」を参照のこと。

書式	ER　tk_opn_dev (CONST UB *devnm, UINT omode) ;	
引数	CONST UB *devnm	デバイス名
	UINT omode	オープンモード
戻り値	エラーコード	
説明	devnm で指定したデバイスを omode のモードでオープンする。 デバイス名 devnm は「表 3-3 I2C 通信デバイスとハードウェアの対応」を参照。	

同一デバイスの多重オープンは許されるが、実際にオープン処理が実行されるのは最初のオープンのみである。

本デバイスはオープンモードの指定は無効である。オープンモードに関係なく、送受信が可能である。

オープン時の I2C 通信デバイスの設定は、コンフィギュレーションで指定された初期値に設定される。詳細は「」を参照。

(3) I2C 通信のデータ送信

I2C 通信デバイスは同期アクセスのみに対応する。よって、データの書込みには API **tk_swri_dev** を使用する。

API(tk_swri_dev)を用いて対象デバイスデバイスへ固有データを書き込むことにより、データ送信を行うことができる。

データは I2C バスで接続された外部デバイスに送信される。送信先の外部デバイスの I2C アドレスを、API の固有データの値に指定する。

以下に API の概要を記す。詳細は「μT-Kernel3.0 仕様書」を参照のこと。

書式	ER ercd = tk_swri_dev(ID dd, W start, CONST void *buf, SZ size, SZ *asize);	
引数	ID dd	対象デバイスのデバイスディスクリプタ (tk_opn_deb の戻り値)
	W start	0 以上:固有データ(送信先外部デバイスの I2C アドレス) 0 未満:属性データ
	void *buf	送信データを格納する領域の先頭アドレス
	SZ size	要求する送信データ数(バイト単位)
	SZ &asize	実際に送信したデータ数を返す領域のアドレス
戻り値	ER	エラーコード
説明	dd で指定した I2C 通信デバイスにより、*buf に格納されたデータを size バイト、start で指定された I2C アドレスの外部デバイスへ送信する。実際に送信されたデータ数は*asize に返される。	

API(tk_srea_dev)の処理の処理中に I2C による通信が実行される。デバイス(ハードウェア)のアクセスの待ちは、割込みを用いて制御する。よって、API の実行中に、API を呼び出したタスクは待ち状態となる。割込み待ちのタイムアウト時間は、デバイスの属性データで指定できる。また、初期値はコンフィギュレーションにて指定される。

(4) I2C 通信データの受信

I2C 通信デバイスは同期アクセスのみに対応する。よって、データの読込みには API **tk_srea_dev** を使用する。

API(tk_srea_dev)を用いて対象デバイスから固有データを読み込むことにより、データ受信を行うことができる。

データは I2C バスで接続された外部デバイスから受信される。受信元の外部デバイスの I2C アドレスを、API の固有データの値に指定する。

以下に API の概要を記す。詳細は「μT-Kernel3.0 仕様書」を参照のこと。

書式	ER ercd = tk_srea_dev_d(ID dd, D start_d, void *buf, SZ size, SZ *asize);	
引数	ID dd	対象デバイスのデバイスディスクリプタ (tk_opn_deb の戻り値)
	W start	0 以上:固有データ(受信本外部デバイスの I2C アドレス) 0 未満:属性データ
	void *buf	受信データを格納する領域の先頭アドレス
	SZ size	要求する受信データ数(バイト単位)
	SZ &asize	実際に受信したデータ数を返す領域のアドレス
戻り値	ER	エラーコード
説明	dd で指定した I2C 通信デバイスにより、size バイトのデータを、start で指定された I2C アドレスの外部デバイスから受信する。実際に受信されたデータは*buf に格納され、受信データ数は*asize に返される。	

API(tk_srea_dev)の処理の処理中に I2C による通信が実行される。デバイス(ハードウェア)のアクセスの待ちは、割込みを用いて制御する。よって、API の実行中に、API を呼び出したタスクは待ち状態となる。割込み待ちのタイムアウト時間は、デバイスの属性データで指定できる。また、初期値はコンフィギュレーションにて指定される。

(5) I2C 通信データの連続した送受信

データの送信に続いて、リスタートコンディションを生成し、続いてデータの受信を行うことができる。

μT-Kernel 3.0 の API では、送受信を同時に行うことはできないので、この機能は、API(tk_swri_dev)を用いて、特定の属性データを書き込むことにより実行される。

属性データは以下のように定義される。

名称	意味	値
TDN_I2C_EXEC	I2C 通信によるの連続した送受信の実行	-100

TDN_I2C_EXEC 属性データは以下のように定義される。

```
typedef struct {
        UW              sadr;           // I2C アドレス
        SZ              snd_size;       // 送信データ数
        UB              *snd_data;      // 送信データ
        SZ              rcv_size;       // 受信データ数
        UB              *rcv_data;      // 受信データ
} T_I2C_EXEC;
```

TDN_I2C_EXEC 属性データの内容を以下に記す。

型	名称	意味
UW	sadr	通信対象のデバイスの I2C アドレス
SZ	snd_size	送信するデータ数(単位:バイト)
UB*	snd_data	送信データの先頭アドレス
SZ	rcv_size	受信するデータ数(単位:バイト)
UB*	rcv_data	受信データを格納する領域の先頭アドレス

API(tk_srea_dev)の処理の処理中に I2C による通信が実行される。デバイス(ハードウェア)のアクセスの待ちは、割込みを用いて制御する。よって、API の実行中に、API を呼び出したタスクは待ち状態となる。割込み待ちのタイムアウト時間は、デバイスの属性データで指定できる。また、初期値はコンフィギュレーションにて指定される。

API(tk_srea_dev)が引数 asize に返す値は、書き込んだ属性データ TDN_I2C_EXEC のサイズである。通信データのサイズでないので注意すること。

(6) デバイスのクローズ

デバイスの使用終了にあたり、μT-Kernel3.0 の API **tk_cls_dev** により対象デバイスをクローズする。以降、対象デバイスへのアクセスが不可となる。

以下に API の概要を記す。詳細は「μT-Kernel3.0 仕様書」を参照のこと。

書式	ER ercd = tk_cls_dev(ID dd, UINT option);
引数	ID dd　　対象デバイスのデバイスディスクリプタ (tk_opn_deb の戻り値)
	UINT option　　クローズオプション
戻り値	エラーコード

説明	dd で指定したデバイスをクローズする。 本デバイスドライバは option を無視する。

多重オープンされている場合、オープンに対応したクローズが必要である。すべてクローズされるとデバイスドライバは処理を終える。
クローズされたデバイスは、再びオープンされることにより使用可能となる。

3.4.4 属性データ

デバイスの属性データは、API **tk_srea_dev** および **tk_seri_dev** でデータ番号を指定することにより読み書き可能である。

本デバイスの属性データを以下に示す。

種別	名称	データ番号	型	意味
共通属性	TDN_EVENT	-1	ID	事象通知用メッセージバッファ ID ※1
デバイス種別属性	TDN_I2C_EXEC	-100	※2	連続した送受信

※1 本デバイスはデバイス事象通知をサポートしない。この属性データは将来の拡張のためにある。
※2 デバイス種別属性データ TDN_I2C_EXEC については「3.4.3 使用方法 (5)I2C 通信データの連続した送受信」を参照。

3.4.5 コンフィギュレーション

シリアル通信デバイスの各初期設定をコンフィギュレーションにより指定できる。
コンフィギュレーションはハードウェアに依存しない共通コンフィグレーションと、依存するデバイス固有コンフィギュレーションがある。

デバイス固有コンフィギュレーションは「3.4.6 ハードウェア依存仕様」に記す。
共通コンフィギュレーションは、コンフィギュレーション定義ファイル i2c_cnf.h に記述される。内容を以下に記す。

名称	初期値	意味
DEVCNF_I2C_DEVNAME	“iic”	デバイス名
DEVCNF_I2C_MAX_SDATSZ	100	最大送信データサイズ(バイト単位)
DEVCNF_I2C_MAX_RDATSZ	100	最大送信データサイズ(バイト単位)

3.4.6 ハードウェア依存仕様

(1) TX03-M367 固有仕様

- マイコン内蔵のシリアルバスインタフェース(SBI)を使用する。SBI の I2C バスモードにのみ対応する。クロック同期 SIO モードには非対応である。
- コンフィギュレーション定義ファイル i2c_cnf_m367.h にて以下のハードウェアに依存するデバイス固有コンフィギュレーションを定義する。

名称	初期値	意味
DEVCNF_I2C0_SCK	0	SBI0 の SCL 出力クロックの周波数選択 ※1
DEVCNF_I2C1_SCK	0	SBI1 の SCL 出力クロックの周波数選択 ※1
DEVCNF_I2C2_SCK	0	SBI2 の SCL 出力クロックの周波数選択 ※1
DEVCNF_I2C0_INTPRI	5	SBI0 の I2C 通信の割込み優先順位
DEVCNF_I2C1_INTPRI	5	SBI1 の I2C 通信の割込み優先順位
DEVCNF_I2C2_INTPRI	5	SBI2 の I2C 通信の割込み優先順位
DEVCNF_I2C0_TMO	1000	SBI0 の通信タイムアウト時間(単位:ミリ秒) ※2
DEVCNF_I2C1_TMO	1000	SBI1 の通信タイムアウト時間(単位:ミリ秒) ※2
DEVCNF_I2C2_TMO	1000	SBI2 の通信タイムアウト時間(単位:ミリ秒) ※2

※1 この値は SBIxCR1 レジスタの SCK に設定される。

※2 この値は通信割込み待ちのタイムアウト時間に使用される。この時間を超えるとAPIはタイムアウトエラーとなる。このタイムアウト時間は、API 処理内の待ちの合計時間ではない。API 処理中に生じる個々のタスクの待ち状態のタイムアウト時間である。

- SBI が使用するマイコンの端子には複数の機能が割り当てられているので、SBI 用に初期化が必要である。本デバイスドライでは端子の初期化は行わないので、マイコンの初期化時などに、本デバイスを使用する前に使用する端子の初期化を行う必要がある。

  ただし、μT-Kernel 3.0 の IoT-Engine 向けの標準の実装では、I2C 通信は、IoT-Engine 用の I2C デバイスの制御用に SBI1 の端子 SDA1 と SCK1は初期化されている。他の UUBI を使用する場合は、マイコンの初期化処理などで、使用する端子の設定が必要である。

3.4.7 その他

シリアル通信デバイスは、以下の機能に対応しない。

- ドライバ要求イベント
- デバイス事象通知

以上