

# TOPPERS基礎実装セミナー

(STM32F4\_Discovery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

2016/10/09

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。

2016/10/09

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(STM32F4-Discovery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) 2008-2015 by 竹内良輔 (株)リコー

Copyright (C) 2013-2015 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。

2016/10/09

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    | 0.3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 |

2016/10/09

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

2016/10/09

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. タスク間の同期と通信

2016/10/09

TOPPERSプロジェクト認定

6



## 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

2016/10/09

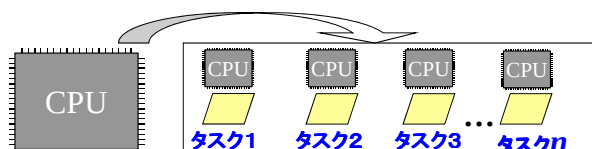
TOPPERSプロジェクト認定

7



## 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



2016/10/09

TOPPERSプロジェクト認定

8



## 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≡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

2016/10/09

TOPPERSプロジェクト認定

9



## 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

2016/10/09

TOPPERSプロジェクト認定

10



# 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. 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

## 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/VDX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

## 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



# 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

2016/10/09

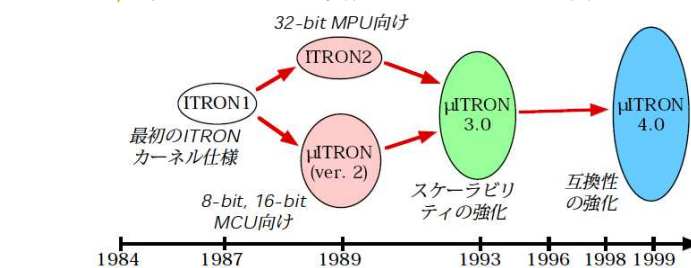
TOPPERSプロジェクト認定

17



## 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仕様を策定・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



2016/10/09

TOPPERSプロジェクト認定

18



## 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
  - 組み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



2016/10/09

TOPPERSプロジェクト認定

19



## ITRON仕様のカーネルの機能(μITRON4.0仕様)

- |                                                                                                                                                                                                                                                                                                                                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>• カーネルの機能               <ul style="list-style-type: none"> <li>– タスク管理機能</li> <li>– タスク付属同期機能</li> <li>– タスク例外処理機能</li> <li>– 同期・通信機能</li> <li>– 拡張同期・通信機能</li> <li>– メモリプール機能</li> <li>– 時間管理機能</li> <li>– システム状態管理機能</li> <li>– 割り込み管理機能</li> <li>– サービスコール管理機能</li> </ul> </li> </ul> | <ul style="list-style-type: none"> <li>• サービスコール数               <ul style="list-style-type: none"> <li>– フルセット                   <ul style="list-style-type: none"> <li>• サービスコール : 166</li> <li>• 静的API : 21</li> </ul> </li> <li>– スタンダードプロファイル                   <ul style="list-style-type: none"> <li>• サービスコール : 70</li> <li>• 静的API : 11</li> </ul> </li> <li>– 自動車制御用プロファイル                   <ul style="list-style-type: none"> <li>• サービスコール : 43</li> <li>• 静的API : 8</li> </ul> </li> <li>– 最小セット</li> </ul> </li> </ul> |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|



**! I/O操作のための機能は規定されていない**

2016/10/09

TOPPERSプロジェクト認定

20



## 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`

操作対象を表す. この場合はタスク(task)

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

2016/10/09

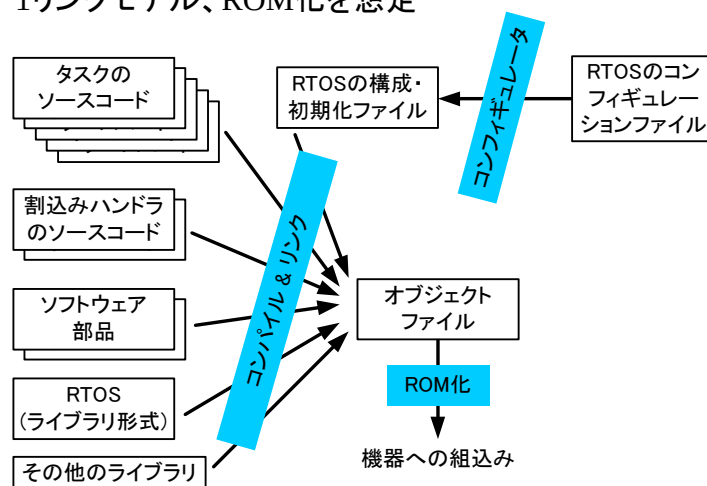
TOPPERSプロジェクト認定

21



## 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



2016/10/09

TOPPERSプロジェクト認定

22



## 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

2016/10/09

TOPPERSプロジェクト認定

23



## 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

2016/10/09

TOPPERSプロジェクト認定

24



## 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       |

2016/10/09

TOPPERSプロジェクト認定

25



## マルチタスク機構

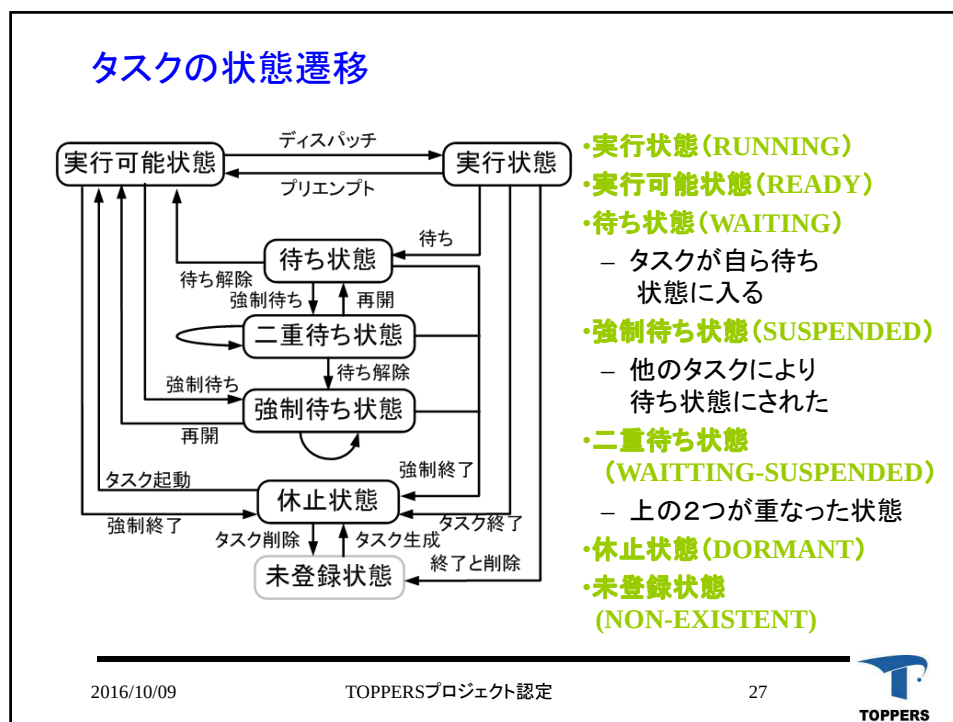
- スケジューリング方式
  - $\mu$ ITRON4.0では、プリエンプティブな優先度ベーススケジューリングによるマルチタスク機構をサポート
  - 同優先度のタスクはFCFS (First Come First Served) 方式でスケジューリング. 同優先度のタスク内での実行順序を変更する機能を用意. これを用いて、同優先度内でのラウンドロビンスケジューリングも実現可能
  - 優先度付けは静的が基本だが、優先度を動的に変更する機能も用意
- タスク状態
  - 実行、実行可能、休止、待ち、強制待ち、二重待ち、未登録の7つの状態をサポート

2016/10/09

TOPPERSプロジェクト認定

26





## 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)

## 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)

2016/10/09

TOPPERSプロジェクト認定

29



## 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
  - μITRON4.0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)
    - オーバーランハンドラ(オーバーラン時に呼ばれる)
- 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

2016/10/09

TOPPERSプロジェクト認定

30



## 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  |

2016/10/09

TOPPERSプロジェクト認定

31



## 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?     |

2016/10/09

TOPPERSプロジェクト認定

32





## 割込み処理と割込み管理機能

- 割込み処理 (外部割込み)
  - $\mu$ ITRON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

2016/10/09

TOPPERSプロジェクト認定

33



## 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、タスクに対して通知を行うことが可能
  - 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   |

2016/10/09

TOPPERSプロジェクト認定

34



## 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?  |

## 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

## 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

2016/10/09

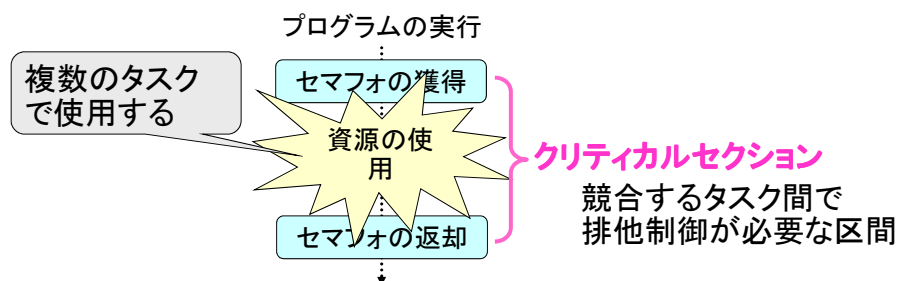
TOPPERSプロジェクト認定

37



## 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。



2016/10/09

TOPPERSプロジェクト認定

38



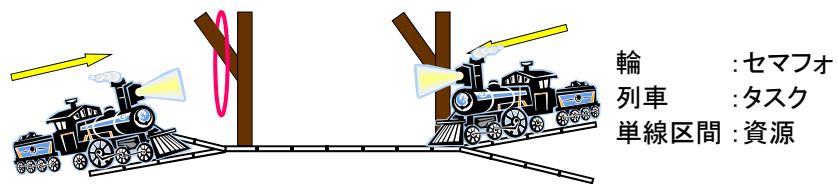
## セマフォ(Semaphore) : サービスコール&語源

### • 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



2016/10/09

TOPPERSプロジェクト認定

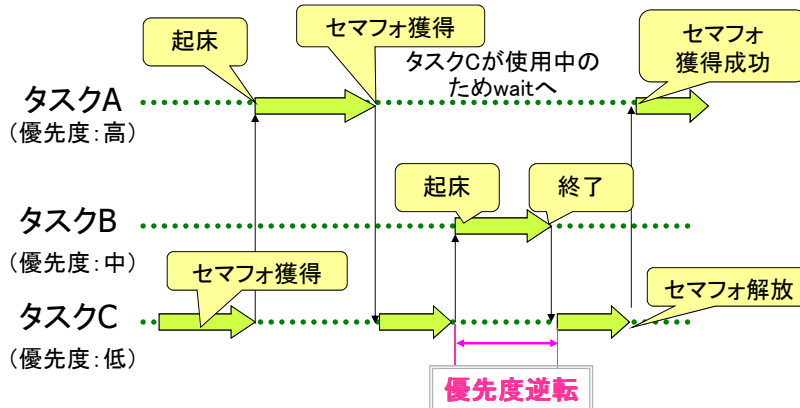
39



## 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)



2016/10/09

TOPPERSプロジェクト認定

40

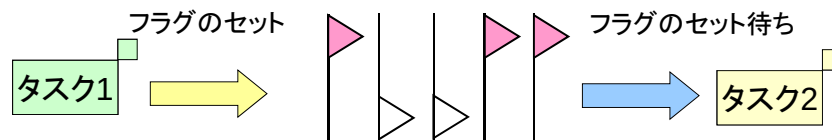


## イベントフラグ (EventFlag)

- 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   |



2016/10/09

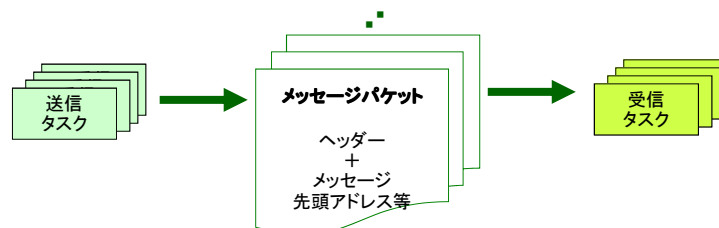
TOPPERSプロジェクト認定

41

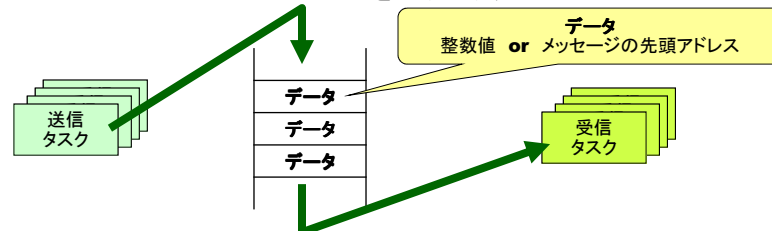


## メールボックス と データキュー:概要

- 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



2016/10/09

TOPPERSプロジェクト認定

42

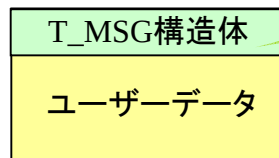


## メールボックス (Mail Box)

- 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



カーネルが  
使用する領域

2016/10/09

TOPPERSプロジェクト認定

43



## 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    |
| rcv_dtq, prcv_dtq, trcv_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 |

2016/10/09

TOPPERSプロジェクト認定

44



## 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 1.4.4
- 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

2016/10/09

TOPPERSプロジェクト認定

47



## 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

2016/10/09

TOPPERSプロジェクト認定

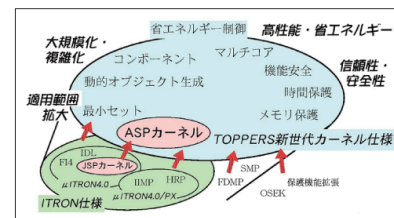
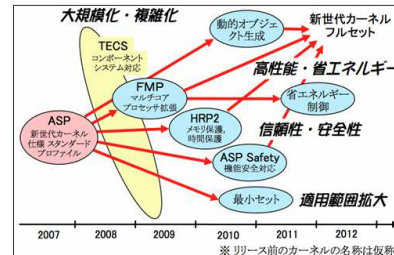
48





## 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



2016/10/09

TOPPERSプロジェクト認定

49



## 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
  - ・ 割り込み処理機能を「TOPPERS準拠割り込み処理モデル」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

2016/10/09

TOPPERSプロジェクト認定

50



## 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
- $\mu$ ITRON4. 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

2016/10/09

TOPPERSプロジェクト認定

51



## 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)

2016/10/09

TOPPERSプロジェクト認定

52



### 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

2016/10/09

TOPPERSプロジェクト認定

53



### TOPPERS/ASPでの型宣言

- TOPPERS新世代カーネルでは、ITRON仕様のいくつかの型を廃止し、明示的に型内容の分かる型に変更した
- ITRON仕様型、定数についても、itron.hインクルードファイルをインクルードすれば使用が可能です

| ITRON型 | 新しい型     |    |          |        |          |
|--------|----------|----|----------|--------|----------|
| B      | int8_t   | W  | int32_t  | VP     | void *   |
| UB     | uint8_t  | UW | uint32_t | INT    | int_t    |
| VB     | uint8_t  | VW | uint32_t | UINT   | uint_t   |
| H      | int16_t  | D  | int64_t  | BOOL   | bool_t   |
| UH     | uint16_t | UD | uint64_t | VP_INT | intptr_t |
| VH     | uint16_t | VD | uint64_t |        |          |

2016/10/09

TOPPERSプロジェクト認定

54



## その他の型とTOPPERS/ASPでの定数の変更

- 従来通り使用するITRON型は以下のとおりです。これらはRTOSの仕様に依存した重要な型です

|      |               |        |             |
|------|---------------|--------|-------------|
| FN   | 機能コード         | SIZE   | メモリ領域のサイズ   |
| ER   | エラーコード        | TMO    | タイムアウト設定    |
| ID   | オブジェクトID番号    | RELTIM | 相対時間        |
| ATR  | オブジェクトの属性     | SYSTIM | システム時刻      |
| STAT | オブジェクトの状態     | SYSUTM | 性能測定用システム時間 |
| MODE | サービスコールの動作モード | FP     | プログラムの起動番地  |
| PRI  | 優先度           |        |             |

- 真偽判定定数は以下のように変更します

TRUE → true

FALSE → false

2016/10/09

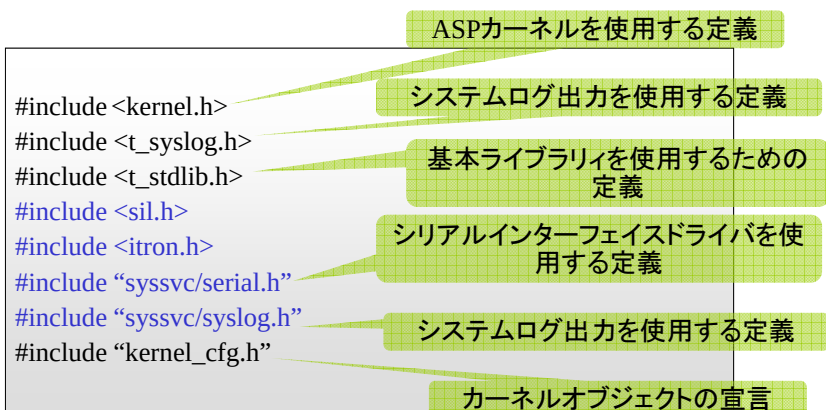
TOPPERSプロジェクト認定

55



## TOPPERS/ASPの環境インクルード手順

- 以下にTOPPERS/ASPを使用するためのインクルードファイルを示します



2016/10/09

TOPPERSプロジェクト認定

56



## 開発環境の構築

2016/10/09

TOPPERSプロジェクト認定

57



### ボードの確認とソフトウェアのセットアップ

- パソコン上に開発環境がセットアップされていない場合は、開発環境をセットアップする必要がある
- 開発環境のセットアップ手順は開発環境編、開発環境の確認「ソフトウェア環境の構築」の記載してあります



2016/10/09

TOPPERSプロジェクト認定

58



# 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カーネルのビルドについての説明
- 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)
  - STM32F4-Discovery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

2016/10/09

TOPPERSプロジェクト認定

61



## TOPPERS/ASPカーネルのセットアップ

- パソコンにTOPPERS/ASP開発環境がセットアップされていない場合は、セットアップを行う必要がある
- TOPPERS/ASP開発環境のセットアップは、開発環境編、「TOPPERS/ASPカーネルの導入」に記載してあります

2016/10/09

TOPPERSプロジェクト認定

62



# 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

2016/10/09

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-1.9.2の個別パッケージをダウンロードします
- 同様にARM Coretex-M4アーキテクチャ・GCC依存部パッケージasp\_arch\_arm\_m4\_gcc-1.9.3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、ターゲットドライバ依存部(PD/C))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、こちらからします。

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、会員としての扱いですが、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-1.9.2.tar.gz | 2015-05-30            |
|                  | Release               |
| asp-1.9.1.tar.gz | 2014-11-17            |
|                  | Release 1.3.0以前のバージョン |

| ターゲット依存部                                               |                                                                                                                                                                                                       |                |            |
|--------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|------------|
| ARM Cortex-M7アーキテクチャ・GCC依存部パッケージ(担当:TOPPERSプロジェクト教育WG) |                                                                                                                                                                                                       |                |            |
| パッケージ                                                  | ダウンロード                                                                                                                                                                                                | 対応する非依存部のバージョン | リリース日      |
| asp_arch_arm_m7_gcc-1.9.2.tar.gz                       | arch/arm_m_gcc<br>target/stm32f746discovery_gcc<br>target/stm32f746nucleo144_gcc<br>tools/rommon                                                                                                      | 1.9 *          | 2016-07-23 |
| ARM Cortex-M4アーキテクチャ・GCC依存部パッケージ(担当:TOPPERSプロジェクト教育WG) |                                                                                                                                                                                                       |                |            |
| パッケージ                                                  | ダウンロード                                                                                                                                                                                                | 対応する非依存部のバージョン | リリース日      |
| asp_arch_arm_m4_gcc-1.9.3.tar.gz                       | arch/arm_m_gcc<br>target/stm32407_gcc<br>target/stm32464discovery_gcc<br>target/stm32464nucleo_gcc<br>target/stm32464nucleo64_gcc<br>target/stm32464nucleo144_gcc<br>tools/rommon<br>tools/TrueSTUDIO | 1.9 *          | 2016-07-22 |
| asp_arch_arm_m4_gcc-1.9.2.tar.gz                       | arch/arm_m_gcc<br>target/stm32407_gcc<br>target/stm32464discovery_gcc<br>target/stm32464nucleo_gcc<br>target/stm32464nucleo64_gcc<br>target/stm32464nucleo144_gcc<br>tools/rommon<br>tools/TrueSTUDIO | 1.9 *          | 2016-01-03 |

2016/10/09

TOPPERSプロジェクト認定

64





## TOPPERS/ASPを構築するディレクトリの位置

- TOPPERS/ASPの開発ディレクトリは基礎2、基礎3で共通に使用します
- 2日目の実習プログラムのディレクトリ(base2)と並列の位置にaspのディレクトリを作成してください



base2の実習ディレクトリとaspは並列に作成

2016/10/09

TOPPERSプロジェクト認定

65



## TOPPERS/ASPの解凍

- 開発用のディレクトリにasp-1.9.2.tar.gz、asp\_arch\_arm\_m4\_gcc-1.9.3.tar.gzと、添付のBASE PLATFORMV1.1(asp\_baseplatformv1.1\_mmddyy.tar.gz)を置きます
- asp-1.9.2.tar.gzとasp\_arch\_arm\_m4-gcc-1.9.3.tar.gzの解凍後、asp\_baseplatformv1.0\_mmddyy.tar.gzを解凍します

```
$ ls
asp_arch_arm_m4_gcc-1.9.2.tar.gz asp-1.9.3.tar.gz
asp_base2_m4-1.9.2.tar.gz
$ tar zxvf asp-1.9.2.tar.gz
$ tar zxvf asp_arch_m4_gcc-1.9.3.tar.gz
$ tar zxvf asp_baseplatformv1.1_mmddyy.tar.gz
```

BASE PLATFORMのmmddyyは、リリース日付

2016/10/09

TOPPERSプロジェクト認定

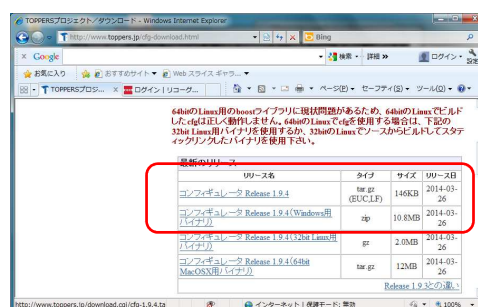
66



## 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/cfgフォルダの下に置きます
- LINUXやCygwinでは、cfgの下に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もよい

```
$ tar zxvf cfg-1.9.1.tar.gz
$ cd cfg
$ ./configure
$ make
```



2016/10/09

TOPPERSプロジェクト認定

67



## TOPPERS/ASPのワークスペースの作成

- aspディレクトリの下にOBJ/STM32F4DISCOVERY\_GCC/SAMPLE1ディレクトリを作成します
- 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

```
$ mkdir OBJ
$ cd OBJ
$ mkdir STM32F4DISCOVERY_GCC
$ cd STM32F4DISCOVERY_GCC
$ mkdir SAMPLE1
$ cd SAMPLE1
$ ../../configure -T stm32f4discovery_gcc
$ ls
Makefile sample1.c sample1.cfg sample1.h
```

2016/10/09

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DfuSeDemoを使って書き込みました
- STM32F4-Discoveryをジャンパーコネクタを使ってUSBシリアルケーブルに接続
- TeraTermを以下の設定で起動します
  - COMn:nはUSBシリアルのCOM番号
  - 115200bps
  - 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

2016/10/09

TOPPERSプロジェクト認定

69



## ROM実行版の確認

- DfuSeDemoで書き込みを行うには.dfuフォーマットのプログラムイメージが必要です
- Makefileに.dfu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DFU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 srec2dfu $(OBJFILE)
325 $(CFG) --pass 3 --kernel asp $(INCLUDES) ¥
326                                     --rom-image $(OBJNAME).srec --symbol-table $(OBJNAME).syms ¥
327                                     -T $(TARGETDIR)/target_check.tf $(CFG_TABS) $<

```

2016/10/09

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
- Discovery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=RAMまたはなしでRAM実行版
  - DBGENV=ROMでROM実行版
- 今回はROM実行版を作成します
- 確認のためにROMファイルを作成します

```
$ make DBGENV=ROM
$ arm-none-eabi-objdump -h -t asp.exe > asp.map
$ ls asp.dfu
asp.dfu
```

2016/10/09

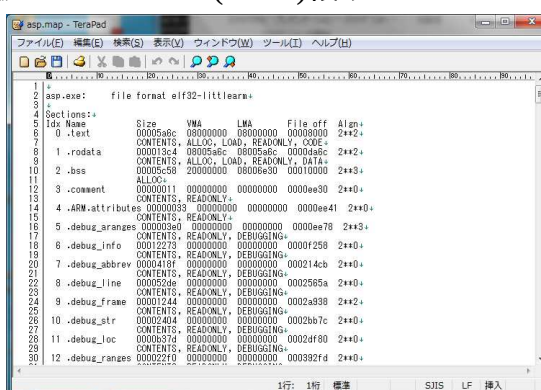
TOPPERSプロジェクト認定

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0x08000000(ROM)領域となっています
  - RAM実行版は0x20000000(RAM)領域



2016/10/09

TOPPERSプロジェクト認定

72



## DfuSeDemoでプログラムを書き込む

- USBケーブルを接続、BOOT0をVDDにショート
- RESETボタンを押すとPCが認識します
- DfuSeDemoでasp.dfuを選択して書き込み



BOOT0とVDDを  
ショート

2016/10/09

TOPPERSプロジェクト認定

73



## ROM実行版の実行

- BOOT0のジャンパーピンをはずし
- RESETボタンを押すと、SAMPLE1が実行します
- この後の実習はROM版で行いますが、以後の実習はRAM版で行いますのでRAM実行版を確認します

```
COM10:115200baud - Tera Term V1
ファイル(F) 編集(E) 設定(S) コントロール(C) ウィンドウ(W) ヘルプ(H)
TOPPERS/ASP Kernel Release 1.9.2 for stm32f4-discovery(Cortex-M4) (Dec 5 2015, 21:38:40)
Copyright (C) 2000-2003 by Embedded and Real-Time Systems Laboratory
Toyohashi Univ. of Technology, JAPAN
Copyright (C) 2004-2014 by Embedded and Real-Time Systems Laboratory
Graduate School of Information Science, Nagoya Univ., JAPAN
Copyright (C) 2015-2016 by Education Working Group TOPPERS PROJECT

System logging task is started on port 1.
Sample program starts (exinf = 0).
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).
```

2016/10/09

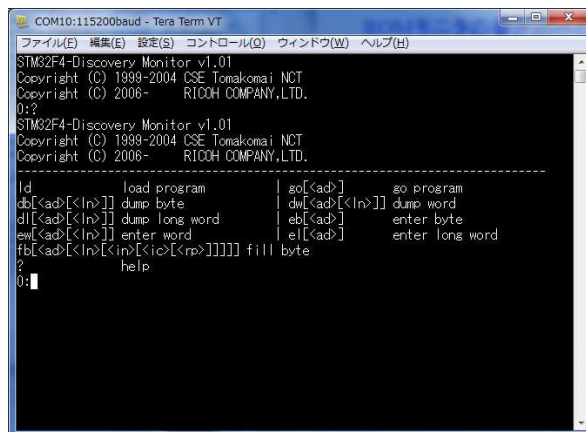
TOPPERSプロジェクト認定

74



## ROMモニタの書き込みと開発環境の確認

- DfuSeDemoを使って、rommon\_discovery.dfuを書き込みます
- RESETキーを押して、ROMモニタの確認を行う



2016/10/09

TOPPERSプロジェクト認定

75



## RAM版のビルド

- Makeコマンドで再ビルドして、RAM実行するasp.srecを作成する

```
$ make clean
$ make
```

2016/10/09

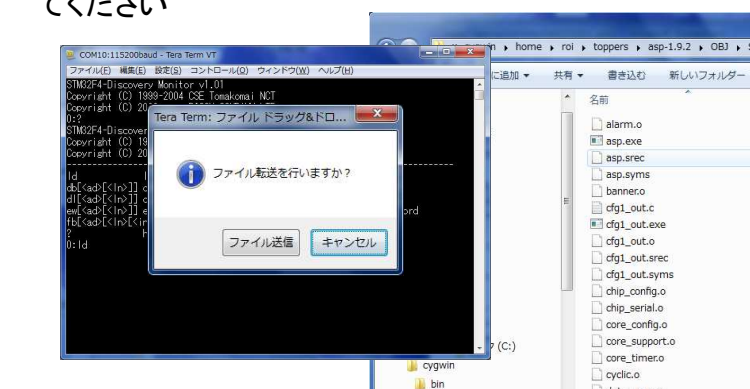
TOPPERSプロジェクト認定

76



## 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



2016/10/09

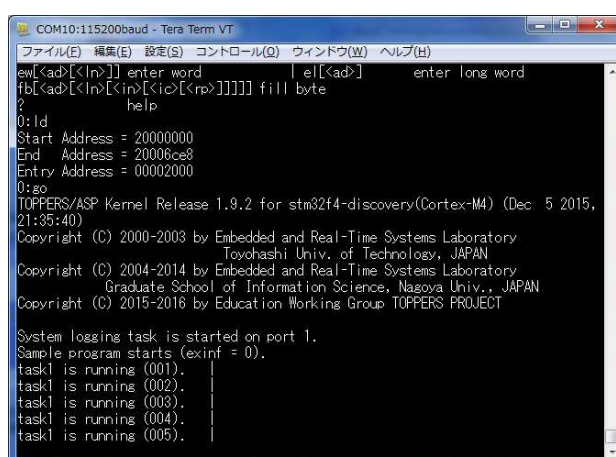
TOPPERSプロジェクト認定

77



## 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



2016/10/09

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

2016/10/09

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                                                |

2016/10/09

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                   |

2016/10/09

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 |

2016/10/09

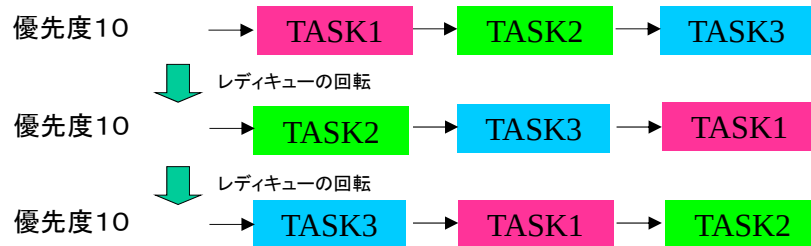
TOPPERSプロジェクト認定

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



2016/10/09

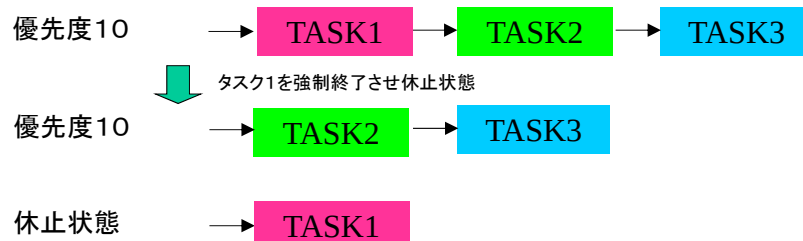
TOPPERSプロジェクト認定

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



2016/10/09

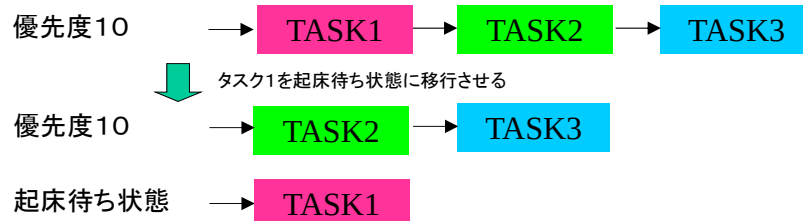
TOPPERSプロジェクト認定

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



2016/10/09

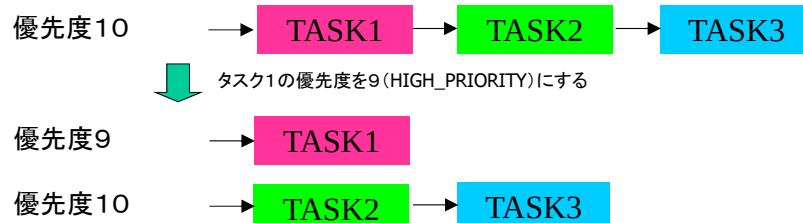
TOPPERSプロジェクト認定

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



2016/10/09

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

2016/10/09

TOPPERSプロジェクト認定

87



## タスクモニタの導入

- RTOSを使用した組込みシステムのメンテナンス性を向上するために、タスクモニタを使用することが有用な手段である
- TOPPERS/ASPにタスクモニタを導入する手順を、開発環境編、「タスクモニタの導入」に記載している
- 以下の演習は、タスクモニタの導入後に行ってください

2016/10/09

TOPPERSプロジェクト認定

88



## 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付をサンプルプログラムを作成する
- デバイスドライバ取り込みを行います
  - モニタプログラムの生成
  - Makefileを修正します
  - コンフィグレーションファイルを修正します

2016/10/09

TOPPERSプロジェクト認定

89



## 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のASP対応版を `pdic/stm32f4xx` 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            |
|--------------------------------|----------------------|-------------------------------|---------------|
| <code>led_init</code>          | <code>void</code>    | <code>inptr_t exinf</code>    | LEDデバイスの初期化   |
| <code>led_in</code>            | <code>uint8_t</code> | <code>void</code>             | LEDの設定値の取り出し  |
| <code>led_out</code>           | <code>void</code>    | <code>uint8_t led_data</code> | LED点灯、消灯を設定   |
| <code>switch_dip_init</code>   | <code>void</code>    | <code>inptr_t exinf</code>    | ディップスイッチの初期化  |
| <code>switch_dip_sense</code>  | <code>uint8_t</code> | <code>void</code>             | ディップスイッチの状態取得 |
| <code>switch_push_init</code>  | <code>void</code>    | <code>inptr_t exinf</code>    | プッシュスイッチの初期化  |
| <code>switch_push_state</code> | <code>uint8_t</code> | <code>void</code>             | プッシュスイッチの状態取得 |

2016/10/09

TOPPERSプロジェクト認定

90



## モニタプログラムの生成

- aspディレクトリの下にOBJ/STM32F4DISCOVERY/MONディレクトリを作成します
- MONディレクトリに入り、-t オプション付きのconfigureコマンドを使用してモニタ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 ..
$ mkdir MON
$ cd MON
$ ../../configure -T stm32f4discovery_gcc -t ../../monitor
$ ls
Makefile sample1.c sample1.cfg sample1.h
```

2016/10/09

TOPPERSプロジェクト認定

91



## デバイスドライバを取り込む

- デバイスドライバのプログラム(device.c)をシステムサービスとして取り込む

```
177
178 #
179 # システムサービスに関する定義
180 #
181 SYSSVC_DIR := $(SYSSVC_DIR) $(SRCDIR)/syssvc ¥
182 SYSSVC_ASMOBJS := $(SYSSVC_ASMOBJS)
183 SYSSVC_COBJS := $(SYSSVC_COBJS) banner.o syslog.o serial.o logtask.o ¥
184 log_output.o vasylog.o t_perror.o strerror.o ¥
185 SYSSVC_CFLAGS := $(SYSSVC_CFLAGS)
186 SYSSVC_LIBS := $(SYSSVC_LIBS)
187 INCLUDES := $(INCLUDES) -I$(SRCDIR)/pdic/stm32f4xx
```

追加

追加分の追加:   
 182 SYSSVC\_ASMOBJS := \$(SYSSVC\_ASMOBJS) **\$(SRCDIR)/pdic/stm32f4xx**   
 183 SYSSVC\_COBJS := \$(SYSSVC\_COBJS) banner.o syslog.o serial.o logtask.o ¥   
 log\_output.o vasylog.o t\_perror.o strerror.o ¥   
 184 **armv7m.o device.o \$(CXXRTS)**

2016/10/09

TOPPERSプロジェクト認定

92



## タスクモニタの取り込み(CFGファイルの修正)

- sample1.cfgに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("syssvc/syslog.cfg");
INCLUDE("syssvc/banner.cfg");
INCLUDE("syssvc/serial.cfg");
INCLUDE("syssvc/logtask.cfg");
INCLUDE("monitor/monitor.cfg");
INCLUDE("pdic/stm32f4xx/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 });

```

2016/10/09

TOPPERSプロジェクト認定

93



## SAMPLE1との違いの確認

- SAMPLE1との違いを確認します
- モニタ用のプログラムの追加が確認できます

```
$ diff ../SAMPLE1 ../MON
```

```

~/toppers/asp-1.9.2/OBJ/STM32F4DISCOVERY_GCC
$ diff ../SAMPLE1 ../MON
diff: ../SAMPLE1/Makefile: MON/Makefile
174d175,176
> # Makefile
> include ../../monitor/Makefile.config
178c182
< SYSSVC_DIR := $(SYSSVC_DIR) $(SRCDIR)/syssvc $(SRCDIR)/library
> SYSSVC_DIR := $(SYSSVC_DIR) $(SRCDIR)/syssvc $(SRCDIR)/library $(SRCDIR)/pdic/
stm32f4xx
180,181c184,185
< SYSSVC_COBJ5 := $(SYSSVC_COBJ5) banner.o syslog.o serial.o logtask.o \
$(CXXRTS)
> SYSSVC_COBJ5 := $(SYSSVC_COBJ5) banner.o syslog.o serial.o logtask.o armv7m.o
device.o $(CXXRTS)
184c188
< INCLUDES := $(INCLUDES)
> INCLUDES := $(INCLUDES) -I$(SRCDIR)/pdic/stm32f4xx
diff: ../SAMPLE1/sample1.cfg: MON/sample1.cfg
12a13,14
> INCLUDE("monitor/monitor.cfg");
> INCLUDE("pdic/stm32f4xx/device.cfg");
18c20
< CRE_TSK(MAIN_TASK, { TA_ACT, 0, main_task, MAIN_PRIORITY, STACK_SIZE, NULL });

```

2016/10/09

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        | ARMV7依存部の追加   |
|-----------------|------------------|---------------|
| stdio/stddev.c  | monitor.c        | arch/armv7m.c |
| stdio/printf.c  | task_expansion.c |               |
| stdio/sprintf.c |                  |               |
| stdio/scanf.c   |                  |               |
| stdio/sscanf.c  |                  |               |

2016/10/09

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
- ダウンロードして、goコマンドでタスクモニタが起動します

```
$ make
```

```

COM1:38400baud - Tera Term V2
ファイル(F) 編集(E) 設定(S) コントロール(C) ウィンドウ(W) ヘルプ(H)
TOPPERS/ASP Kernel Release 1.7.0 for LPC2388(LPC2388(ARM)) (Jun. 8 2013, 18:49:00)
Copyright (C) 2000-2003 by Embedded and Real-Time Systems Laboratory
    Toyohashi Univ. of Technology, JAPAN
Copyright (C) 2004-2011 by Embedded and Real-Time Systems Laboratory
    Graduate School of Information Science, Nagoya Univ., JAPAN
System loading task is started on port 1.
Change task priority 3 to 4 !
ASP Task Monitor Release 1.1.1 for LPC2388(LPC2388(ARM)) (Jun. 8 2013, 18:49:50)
Copyright (C) 2003-2008 by S3 Business Division RICOH COMPANY, LTD., JAPAN
mon: t
cur id pri state pc stack initsize initsize
001 4 WAITING 00007998 40000000 4000090c 1024
* mon 3 RUNNING 40001504 4000040c 2048
002 10 DORMANT 00000248 00000000 40001f0c 4096
004 10 DORMANT 00000248 00000000 40002f0c 4096
005 10 DORMANT 00000248 00000000 40003f0c 4096
006 5 DORMANT 0000061c 00000000 40004f0c 4096
mon:
  
```

2016/10/09

TOPPERSプロジェクト認定

96





## 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]
```

2016/10/09

TOPPERSプロジェクト認定

97



## 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

```

COM10:115200baud - Tera Term VT
コンソール 実行中 終了 接続中 接続中 接続中 接続中
System loading task file started on port 1
dances log task priority 2 to 4
ASP TASK Monitor Release 1.2.0 for stm32f4-discovery(Cortex-M4) (Dec 4 2015, 23:38:15)
Copyright (C) 2003-2008 by GI Business Division RICOH COMPANY, LTD., JAPAN
mon> display task
cur id pri state ps stack initack initize
001 4 WAITING 2000467 2000600 2000608 1024
* non 3 RUNNING 2000c00 2000c08 2048
002 10 DORMANT 2000020 0000000 2000c18 4096
004 10 DORMANT 2000020 0000000 2000d18 4096
006 10 DORMANT 2000020 0000000 2000e18 4096
008 3 DORMANT 200078d 0000000 2000f18 4096
mon> set task 6
006(DORMANT)>task activate
* execute act_tsk(6) ; result = E_OK 1
Sample program starts (xend = 0).
task1 is running (001). |
008(WAITING)>task1 is running (002). |
task1 is running (003). |
task1 is running (004). |

```

2016/10/09

TOPPERSプロジェクト認定

98



## 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

```

COM10:115200baud - Tera Term VT
ファイル(F) 編集(E) 設定(S) コントロール(C) ウィンドウ(W) ヘルプ(H)

System loading task is started on port 1.
change log task priority 5 to 4 !

ASP TASK Monitor Release 1.2.0 for stm324-discovery(Cortex-M4) (Dec 4 2015, 23:38:15)
Copyright (C) 2003-2008 by GJ Business Division RICOH COMPANY,LTD., JAPAN
mon's t 6
006(DORMANT)>t a
  execute act_tsk(6) :: result = E_OK !
Sample program starts (exinf = 0).
task1 is running (001).
006(WAITING)>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).
set logmask=LOG_WARNING logmask=LOG_EMERG !
006 (WAITING):
  
```

2016/10/09

TOPPERSプロジェクト認定

99



## 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

```

COM10:115200baud - Tera Term VT
ファイル(F) 編集(E) 設定(S) コントロール(C) ウィンドウ(W) ヘルプ(H)

set logmask=LOG_WARNING logmask=LOG_EMERG !
006 (WAITING):task terminate
  execute ter_tsk(6) :: result = E_OK !
006 (DORMANT):set task 5
005 (RUNNABLE):task term
  execute ter_tsk(5) :: result = E_OK !
005 (DORMANT):set task 4
004 (RUNNABLE):task term
  execute ter_tsk(4) :: result = E_OK !
004 (DORMANT):set task 3
003 (RUNNABLE):task term
  execute ter_tsk(3) :: result = E_OK !
003 (DORMANT):display task
cur id pri state pc stack inistack inisize
001 4 WAITING 20004e67 20008320 20008560 1024
002 5 RUNNING 2000c080 20008988 2048
x 003 10 DORMANT 20000231 2000d10c 2000c1b8 4096
004 10 DORMANT 20000231 2000a1b8 2000d1b8 4096
005 10 DORMANT 20000231 2000f1b8 2000a1b8 4096
006 5 DORMANT 2000073d 200101f0 2000f1b8 4096
003 (DORMANT):
  
```

2016/10/09

TOPPERSプロジェクト認定

100



## 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、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

2016/10/09

TOPPERSプロジェクト認定

101



## 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Dのポートレジスタ(0x40020c18/0x40020c1a)に直接書き込み、動作を確認する

2016/10/09

TOPPERSプロジェクト認定

102



## 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

2016/10/09

TOPPERSプロジェクト認定

103



### 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**

2016/10/09

TOPPERSプロジェクト認定

104



## 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
 */
ALABEL(dispatcher)
/*
 *
 */
#ifdef TOPPERS_FPU_CONTEXT
    mrs r3, control /* ディスパッチャ実行時はFPCAを一律クリアする */
    bic r3, r3, #CONTROL_FPCA
    msr control, r3
#endif /* TOPPERS_FPU_CONTEXT */
#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 */

```

2016/10/09

TOPPERSプロジェクト認定

105



## 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

```

ALABEL(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 pc, [r1, #TCB_pc] /* 実行再開番地を復帰 */
ALABEL(dispatcher_1)

```

2016/10/09

TOPPERSプロジェクト認定

106



## 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

```

COM10:115200baud - Tera Term VT
[ファイル(F) 編集(E) 設定(S) コントロール(C) ウィンドウ(W) ヘルプ(H)]
006(WAITING)>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).
set logmask=LOG_WARNING logmask=LOG_EMERG !
006(WAITING)>log task 1000 10
time --id=0003 --id=0004 --id=0005 --others --"vacancy"
000049211 1539 18884.9 0000 00000.0 0000 00000.0 4520 00109.3 002437.4
000050218 0092 01000.3 0000 00000.0 0000 00000.0 0092 00006.1 0000000.0
000051216 0091 00999.5 0000 00000.0 0000 00000.0 0092 00003.5 0000000.0
000052215 0092 00998.7 0000 00000.0 0000 00000.0 0092 00003.3 0000000.0
000053215 0092 00998.5 0000 00000.0 0000 00000.0 0092 00003.5 0000000.0
000054216 0092 00998.6 0000 00000.0 0000 00000.0 0092 00003.4 0000000.0
000055215 0091 00998.7 0000 00000.0 0000 00000.0 0091 00003.3 0000000.0
000056215 0091 00998.5 0000 00000.0 0000 00000.0 0092 00003.5 0000000.0
000057216 0092 00998.7 0000 00000.0 0000 00000.0 0092 00003.3 0000000.0
000058215 0092 00998.7 0000 00000.0 0000 00000.0 0092 00003.3 0000000.0
006(WAITING)>

```

LOG TASK <時間(単位MS)> <回数>

2016/10/09

TOPPERSプロジェクト認定

107



## 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

```

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

```

2016/10/09

TOPPERSプロジェクト認定

108



## 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

2016/10/09

TOPPERSプロジェクト認定

109



## 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](#)

2016/10/09

TOPPERSプロジェクト認定

110



### 演習:タスクモニタのコマンドの拡張を行う

- タスクモニタを組み込みシステム用に拡張する
- タスクモニタはカテゴリとコマンドを追加できる
- コマンド拡張機能でLEDコマンドを追加して、LEDの点灯、消灯を行う
- カテゴリ:DEVICE、コマンド:LED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(1-4)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
  - 以下、DEVICEカテゴリにLEDコマンド等々を追加

DEVICE LED (LED番号1-4) (消灯-0、点灯-1)  
DIP (DIPSW番号1-4) (OFF-0、ON-1)

2016/10/09

TOPPERSプロジェクト認定

111



### 演習:カテゴリとコマンドの追加

- コマンドリンク構造体を作成し、setup\_commnad関数を使用して新たなカテゴリとコマンドの追加ができる

以下、monitor.hのコマンドリンク構造体記述

```
/*
 * コマンドデスバッチ用の構造体定義
 */
typedef struct _COMMAND_INFO {
    const char *command;          /* コマンド文 */
    int_t (*func)(int argc, char **argv); /* 実行関数 */
} COMMAND_INFO;

typedef struct _COMMAND_LINK COMMAND_LINK;
struct _COMMAND_LINK {
    COMMAND_LINK *pcnext;
    int num_command;
    const char *command;          /* 主コマンド文 */
    int_t (*func)(int argc, char **argv); /* 実行関数 */
    const char *help;             /* ヘルプ文 */
    const COMMAND_INFO *pcinfo;
};
```

2016/10/09

TOPPERSプロジェクト認定

112





## 演習: command.cの取り込み

- 拡張コマンドのソースファイルcommand.cはbase2ディレクトリにあります、command.cをOBJ/STM32FDISCOVERY\_GCC/MONディレクトリにコピーする
- コンパイルの対象となるようにMakefileを書き換えます

```

153 #
154 # アプリケーションプログラムに関する定義
155 #
156 APPLNAME = sample1
157 APPLDIR =
158 APPL_CFG = $(APPLNAME).cfg
159
160 APPL_DIR = $(APPLDIR) $(SRCDIR)/library
161 APPL_ASMOBS =
162 ifdef USE_CXX
163   APPL_CXXOBS = $(APPLNAME).o
164   APPL_COBS = command.o
165 else
166   APPL_COBS = $(APPLNAME).o command.o
167 endif

```

追加

2016/10/09

TOPPERSプロジェクト認定

113



## 演習: command.cの実装

- DEVICEカテゴリを実行するcommand.cを参照する

```

024 static const COMMAND_INFO device_command_info[NUM_DEVICE_CMD] = {
025   {"LED",          led_func},
026   {"DIP",          dip_func},
027   {"CUP",          cup_func}
028 };
029
030 static const char device_name[] = "DEVICE";
031 static const char device_help[] =
032   " Device LED (no) (on-1,off-0) led control\n"
033   "      DIP (no) (on-1,off-0) dipsw control\n"
034   "      CUP command      cup control\n";
035
036 static const uint16_t led_pattern[4] = {
037   LED01, LED02, LED03, LED04
038 };
039
040 static COMMAND_LINK device_command_link = {
041   NULL,
042   NUM_DEVICE_CMD,
043   device_name,
044   NULL,
045   device_help,
046   &device_command_info[0]
047 };

```

コマンドリンク  
構造体

2016/10/09

TOPPERSプロジェクト認定

114



## 演習: command.cの参照

- 個々のコマンドは引数をargc(引数の数),argv(引数の文字列のポインタ配列)の形状で渡される

```

/*
 * LED設定コマンド関数
 */
static int_t led_func(int argc, char **argv)
{
    int_t arg1=0, arg2;
    uint16_t reg;

    if(argc < 3)
        return -1;
    arg1 = a2i(argv[1]);
    arg2 = a2i(argv[2]);
    if(arg1 >= 1 && arg1 <= 4){
        reg = led_in();
        if(arg2 != 0)
            reg |= led_pattern[arg1-1];
        else
            reg &= ~led_pattern[arg1-1];
        led_out(reg);
    }
    return 0;
}

```

引数1, 2から数値を取り出す

デバイスドライバ関数を呼び出す

2016/10/09

TOPPERSプロジェクト認定

115



## 演習: command.cを参照

- コマンドリンク構造体をseup\_command関数を用いて、タスクモニタに登録すれば新たなコマンドが追加できます
- コマンドの追加はATT\_INIサービスコールを使用する
- ATT\_INIサービスコールは初期化ルーチンを実行する、コマンドの追加処理を初期化で実行する

```

061 /*
062 * DEVICEコマンド設定関数
063 */
064 void device_info_init(intptr_t exinf)
065 {
066     setup_command(&device_command_link);
067 }

```

属性(Cかアセンブラか)

拡張情報  
(初期化ルーチンへの引数)初期化ルーチンの  
起動番地

ATT\_INI({ATR intatr, intptr\_t exinf, FP intrtn});

2016/10/09

TOPPERSプロジェクト認定

116



## 演習: sample.hの書き換え

- ATT\_INIサービスコールに初期化関数を認識させるために、device\_info\_init関数のプロトタイプ宣言をsample.hに追加する

```
079 /*
080 * 関数のプロトタイプ宣言
081 */
082 #ifndef TOPPERS_MACRO_ONLY
083
084 extern void      task(intptr_t exinf);
085 extern void      main_task(intptr_t exinf);
086 extern void      tex_routine(TEXPTN texptn, intptr_t exinf);
087 #ifdef CPUEXEC1
088 extern void      cpuexc_handler(void *p_excinf);
089 #endif /* CPUEXEC1 */
090 extern void      cyclic_handler(intptr_t exinf);
091 extern void      alarm_handler(intptr_t exinf);
092 extern void      device_info_init(intptr_t exinf);
093
094 #endif /* TOPPERS_MACRO_ONLY */
```

追加

2016/10/09

TOPPERSプロジェクト認定

117



## 演習: sample1.cfgの修正

- コンフィギュレーションファイルsample1.cfgを修正し device\_info\_init関数を初期化関数と実行するように ATT\_INIサービスコールを追加する

```
016 #include "sample1.h"
017 CRE_TSK(TASK1, { TA_NULL, 1, task, MID_PRIORITY, STACK_SIZE, NULL });
018 CRE_TSK(TASK2, { TA_NULL, 2, task, MID_PRIORITY, STACK_SIZE, NULL });
019 CRE_TSK(TASK3, { TA_NULL, 3, task, MID_PRIORITY, STACK_SIZE, NULL });
020 CRE_TSK(MAIN_TASK, { TA_NULL, 0, main_task, MAIN_PRIORITY, STACK_SIZE, NULL });
021 DEF_TEX(TASK1, { TA_NULL, tex_routine });
022 DEF_TEX(TASK2, { TA_NULL, tex_routine });
023 DEF_TEX(TASK3, { TA_NULL, tex_routine });
024 CRE_CYC(CYCHDR1, { TA_NULL, 0, cyclic_handler, 2000, 0 });
025 CRE_ALM(ALMHDR1, { TA_NULL, 0, alarm_handler });
026 ATT_INI({ TA_NULL, 0, device_info_init });
027 #ifdef CPUEXEC1
028 DEF_EXC(CPUEXEC1, { TA_NULL, cpuexc_handler });
029 #endif /* CPUEXEC1 */
```

2016/10/09

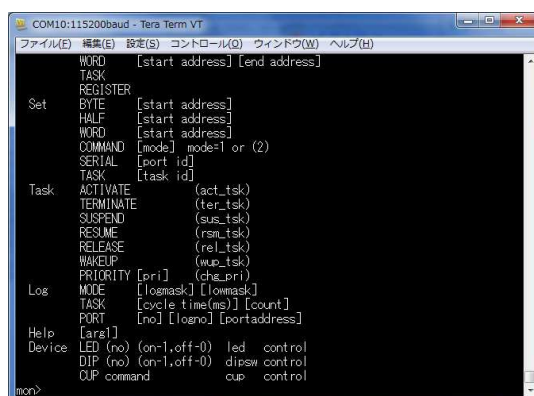
TOPPERSプロジェクト認定

118



## 演習:ビルドとFLASH-ROM書き込み

- ビルドを行い、FLASH-ROMに書き込む
- 実行後、helpコマンドを実行すると、DEVICEカテゴリのLEDコマンドが追加されている



```

COM10:115200baud - Tera Term VT
ファイル(F) 編集(E) 設定(S) コントロール(C) ウィンドウ(W) ヘルプ(H)

TASK [start address] [end address]
REGISTER
Set BYTE [start address]
    HALF [start address]
    WORD [start address]
    COMMAND [mode] mode=1 or (2)
    SERIAL [port id]
    TASK [task id]
Task ACTIVATE (act_tsk)
    TERMINATE (ter_tsk)
    SUSPEND (sus_tsk)
    RESUME (rsm_tsk)
    RELEASE (rel_tsk)
    WAKEUP (wup_tsk)
    PRIORITY [pri] (che_pri)
Log MODE [logmask] [lowmask]
    TASK [cycle time(ms)] [count]
    PORT [no] [logno] [portaddress]
Help [arg1]
Device LED (no) (on-1,off-0) led control
    DIP (no) (on-1,off-0) dipsw control
    CUP command cup control
mon>
  
```

2016/10/09

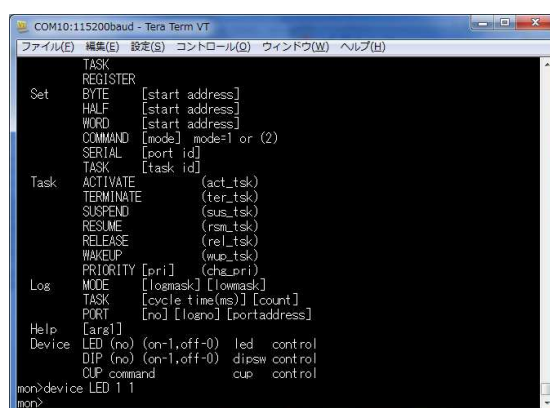
TOPPERSプロジェクト認定

119



## 演習:LEDコマンドの実行確認

- DEVICE LED 1 1(return)でLED1が点灯することを確認してください
- 他のLEDのON/OFFを行ってください



```

COM10:115200baud - Tera Term VT
ファイル(F) 編集(E) 設定(S) コントロール(C) ウィンドウ(W) ヘルプ(H)

TASK [start address] [end address]
REGISTER
Set BYTE [start address]
    HALF [start address]
    WORD [start address]
    COMMAND [mode] mode=1 or (2)
    SERIAL [port id]
    TASK [task id]
Task ACTIVATE (act_tsk)
    TERMINATE (ter_tsk)
    SUSPEND (sus_tsk)
    RESUME (rsm_tsk)
    RELEASE (rel_tsk)
    WAKEUP (wup_tsk)
    PRIORITY [pri] (che_pri)
Log MODE [logmask] [lowmask]
    TASK [cycle time(ms)] [count]
    PORT [no] [logno] [portaddress]
Help [arg1]
Device LED (no) (on-1,off-0) led control
    DIP (no) (on-1,off-0) dipsw control
    CUP command cup control
mon>device LED 1 1
mon>
  
```

2016/10/09

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

2016/10/09

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バリエーションを持つため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

2016/10/09

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。