

μ T-Kernel 3.0 共通デバイスドライバ説明書

Rev 3.00.00

April, 2023



μT-Kernel 3.0

目次

1. 概要	2
1.1. 本書について	2
1.2. 目的と位置づけ	2
1.3. 特徴	2
1.4. 対象とするハードウェア	3
1.5. ソースコードの基本構成	4
1.6. 制限事項	5
1.7. 表記について	5
2. 使用方法	6
2.1. OS への組み込み方法	6
2.2. コンフィギュレーション	6
2.3. ユーザプログラムからの使用	7
3. デバイスドライバ	8
3.1. 共通仕様	8
3.1.1. デバイスの初期化	8
3.1.2. デバイスのオープンモード	8
3.1.3. 多重オープンと排他制御	8
3.1.4. 同期アクセスと割込み制御	8
3.2. シリアル通信ドライバ	9
3.2.1. 概要	9
3.2.2. 基本仕様	9
3.2.3. 使用方法	9
3.2.4. 属性データ	12
3.2.5. コンフィギュレーション	14
3.2.6. ハードウェア依存仕様	15
3.2.7. その他	15
3.3. A/D 変換デバイスドライバ	16
3.4. I2C 通信デバイスドライバ	16

1. 概要

本品は μT-Kernel 3.0 の BSP(Board Support Package)用のサンプルデバイスドライバである(以下、「本ソフト」とする)。対象はユーシーテクノロジー株式会社(以下、UCT)が開発した BSP に付属のデバイスドライバに限定される。

1.1. 本書について

本書は、UCT が開発した BSP に付属するデバイスドライバの仕様などについて説明する。

本書で説明するデバイスドライバは、μ T-Kernel 3.0 向けにトロンフォーラムが開発したデバイスドライバをベースとしている。

トロンフォーラムが開発・公開しているデバイスドライバについては、トロンフォーラムの資料「μ T-Kernel 3.0 デバイスドライバ説明書」を参照のこと。

1.2. 目的と位置づけ

トロンフォーラムからは、以下を目的としてサンプルコードが公開されている。

- μ T-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。
- μ T-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。

μ T-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ていない。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 μ T-Kernel 3.0 の仕様に含まれていない。

なお、トロンフォーラムのサンプルコードは T-License に基づいて公開されている。

1.3. 特徴

前述の目的より、本ソフトは以下の特徴を持つ。

- μ T-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容易である。

1.4. 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用のデバイス(内蔵デバイス、拡張ボード)に対応する。
ターゲットボードは、実際に動作確認を行った実機である。

表 1-1 対象デバイス

マイコン (マイコンメーカー)	ターゲットボード	対象デバイス
CY8C624ABZI-S2D44A0 (Infineon)	CY8CKIT-062S2-43012	シリアル通信(UART)
TMPM4KNFYAFG (東芝デバイス&ストレージ)	SBK-M4KN	シリアル通信(UART)
TMPM4MNFYAFG (東芝デバイス&ストレージ)	SBK-M4KN	シリアル通信(UART)

- ◆ 本ソフトは基本的にハードウェア依存部に対して実装されているが、各ハードウェア(デバイス)の特性によっては共通化が難しいため変更されている場合がある。このため、本ソフトを利用する場合は、対象とするターゲットボードに対応したブランチを選択したうえで利用する必要がある。

他のターゲットボード用のブランチを選択した場合、正常にビルド・実行できるとは限らないので注意が必要である。

1.5. ソースコードの基本構成

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
└─drvif/	mSDI(Simple Device driver I/F)
└─ser/	シリアル通信ドライバ(UART)
└─sysdepend/	ハードウェア依存部
└─<CPU#1>	CPU#1 依存部
└─<CPU#2>	CPU#2 依存部
└─<CPU#3>	CPU#3 依存部
:	:
└─<CPU#n>	CPU#n 依存部
└─adc/	A/D 変換ドライバ
└─sysdepend/	ハードウェア依存部
└─i2c/	I2C 通信デバイスドライバ
└─sysdepend/	ハードウェア依存部

- ❶ 各デバイスの機種依存部のコードは、**sysdepend/**以下に格納されるが、**sysdepend/**以下は**ser/sysdepend/**と同じ構成なので**net/**以下では省略している。
- ❷ 各デバイスドライバのルートとなるディレクトリ(**ser/**など)に含まれるソースコードは原則としてハードウェアに依存しない。ただし、各ハードウェア(デバイス)の特性によっては変更される場合があるので注意が必要である。
- ❸ 上記にはトロンフォーラムが配布するデバイスドライバ(**adc/**と**i2c/**)も含めているが、これらは UCT が提供するものではない。

各ディレクトリの内容は以下の通りである。

(A) include/ 定義ファイル

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」を参照のこと。

(B) common/ デバイス共通ファイル

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、デバイスドライバを開発する際に使用する。

デバイスドライバのユーザ(μ T-Kernel 3.0 用のアプリケーション開発者)が直接使用することはない。

(C) ser/ シリアル通信ドライバ

シリアル通信デバイスのソースコードを格納する。

(D) `adc/` A/D 変換ドライバ
A/D 変換デバイスのソースコードを格納する。



(E) `i2c/` 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。

1.6. 制限事項

本ソフトは、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。

1.7. 表記について

表記	説明
<code>[]</code>	<code>[]</code> はソフトウェア画面のボタンやメニューを表す。
<code>「」</code>	<code>「」</code> はソフトウェア画面に表示された項目などを表す。
	注意が必要な内容の場合に記述する。
	補足やヒントなどの内容の場合に記述する。
<code><TARGET></code>	ターゲットボード用のディレクトリ名を表す。
<code><CPU></code>	CPU 用のディレクトリ名を表す。
<code><CORE></code>	CPU コア用のディレクトリ名を表す。

2. 使用方法

2.1. OS への組み込み方法

本ソフトは、μT-Kernel3.0 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 config_device.h を変更する。本ファイルは、以下の OS のコンフィギュレーション情報ディレクトリに格納されている。

config/config_device.h

config_device.h は C 言語のソースコードであり、各設定項目の値をマクロとして定義している。

config/config_device.h

```

/* Device usage settings
 * 1: Use 0: Do not use
 */
#define DEVCNF_USE_SER      1      // Serial communication device
#define DEVCNF_USE_ADC      0      // A/D conversion device
#define DEVCNF_USE_IIC      0      // I2C communication device
#define DEVCNF_USE_CDC      1      // USB CDC communication device
#define DEVCNF_USE_NET      1      // Ethernet communication device

```

リスト 1 デバイスドライバ用の設定

各マクロによってそれぞれのデバイスを使用する(1)か、使用しない(0)かを指定する。

- ❶ 実装によってはマクロが定義されていない場合がある。
マクロが定義されていないデバイスドライバは非対応である。
- ❶ マクロの設定値とデバイスドライバの対応状況(対応／非対応)は一致していない。
このため、config_device.h においてマクロ定義を 1(使用する)に設定しても、対象のデバイスドライバが利用可能になるとは限らない。
対象とするマイコンにおけるデバイスドライバの対応状況については、「1.4. 対象とするハードウェア」を参照のこと。

2.3. ユーザプログラムからの使用

アプリケーションプログラムから本ソフトを使用するには、以下のようにデバイスドライバ定義ファイルをインクルードする。

```
#include <tk/device.h>
```

アプリケーションプログラムでは、<tk/device.h>をインクルードすれば、コンフィギュレーションに合わせて必要な各デバイスの API 定義ファイルがインクルードされる。

各デバイスの API 定義ファイルは以下のディレクトリ中に存在する。

```
device/include/
```

API 定義ファイルの一覧を以下に示す。

表 2-1 API 定義ファイル一覧

名称	説明
device.h	全デバイスドライバの API 定義ファイル
dev_ser.h	シリアル通信デバイスドライバの API 定義ファイル
dev_adc.h	A/D 変換デバイスドライバの API 定義ファイル
dev_iic.h	I2C 通信デバイスドライバの 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、割り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、読み専用 TD_READ、書き専用 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。読み専用の場合は、固有データの書き込みが不可となり、書き専用の場合は固有データの読み込みが不可となる。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. 同期アクセスと割り込み制御

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 である `tk_srea_dev` および `tk_swri_dev` を使用する。非同期 API である `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、原則として割り込みを使用する。よって、API の処理内において割り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)シリアル通信を用いて、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は以下の通りである。

表 3-1 シリアル通信デバイスとハードウェアの対応

ターゲットボード	デバイス	ユニット番号	デバイス名	TM
SBK-M4KN	UART0	0	sera	○
CY8CKIT-062S2-43012	SCB1	1	serb	
	SCB2	2	serc	
	SCB5	5	serf	○

一覧の「TM」が○になっているデバイスは、T-Monitor 互換機能がコンソールとして利用しているデバイスである。

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。

表 3-2 シリアル通信ドライバ初期化関数

書式	ER ercd = dev_init_ser(UW unit);
引数	UW unit ユニット番号(0, 1, 2, ...)
戻値	エラーコード
説明	unit で指定したデバイスの初期化と登録を行う。 ユニット番号とデバイスの対応は「表 3-1」を参照のこと。

(2) デバイスのオープン

デバイスの使用を開始するには、μ T-Kernel 3.0 の API である **tk_opn_dev** を用いて対象デバイスをオープンする。

tk_opn_dev でのデバイスのオープン後、対象デバイスへのアクセスが可能となる。

対象デバイスへのアクセスには本関数の戻値であるデバイスディスクリプタを使用する。

表 3-3 デバイスオープン関数

書式	ID dd = 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
戻値	デバイスディスクリプタ、またはエラーコード
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。 指定したデバイスの初期化と OS への登録を行う。 デバイス名とデバイスの対応は「表 3-1」を参照のこと。

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(詳細は「μT-Kernel 3.0 仕様書」を参照のこと)。読み込み専用でオープン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。書き込み専用でオープンした場合はデバイスの固有データの読み込み(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に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」を参照のこと。

(3) シリアル通信データの受信(固有データの読み込み)

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、データの読み込みには tk_srea_dev を使用する。

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。

表 3-4 デバイスの同期読み込み関数

書式	ER ercd = tk_srea_dev(ID dd, W start, 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_dev の戻値)
	W start ≥ 0 : 固有データ(値は任意) < 0 : 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
戻値	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。

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。

`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み込みである。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。

`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、読み込み可能なデータ数が `*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) シリアル通信データの送信(固有データの書き込み)

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、データの書き込みには `tk_swri_dev` を使用する。

`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。

表 3-5 デバイスの同期書き込み関数

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_dev の戻値)	
	W start	≥ 0 : 固有データ(値は任意) ＜ 0 : 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	
戻値	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は*asize に返される。		

`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。

`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き込みである。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。

`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-Kernel 3.0 の API である `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。

表 3-6 デバイスクローズ関数

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_dev の戻値)	
	UINT option	クローズオプション	
戻値	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、`tk_srea_dev` および `tk_swri_dev` で `start` に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は以下の通りである。

表 3-7 シリアル通信デバイスの属性データ

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

❶ 本ソフトでは `TDN_EVENT`(デバイス事象通知)はサポートしていない。

この属性データは将来の拡張用に定義してある。

❶ シリアル通信モード(`TDN_SER_MODE`)およびシリアル通信速度(`TDN_SER_SPEED`)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一旦クローズしたのち、再度オープンしなければならない。

シリアル通信デバイスの属性データの詳細は以下の通りである。

(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

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)を数値で指定する。

指定可能な通信速度は以下のとおりである。

表 3-8 シリアル通信デバイスドライバがサポートしている通信速度

ターゲットボード	通信速度
CY8CKIT-062S2-43012	115200, 57600, 38400, 19200, 9600, 2400, 1200
SBK-M4KN	115200, 57600, 38400, 19200, 9600, 4800, 2400, 1200

 上記は本ソフトで対応している通信速度である。

各ターゲットボードでは更に多くの通信速度にも対応している。

(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。本属性データは読込みのみ可能である。

読み込むと本属性データは 0 にクリアされる。

データの内容はハードウェア(マイコン)毎に規定される。ただし、以下のエラーはすべてのハードウェアで共通である。

表 3-9 受信バッファ・オーバーフローのマクロ定義

名称	設定値	ビット	意味
DEV_SER_ERR_ROVR	0x00000080	7	受信バッファ・オーバーフロー

(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、読込みはできない(エラーとなる)。

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ギュレーションと、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」を参照のこと。

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` において以下の様に実装されている。

`device/ser/ser_cnf.h`

```
#define DEVCNF_SER_DEVNAME    "ser"        // Device name ("ser")

#define DEVCNF_SER_BUFFSIZE   50           // Communication data buffer size

/* Default value for attribute data */
#define DEVCNF_SER_SPEED      115200       // Communication speed (baud rate)
#define DEVCNF_SER_MODE      (DEV_SER_MODE_CTSN | DEV_SER_MODE_RTSEN | ¥
                             DEV_SER_MODE_8BIT | DEV_SER_MODE_1STOP | ¥
                             DEV_SER_MODE_PNON )
                                     // Mode: Hard flow control enable,
                                     // data 8bit, stop 1bit, no parity
#define DEVCNF_SER_SND_TMO    TMO_FEVR     // Send timeout
#define DEVCNF_SER_RCV_TMO    TMO_FEVR     // Receive timeout
```

リスト 2 共通コンフィギュレーション定義

通信速度、通信モード、タイムアウト時間はシリアル通信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

シリアル通信ドライバのハードウェア依存仕様に関しては、各ターゲットボードの「実装仕様書」を参照のこと。

3.2.7. その他

シリアル通信デバイスドライバは、以下の機能には対応していない。

- 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

本ソフトには含まれないので省略する。

トロンフォーラムが配布する BSP に含まれる A/D 変換デバイスドライバについては、「μT-Kernel 3.0 デバイスドライバ説明書」を参照のこと。

3.4. I2C 通信デバイスドライバ

本ソフトには含まれないので省略する。

トロンフォーラムが配布する BSP に含まれる I2C 通信デバイスドライバについては、「μT-Kernel 3.0 デバイスドライバ説明書」を参照のこと。

μ T-Kernel 3.0

共通デバイスドライバ説明書

Rev 3.00.00 (April, 2023)

ユーシーテクノロジー株式会社

141-0031 東京都品川区西五反田 2-12-3 第一誠実ビル 9F

©2023 Ubiquitous Computing Technology Corporation All Rights Reserved.