文書番号：

TOPPERS Automotive Kernel
アプリケーションノート

Ver.3.00

2008/10/20

株式会社ヴィッツ　組込みソフトウェア開発部

TOPPERS Automotive Kernel
　　Toyohashi Open Platform for Embedded Real-Time Systems Automotive Kernel

＜目次＞

1. 概要

1.1. はじめに

本アプリケーションノートは、TOPPERS Automotive Kernel について、その機能を体験するために付属しているサンプルアプリケーションの利用方法を記載します。なお、本アプリケーションノートで記載するのは、カーネル機能の使用法の一例であり、安全なシステム構築を保障するものではないことをご了承ください。

1.2. 関連文書

カーネルの機能詳細は「TOPPERS Automotive Kernel 外部仕様書」を参照してください。

カーネル用 SG の使用方法は「TOPPERS Automotive Kernel SG 取扱説明書」を参照してください。

開発環境の構築手順など機種依存の情報は「TOPPERS Automotive Kernel アプリケーションノート "ターゲットボード名"」を参照してください。

OSEK/VDX 仕様：OSEK/VDX Operating System Ver2.2.1
OSEK/VDX OIL Specification Version 2.5
OSEK/VDX Binding Specification Version 1.4.2

※OSEK/VDX が公開している各仕様書は http://www.osek-vdx.org/よりダウンロード可能です。

2. TOPPERS Automotive Kernelの階層

TOPPERS Automotive Kernel 開発階層の構成について説明します。

```
//root
        /config
                /CPU 依存部
                        /システム依存部
        /include
        /kernel
        /sample
        /sg
                /impl_oil
        /syslib
                /ライブラリ CPU 依存部
                        /ライブラリシステム依存部
        /tools
                /ワークスペース
                        /プロジェクト
                                /output
```

- root：ルート階層
- config：機種依存階層
- CPU 依存部：CPU・開発環境依存部階層（ターゲット CPU・開発環境ごとに複数存在）
- システム依存部：システム依存部階層（ターゲットシステムごとに複数存在）
- include：アプリケーションインクルードファイル階層
- kernel：カーネル共通部階層
- sample：サンプルプログラム階層
- sg：システムジェネレータ階層
- impl_oil：OIL 記述の実装定義部ファイル階層
- syslib：サンプルプログラム用システムライブラリ階層
- ライブラリ CPU 依存部：CPU 依存部階層（ターゲット CPU ごとに複数存在）
- ライブラリシステム依存部：システム依存部階層（ターゲットシステムごとに複数存在）
- tools：メーカー開発環境階層

- ワークスペース：ワークスペース階層（ワークスペースごとに複数存在）
- プロジェクト：プロジェクト階層（プロジェクトごとに複数存在）
- output：オブジェクトファイル・ロードファイル出力階層

3. サンプルアプリケーション

サンプルアプリケーションについて説明します。

3.1. サンプルアプリケーションの概要

3.1.1. 構成概要

サンプルアプリケーションは 8 個のタスク、5 個の割込み、3 個のリソース、3 個のイベント、2 個のカウンタ、5 個のアラーム、3 個のアプリケーションモードで構成しています。

タスクはコマンドを受信するメインタスクと、ユーザーからの操作で動作するタスクから構成されています。タスクへの操作はユーザーからのコマンド入力、または OS オブジェクトにより行われます。

割込みはコマンドを受信する割込みと、OS オブジェクト（カウンタ）を動作させる割込みと、リソース管理・割込み制御機能の動作確認用の割込みから構成されています。

リソースはタスクータスク間排他用のリソースと、割込みータスク間排他用のリソースと、タスクグループを構成するためのリソースから構成されている。

アラームはメインタスク周期アラームと、アラーム機能の動作確認用アラーム、カウンタ機能動作確認アラームから構成されている。

アプリケーションモードはユーザー操作タスクを自動起動しないモードと、自動起動するモードから構成されている。

サンプルアプリケーションは上記で構成されているシステムに対し、ユーザーから任意のコマンドを入力し、動作を確認できるプログラムとなっています。当然、仕様に合わない指令を行った場合、エラーとなることも確認可能となっています。

3.1.2. 動作概要

サンプルアプリケーションはターゲットボードとターミナル PC をシリアル通信（RS-232C）で接続し、PC 上のシリアルターミナルからコマンドを入力することで動作します。また、サンプルアプリケーションの動作ログを PC 上のシリアルターミナルに表示します。PC 上のシリアルターミナルの設定およびサンプルアプリケーションのコマンドについては次節以降を参照してください。

ターゲットボードとターミナル PC をシリアルケーブルで接続し、PC 上でターミナルソフトを起動後、ターゲットボードの電源を ON にすると以下のログが表示されます。

```
TOPPERS　Automotive Kernel Release 1.1 for H8S/HSB8S2638F
Sample System StartUp

Input Command:
```

Input Command: のプロンプトが表示されたら、前述のコマンドを入力することができます。

＜タスクを起動する＞

タスクを起動させるコマンドは'a'ですので、タスク 1 を起動するには指令対象タスク指定コマンドの

```
Input Command:1
Input Command:a
Call ActivateTask(Task1)
Task1 ACTIVATE
Input Command:
```

'1'とタスク起動コマンドの'a'を入力します。コマンド入力の結果以下のログが表示されます。

＜タスクを終了する＞

タスクを終了させるコマンドは'A'ですので、上記で起動したタスクを終了させるには、タスク終了コマンド'A'を入力します。コマンド入力の結果以下のログが表示されます。

```
Input Command:A
Task1 TERMINATE
Input Command:
```

上記では指令対象タスク指定コマンドの'1'を入力していませんが、タスク終了コマンドは指令したいタスク 1 に対して行われています。これは指令対象タスク指定コマンドが、一度入力すると以降変更されるまでの間保持されているためです。

＜最終コマンドの保持＞

タスクが起動していない状態でのタスクの終了コマンドなど、即実行ができないコマンドについては、各タスクに対して最後に指令されたコマンドを保持しています。このため、タスクが起動していない状態でタスク終了コマンド'A'を、それに続けてタスク起動コマンド'a'を入力すると、以下のログが表示されます。

```
Input Command:A
Input Command:a
Call ActivateTask(Task1)
Task1 ACTIVATE
Task1 TERMINATE
Input Command:
```

なお、タスク起動のような即実行可能なコマンドは保持されません。

3.2. サンプルアプリケーションの動作環境

サンプルアプリケーションを動作させるための環境は、ターゲットシステムごとの実装依存です。詳細は「TOPPERS Automotive Kernel アプリケーションノート"ターゲットボード名"」を参照してください。

3.3. コマンド一覧

サンプルアプリケーションを操作するコマンドの一覧を以下に記載します。

コマンド種別	コマンド	動作概要
タスク指定	1	以降のコマンドを Task1 に対して行う。
	2	以降のコマンドを Task2 に対して行う。
	3	以降のコマンドを Task3 に対して行う。
	4	以降のコマンドを Task4 に対して行う。
	5	以降のコマンドを Task5 に対して行う。
タスク管理機能	a	ActivateTask にてタスクを起動する。
	A	TerminateTask にてタスクを終了する。
	!	ChainTask にてタスクを終了し、Task1 を起動する。(Shift+1)
	"	ChainTask にてタスクを終了し、Task2 を起動する。(Shift+2)
	#	ChainTask にてタスクを終了し、Task3 を起動する。(Shift+3)
	$	ChainTask にてタスクを終了し、Task4 を起動する。(Shift+4)
	%	ChainTask にてタスクを終了し、Task5 を起動する。(Shift+5)
	s	ノンプリエンプティブ属性であるタスク MainTask にて最高優先度タスク HighPriorityTask を起動し、Schedule にて再スケジューリングを行う。

	S	ノンプリエンプティブタスク NonPriTask を起動する。ノンプリエンプティブタスク NonPriTask にて最高優先度タスク HighPriorityTask を起動し、タスク終了する。
	z	GetTaskID にて自タスクの ID を取得する。
	Z	GetTaskState にてタスク状態を取得する。
割込み管理機能	d	DisableAllInterrupts を実行後、ハードウェアカウンタ値を 3 回表示し、EnableAllInterrupts を実行する。
	D	SuspendAllInterrupts を実行後、ハードウェアカウンタ値を 3 回表示し、さらに SuspendAllInterrupts を実行後、ハードウェアカウンタ値を 3 回表示し、ResumeAllInterrupts を実行後、ハードウェアカウンタ値を 3 回表示し、ResumeAllInterrupts を実行する。
	f	SuspendOSInterrupts を実行後、ハードウェアカウンタ値を 3 回表示し、さらに SuspendOSInterrupts を実行後、ハードウェアカウンタ値を 3 回表示し、さらに SuspendAllInterrupts を実行後、ハードウェアカウンタ値を 3 回表示し、ResumeAllInterrupts を実行後、ハードウェアカウンタを 3 回表示し、ResumeOSInterrupts を実行後ハードウェアカウンタ値を 3 回表示し、ResumeOSInterrupts を実行する。
	T	ハードウェアカウンタ値を 3 回表示する。
リソース管理機能	k	GetResource にてリソース TskLevelRes を取得する。なお、Task3 は、このリソースより優先度が高いためエラーとなる。
	K	ReleaseResource にてリソース TskLevelRes を解放する。
	l	GetResource にてスケジューラリソースを取得後、最高優先度タスク HighPriorityTask を起動し、ReleaseResource にてスケジューラリソースを解放する。
	i	GetResource にてリソース IntLevelRes を取得後、ハードウェアカウンタ値を 3 回表示し、ReleaseResource にてリソース IntLevelRes を解放する。
イベント制御機能	e	SetEvent にてイベントを設定する。Task2 と Task3 以外は割り当てがないためエラーとなる。
	w	ClearEvent にて自タスクのイベントをクリアする。Task2 と Task3 以外は割り当てがないためエラーとなる。
	E	GetEvent にてイベント状態を取得する。Task2 と Task3 以外は割り当てがないためエラーとなる。
	W	WaitEvent にて自タスクのイベントを待つ。Task3 と Task4 以外はエラーとなる。
アラーム機能	b	GetAlarmBase にてアラーム MainCycArm のアラームベース情報を取得する。

機能	B	GetAlarm にてアラーム MainCycArm の残りカウント値を 2 回連続で取得する。
	v	SetRelAlarm にてアラーム ActTskArm を起動し、500ms 後にタスク Task1 を起動する。
	V	SetRelAlarm にてアラーム SetEvtArm を起動し、500ms 後にイベント T3Evt を設定する。
	n	SetRelAlarm にてアラーム CallBackArm を、パラメータ 900ms 後に満了・単発アラーム指定で設定する。
	N	SetRelAlarm にてアラーム CallBackArm を、パラメータ 900ms 後に満了・500ms 周期アラーム指定で設定する。
	m	SetAbsAlarm にてアラーム CallBackArm を、パラメータカウンタ値 900 に満了・単発アラーム指定で設定する。
	M	SetAbsAlarm にてアラーム CallBackArm を、パラメータカウンタ値 900 に満了・ 500ms 周期アラームで設定する。
	h	CancelAlarm にてアラーム CallBackArm をキャンセルする。
カウンタ操作機能	c	カウンタ通知用割込みを起動し、割込み内にて SignalCounter を実行し、カウンタ SampleCnt にシグナル通知する。1 シグナルでアラーム SampleArm が満了し、コールバックを実行する。
OS 実行制御機能	p	GetActiveApplicationMode にてアプリケーションモードを取得する。
	q	ShutdownOS をコード E_OK で実行し、サンプルプログラムを終了する。
	Q	ShutdownOS をコード E_OS_STATE で実行し、サンプルプログラムを終了する。

3.4. オブジェクト一覧

サンプルアプリケーションで定義しているオブジェクトの一覧を以下に記載します。

オブジェクト種別	オブジェクト概要	オブジェクトパラメータ
OS	OS システム	スタートアップフック：使用 シャットダウンフック：使用 エラーフック：使用 プレタスクフック：未使用 ポストタスクフック：未使用 スケジューラリソース：使用

タスク	メインタスク	タスク ID：MainTask 優先度：14 多重起動数：1 スケジュール：ノンプリエンプティブ 自動起動：AppMode1, AppMode2, AppMode3 概要：ユーザインタフェース（シリアル IO よりコマンドを受信し、それに対応した動作を行なう)。周期アラーム MainCycArm により、100ms ごとに待ち解除しコマンドの受信有無をポーリングする。イベント（ID：MainEvt）を関連付けている。
	最高優先度タスク	タスク ID：HighPriorityTask 優先度：15 多重起動数：1 スケジュール：フルプリエンプティブ 自動起動：なし 概要：起動ログを出力して終了する。ノンプリエンプティブタスクから起動され、プリエンプトしているかどうかの確認用。
	ノンプリエンプティブタスク	タスク ID：NonPriTask 優先度：1 多重起動数：8 スケジュール：ノンプリエンプティブ 自動起動：なし 概要：起動ログを出力し、最高優先度タスク HighPriorityTask を起動後、終了ログを出力してタスクを終了する。
	同時実行タスク 1	タスク ID：Task1 優先度：4 スケジュール：フルプリエンプティブ 自動起動：AppMode2 多重起動数：8 概要：並列処理タスク（メインタスクからの指令により動作)。起動されすると無限ループに入り、コマンド処理を実行する。リソース TskLevelRes を関連付けている。リソース IntLevelRes を関連付けている。

	同時実行タスク 2	タスク ID：Task2 優先度：7 多重起動数：1 スケジュール：フルプリエンプティブ 自動起動：なし 概要：並列処理タスク（メインタスクからの指令により動作)。起動されすると無限ループに入り、コマンド処理を実行する。リソース TskLevelRes を関連付けている。リソース IntLevelRes を関連付けている。イベント T2Evt を関連付けている。
	同時実行タスク 3	タスク ID：Task3 優先度：12 多重起動数：1 スケジュール：フルプリエンプティブ 自動起動：AppMode3 概要：並列処理タスク（メインタスクからの指令により動作)。起動されすると無限ループに入り、コマンド処理を実行する。イベント待ちすることが可能である。リソース IntLevelRes を関連付けている。イベント T3Evt を関連付けている。
	同時実行タスク 4	タスク ID：Task4 優先度：6 多重起動数：5 スケジュール：フルプリエンプティブ 自動起動：なし 概要：並列処理タスク（メインタスクからの指令により動作)。起動されすると無限ループに入り、コマンド処理を実行する。リソース TskLevelRes を関連付けている。リソース IntLevelRes を関連付けている。内部リソース GroupRes を関連付けている。
	同時実行タスク 5	タスク ID：Task5 優先度：9 多重起動数：5 スケジュール：フルプリエンプティブ 自動起動：なし 概要：並列処理タスク（メインタスクからの指令により動作)。起動されすると無限ループに入り、コマンド処理を実行する。リソース TskLevelRes を関連付けている。リソース IntLevelRes を関連付けている。内部リソース GroupRes を関連付けている。

ISR	シリアルIO受信割込み	ISRID：RxHwSerialInt 優先度：6（実装依存：優先度関係の目安にしてください） カテゴリ：1 概要：コマンドを受信する。
	シリアルIO受信エラー割込み	ISRID：ErrHwSerialInt 優先度：6（実装依存：優先度関係の目安にしてください） カテゴリ：1 概要：受信エラー処理を行う。なお、受信割込みと受信エラー割込みが1つの割込み要因となっている場合、本割込みの設定は不要となりす。
	ハードウェアカウンタ1割込み	ISRID：HwCnt1Int 優先度：5（実装依存：優先度関係の目安にしてください） カテゴリ：1 概要：割込み禁止確認用カウンタ1を加算する。
	システムタイマ割込み	ISRID：SysTimerInt 優先度：4（実装依存：優先度関係の目安にしてください） カテゴリ：2 概要：SignalCounter(SysTimerCnt)を実行する。
	カウンタソース割込み	ISRID：CounterInt 優先度：3（実装依存：優先度関係の目安にしてください） カテゴリ：2 概要：SignalCounter(SampleCnt)を実行する。
	ハードウェアカウンタ2割込み	ISRID：HwCnt1Int 優先度：2（実装依存：優先度関係の目安にしてください） カテゴリ：2 概要：割込み禁止確認用カウンタ 2 を加算する。リソース（ID：IntLevelRes）を関連付けている.
リソース	タスクレベルリソース	リソースID：TskLevelRes プロパティ：標準
	割込みレベルリソース	リソースID：IntLevelRes プロパティ：標準
	タスクグループリソース	リソースID：GroupRes プロパティ：内部
イベント	メインタスクイベント	イベントID：MainEvt
	タスク2イベント	イベントID：T2Evt

	タスク 3 イベント	イベント ID：T3Evt
カウンタ	システムタイマカウンタ	カウンタ ID：SysTimerCnt カウント値：0～999 加算値：1
	サンプルカウンタ	カウンタ ID：SampleCnt カウント値：0～99 加算値：10
アラーム	メイン周期アラーム	アラーム ID：MainCycArm ベースカウンタ ID：SysTimerCnt アクション：イベント設定 MainEvt 自動起動：AppMode1, AppMode2, AppMode3 設定：カウンタ 100，周期 100
	タスク起動アラーム	アラーム ID：ActTskArm ベースカウンタ ID：SysTimerCnt アクション：タスク起動 Task1 自動起動：なし
	イベント設定アラーム	アラーム ID：SetEvtArm ベースカウンタ ID：SysTimerCnt アクション：イベント設定 T3Evt 自動起動：なし
	コールバック実行アラーム	アラーム ID：CallBackArm ベースカウンタ ID：SysTimerCnt アクション：コールバック関数実行 自動起動：なし
	SignalCounter 確認用アラーム	アラーム ID：SampleArm ベースカウンタ ID：SampleCnt アクション：コールバック関数実行 自動起動：なし
アプリケーションモード	自動起動なしモード	アプリケーションモード ID：AppMode1
	Task1 自動起動モード	アプリケーションモード ID：AppMode2
	Task3 自動起動モード	アプリケーションモード ID：AppMode3

4. サンプル用ライブラリ

サンプルアプリケーションを動作させるための、ミドルウェアについて説明します。

4.1. システムタイマモジュール

システムタイマカウンタを動作させるためのタイマモジュールについて説明します。本モジュールでは次項以降に記載するミドルウェア API を提供しています。

4.1.1. タイマモジュールの初期化

- 関数名

  void InitSysTimer(void)
- パラメータ

  なし
- 機能詳細

  タイマモジュールの初期化を行います。
- 備考

  本関数は割込みが禁止された状態で実行してください。

4.1.2. タイマモジュールの停止

- 関数名

  void TermSysTimer(void)
- パラメータ

  なし
- 機能詳細

  タイマモジュールの停止処理を行います。
- 備考

  本関数は割込みが禁止された状態で実行してください。

4.1.3. システムタイマ割込みサービスルーチン

- 関数名

  ISR(SysTimerInt)
- パラメータ

  なし
- 機能詳細

  OS にシステム時刻を通知するためのカウンタ加算処理を実行します。
- 備考

  特になし。

4.2. シリアルI/Oモジュール

シリアル通信によるログの送信・コマンドの受信や数値データの文字列変換処理を行うための、シリアル I/O モジュールについて説明します。本モジュールでは次項以降に記載するミドルウェア API を提供しています。

4.2.1. シリアルI/Oモジュールの初期化

- 関数名

  void InitSerial(void)
- パラメータ

  なし
- 機能詳細

  シリアル I/O モジュールの初期化を行います。
- 備考

  本関数は割込みが禁止された状態で実行してください。

4.2.2. シリアルI/Oモジュールの停止

- 関数名

  void TermSerial(void)
- パラメータ

  なし
- 機能詳細

  シリアル I/O モジュールの停止処理を行います。
- 備考

  本関数は割込みが禁止された状態で実行してください。

4.2.3. シリアルI/O文字列送信処理

- 関数名

  void SendSerialStr(const UINT8 * str)
- パラメータ

  str　　(I)　　送信する文字列バッファへのポインタ
- 機能詳細

  文字列終端コードが見つかるまで、シリアル I/O デバイスドライバ文字送信処理を繰返し実行します。
- 備考

  本関数は割込みが許可された（禁止されていない）状態で実行してください。

  本関数はサンプルアプリケーション内では PutSysLog()というシンボルで使用されていま

す。実際にはマクロにて名前変換をしており、上記シンボルの実態は本関数となります。

4.2.4. 割込み禁止中シリアルI/O文字列送信処理

- 関数名

  void SendIntSerialStr(const UINT8 * str)

- パラメータ

  str　　(I)　　送信する文字列バッファへのポインタ

- 機能詳細

  文字列終端コードが見つかるまで、シリアル I/O デバイスドライバ文字送信処理を繰返し実行します。

- 備考

  本関数は割込みが禁止された状態で実行してください。

  本関数はサンプルアプリケーション内では PutIntSysLog()というシンボルで使用されています。実際にはマクロにて名前変換をしており、上記シンボルの実態は本関数となります。

4.2.5. シリアルI/O　1バイト受信処理

- 関数名

  void RecvPolSerialChar(UINT8 * character)

- パラメータ

  character　　(O)　　受信したデータを格納するバッファへのポインタ

- 機能詳細

  データを受信していれば、そのデータを引数で指したバッファに格納します。データを受信していなければ、'¥0'を引数で指したバッファに格納します。

- 備考

  特になし。

4.2.6. 32bitデータ 10 進文字列変換処理

- 関数名

  void ConvLong2DecStr(UINT8 *dst, UINT32 src)

- パラメータ

  dst　　(O)　　変換した文字列を格納するバッファへのポインタ

  src　　(I)　　変換する 32bit データ

- 機能詳細

  32bit のデータを 10 進数の文字列に変換します。

- 備考

  引数 dst に指定するバッファは 11Byte 以上の領域が確保されている必要があります。

4.2.7. 16bitデータ 10 進文字列変換処理

- 関数名

  void ConvShort2DecStr(UINT8 *dst, UINT16 src)

- パラメータ

  dst　(O)　変換した文字列を格納するバッファへのポインタ

  src　(I)　変換する 16bit データ

- 機能詳細

  16bit のデータを 10 進数の文字列に変換します。

- 備考

  引数 dst に指定するバッファは 6Byte 以上の領域が確保されている必要があります。

4.2.8. 8bitデータ 10 進文字列変換処理

- 関数名

  void ConvByte2DecStr(UINT8 *dst, UINT8 src)

- パラメータ

  dst　(O)　変換した文字列を格納するバッファへのポインタ

  src　(I)　変換する 8bit データ

- 機能詳細

  8bit のデータを 10 進数の文字列に変換します。

- 備考

  引数 dst に指定するバッファは 4Byte 以上の領域が確保されている必要があります。

4.2.9. 32bitデータ 16 進文字列変換処理

- 関数名

  void ConvLong2HexStr(UINT8 *dst, UINT32 src)

- パラメータ

  dst　(O)　変換した文字列を格納するバッファへのポインタ

  src　(I)　変換する 32bit データ

- 機能詳細

  32bit のデータを 16 進数の 8 桁 0 詰め文字列に変換します。

- 備考

  引数 dst に指定するバッファは 9Byte 以上の領域が確保されている必要があります。

4.2.10. 16bitデータ 16 進文字列変換処理

- 関数名

  void ConvShort2HexStr(UINT8 *dst, UINT16 src)

- パラメータ

dst　(O)　変換した文字列を格納するバッファへのポインタ
src　(I)　変換する 16bit データ

- 機能詳細

  16bit のデータを 16 進数の 4 桁 0 詰め文字列に変換します。

- 備考

  引数 dst に指定するバッファは 5Byte 以上の領域が確保されている必要があります。

4.2.11. 8bitデータ 16 進文字列変換処理

- 関数名

  void ConvByte2HexStr(UINT8 *dst, UINT8 src)

- パラメータ

  dst　(O)　変換した文字列を格納するバッファへのポインタ
  src　(I)　変換する 8bit データ

- 機能詳細

  8bit のデータを 16 進数の 2 桁 0 詰め文字列に変換します。

- 備考

  引数 dst に指定するバッファは 3Byte 以上の領域が確保されている必要があります。

4.3.　サンプルドライバモジュール

サンプルアプリケーションの各種動作を行うための、サンプルドライバモジュールについて説明します。本モジュールでは次項以降に記載するデバイスドライバ API を提供しています。

4.3.1.　アプリケーションモード判定情報提供ポートドライバ

サンプルアプリケーションにおける、アプリケーションモード判定情報入力ドライバの機能について説明します。

4.3.1.1.　アプリケーションモード情報取得処理

- 関数名

  UINT8 GetAppModeInfo(void)

- パラメータ

  戻り値（O)　　アプリケーションモード情報
  　　0～2

- 機能詳細

  ターゲットごとの入力装置より起動情報を読込み、0～2 の値を戻します。

- 備考

  詳細は「TOPPERS Automotive Kernel アプリケーションノート"ターゲットボード名"」を参照してください。

4.3.2. カーネルAPI「SignalCounter()」実行割込みドライバ

サンプルアプリケーションにおける、カーネル API「SignalCounter()」をテスト実行するための割込み制御ドライバの機能について説明します。

4.3.2.1. カウンタ ソース割込みの初期化処理

- 関数名

  void InitCounterInt(void)

- パラメータ

  なし

- 機能詳細

  カーネル API「SignalCounter()」を実行している割込み要因の初期化を行います。

- 備考

  本機能はターゲットシステムごとの実装依存です。

4.3.2.2. カウンタ ソース割込みの停止処理

- 関数名

  void TermCounterInt(void)

- パラメータ

  なし

- 機能詳細

  カーネル API「SignalCounter()」を実行している割込み要因の停止処理を行います。

- 備考

  本機能はターゲットシステムごとの実装依存です。

4.3.2.3. カウンタ ソース割込み稼動処理

- 関数名

  void ActCounterInt(void)

- パラメータ

  なし

- 機能詳細

  カーネル API「SignalCounter()」を実行している割込み要因の稼動処理を行います。

- 備考

  本機能はターゲットシステムごとの実装依存です。

4.3.2.4. シグナル通知実行割込みサービスルーチン

- 関数名

ISR(CounterInt)

- パラメータ

  なし

- 機能詳細

  割込み要因の停止処理と、カーネル API「SignalCounter()」を実行します。なお、カウンタ ID は「SampleCnt」です。

- 備考

  本機能はターゲットシステムごとの実装依存です。

4.3.3. 割込み制御機能確認用ハードウェアカウンタドライバ

サンプルアプリケーションにおける、割込み制御機能やタスク・割込み間のリソース管理機能をテストするための、ハードウェアカウンタドライバの機能について記載します。

4.3.3.1. ハードウェアカウンタバッファ

カテゴリ 1 割込みカウンタバッファとカテゴリ 2 割込みカウンタバッファを用意します。

4.3.3.2. ハードウェアカウンタの初期化処理

- 関数名

  void InitHwCntInt(void)

- パラメータ

  なし

- 機能詳細

  カウンタバッファの初期化と、カウント割込み要因の初期化を行います。

- 備考

  本機能はターゲットシステムごとの実装依存です。

4.3.3.3. ハードウェアカウンタの停止処理

- 関数名

  void TermHwCntInt(void)

- パラメータ

  なし

- 機能詳細

  カウント割込み要因の停止処理を行います。

- 備考

  本機能はターゲットシステムごとの実装依存です。

4.3.3.4. ハードウェアカウンタ値の取得処理

- 関数名

  void GetHwCnt(UINT8 *isr1_cnt,UINT8 *isr2_cnt)

- パラメータ

isr1_cnt	(O)	カテゴリ 1 割込みカウンタ値取得バッファへのポインタ
isr2_cnt	(O)	カテゴリ 2 割込みカウンタ値取得バッファへのポインタ

- 機能詳細

  カテゴリ 1 割込みカウンタバッファとカテゴリ 2 割込みカウンタバッファの値を、引数で渡されたバッファにそれぞれ格納します。

- 備考

  本機能はターゲットシステムごとの実装依存です。

4.3.3.5. カウンタ加算カテゴリ 1 割込みサービスルーチン

- 関数名

  void HwCnt1Int(void)

- パラメータ

  なし

- 機能詳細

  カテゴリ 1 割込みカウンタ値を加算します。

- 備考

  本機能はターゲットシステムごとの実装依存です。

4.3.3.6. カウンタ加算カテゴリ 2 割込みサービスルーチン

- 関数名

  ISR(HwCnt2Int)

- パラメータ

  なし

- 機能詳細

  カテゴリ 2 割込みカウンタ値を加算します。

- 備考

  本機能はターゲットシステムごとの実装依存です。

変更履歴

Version	Date	Detail	Editor
1.00	2006/04/18	・ 新規作成	ヴィッツ
1.10	2006/05/30	・ 構成変更 ・ サンプルライブラリ追記 ・ TOPPERS/OSEK カーネル(旧称)の階層追記 ・ TOPPERS ライセンスヘッダ追記	ヴィッツ
2.00	2006/05/30	・ TOPPERS/OSEK 公開用にバージョンアップ	ヴィッツ
2.01	2007/11/20	・ 誤記修正	ヴィッツ
3.00	2008/10/20	・ カーネル名称変更	ヴィッツ