

TOPPERS基礎実装セミナー

(LPC2388版:基本)

基礎2編:1日目

TOPPERSプロジェクト
教育ワーキング・グループ

2013/06/16

TOPPERSプロジェクト認定

1



本教材の利用条件

NEXCESS基礎コース01 組込みソフトウェア開発技術の基礎

Copyright (C) 2006-2007 by 名古屋大学 組込みソフトウェア技術者人材養成プログラム
Copyright (C) 2006-2007 by 本田晋也, 高田広章

上記著作権者は、以下の(1)~(4)の条件を満たす場合に限り、本コンテンツ(本コンテンツを改変・翻訳したものを含む、以下同じ)を使用・複製・改変・翻訳・再配布(以下、利用と呼ぶ)することを無償で許諾する。

- (1) この枠内の著作権表記等が、そのままの形でコンテンツ中に含まれていること。
- (2) 本コンテンツを再配布する場合には、再配布の形態等を、以下のウェブサイトから報告すること。
<http://www.nces.is.nagoya-u.ac.jp/NEXCESS/REPORT/>
- (3) 本コンテンツを改変・翻訳する場合には、コンテンツを改変・翻訳した旨の記述を、コンテンツ中に含めること。また、改変・翻訳者の著作権表記等は、この枠内の著作権表記等とは別に行うこと。
- (4) 本コンテンツの利用により直接的または間接的に生じるいかなる損害からも、上記著作権者を免責すること。

※ 本コンテンツの一部は、文部科学省 科学技術振興調整費により、名古屋大学 組込みソフトウェア技術者人材養成プログラム(NEXCESS)の一環として作成しました。

※ 本コンテンツ中に記載されている商品名やサービス名などは、各社の商標または登録商標です。

2013/06/16

TOPPERSプロジェクト認定

2



本ドキュメントに関して

1. 著作権に関する表記

＜TOPPERS基礎実装セミナー(LPC2388版:基本2)1日目＞

Copyright (C) 2006-2009 by 名古屋大学 組込みソフトウェア技術者人材養成プログラム

Copyright (C) 2006-2009 by 本田晋也、高田広章 名古屋大学

Copyright (C) 2008-2013 by 竹内良輔 (株)リコー

Copyright (C) 2013 by 森田浩

上記著作権者は、以下の(1)～(3)の条件を満たす場合に限り、本ドキュメント(本ドキュメントを改変したものを含む。以下同じ)を使用・複製・改変・再配布(以下、利用と呼ぶ)することを無償で許諾する。

(1) 本ドキュメントを利用する場合には、上記の著作権表示、この利用条件および以下の無保証規定が、そのままの形でドキュメント中に含まれていること。

(2) 本ドキュメントを改変する場合には、ドキュメントを改変した旨の記述を、改変後のドキュメント中に含めること。ただし、改変後のドキュメントがTOPPERSプロジェクト指定の開発成果物である場合には、この限りではない。

(3) 本ドキュメントの利用により直接的または間接的に生じるいかなる損害からも、上記著作権者およびTOPPERSプロジェクトを免責すること。また、本ドキュメントのユーザまたはエンドユーザからのいかなる理由に基づく請求からも、上記著作権者およびTOPPERSプロジェクトを免責すること。

本ドキュメントは、無保証で提供されているものである。上記著作権者およびTOPPERSプロジェクトは、本ドキュメントに関して、特定の使用目的に対する適合性も含めて、いかなる保障もしない。また、本ドキュメントの利用により直接的または間接的に生じたいかなる損害に関しても、その責任を負わない。

2. 本ドキュメントに関するご意見・ご提言・ご感想・ご質問等がありましたら、TOPPERSプロジェクト事務局までE-Mailにてご連絡ください。

3. 本ドキュメントの内容は、内容の改善や適正化の目的で予告無く改定することがあります。

本ドキュメントでは、Microsoft社のClip Art Galleryコンテンツを使用しています。

TRONは"The Real-time Operating system Nucleus"の略称です。ITRONは"Industrial TRON"の略称です。
μITRONは"Micro Industrial TRON"の略称です。TOPPERS/JSPはToyohashi Open Platform for Embedded Real-Time System/Just Standard Profile Kernelの略称です。」

本ドキュメント中の商品名及び商標名は、各社の商標または登録商標です。

2013/06/16

TOPPERSプロジェクト認定

3



スケジュール

■ 1日目

- | | |
|-------------------|-------|
| 1. リアルタイムOSの基礎 | 0.5時間 |
| 2. ITRONの仕様について学ぶ | 1.5時間 |
| 3. 開発環境の構築 | 0.5時間 |
| 4. TOPPERS/ASPの導入 | 1.5時間 |
| 5. システム検証モジュールの導入 | 2.0時間 |
| 6. まとめ | 0.5時間 |

2013/06/16

TOPPERSプロジェクト認定

4



概要

リアルタイムOSの基礎及び、 TOPPERS／ASPカーネルの導入と拡張を学ぶ

- リアルタイムOSの基礎
 - リアルタイムOSを用いた組込み開発
 - ITRON仕様
 - TOPPERS／ASPカーネル
- TOPPERS／ASPカーネルの導入
 - 開発環境の構築
 - サンプルプログラムのビルドと実行
- TOPPERS／ASPカーネルの拡張
 - システム検証モジュールの導入と改造

2013/06/16

TOPPERSプロジェクト認定

5



リアルタイムOSの基礎

1. [リアルタイムOSを用いた組込み開発](#)
2. リアルタイムOSのメリットデメリット
3. タスク間の同期と通信

2013/06/16

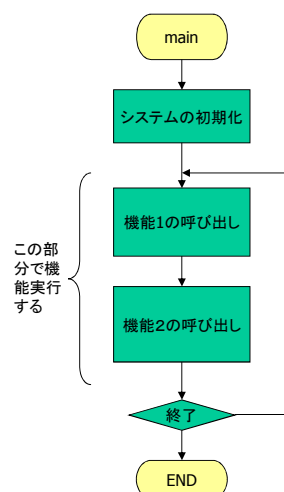
TOPPERSプロジェクト認定

6



小規模組込み開発の開発手法

- 小規模組込み開発では、main関数中のループ部からシステムの制御に必要な関数を周期的に呼び出すことにより、制御を行うのが一般的な方法です
- 1, 2人の開発者でワンチップCPUに収まるソースコード一万行程度までのプログラムならば、十分な作りこみが可能です
- しかし、システムの規模が大きくなると種々の問題が発生します



2013/06/16

TOPPERSプロジェクト認定

7



小規模組込み開発の限界

- 小規模システムでは小規模な機器管理が中心です。開発規模が大きくなると、複雑なUI部、通信機能、ファイルシステムの管理等種々のノウハウを持った開発者が分業してシステムを開発する必要があります
- 多くの機能を組み合わせた場合、main関数からの関数コールではハードリアルタイム性を保持するのが難しくなる
 - ある機能で100msの待ちが発生した場合、main関数からの関数呼び出しでは、その機能中でソフトウェアディレイで待つしかなく、不要なCPUの占有が発生する
 - ソフトウェアディレイ中に他の機能を動かそうとするとシステムが複雑化し、メンテナンス性が落ちる

➡ リアルタイムOSを用いれば問題解決可能

2013/06/16

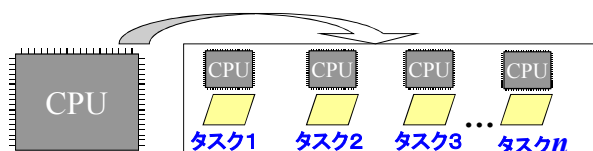
TOPPERSプロジェクト認定

8



中規模組込みシステムとリアルタイムOSの必要性

- リアルタイムOSを用いることにより、個々の機能を独立したタスクとして動作させることが可能となります
 - リアルタイムOSの大きな特徴はマルチタスク機能
- マルチタスク機能とは
 - 複数のタスクを擬似並列に実行するための機能
 - 1CPUのシステムでは、実際には同時に実行できるのは1つのタスクのみです
 - あたかも複数のタスクを同時に実行しているようにみせる機能です



2013/06/16

TOPPERSプロジェクト認定

9



中規模組込み開発とシステム管理者

- 中規模組込みシステムではプラットフォーム部を先行開発しサブシステムを追加していく方式が一般的です、また、商品化後継続的に機能アップを行う必要もあります
- システム管理者はプラットフォーム部とサブシステム間のアーキテクチャを決め、サブシステム間のインターフェイスを明確化する必要があります
- 分業開発では、予測外の問題で開発遅延が発生することが多々あります、これらの問題解決もシステム管理者の仕事です
 - 開発時、遅れたサブシステムをスタブ化してシステム開発の遅れを回避
 - 問題発生時、原因がどの部署に起因するかを解析

2013/06/16

TOPPERSプロジェクト認定

10



リアルタイムOSの内容を理解する技術者の必要性

- 中規模組込みシステムでは、リアルタイムOSに熟達した技術者を社内または社外の協力会社に用意する必要があります

システム開発中にアーキテクチャに関する問題解決が必要となるケースが多く発生する

- 中規模システムでは、汎用ミドルウェアの導入やデバイスドライバの導入が必要となります、この場合、リアルタイムOSに対して最適化を行う必要があります
- アプリケーションの障害でも、リアルタイムOS中でエクセプションが発生することがあります
 - ソースコードのないリアルタイムOSでは解析が困難
- 省エネ対応のためにリアルタイムOSの改造やSoC化によって何度も、検証用のデバイスドライバを書き換える必要があります

2013/06/16

TOPPERSプロジェクト認定

11



大規模組み込みシステムとの比較

- UNIXやWindowsの汎用OSを組込みシステムとして用いる場合とリアルタイムOSを用いる場合の比較を行いましょ
- 商品は品質、機能、コストを最適にすることにより商品性が向上します
- 汎用OSはシステムの安全性を向上させる点からは有用ですが、OSの実行管理のために多くのCPUパワーやメモリが必要となり性能やコストの面で問題があります
- 汎用OS自体の工数が多いため、それを最適化し、管理するためそれなりの技術者も必要となる(リアルタイムOSより複雑でノウハウは高い)

簡単な例として、パソコンの周辺機の管理に、パソコンと同等のシステムを用いて行うのは無理があります

2013/06/16

TOPPERSプロジェクト認定

12



リアルタイムOSの基礎

1. リアルタイムOSを用いた組込み開発
2. リアルタイムOSのメリットデメリット
3. タスク間の同期と通信

リアルタイムOSを利用するメリット

- ソフトウェアの構造化により生産性、保守性、信頼性が向上する(システム規模が大きくなると重要な問題となる)
ソフトウェアの構造化 $\rightleftharpoons$ **モジュール化**
- 時間制約を持った多重処理システムの構築が容易になる
 - 論理的な処理順序と時間的な処理順序の分離により、時間制約を持ったソフトウェアの保守性・再利用性が向上
ソフトウェア部品を用いたソフトウェア開発 (コンポーネントベース開発)の基盤
- ハードウェア(特にプロセッサ)の違いを隠蔽できる
 - ハードウェアの細部を知らない技術者でも、アプリケーションが構築できる

リアルタイムOSを使用するデメリット

- リアルタイムOS自身にオーバーヘッドがある
 - オーバーヘッドによる実行性能の低下
 - RTOS自身のワークエリアによりメモリ消費
- マルチタスク機構による問題点
 - タスクのスタックエリアによるメモリ消費
 - 非決定性が増すことによるデバッグ・テストの困難化
行儀のよいタスク間インターフェイス設計が必要
- リアルタイムOSのブラックボックス化により、問題発生時の解析が困難となる
 - リアルタイムOSに対応したツールの利用で改善は可能
 - リアルタイムOSの原理、内部構造を知る技術者は必要

リアルタイムOSの基礎

1. リアルタイムOSを用いた組込み開発
2. リアルタイムOSのメリットデメリット
3. タスク間の同期と通信

タスク間の通信と同期

- タスク間の通信・同期の必要性
 - タスクが協調して動作するためには、タスク間でデータをやりとりすること(=タスク間通信)が必要
 - タスク間通信の際には、タスク同士で動作タイミングをあわせること(=タスク間同期)が必要
 - 複数のタスクが同一の資源を取り合う場合にも、タスク間の同期が必要
- タスク間通信・同期のタイプ

タスクを仮想化されたプロセッサと捉えるなら、タスク間の通信・同期はプロセッサ間の通信・同期を仮想化したもの

共有メモリによる通信 or メッセージによる通信

共有メモリによる通信

- 複数のタスクからアクセスできるメモリ領域上に受け渡しするデータ(共有データ)置く
- C言語のグローバル変数による実現
 - 共有データを読み書きする際には、RTOSの機能を用いて、排他制御することが必要
 - 割込み/ディスパッチの禁止
 - セマフォ/ミューテックス
 - ⚠ デッドロックと優先度逆転に注意
 - タスクの優先度の変更
 - 共有データを速やかに処理させたい場合には、事象の発生を知らせる機能を用いる
 - タスクの起動/起床
 - イベントフラグ/Condition Variable(条件変数)

メッセージによる通信

RTOSが持つメッセージ通信のための機能を用いて、
タスク間でメッセージを受け渡す

- メッセージ通信機能のタイプ
 - コネクションあり or なし、単方向 or 双方向、
1対1(送信できるタスクが固定) or n対1 or n対n
 - 通信相手の指定方法
 - 通信オブジェクトを指定 or 相手タスクを指定 or ……
 - 同期メッセージ通信 or 非同期メッセージ通信
 - (非同期通信で)バッファにメッセージが無い場合の振る舞い
 - 待つ(ブロッキング) or 待たない(ノンブロッキング)
 - (非同期通信で)バッファがフルの時の振る舞い
 - 待つ(ブロッキング) or 待たない(ノンブロッキング)
or 上書きする

2013/06/16

TOPPERSプロジェクト認定

19



メッセージ通信機能のタイプ(続き)

- メッセージが固定長 or 可変長(任意長)
- (可変長の時に)パケットの単位がある or ない
- ポインタを渡す or コピーする
- (非同期通信で)メッセージのキューイング順序
 - FIFO順 or 優先度順
- 受信/送信待ちタスクのキューイング順序
 - FIFO順 or 優先度順
- その他特殊な機能
 - マルチキャスト/ブロードキャスト
 - 状態メッセージ(OSEK/VDK仕様の unqueued message)

RTOSの持つメッセージ通信機能(複数の機能を持つ
場合も多い)の性質をよく理解することが必要

2013/06/16

TOPPERSプロジェクト認定

20



タスク間通信・同期の設計

- 共有メモリ vs. メッセージ通信
 - 基本的には、一方で実現できることは、もう片方でも実現できる
 - 一般には共有メモリの方がオーバーヘッドが小さい
 - システム検証には、通信の粒度が大きくRTOSレベルで監視できるメッセージ通信の方が扱いやすい
 - 例えば、問題の切り分けが容易
 - 状況により、どちらかが有利/便利な状況がある
 - 情報の流れが 1:n の場合（ブロードキャストを含む）や、情報が必要なタイミングが不定の場合には、共有メモリが有利
 - 情報のキューイングが必要な時には、メッセージ通信が便利

2013/06/16

TOPPERSプロジェクト認定

21



ITRONの基礎知識

1. [ITRON仕様](#)
2. ITRONの同期・通信機能
3. TOPPERS/ASPカーネル

2013/06/16

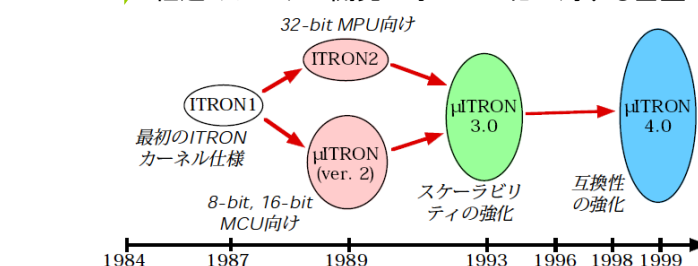
TOPPERSプロジェクト認定

22



ITRON仕様

- ITRONプロジェクト
 - TRONプロジェクトのサブプロジェクトの1つ
 - ITRON仕様
 - ITRONプロジェクトで策定されたリアルタイムカーネル仕様
 - これまでに4世代のITRON仕様に策定・公表
 - 現世代のリアルタイムカーネル技術の範囲では、完成度の高い仕様に
- ➡ 組込みシステム開発のオープン化に対する基盤



2013/06/16

TOPPERSプロジェクト認定

23



ITRON仕様の特徴

- OSの小型軽量化が可能
 - ワンチップマイコンにも適用可能
- 仕様の理解が容易
 - 技術者教育のための標準化の側面を重視
- 完全にオープンな標準仕様
 - ロイヤリティなしで実装することができる
- 多種多様なプロセッサ用に実装できる/されている
 - 8bitワンチップマイコンから32bit RISCマイコンまで
 - 異なるプロセッサへの移行が容易に
- 多くの機器で使用実績がある
 - 組込みシステム分野で最も広く使われているOS仕様
- 多くのメーカ/ベンダがサポート



2013/06/16

TOPPERSプロジェクト認定

24



ITRON仕様のカーネルの機能(μ ITRON4.0仕様)

- カーネルの機能
 - タスク管理機能
 - タスク付属同期機能
 - タスク例外処理機能
 - 同期・通信機能
 - 拡張同期・通信機能
 - メモリプール機能
 - 時間管理機能
 - システム状態管理機能
 - 割り込み管理機能
 - サービスコール管理機能
- サービスコール数
 - フルセット
 - サービスコール : 166
 - 静的API : 21
 - スタンダードプロファイル
 - サービスコール : 70
 - 静的API : 11
 - 自動車制御用プロファイル
 - サービスコール : 43
 - 静的API : 8
 - 最小セット



! I/O操作のための機能は規定されていない

2013/06/16

TOPPERSプロジェクト認定

25



ITRON仕様のサービスコール(API)

- サービスコール名称のガイドライン
 - 例) `act_tsk`
 - `act` 操作対象を表す. この場合はタスク(task)
 - `tsk` 操作方法を表す. この場合は起動(activate)
 - サービスコールが成功するとE_OKが戻り値として返される
- 割り込みハンドラ(厳密には)から呼び出せるサービスコールは"i"で始まる名称として区別
 - 例) `iact_tsk`
- 割り込みハンドラから呼び出せないAPIがある
 - 待ち状態に入るAPI

2013/06/16

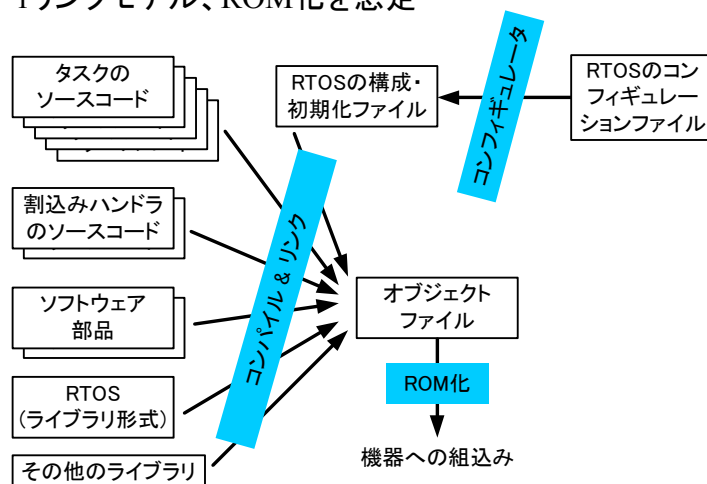
TOPPERSプロジェクト認定

26



RTOSを用いた組み込みソフトウェアの構築方法

- 1リンクモデル、ROM化を想定



2013/06/16

TOPPERSプロジェクト認定

27



RTOSのコンフィギュレーション

- RTOSの構成や初期状態を決める手順
- どのような構成が変更できるかはRTOS毎. 例えば、
 - 用いる機能/用いない機能
 - 各オブジェクト(タスクやセマフォ)などの最大値
 - タスク優先度の段数
 - **各オブジェクトの生成・定義情報**
(オブジェクトを静的に生成する場合)
 - コンフィギュレーション機構
 - コンフィギュレーションによりRTOSのコードを変える方法(条件コンパイルなどを使う)
 - コンフィギュレーション結果より、1つ(または複数)のソースコードに生成する方法

2013/06/16

TOPPERSプロジェクト認定

28



μITRON4.0仕様におけるコンフィギュレーション

- カーネルオブジェクトは静的に生成
 - オブジェクト生成情報の、コンフィギュレーションファイルの中での記述方法を標準化(静的API)
- 静的APIの例

```
CRE_TSK(tskid, {tskatr, exinf, task, itskpri, stksz, stk});
DEF_INH(inhno, {inhatr, inthdr});
```

- コンフィギュレータは静的APIを解析して、カーネルの内部情報を生成する

2013/06/16

TOPPERSプロジェクト認定

29



静的APIの詳細

CRE_TSK	タスク生成(静的API)	【スタンダード】
cre_tsk	タスク生成	【スタンダード外】

【静的API】

```
CRE_TSK(ID tskid, {ATR tskatr, VP_INT exinf, FP task,
                  PRI itskpri, SIZE, stksz, VP stk})
```

【C言語API】

```
ER ercd = cre_tsk(ID tskid, T_CTSK *pk_ctsk)
```

【パラメータ】

ID	tskid	生成対象のタスクのID番号
T_CTSK	*pk_ctsk	タスク生成情報を入れたパケット
ATR	tskatr	タスク属性(記述言語、初期状態)
VP_INT	exinf	タスクの拡張情報(タスクへのパラメータ)
FP	task	タスクの起動番地
PRI	itskpri	タスクの起動優先度
SIZE	stksz	タスクのスタック領域のサイズ(バイト数)
VP	stk	タスクのスタック領域の先頭番地

2013/06/16

TOPPERSプロジェクト認定

30



マルチタスク機構

- スケジューリング方式
 - μITRON4.0では、プリエンプティブな優先度ベーススケジューリングによるマルチタスク機構をサポート
 - 同優先度のタスクはFCFS (First Come First Served) 方式でスケジューリング. 同優先度のタスク内での実行順序を変更する機能を用意. これを用いて、同優先度内でのラウンドロビンスケジューリングも実現可能
 - 優先度付けは静的が基本だが、優先度を動的に変更する機能も用意
- タスク状態
 - 実行、実行可能、休止、待ち、強制待ち、二重待ち、未登録の7つの状態をサポート

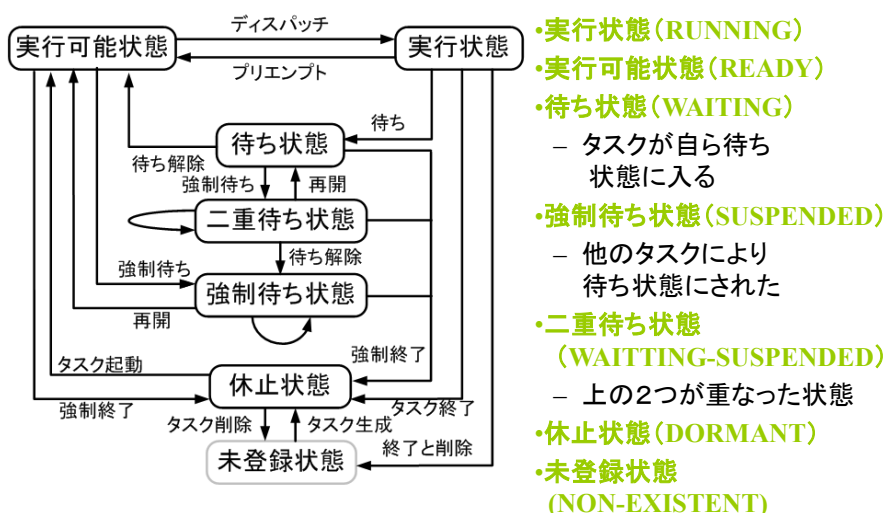
2013/06/16

TOPPERSプロジェクト認定

31



タスクの状態遷移



2013/06/16

TOPPERSプロジェクト認定

32



タスク管理機能

タスクの状態を直接的に操作するための機能

- タスク管理機能のサービスコール

cre_tsk	タスクの生成
del_tsk	タスクの削除
act_tsk, iact_tsk	タスクの起動
can_act	タスクの起動要求の無効化
ext_tsk	自タスクの終了
ter_tsk	タスクの強制終了
chg_pri	タスクの優先度の変更
get_pri	タスクの優先度の参照
ref_tsk	タスクの状態参照

- タスクの起動要求 (act_tsk) のキューイングとその無効化 (can_act)

2013/06/16

TOPPERSプロジェクト認定

33



タスク付属同期機能

タスクを直接操作して同期を実現するための機能

- タスク付属同期機能のサービスコール

slp_tsk, tslp_tsk	自タスクを起床待ち状態に
wup_tsk, iwup_tsk	他タスクの起床
can_wup	起床要求の無効化
rel_wai, irel_wai	タスクの待ち状態の強制解除
sus_tsk	タスクを強制待ちに
rsm_tsk, frsm_tsk	タスクを強制待ちからの再開
dly_tsk	自タスクを遅延

- タスクの起床要求 (wup_tsk) のキューイングとその無効化 (can_wup)
- タイムアウト付きの起床待ち (tslp_tsk) と自タスクの遅延 (dly_tsk)

2013/06/16

TOPPERSプロジェクト認定

34



時間管理機能：概要

- システム時刻管理
 - システム時刻(カーネルが管理する現在時刻)を管理するための機能
- タイムイベントハンドラ
 - 時間の経過をきっかけに呼び出されるルーチン
 - μ ITRON4.0仕様では以下のタイマハンドラを用意
 - 周期ハンドラ(周期的に呼ばれる)
 - アラームハンドラ(指定した時刻に呼ばれる)
 - オーバランハンドラ(オーバラン時に呼ばれる)
- 時間同期のためのその他の機能
 - タスクの遅延(タスクを一定時間待たせる)
 - 待ちに入るサービスコールのタイムアウト機能

2013/06/16

TOPPERSプロジェクト認定

35



時間管理機能：サービスコール

- システム時刻管理のためのサービスコール

set_tim	システムクロックの設定
get_tim	システムクロックの読み出し
isig_tim	タイムティック供給

- 周期ハンドラ操作のためのサービスコール

cre_cyc	周期ハンドラ生成
del_cyc	周期ハンドラの削除
sta_cyc	周期ハンドラの起動
stp_cyc	周期ハンドラ停止

2013/06/16

TOPPERSプロジェクト認定

36



システム状態管理機能

システム状態を参照/変更するための機能

- システム状態管理機能のサービスコール

rot_rdq, irot_rdq	タスクの実行順序の回転
get_tid,iget_tid	実行状態のタスクのIDの取り出し
loc_cpu, iloc_cpu	CPUロック状態に
unl_cpu_iunl_cpu	CPUロック状態の解除
dis_dsp	ディスパッチ禁止
ena_dsp	ディスパッチ許可
sns_ctx	非タスクコンテキスト実行中か?
sns_loc	CPUロック状態か?
sns_dsp	ディスパッチ禁止状態か?
sns_dpn	ディスパッチ保留状態か?

2013/06/16

TOPPERSプロジェクト認定

37



割込み処理と割込み管理機能

- 割込み処理 (外部割込み)

- μITRON仕様では、割込みハンドラをアプリケーション側で記述できる
- 割込みが発生すると、カーネル内の出入り口処理を経由して、割込みハンドラが呼び出される
- 割込みハンドラの中でサービスコールを呼び出して、タスクの起動/起床などが可能
- 割込みハンドラ処理はタスクよりも優先
 - ディスパッチが遅延する

- 割込み管理機能のサービスコール

def_inh	割込みハンドラの定義
---------	------------

- この他に、割込みを個別に禁止/許可する機能など

2013/06/16

TOPPERSプロジェクト認定

38



システム構成管理機能

- プロセッサレベルの例外処理 (CPU例外)
 - 割込みと類似
 - CPU例外が発生すると、カーネル内の出入口処理を経由して、CPU例外ハンドラが呼び出される
 - CPU例外ハンドラの中でサービスコールを呼び出して、タスクに対して例外を発生させることが可能
 - CPU例外ハンドラの処理はタスクよりも優先
- ⚠ タスク例外との違いに注意
- 初期化ルーチンの定義
 - 初期化ルーチンは、システム初期化時に実行される

- システム構成管理機能のサービスコール

def_exc	CPU例外ハンドラの定義
ATT_INI	初期化ルーチンの追加

2013/06/16

TOPPERSプロジェクト認定

39



タスク例外処理

- タスクに対する割込み/例外に対応 (仮想化された割込み)
- 明示的なサービスコール呼び出しにより、タスクに例外を発生させる
- 次にそのタスクが実行されるタイミングで、タスク例外処理ルーチンが呼び出される
- UNIXのシグナル処理/シグナルハンドラに相当
- タスク例外処理サービスコール

def_tex	タスク例外処理ルーチンの定義
ras_tex, iras_tex	タスク例外処理の要求
dis_tex	タスク例外処理の禁止
ena_tex	タスク例外処理の許可
sns_tex	タスク例外処理禁止状態か?

2013/06/16

TOPPERSプロジェクト認定

40



ITRONの基礎知識

1. ITRON仕様
2. [ITRONの同期・通信機能](#)
3. TOPPERS/ASPカーネル

2013/06/16

TOPPERSプロジェクト認定

41



ITRONの同期・通信機能

μITRON4.0では、タスク間同期・通信のために
以下の機能を用意

- (主に) 共有メモリによる通信のための機能
 - [セマフォ*](#) (排他制御など)
 - [イベントフラグ*](#) (事象通知)
 - ミューテックス (排他制御)
- メッセージ通信のための機能
 - [データキュー*](#) (同期・非同期、1ワードのメッセージ)
 - [メールボックス*](#) (非同期、ポインタを送受信)
 - メッセージバッファ (同期・非同期、可変長メッセージ)
 - ランデブ (同期、双方向に通信、可変長メッセージ)

*スタンダードプロファイルに含まれる機能

2013/06/16

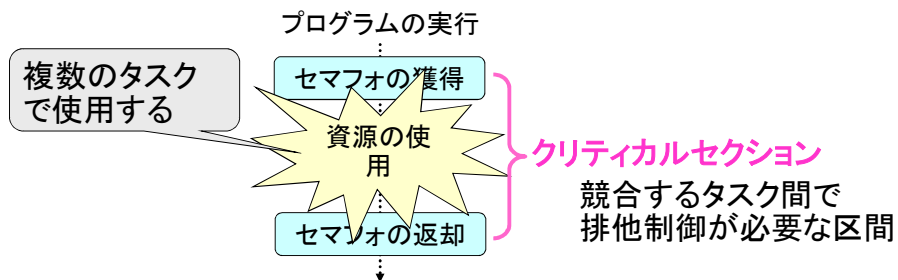
TOPPERSプロジェクト認定

42



セマフォ(Semaphore)

- 使用されていない資源の有無や数量をセマフォの資源数として管理することにより排他制御を実現
- 資源を使用する前にセマフォ資源を獲得し、資源を使用した後にセマフォ資源を返却する。
- 資源が全て使用されていれば、セマフォ資源獲得の時点で待ち状態になる。



2013/06/16

TOPPERSプロジェクト認定

43



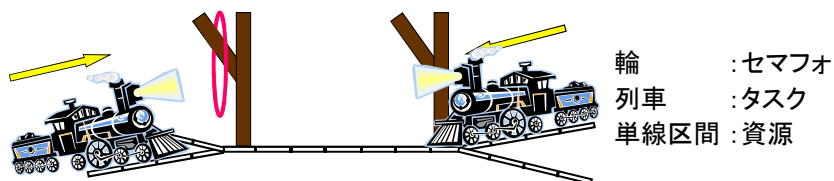
セマフォ(Semaphore) : サービスコール&語源

• サービスコール

cre_sem	セマフォの生成
del_sem	セマフォの削除
sig_sem, isig_sem	セマフォ資源の返却
wai_sem, pol_sem, twai_sem	セマフォ資源獲得

• 語源

鉄道の単線区間で進入制御に用いていた「輪」。
単線区間に入る場合は、この輪(セマフォ)を取る



2013/06/16

TOPPERSプロジェクト認定

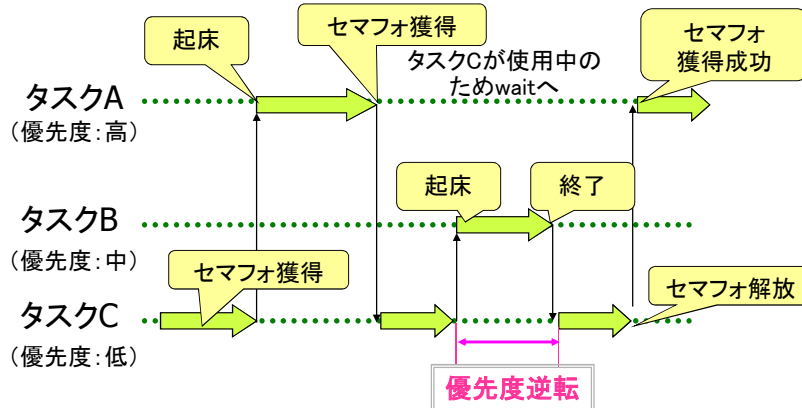
44



セマフォ(Semaphore): 優先度逆転

- 高優先度のタスクが低優先度のタスクの実行により待たされること

例) 高優先度のタスクAがそれより低優先度のタスクBの実行により待たされる。(注 タスクCに待たされるのは本質的)



2013/06/16

TOPPERSプロジェクト認定

45

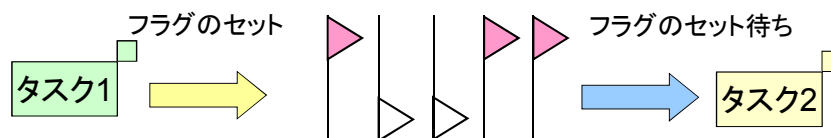


イベントフラグ(EventFlag)

- タスク間でイベントの発生を伝えることで同期するための機構. 1つのイベントフラグは、複数のフラグで構成

- サービスコール

cre_flg	イベントフラグの生成
del_flg	イベントフラグの削除
set_flg, iset_flg	イベントフラグのセット
clr_flg	イベントフラグのクリア
wai_flg, pol_flg, twai_flg	イベントフラグ待ち



2013/06/16

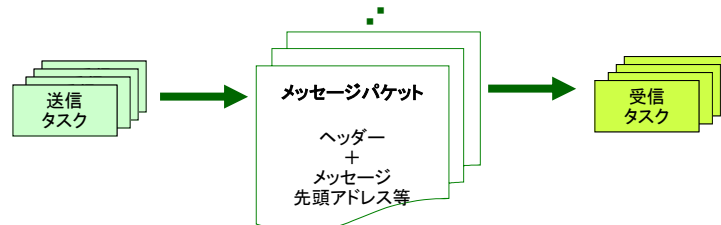
TOPPERSプロジェクト認定

46

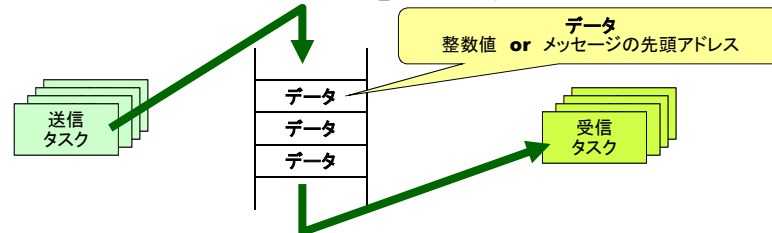


メールボックス と データキュー: 概要

- メールボックス：共有メモリ上に置かれたデータをやり取りする



- データキュー：1ワードのメッセージをやり取りする



2013/06/16

TOPPERSプロジェクト認定

47

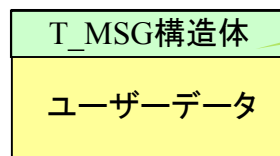


メールボックス (Mail Box)

- 可変長メッセージの非同期メッセージ通信機構
 - カーネルがリンクに用いるための領域を、メッセージの先頭にアプリケーション側で用意
- サービスコール

cre_mbx	メールボックスの生成
del_mbx	メールボックスの削除
snd_mbx	メールボックスへの送信
rcv_mbx, prcv_mbx, trcv_mbx	メールボックスからの受信

受け渡す
メッセージ



2013/06/16

TOPPERSプロジェクト認定

48



データキュー (Data Queue)

- 1ワードメッセージの非同期メッセージ通信機構
- メッセージは, 内部バッファにコピーされる
- メッセージが無い/フルの時は, 待ち合わせる
- データキュー領域のサイズを0に設定すると, 同期メッセージ通信も実現可能
- サービスコール

cre_dtq	データキューの生成
del_dtq	データキューの削除
snd_dtq, psnd_dtq, tsnd_dtq, isnd_dtq	データキューへの送信
rev_dtq, prev_dtq, trev_dtq	データキューからの受信
fsnd_dtq, ifsnd_dtq	データキューへの上書き送信

まとめ

- ITRONの同期・通信機能
 - セマフォ
 - イベントフラグ
 - メールボックス
 - データキュー
- それぞれの機能の性質を理解して, 使用目的に最適な機能を選択することが重要.
- 細かいパラメータの設定で, 振る舞いが変わるので, プログラミングの際には注意する.

ITRONの基礎知識

1. ITRON仕様
2. ITRONの同期・通信機能
3. [TOPPERS/ASPカーネル](#)



TOPPERS/JSPカーネルについて

- JSPとは
 - JSP = **J**ust **S**tandard **P**rofile
 - TOPPERSプロジェクトで開発されているμITRON4.0仕様のスタンダードプロファイルに準拠した**オープンソース**のリアルタイムOS.
最新版は Release 1.4.3
- 開発の目的
 - μITRON4.0仕様の評価、リファレンス実装
 - **研究・教育機関における研究・教育のプラットフォーム**
 - ソフトウェア部品 (IP) 開発のプラットフォーム
 - 評価目的・プロトタイプ開発への利用
 - 実製品への適用
- ターゲット環境
 - SH1/2, SH3, H8, ARMv4, MIPS, PowerPC



TOPPERS/JSPカーネルの特徴

- 読みやすく改造しやすいソースコード
 - 定量的な評価は難しいが、読みやすさには自身あり
 - 可能な限り #ifdef を追放
 - 様々な改造・拡張の実績あり
- 他のターゲットへのポーティングが容易な構造
 - 実行性能を落とさないプロセッサの抽象化
 - 新しいプロセッサへのポーティングを2日間で行った例
- 高い実行性能と小さいRAM使用量
 - **!** 大部分をC言語で記述したカーネルとしては
- LinuxおよびWindows上でのシミュレーション環境
 - プロトタイプ開発、教育用に最適
- オープンソースソフトウェアのみで開発環境まで

2013/06/16

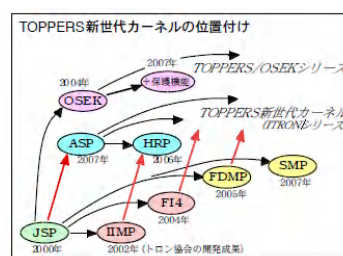
TOPPERSプロジェクト認定

53

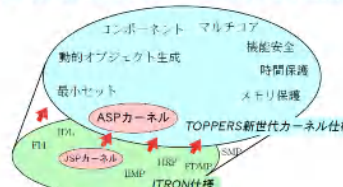


TOPPERS/ASPカーネルについて

- これまでの経験を反映して、次世代リアルタイムOSの普及版の開発を行う
- 開発方針
 - μITRON4.0仕様をベースに拡張・改良を行う。
 - ソフトウェアの再利用性を重視する
 - 高信頼性・安全なシステム構築を支援する
 - アプリケーションシステム構築に必要な機能は積極的に取り組む



TOPPERS新世代カーネル ~ ITRONからの継承



2013/06/16

TOPPERSプロジェクト認定

54



TOPPERS/ASPカーネルの概要

- 仕様の概要

TOPPERS/JSPカーネル(μITRON4. 0仕様スタンダードプロファイル準拠)に対して、信頼性・安全性・ソフトウェアポータビリティ向上のための各種の拡張・改良

- μITRON4. 0仕様からの主な拡張・改良点

- ・割込み処理機能を「TOPPERS準拠割込み処理モデル」によりプロセッサによらず標準化
- ・いくつかの独自の拡張機能(同期・通信オブジェクトの再初期化、優先度データキューなど)
- ・システムコンフィギュレーションの仕組みの見直し
- ・信頼性・安全性の向上については細かな改良の積み重ね
- ・TOPPERS組込みコンポーネント仕様の導入(将来拡張)

- μITRON4. 0仕様の上位互換ではない

2013/06/16

TOPPERSプロジェクト認定

55



TOPPERS/ASP仕様拡張の詳細1

- スタンダードプロファイル外の機能の一部導入

- イベントハンドラに複数待ち機能(TA_WMUL)
- アラームハンドラ
- 割込みサービスルーチン
- 割込み管理機能
- オブジェクトの状態参照機能

- μITRON4. 0仕様に対する変更

- ITRON標準データの見直し
- 非タスクコンテキストからのext_tsk
- CPU例外ハンドラで行える操作
- カーネルの用いる管理領域の分離
- 処理単位とメモリ領域のデータ型の見直し
- 値がゼロの定数(オブジェクト属性等)の見直し
- 強制待ち要求ネストの廃止
- システム時刻の設定機能(set_tim)の廃止

2013/06/16

TOPPERSプロジェクト認定

56



TOPPERS/ASP仕様拡張の詳細2

- JSPカーネルにおける独自の拡張機能
 - 性能評価用システム時刻参照機能(get_utm)
 - 終了処理ルーチン機能(ATT_TER)
 - カーネル動作状態の参照(sns_ker)
- ASPカーネルにおける独自の拡張
 - 割込み要求ラインの属性の設定(CFG_INT)
 - 同期・通信オブジェクトの再初期化機能
 - 優先度付きデータキューの新設
 - 自タスクの拡張情報の参照(get_inf)
 - カーネルの終了(ext_ker)
 - 非タスクコンテキスト用のスタック領域の設定(DEF_ICS)



TOPPERS/ASP仕様拡張の詳細3

- JSPにおける実装定義／実装依存部規定からの変更
 - アプリケーション向けのインクルードファイルの構成の整理
 - 割込み処理／例外関連の型の定義変更
 - 処理単位の実行開始／リターン時のシステム状態の規定
 - iset_timの廃止
 - カーネルの用いる領域の指定方法
 - カーネル管理外の割込みの扱いの規定
- システムコンフィギュレーション処理の見直し
 - システムコンフィギュレーション処理を全面的に見直した
 - 自動生成される標準的なファイルは以下の2つとなった
 - kernel_cfg.c:カーネル構成初期化ファイル
 - kernel_cfg.h:カーネル構成ヘッダファイル



開発環境の構築

ボードの確認とソフトウェアのセットアップ

- 開発環境のセットアップ手順は「基礎1:1日目」の以下の章に記載されています。これを参考に開発環境のセットアップを行ってください。
 - 「開発環境の確認」の
 - 1. ハードウェア・ソフトウェア環境の構築



TOPPERS/ASPカーネルの導入

1. [実習内容の説明](#)
2. 開発環境の構築
3. ビルドと実行
4. サンプルプログラムの確認



ITRON導入実習 : 実習内容

- TOPPERS/ASPカーネルの導入実習を行う
 - TOPPERS/ASPカーネルについての説明
 - カーネルのダウンロードと環境づくり
 - ダウンロードとツールを作成します
 - 開発・実行環境の使用方法についての説明
 - 開発環境と2種類の実行環境の使用方法について解説
 - カーネルのビルドと実行(デバグ使用版)
 - LPC2388用カーネルをビルドし、サンプルプログラムを実行します
 - サンプルプログラムの説明
 - サンプルプログラムの説明と実行確認



TOPPERS/ASPカーネルの導入

1. 実習内容の説明
2. 開発環境の構築
3. ビルドと実行
4. サンプルプログラムの確認

2013/06/16

TOPPERSプロジェクト認定

63



TOPPERS/ASPのダウンロード

- TOPPERSプロジェクトのWebからasp-1.7.0の個別パッケージをダウンロードします
- 同様にARMアーキテクチャ・GCC依存部パッケージ asp_arch_arm_gcc-1.7.0をダウンロードします

TOPPERS/ASPカーネル 個別パッケージのダウンロード

TOPPERS/ASPカーネルの最新リリースの個別パッケージを配布しています。各ターゲットカーネル非依存部パッケージ、依存部パッケージ(アーキテクチャ依存部、ターゲットビルドライブラリ依存部(BDNC))パッケージを個別にダウンロードできます。

ターゲットごとに必要なソースコードを一つにまとめた簡易パッケージは、こちらからります。

一般公開以前のバージョン(Release 1.3.0以前)に対応した個別パッケージは、当頁として提供していますが、こちらからダウンロードできます。

ターゲット非依存部

TOPPERS/ASPカーネル ターゲット非依存部パッケージ(担当:名古屋大学)

パッケージ	リリース日
asp-1.8.0.tar.gz	2012-12-26
asp-1.7.0.tar.gz	2011-05-09

ARMアーキテクチャ・GCC依存部パッケージ (担当:名古屋大学)			
パッケージ	ディレクトリ	対応する非依存部のバージョン	リリース日
asp_arch_arm_gcc-1.7.0.tar.gz	arch/arm_gcc/common target/at91skyeye_gcc	1.7.*	2011-05-19
asp_arch_arm_gcc-1.4.0.tar.gz	arch/arm_gcc/common arch/arm_gcc/at91sam7s target/at91skyeye_gcc target/bic090_gcc	1.4.*	2009-05-18
asp_arch_arm_gcc-1.3.2.tar.gz	arch/arm_gcc/common arch/arm_gcc/at91sam7s target/at91skyeye_gcc target/bic090_gcc	1.3.*	2008-08-27

2013/06/16

TOPPERSプロジェクト認定

64



TOPPERS/ASPの解凍

- 開発用のディレクトリにasp-1.7.0.tar.gz、asp_arch_gcc-1.7.0.tar.gzと、添付の開発環境asp-1.7.0-030913.tar.gzを置きます
- asp-1.7.0.tar.gzとasp_arch_arm_gcc-1.7.0.tar.gzの解凍後、解凍されたaspディレクトリ名をasp-1.7.0に変更します
- その後、asp-1.7.0-030913.tar.gzを解凍します

```
$ ls
asp_arch_arm_gcc-1.7.0.tar.gz asp-1.7.0.tar.gz asp-1.7.0-070812.tar.gz
$ tar zxvf asp-1.7.0.tar.gz
$ tar zxvf asp_arch_arm_gcc-1.7.0.tar.gz
$ mv asp-1.7.0-
$ tar zxvf asp-1.7.0-030913.tar.gz
```

2013/06/16

TOPPERSプロジェクト認定

65



コンフィギュレータの構築

- Webより、コンフィギュレータをダウンロードしasp-1.7.0フォルダの下に置きます
- LINUXやCygwinでは、cfgの下でMakefileを用いてコンフィギュレータのビルドを行う
- コンフィギュレータの構築はCygwinのバージョンにも関係するため、構築済みの実行形式を利用してよい

```
$ tar zxvf cfg-1.8.0.tar.gz
$ cd cfg
$ ./configure
$ make
```

TOPPERS新世代カーネル用コンフィギュレータ

TOPPERS新世代カーネル用コンフィギュレータの最新リリースを配布しています。コンフィギュレータ自体についてはこちらを、構築方法等についてはアーカイブに含まれる README.txtをご覧ください。

最新のリリース			
リリース名	タイプ	サイズ	リリース日
コンフィギュレータ Release 1.9.1	tar.gz(EUC-LF)	139KB	2013-04-05
コンフィギュレータ Release 1.9.1(Windows用バイナリ)	zip	10.8MB	2013-04-05
Release 1.9.0Cの通り			

過去のリリース			
リリース名	タイプ	サイズ	リリース日
コンフィギュレータ Release 1.8.0	tar.gz(EUC-LF)	139KB	2012-12-26
コンフィギュレータ Release 1.8.0(Windows用バイナリ)	zip	10.7MB	2012-12-26
Release 1.8.0Cの通り			
コンフィギュレータ Release 1.8.0	tar.gz(EUC-LF)	91KB	2012-05-09
コンフィギュレータ Release 1.8.0(Windows用バイナリ)	zip	1.0MB	2012-05-09
コンフィギュレータ Release 1.8.0(Cygwin用バイナリ)	tar.gz	450KB	2012-05-09

2013/06/16

TOPPERSプロジェクト認定

66



TOPPERS/ASPのワークスペースの作成

- asp-1.7.0ディレクトリの下にOBJ/LPC2388/SAMPLE1ディレクトリを作成します
- 教材では基礎3講座用のSAMPLE1のプログラムが用意されていますが、解説のため、これを削除し再生成します
- SAMPLE1ディレクトリに入り、configureコマンドを使用してSAMPLE1プログラムを生成します
- 以下のファイルが生成されます
 - Makefile, sample1.cfg, sample1.c, sample1.h

```
$ cd ../OBJ/LPC2388_GCC/SAMPLE1
$ rm *
$ ../../configure -T lpc2388_gcc
$ ls
Makefile sample1.c sample1.h
```

2013/06/16

TOPPERSプロジェクト認定

67



TOPPERS/ASPカーネルの導入

1. 実習内容の説明
2. 開発環境の構築
- [3. ビルドと実行](#)
4. サンプルプログラムの確認

2013/06/16

TOPPERSプロジェクト認定

68



開発環境の確認

- 基礎1講座で、ROMモニタをFlash Magicを使って書き込みました
- LPC2388ボードをUSBシリアルポートをUART0に接続
- TeraTermを以下の設定で起動します
 - COMn:nはUSBシリアルCOM番号
 - 38600bps
 - データ8bit
 - パリティなし
 - ストップビット1
 - フロー制御なし

2013/06/16

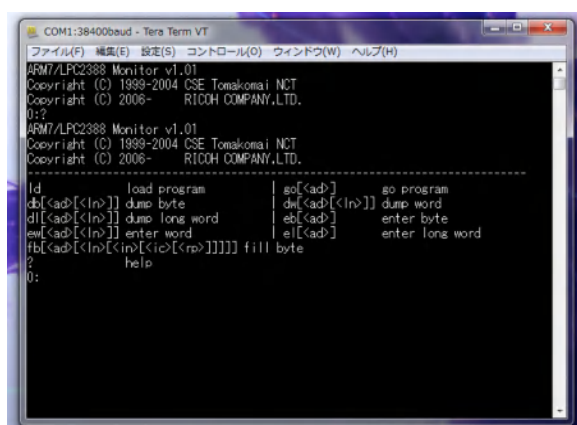
TOPPERSプロジェクト認定

69



開発環境の確認

- 電源を入れて、ROMモニタが起動することを確認してください



2013/06/16

TOPPERSプロジェクト認定

70



SAMPLE1のビルド

- ワークスペースで、SAMPLE1プログラムのビルドを行います
- LPC2388用のTOPPERS/ASPはDBGENV環境スイッチの指定で実行環境を変えられます
 - DBGENV=ROMまたはなしでROM実行版
 - DBGENV=RAMでRAM実行版
- 今回はRAM実行版を作成します
- 確認のためにMAPファイルを作成します

```
$ make depend DBGENV=RAM
$ make DBGENV=RAM
$ arm-elf-objdump -h -t asp.exe > asp.map
```

2013/06/16

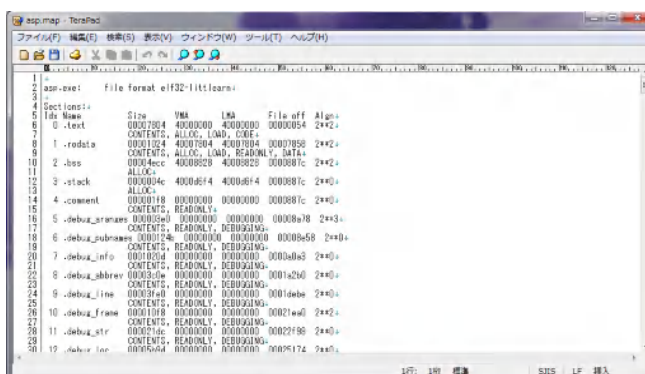
TOPPERSプロジェクト認定

71



実行アドレスの確認

- asp.mapファイルをエディタを使って開いてください
- .textのアドレスが0x40000000(RAM)領域となっています
 - ROM実行版は0x00000000(ROM)領域



2013/06/16

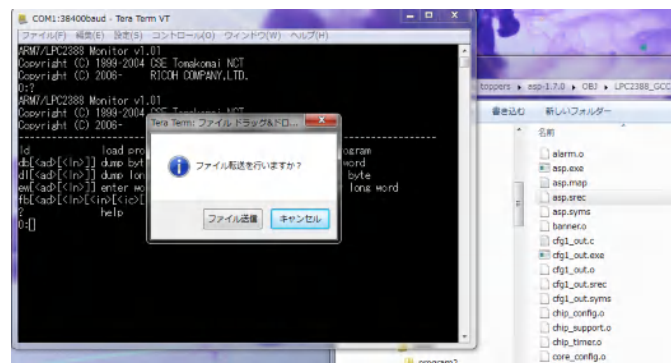
TOPPERSプロジェクト認定

72



ダウンロード

- ROMモニタ上でldコマンドを実行して、ダウンロードモードにします
- ビルドされたasp.srecをTeraTermに重ねてダウンロードしてください



2013/06/16

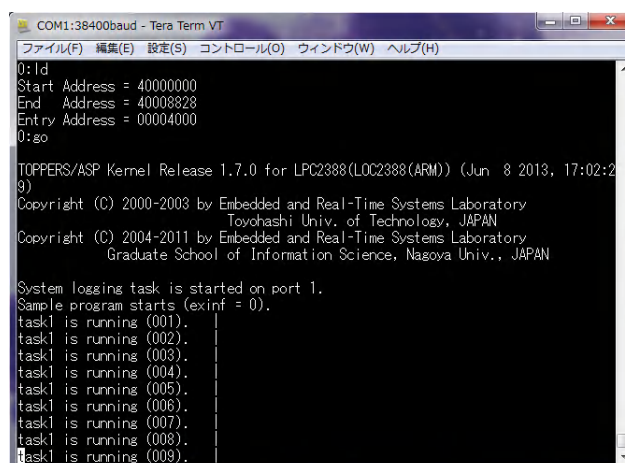
TOPPERSプロジェクト認定

73



実行

- ダウンロードが終了したら、goコマンドで実行してください



2013/06/16

TOPPERSプロジェクト認定

74



ROM実行版の確認

- Flash Magicにて書き込みを行うには.hexフォーマットのプログラムイメージが必要です
- Makefileに.hexフォーマットの出力を行うように修正します
- 全体のリンクの指定にインテルHEXファイルを追加します

```

315 #
316 # 全体のリンク
317 #
318 $(OBJFILE): $(APPL_CFG) kernel_cfg.timestamp $(ALL_OBJS) ¥
319             $(filter %.a,$(ALL_LIBS))
320 $(LINK) $(CFLAGS) $(LDFLAGS) -o $(OBJFILE) $(START_OBJS) ¥
321 $(APPL_OBJS) $(SYSSVC_OBJS) $(CFG_OBJS) $(ALL_LIBS) $(END_OBJS)
322 $(NM) -C $(OBJFILE) > $(OBJNAME).syms
323 $(OBJCOPY) -O srec -S $(OBJFILE) $(OBJNAME).srec
324 $(OBJCOPY) -O ihex -S $(OBJFILE) $(OBJNAME).hex
325 $(CFG) --pass 3 --kernel asp $(INCLUDES) ¥
326             --rom-image $(OBJNAME).srec --symbol-table $(OBJNAME).syms ¥
327             -T $(TARGETDIR)/target_check.tf $(CFG_TABS) $<

```

2013/06/16

TOPPERSプロジェクト認定

75



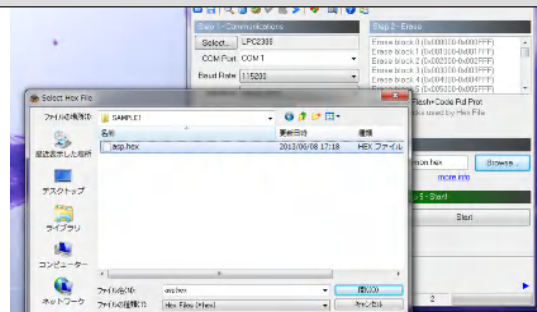
ROM版のビルドと書き込み

- Makeコマンドで再ビルドして、作成されたasp.hexをFlash Magicを使って、ボードに書き込み実行を確認します

```

$ make clean
$ make
$ ls -la asp.hex
--rw-r--r-- 1 roi None 98147 6月  8 17:18 asp.hex

```



2013/06/16

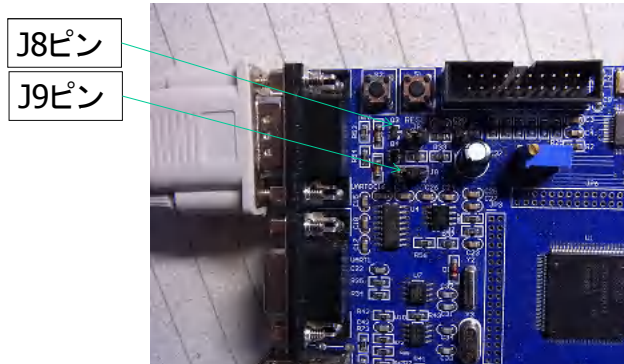
TOPPERSプロジェクト認定

76



Flash Magicでプログラムを書き込む

- TeraTermを閉じ、J8ピンを接続、J9ピンの左の2つのピンを接続する
- ボードの電源を入れてメニューの「Start」ボタンを押すと書き込みが始まります



2013/06/16

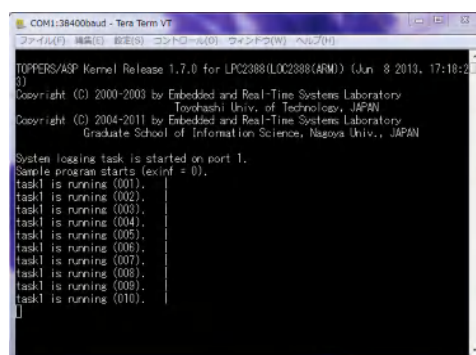
TOPPERSプロジェクト認定

77



ROM実行版の実行

- 電源OFFし、J8、J9をもとに戻し、TeraTermを立ち上げ
- 電源投入でSAMPLE1が実行する
- この後の実習はROM版で行いますが、2日目の実習はRAM版で行います、ROMモニタに戻してください



2013/06/16

TOPPERSプロジェクト認定

78



TOPPERS/ASPカーネルの導入

1. 実習内容の説明
2. 開発環境の構築
3. ビルドと実行
4. [サンプルプログラムの確認](#)

2013/06/16

TOPPERSプロジェクト認定

79



sample1プログラムの内容

- Sample1プログラムはひとつのメインタスクと3つの並列実行タスクで構成される
- メインタスクは種々のRTOS評価用メッセージを並列実行タスクに送り、並列実行タスクが表示するメッセージにてTOPPERS/ASPのサービスコールと動作の評価を行う
- 並列実行タスクは同一優先度で、待ち状態を作らずに実行する

タスクID	優先度	タスク関数	引数	機能
TASK1	10	task	1	並列実行されるタスクは、task_loop 回空ループを実行する度に、タスク * が実行中であることをあらわすメッセージを表示する。
TASK2	10	task	2	
TASK3	10	task	3	
MAIN_TASK	5	main_task	0	コマンドの読み込みと並列タスクへの通知

2013/06/16

TOPPERSプロジェクト認定

80



sample1用コマンド1

- sample1のコマンドは、並列実行タスクに対するコマンドと、全体の制御を行うコマンドがあります
- 以下に全体の制御に関するコマンドを記載します

コマンド	機能
1	以後のコマンドの対称を並列実行タスク1とする
2	以後のコマンドの対称を並列実行タスク2とする
3	以後のコマンドの対称を並列実行タスク3とする
v	発行したシステムコールを表示する(デフォルト)
q	発行したシステムコールを表示しない
r	三つの優先度(9、10、11)のレディキューを回転させる
Q	プログラムを終了する
Ctrl-C	プログラムを終了する

2013/06/16

TOPPERSプロジェクト認定

81



sample1コマンド2

- 並列実行タスクに対する簡単なコマンドは以下の通り

コマンド	機能
a	タスクをact_tskにより起動する
A	タスクに対する起動要求をcan_actによりキャンセルする
e	タスクにext_tskを呼び出させ、終了させる
t	タスクをter_tskにより強制終了させる
s	タスクにslp_tskを呼び出させ、起床待ちにさせる
S	タスクにtslp_tsk(10秒)を呼び出させ、起床待ちにさせる
w	タスクをwup_tskにより起床させる
W	タスクに対する起床要求をcan_wupによりキャンセルする
l	タスクをrel_waiにより強制的に待ち状態にする
>	タスクの優先度を9(HIGH_PRIORITY)にする
=	タスクの優先度を10(MID_PRIORITY)にする
<	タスクの優先度を11(LOW_PRIORITY)にする
d	タスクにdly_tsk(10秒)を呼び出させ、時間経過待ちにさせる

2013/06/16

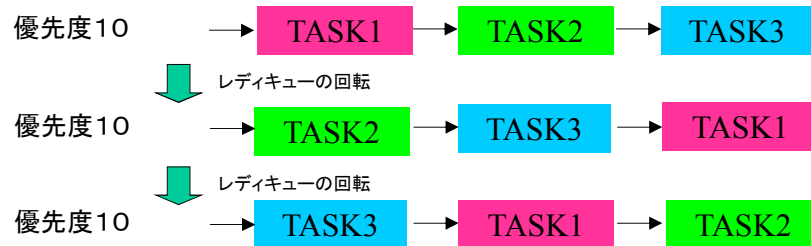
TOPPERSプロジェクト認定

82



演習: レディキューの回転

- 起床時、3つの並列実行タスクは優先度10でレディ状態ですが、並列実行タスクには待ち状態がないため、レディキューの先頭のTASK1が常に実行されます
- rコマンドでレディキューを回転させ、他のタスクの実行を確認しましょう
- 起動時の優先度10 (MID_PRIORITY) のレディキューの状態



2013/06/16

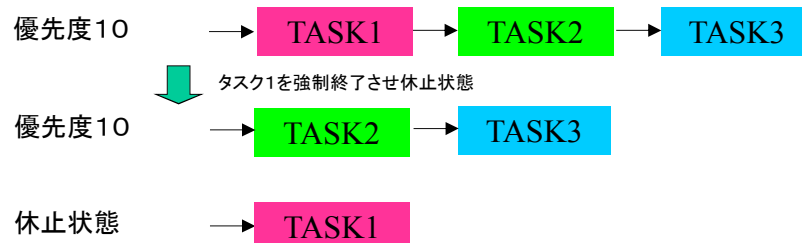
TOPPERSプロジェクト認定

83



演習: タスクの起床と終了

- tコマンドを用いて、TASK1を強制終了させ、レディキューの回転を行っても、TASK1が実行しないことを確認してください
- aコマンドでTASK1を起動させ、レディキューの回転で、TASK1が実行されることを確認してください



2013/06/16

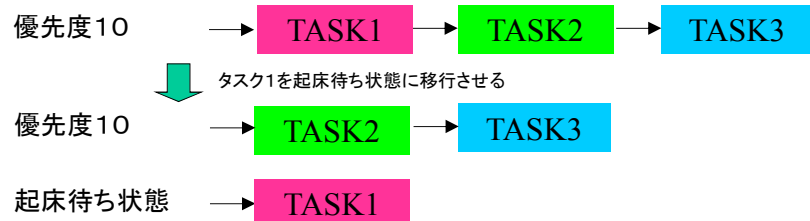
TOPPERSプロジェクト認定

84



演習:タスクの起床待ち

- sコマンドでTASK1を起床待ち状態に、レディキューの回転でTASK1が実行されないことを確認してください
- wコマンドでTASK1を起床させ、レディキューの回転でTASK1が実行されることを確認してください
- dコマンドでTASK1を時間待ち状態にさせ、10秒間はレディキューの回転を行っても実行しないことを確認してください



2013/06/16

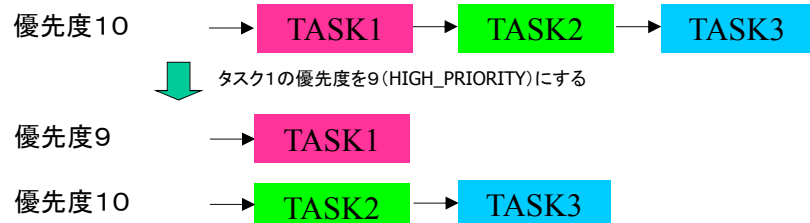
TOPPERSプロジェクト認定

85



演習:優先順位の変更

- >コマンドでTASK1の優先度を高くして、レディキューの回転を行っても、TASK1しか実行しないことを確認してください
- <コマンドでTASK1の優先度を低くして、レディキューの回転を行っても、TASK1が実行されないことを確認してください



2013/06/16

TOPPERSプロジェクト認定

86



システム検証モジュールの導入

1. タスクモニタの導入

2. タスクモニタの改造

2-1. タスク実行時間の測定

2-2. LED用拡張コマンド

2013/06/16

TOPPERSプロジェクト認定

87



タスクモニタの重要性1

- 中規模システムでは、複数のサブシステム分離開発する。サブシステムの結合を行うと種々の原因でいろいろな問題が発生する。ひとつのサブシステムの不具合により、別のサブシステムが誤動作する可能性もある
- 評価専門の部署で商品評価を行う。デバッグ環境の整っていない環境で、不具合の一時調査を行う必要がある
- いろいろな開発、評価の部署で問題が発生した場合、商品に近い開発形態で問題解決する手段が必要となる



商品形態でタスクモニタを実装することにより、問題調査を行うことができる

2013/06/16

TOPPERSプロジェクト認定

88



タスクモニタの重要性2

- 中期規模システムでは、結合環境でサブシステムや専用ミドルウェアの状態表示やモード設定のためのインターフェイスがあったほうが、問題解決を行いやすい
- 開発メーカ供給のデバッガやICEでは、ユーザーシステムに特化した機能を組み込めない
 - このような開発環境は基本的に改造ができない



ソース公開されているタスクモニタに、ユーザーシステム専用の機能を組み込む改造することにより、開発効率を上げることができる

2013/06/16

TOPPERSプロジェクト認定

89



タスクモニタの取り込み1

- 教育WGで開発したタスクモニタをサンプルプログラムに取り込みを行う
- program2ディレクトリ以下の3つのファイルをasp-1.7.0/pdic/lpc23xxディレクトリの下にコピーする
- タスクモニタを取り込みには以下の変更が必要です
 - タスク化するため、コンフィギュレーションファイルの変更を行います
 - インクルードパスの追加と部品となるファイルの追加を行います
 - Makefileの修正を行います

2013/06/16

TOPPERSプロジェクト認定

90



タスクモニタの取り込み2(デバイスドライバ)

- この演習で使用するデバイス(LED、ディップスイッチ、プッシュスイッチ)のデバイスドライバを取り込みます
- 基礎1で使用したデバイスドライバをdeviceディレクトリ以下に用意してあります

関数名	型	引数	機能
led_init	void	inptr_t exinf	LEDデバイスの初期化
led_in	uint8_t	void	LEDの設定値の取り出し
led_out	void	uint8_t led_data	LED点灯、消灯を設定
switch_dip_init	void	inptr_t exinf	ディップスイッチの初期化
switch_dip_sense	uint8_t	void	ディップスイッチの状態取得
switch_push_init	void	inptr_t exinf	プッシュスイッチの初期化
switch_push_state	uint8_t	void	プッシュスイッチの状態取得

2013/06/16

TOPPERSプロジェクト認定

91



デバイスドライバを取り込む

- C言語の標準ライブラリを使用するため、LIBSに-lcの設定を追加します

```

130 #
131 # 共通コンパイルオプションの定義
132 #
133 COPTS := $(COPTS) -g
134 ifndef OMIT_WARNING_ALL
135 COPTS := $(COPTS) -Wall
136 endif
137 ifndef OMIT_OPTIMIZATION
138 COPTS := $(COPTS) -O2
139 endif
140 CDEFS := $(CDEFS)
141 INCLUDES := -I. -I$(SRCDIR)/include -I$(SRCDIR)/arch -I$(SRCDIR) $(INCLUDES)
142 LDFLAGS := $(LDFLAGS)
143 LIBS := $(LIBS) -lc $(CXXLIBS)
144 CFLAGS = $(COPTS) $(CDEFS) $(INCLUDES)

```

2013/06/16

TOPPERSプロジェクト認定

92



タスクモニタとデバイスドライバを取り込み

- Makefileにmonitorディレクトリ中のMakefile.configをインクルードすることによりパスや部品の設定が行える
- デバイスドライバのプログラム(device.c)とmonitorのボード依存部(armv4.c)をシステムサービスとして取り込む

```

167 #
168 # ミドルウェアの Makefile のインクルード
169 #
170 include $(SRCDIR)/monitor/Makefile.config
171
172 #
173 # システムサービスに関する定義
174 #
175 SYSSVC_DIR := $(SYSSVC_DIR) $(SRCDIR)/sysvc $(SRCDIR)/library $(SRCDIR)/pdic/lpc23xx
176 SYSSVC_ASMOBS := $(SYSSVC_ASMOBS)
177 SYSSVC_COBJS := $(SYSSVC_COBJS) banner.o syslog.o serial.o logtask.o ¥
178                  log_output.o vasyslog.o t_peror.o strerror.o ¥
179                  armv4.o device.o $(CXXRTS)
180 SYSSVC_CFLAGS := $(SYSSVC_CFLAGS)
181 SYSSVC_LIBS := $(SYSSVC_LIBS)
182 INCLUDES := $(INCLUDES) -I$(SRCDIR)/pdic/lpc23xx

```

2013/06/16

TOPPERSプロジェクト認定

93



タスクモニタの取り込み(CFGファイルの修正)

- sample1.cfgにmonitor.cfgとdevice.cfgを取り込む
- サンプルプログラムが自動実行しないようにTA_ACTを変更

```

INCLUDE("target_timer.cfg");
INCLUDE("sysvc/syslog.cfg");
INCLUDE("sysvc/banner.cfg");
INCLUDE("sysvc/serial.cfg");
INCLUDE("sysvc/logtask.cfg");
INCLUDE("monitor/monitor.cfg");
INCLUDE("pdic/lpc23xx/device.cfg");

```

追加

```

#include "sample.h"
CRE_TSK(TASK1, { TA_NULL, 1, task, MID_PRIORITY, STACK_SIZE, NULL });
CRE_TSK(TASK2, { TA_NULL, 2, task, MID_PRIORITY, STACK_SIZE, NULL });
CRE_TSK(TASK3, { TA_NULL, 3, task, MID_PRIORITY, STACK_SIZE, NULL });
CRE_TSK(MAIN_TASK, { TA_NULL, 0, main_task, MAIN_PRIORITY, STACK_SIZE, NULL });
DEF_TEX(TASK1, { TA_NULL, tex_routine });
DEF_TEX(TASK2, { TA_NULL, tex_routine });
DEF_TEX(TASK3, { TA_NULL, tex_routine });
CRE_CYC(CYCHDR1, { TA_NULL, 0, cyclic_handler, 2000, 0 });
CRE_ARM(ALMHDR1, { TA_NULL, 0, alarm_handler });

```

修正

2013/06/16

TOPPERSプロジェクト認定

94



タスクモニタのソースの確認

- 以下のファイルはmonitor/Makefile.configによりビルド時に自動的に取り込まれます
- タスクモニタの入出力先はstddev.cで関数設定されていますので、関数を入れ替えれば、種々のデバイスに対応が可能です

標準入出力の追加	モニタ基本部の追加	ARMV4依存部の追加
stdio/stddev.c	monitor.c	arch/armv4.c
stdio/printf.c	task_expansion.c	
stdio/sprintf.c		
stdio/scanf.c		
stdio/sscanf.c		

2013/06/16

TOPPERSプロジェクト認定

95

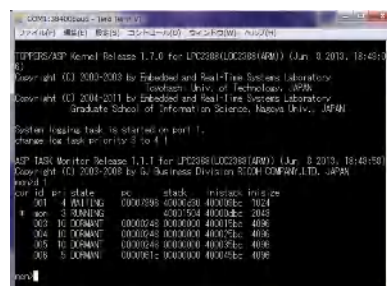


ビルドとタスクモニタの起動を確認

- makeコマンドにて、再ビルドします
- プログラム書き込み後、電源投入でタスクモニタが起動します

```
$ make realclean
$ make depend
$ make
```

構成変更となったため、
realcleanから始める



2013/06/16

TOPPERSプロジェクト認定

96



タスクモニタの使用方法

- タスクモニタが起動されてコンソールにモニタの起動ログを表示し、コマンド待ちを示す `mon>` が表示される
- `help`と入力するとコマンド一覧が表示される

```

task.mon
mon> help
Display [start address] [end address]
Set [start address] [end address] [mode]
Task [act_tsk] [ter_tsk] [sus_tsk] [rsm_tsk] [chg_pri]
Log [logmask] [logaddr]
Help
mon>
  
```

2013/06/16

TOPPERSプロジェクト認定

97



演習 : タスクモニタによるタスク操作

- タスクモニタではタスクに対して次の操作が可能
 - タスクの起動(`act_tsk`) : タスクを実行可能状態に
 - タスクの終了(`ter_tsk`) : タスクを休止状態に
 - タスクのサスペンド(`sus_tsk`) : タスクを強制待ち状態に
 - タスクのレジューム(`rsm_tsk`) : タスクを強制待ち状態から復帰
 - 優先度の変更(`chg_pri`) : タスクの優先度を変更する
- タスクの操作する前に操作対象のタスクを指定する必要がある
 - タスクIDは `kernel_id.h` で定義されている値
 - 一度設定すると、それ以後のタスク操作は指定したタスクに対して行われる

```
mon> set task [タスクID]
```

2013/06/16

TOPPERSプロジェクト認定

98



演習: サンプルプログラムの実行

- ・ タスクモニタからサンプルプログラムを実行する
- ・ 入力はタスクモニタが取り込むため、操作はできない
- ・ display task : タスクの状態を表示
- ・ set task 6 : ID番号6(メインタスク)のタスクを指定
- ・ task activate : タスクの起動
- ・ コマンドは最初の一文字で短縮可能

display task

set task 4

task activate

```

TOPPERS/OS2 Kernel Version 1.1.1 for ARMv7-M (Oct 08 2013) 14:11:45
Copyright (C) 2005-2013 by Software and Real-Time Systems Laboratory
                        Graduate School of Information Science, Nippon
                        University (S) 2005 by Nippon University, JAPAN
Current monitor task is started on host ...
show log task process 0 to 4
=====
task 0: name: main, id: 0, priority: 0, status: running, ppid: 0
task 1: name: task1, id: 1, priority: 1, status: ready, ppid: 0
task 2: name: task2, id: 2, priority: 2, status: ready, ppid: 0
task 3: name: task3, id: 3, priority: 3, status: ready, ppid: 0
task 4: name: task4, id: 4, priority: 4, status: ready, ppid: 0
task 5: name: task5, id: 5, priority: 5, status: ready, ppid: 0
task 6: name: task6, id: 6, priority: 6, status: ready, ppid: 0
=====
set task 4
=====
task 4: name: task4, id: 4, priority: 4, status: running, ppid: 0
=====
task activate
=====
task 4: name: task4, id: 4, priority: 4, status: running, ppid: 0
=====

```

2013/06/16

TOPPERSプロジェクト認定

99



演習: サンプルプログラムの表示の停止

- ・ サンプルプログラムはsyslogのLOG_NOTICEで表示を行っている、タスクモニタからsyslogの表示レベルを変更できる
- ・ 表示を無視して、log mode 4 (return)を入力すると、表示レベルがLOG_WARNINGとなってログ表示しなくなる

log mode 4

```

TOPPERS/OS2 Kernel Version 1.1.1 for ARMv7-M (Oct 08 2013) 14:11:45
Copyright (C) 2005-2013 by Software and Real-Time Systems Laboratory
                        Graduate School of Information Science, Nippon
                        University (S) 2005 by Nippon University, JAPAN
Current monitor task is started on host ...
show log task process 0 to 4
=====
task 0: name: main, id: 0, priority: 0, status: running, ppid: 0
task 1: name: task1, id: 1, priority: 1, status: ready, ppid: 0
task 2: name: task2, id: 2, priority: 2, status: ready, ppid: 0
task 3: name: task3, id: 3, priority: 3, status: ready, ppid: 0
task 4: name: task4, id: 4, priority: 4, status: ready, ppid: 0
task 5: name: task5, id: 5, priority: 5, status: ready, ppid: 0
task 6: name: task6, id: 6, priority: 6, status: ready, ppid: 0
=====
log mode 4
=====
log mode is set to 4 (return)
=====

```

2013/06/16

TOPPERSプロジェクト認定

100



演習: サンプルプログラムの停止

- サンプルプログラムは3～6までの4つのタスクで実行している
- task terminate : 現在選ばれているタスク(6)を終了
- set task とtask terminateで、3～5のタスクを終了させる

リセットは直接実行版では、リセットボタン押下。

ROMモニタが再起動するので、ダウンロードから再実行させる

[illegible]

演習:タスクモニタからのメモリ操作

- ・ タスクモニタから直接、LEDを操作します
- ・ set wordコマンドでLEDが接続されているポート0のポートレジスタ(0x3fffc058/0x3fffc05c)に直接書き込み、動作を確認する

0x3fffc040をOUT設定:設定済

COM1:38400baud - Tera Term VT

ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)

change log task priority 3 to 4 !

ASP TASK Monitor Release 1.1.1 for LPC2888(LOC2888(ARM)) (Jun 16 2013, 09:32:35)
Copyright (C) 2003-2008 by GJ Business Division RICOH COMPANY, LTD. JAPAN

mon>set word 3fff040
3fff040 ffffffff =,
mon>set word 3fff05c
3fff05c 00000000 =ff 00000000
3fff060 00000000 =,
mon>set word 3fff058
3fff058 00000000 =ff 000000ff
3fff05c 00000000 =,
mon>

0x3fff05cに0xfffを書き込み：全LED 消灯

0x3fff058に0xfffを書き込み：全LED 消灯

演習: タスクモニタでのタスク優先度の注意

- タスクモニタの優先度は3で通常はコンソールからの入力待ちで待ち状態になっており、コンソールに入力があると起床して指定されたコマンドを実行する
- メインタスクの優先度2以上に設定すると、メインタスクが優先的にサンプルプログラム操作を行い、タスクモニタの操作ができなくなる

選択されているタスクがメインタスクの状態で、task priority 2 (return)の短縮コマンド t p 2 (return)を発行して優先度2に変更

```

C:\COM1 Test Run V1
Print: 0000 0000 0000 0000 0000 0000 0000 0000
main: task 0
00000000:task activate
00000000:task activate
compute sel_kid(0) : result = 0,OK !
Basic remove actor Genral = 0
00000000:task1 is running (001).
task1 is running (002).
task1 is running (003).
task1 is running (004).
task1 is running (005).
task1 is running (006).
task1 is running (007).
task1 is running (008).
task1 is running (009).
task1 is running (010).
task1 is running (011).
task1 is running (012).
task1 is running (013).
task1 is running (014).
task1 is running (015).
task1 is running (016).
task1 is running (017).
task1 is running (018).
task1 is running (019).
task1 is running (020).
compute sel_pri(0) : result = 0,OK !
00000017:task2 is running (001).
task1 is running (002).
task1 is running (003).
task1 is running (004).
task1 is running (005).
task1 is running (006).
task1 is running (007).
task1 is running (008).
task1 is running (009).
task1 is running (010).
task1 is running (011).
task1 is running (012).
task1 is running (013).
task1 is running (014).
task1 is running (015).
task1 is running (016).
task1 is running (017).
task1 is running (018).
task1 is running (019).
task1 is running (020).

```

システム検証モジュールの導入

1. タスクモニタの導入
 2. タスクモニタの改造
- 2-1.タスク実行時間の測定
- 2-2. LED用拡張コマンド

タスクごとの実行時間の測定

- タスクモニタは、簡単な方法でタスク毎の実行時間を測定する機能を持っています
- タスク毎の実行時間がわかると、システムがハードリアルタイムが満たしているかの確認が行えます
- タスクの優先順位設定や、高速化が必要なタスクの性能目標を立てるのに役立ちます
- **このような機能の実装は、オーバーヘッドの増大や性能の低下を招きますので、必要のない場合は機能の無効化等の管理が必要です**

2013/06/16

TOPPERSプロジェクト認定

105



タスク実行時間の取得

- タスク時間測定は、スケジューラ中でタスクのスイッチングを行う時に実行ログを取る形で測定を行います
- スケジューラソースcore_support.SのLOG_DSP_ENTERとLOG_DSP_LEAVEコンパイルスイッチが設定されたとき、log_dsp_enter/log_dsp_leaveがディスパッチの前後に呼び出されます

```

/*
 * ディスパッチャ本体
 */
dispatcher:
/*
 * このルーチンは、タスクコンテキスト・CPUロック状態・ディスパッチ
 * 許可状態・(モデル上の)割込み優先度マスク全解除状態で呼び出さ
 * れる。実行再開番地へもこの状態のまま分岐する。
 *
 * すなわち、スーパーバイザーモード、IRQ禁止・disdspがfalse・dspflg
 * がtrueとなっている。実行再開番地へもこの状態のまま分岐する。
 */
#ifdef LOG_DSP_ENTER
    ldr r1, =p_runtsk /* p_runtskをパラメータに */
    ldr r0, [r1]
    bl log_dsp_enter
#endif /* LOG_DSP_ENTER */

```

2013/06/16

TOPPERSプロジェクト認定

106



タスク時間の測定

- タスクモニタで、これらの関数の呼び出しで実行時間の測定を行う

```
dispatcher_0:
    ldr r0, =p_schedtsk /* p_schedtskをp_runtskに */
    ldr r1, [r0]
    ldr r2, =p_runtsk
    str r1, [r2]
    cmp r1, #0 /* p_runtskがNULLならdispatcher_1へ */
    beq dispatcher_1
    ldr sp, [r1, #TCB_sp] /* タスクスタックを復帰 */
#ifdef LOG_DSP_LEAVE
    mov r0, r1 /* p_runtskをパラメータに */
    mov r4, r1 /* r1はスクラッチレジスタなので保存 */
    bl log_dsp_leave
    mov r1, r4
#endif /* LOG_DSP_LEAVE */
    ldr r4, [r1, #TCB_pc] /* 実行再開番地を復帰 */
    bx r4
.endif
```

2013/06/16

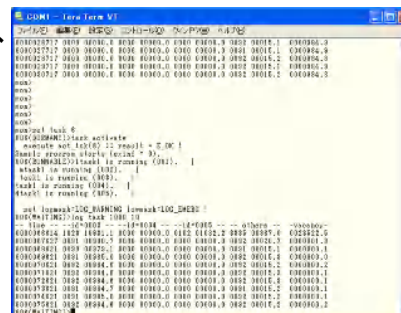
TOPPERSプロジェクト認定

107



タスク実行時間の表示

- スケジューラの切り替え時に保存されたタスク実行時間は、LOG TASKコマンドで設定された時間単位に表示を行い、表示後は実行時間をリセットします
- タスク実行時間の収集はmonitor/task_expansion.c中のlog_tsk_enterとlog_tsk_leave関数で行います。収集したタスク実行時間の取り出しはget_tsklog関数をタスクモニタが周期的に行うことにより実行されます



LOG TASK <時間(単位MS)> <回数>

2013/06/16

TOPPERSプロジェクト認定

108



演習: タスク実行時間ログ機能の追加

- Makefileに「LOG_DSP_ENTER」と「LOG_DSP_LEAVE」を追加する
- core_support.Sはカーネルソースなので、クリア後再ビルドします

```

130 #
131 # 共通コンパイルオプションの定義
132 #
133 COPTS := $(COPTS) -g
134 ifndef OMIT_WARNING_ALL
135 COPTS := $(COPTS) -Wall
136 endif
137 ifndef OMIT_OPTIMIZATION
138 COPTS := $(COPTS) -O2
139 endif
140 CDEFS := $(CDEFS) -DLOG_DSP_ENTER -DLOG_DSP_LEAVE
141 INCLUDES := -I. -I$(SRCDIR)/include -I$(SRCDIR)/arch -I$(SRCDIR) $(INCLUDES)
142 LDFLAGS := $(LDFLAGS)
143 LIBS := $(LIBS) -lc $(CXXLIBS)
144 CFLAGS = $(COPTS) $(CDEFS) $(INCLUDES)

```

2013/06/16

TOPPERSプロジェクト認定

109



演習: TASKの実行時間の確認

- sample1の実行中、TASK1のみ実行していることをタスク実行時間ログ機能を使って確認してみよう
- TASK1がID=3、TASK2がID=4、TASK3がID=5に設定され、その他のタスクの実行時間はothersに計測されます
- MAIN_TASKはID=6なので、実行後set task 6で対象タスクを6に切り替え、task activateでMAIN_TASKを実行します
- 表示が始まった後、log mode 4でログ表示を止めます
- log task 1000 10で1秒単位に実行時間ログを計測します
- ID=3(TASK1)が実行が割り当てられ、ID=4(TASK2)とID=5(TASK3)が実行時間が割り当てられていないことを確認してください

2013/06/16

TOPPERSプロジェクト認定

110



システム検証モジュールの導入

1. タスクモニタの導入
2. タスクモニタの改造
 - 2-1. タスク実行時間の測定
 - 2-2. LED用拡張コマンド

2013/06/16

TOPPERSプロジェクト認定

111



演習: タスクモニタの拡張コマンドでLEDを制御する

- タスクモニタはPIPE以下に新規のコマンドを追加できる
- PIPEコマンドでLEDコマンドを追加して、LEDの点灯、消灯を行う
- PIPEコマンドのサブコマンドとして、LEDコマンドを設ける
- LEDコマンドには2つのパラメータがあり、第1パラメータはLEDの番号(1-8)をセットし、第2パラメータは消灯: 0、点灯: 1を指定する
- LEDコマンドを用いて、LEDを制御する

PIPE LED (LED番号1-8) (消灯-0、点灯-1)

2013/06/16

TOPPERSプロジェクト認定

112



演習: PIPEコマンドの追加

- モニタタスクのコンパイルスイッチSUPPORT_PIPEを定義すると、PIPEコマンドにて、pipe_command関数を呼び出すように設定できる

以下、monitor.cのソースの記述

```
/*
 * メインコマンドテーブル
 */
static struct COMMAND_TABLE const mon_dispatch[] = {
    {"DISPLAY",    display_command }, /* 表示 */
    {"SET",        set_command    }, /* 設定 */
    {"TASK",       task_command   }, /* itron TASK */
    {"LOG",        log_command    }, /* ロギング */
#ifdef SUPPORT_VOLUME
    {"VOLUME",     volume_command }, /* ディスクコマンド */
#endif
#ifdef SUPPORT_PIPE
    {"PIPE",       pipe_command   }, /* 拡張コマンド */
#endif
    {"HELP",       help_command   }, /* ヘルプ */
};
```

2013/06/16

TOPPERSプロジェクト認定

113



演習: SUPPORT_PIPEコンパイルスイッチ

- Makefileを修正し、「SUPPORT_PIPE」定義を追加する
- ビルドを行うと、pipe_command関数がundefinedとなる

```
130 #
131 # 共通コンパイルオプションの定義
132 #
133 COPTS := $(COPTS) -g
134 ifndef OMIT_WARNING_ALL
135 COPTS := $(COPTS) -Wall
136 endif
137 ifndef OMIT_OPTIMIZATION
138 COPTS := $(COPTS) -O2
139 endif
140 CDEFS := $(CDEFS) -DLOG_DSP_ENTER -DLOG_DSP_LEAVE -DSUPPORT_PIPE
141 INCLUDES := -I. -I$(SRCDIR)/include -I$(SRCDIR)/arch -I$(SRCDIR) $(INCLUDES)
142 LDFLAGS := $(LDFLAGS)
143 LIBS := $(LIBS) -lc $(CXXLIBS)
144 CFLAGS = $(COPTS) $(CDEFS) $(INCLUDES)
```

2013/06/16

TOPPERSプロジェクト認定

114



演習: pipe_commandの追加

- command.cファイルをSAMPLE1ディレクトリに追加して、LEDコマンドを実装する
- インクルードファイルとしてstdio.hとmonitor.hを追加する
- need_monitor関数にLEDの初期化を追加する
- pipe_command関数を追加し、以下の機能を実装する
 - LEDコマンドの実装
 - LED以外のコマンドでは、ヘルプメニューを表示する
- LEDコマンドの判定は、タスクモニタ内の関数を流用する
- LEDコマンドからのパラメータの取り出しはsscanf関数を用いる

2013/06/16

TOPPERSプロジェクト認定

115



command.cの追加

- Makefileを修正してcommand.cをアプリケーションに追加します
- APPL_COBJS変数にcommand.oを追加します

```

146 #
147 # アプリケーションプログラムに関する定義
148 #
149 APPLNAME = sample1
150 APPLDIR =
151 APPL_CFG = $(APPLNAME).cfg
152
153 APPL_DIR = $(APPLDIR) $(SRCDIR)/library
154 APPL_ASMOBS =
155 ifdef USE_CXX
156   APPL_CXXOBS = $(APPLNAME).o
157   APPL_COBJS = command.o
158 else
159   APPL_COBJS = $(APPLNAME).o command.o
160 endif
161 APPL_CFLAGS =
162 APPL_LIBS =
163 ifdef APPLDIR
164   INCLUDES := $(INCLUDES) $(foreach dir,$(APPLDIR),-I$(dir))
165 endif

```

2013/06/16

TOPPERSプロジェクト認定

116



演習: command.cの作成(インクルードファイル)

- LEDコマンドのパラメータ取り込み用にsscanf関数を使用するため、プロトタイプ宣言のあるstdio.hをインクルードする
- LEDコマンドの解析用にタスクモニタで使用している構造体や関数を使用するため、monitor.hをインクルードする

```
/*
 * モニタコマンド用ソースファイル
 */
#include <kernel.h>
#include <stdio.h>
#include <t_syslog.h>
#include <t_stdlib.h>
#include "syssvc/serial.h"
#include "syssvc/syslog.h"
#include "kernel_cfg.h"
#include "monitor.h"
#include "device.h"
```

2013/06/16

TOPPERSプロジェクト認定

117



演習: コマンド定義テーブルの取り込み

- タスクモニタ内のSUBCOMMAND_TABLEに定義したコマンドをコマンド番号に変換する機能を用いてLEDコマンドを簡略化します
- compare_word関数は、ここで定義したmon_pipeとコマンド文字列からコマンド番号に変換を行います

```
/*
 * パイプコマンド番号
 */
#define PIPE_LED    0
#define PIPE_HELP   1
/*
 * パイプコマンドテーブル
 */
static struct SUBCOMMAND_TABLE const mon_pipe[] = {
    {"LED",          PIPE_LED }, /* ログモード設定 */
    {"HELP",         PIPE_HELP }
};
```

2013/06/16

TOPPERSプロジェクト認定

118



演習: pipe_command関数の作成

- pipe_command関数は、引数としてPIPE以下の文字列が渡されます
- skip_command文字列“LED”を見つけるとtrueを返します、skip_wordは“LED”を読み飛ばした文字番号をpointにセットします

```

ulong_t pipe_command(B *command)
{
    int_t point=0;
    char_t cmd=1;
    int_t no, count;
    int_t arg1, arg2;
    char_t reg;
    count = sizeof(mon_pipe) / sizeof(struct SUBCOMMAND_TABLE);
    if(command[point]){
        for(no = 0 ; no < count ; no++){
            if(compare_word(mon_pipe[no].subcommand, command, 0)){
                skip_word(command, &point);
                cmd = mon_pipe[no].type;
                break;
            }
        }
    }
}

```

2013/06/16

TOPPERSプロジェクト認定

119



演習: パラメータの解析とコマンド実行

- cmdにセットされたコマンド番号から処理を判別してください
- パラメータ1はarg1、パラメータ2はarg2に設定されますので、LEDの制御は自作してください
- あとは、ダウンロードしてデバッグしてください

```

switch(cmd){
case PIPE_LED: /* LEDの制御 */
    sscanf((char *)&command[point], "%d %d", &arg1, &arg2);
    if(arg1 >= 1 && arg1 <= 8){
        reg = led_in();
        :
        :
        led_out(reg);
    }
    break;
default:
    printf(" LED (no) (on-1,off-0) led control\n");
    break;
}
return 0;
}

```

2013/06/16

TOPPERSプロジェクト認定

120



TOPPERS/ASP導入実習のまとめ

- コンパイルやリンク等の開発環境とターゲットボードがあれば、比較的簡単にTOPPERS/ASPの動作確認が行える
- ソースが供給されているので、RTOSのしくみを理解しやすい、問題発生時もソースを用いた問題解析が可能である
- RTOSを使用する中規模組込み開発では、多人数の開発者が分業して多くのプログラム開発やミドルウェアの導入を行うことが多い、タスクモニタをシステム用に機能拡張し、問題発生時に、問題の要因を簡単に判別できる開発環境を用意していくことが、開発効率や品質の向上につながる



まとめ



リアルタイムOSを用いたシステム設計

- リアルタイムOSは、プログラム規模が大きい中規模組込みシステムで使用すると効果的です
- リアルタイムOSは、CPU処理の代行を行う機能が主で、システム設計を行う場合、機器の機能に対応したベース機能を含んだプラットフォームを構築すると、開発工数の削減や品質の向上につながります
- プラットフォームは、機器の機能に応じたメンテナンス機能を用意した方が、効率的な開発が行えます
 - メンテナンス機能の代表がタスクモニタです
- システム設計管理には、熟練したシステム管理者が必要です

2013/06/16

TOPPERSプロジェクト認定

123



μITRON仕様に関して

- μITRON仕様は、日本の組込みシステムでもっと多く使われているリアルタイムOSの仕様です
- リアルタイムOSは、組込み機器が多くの変種を持つためCPUの機能の代行する機能に限定した仕様となっています
- タスクを用いたシステムでは、同期や通信などの機能を用いて各機能が円滑に動作するようにシステム設計する必要があります
- TOPPERS/JSPはμITRON4.0のスタンダードプロファイルに準じたリアルタイムOSで、TOPPERS/ASPはμITRON4.0仕様に新たな機能を付加し、オープンソースとして供給されています

2013/06/16

TOPPERSプロジェクト認定

124



著者リスト

- リアルタイムOSの基礎
 - 高田 広章(名古屋大学), 本田 晋也(名古屋大学)
 - 竹内 良輔((株)リコー)
- ITRON仕様の基礎知識
 - 高田 広章(名古屋大学), 本田 晋也(名古屋大学)
- TOPPERS/ASPの導入
 - 竹内 良輔((株)リコー)
- システム検証モジュールの導入
 - 竹内 良輔((株)リコー)

謝辞

本教材の開発において、NPO法人組込みソフトウェア管理者・技術者育成研究会およびNPO法人TOPPERSプロジェクトの作成した以下の教材を参考としました。

- OpenSESSAME Seminar組込みソフトウェア技術者・管理者向けセミナー初級者向けテキスト第5版
- TOPPERS初級実装セミナー教材

両団体および上記教材の作者の皆様へ感謝します。

本教材の開発において、NEXCESS初級コースおよび指導者養成コースの受講者の皆様のコメントを参考としました。両コースの受講者の皆様へ感謝します。