ET2007 TOPPERSセッション

TOPPERS新世代カーネルの仕様解説

2007年11月15日

高田 広章

NPO法人 TOPPERSプロジェクト 会長

名古屋大学 大学院情報科学研究科 教授

附属組込みシステム研究センター長

Email: hiro@ertl.jp　URL: http://www.ertl.jp/~hiro/

TOPPERS

TOPPERSプロジェクトの狙い

現世代のリアルタイムOSの決定版の構築

！約20年間に渡るITRON仕様の技術開発成果をベースに

- ITRON仕様がかかえる*過剰な重複投資と過剰な多様性*の問題を解決（または軽減）

次世代のリアルタイムOS技術の開発

- 組込みシステムの要求に合致し，ITRONの良さを継承する次世代のリアルタイムOS技術を開発

  Linuxと類似のOSをもう１つ作っても意味がない！

- オープンソースソフトウェア化により産学官の力を結集

組込みシステム技術者の育成への貢献

- オープンソースソフトウェアを用いた教育コースや教材を開発し，それを用いた教育の場を提供
- 開発した教育コンテンツもオープン化

TOPPERSプロジェクトの現状

！プロジェクトの3つの狙いの進捗をチェック

現世代のリアルタイムOSの決定版の構築

- ソフトウェア開発という意味では一定の成果
- ITRON仕様がかかえる過剰な重複投資と過剰な多様性の問題を解決するには，さらなる普及が必要

次世代のリアルタイムOS技術の開発

- 次世代のリアルタイムOS技術を各方面で探究
  - メモリ保護，時間保護，マルチコアプロセッサ
  - コンポーネント技術，高信頼性・安全性

組込みシステム技術者の育成への貢献

- 各種の教材を開発・オープン化し，一定の成果

➡ ***速度が十分とは言えかもしれないが，順調に進捗***

TOPPERSプロジェクトの次のステップ

!次のステップへ踏み出す時期に来ている

次世代リアルタイムOSの普及版の開発

- 次世代のリアルタイムOS技術に関するこれまでの研究開発成果から，有望な技術が見えてきた
- これまでの経験を反映して，普及版のOSを開発する

現世代のリアルタイムOSの完成度をさらに上げる

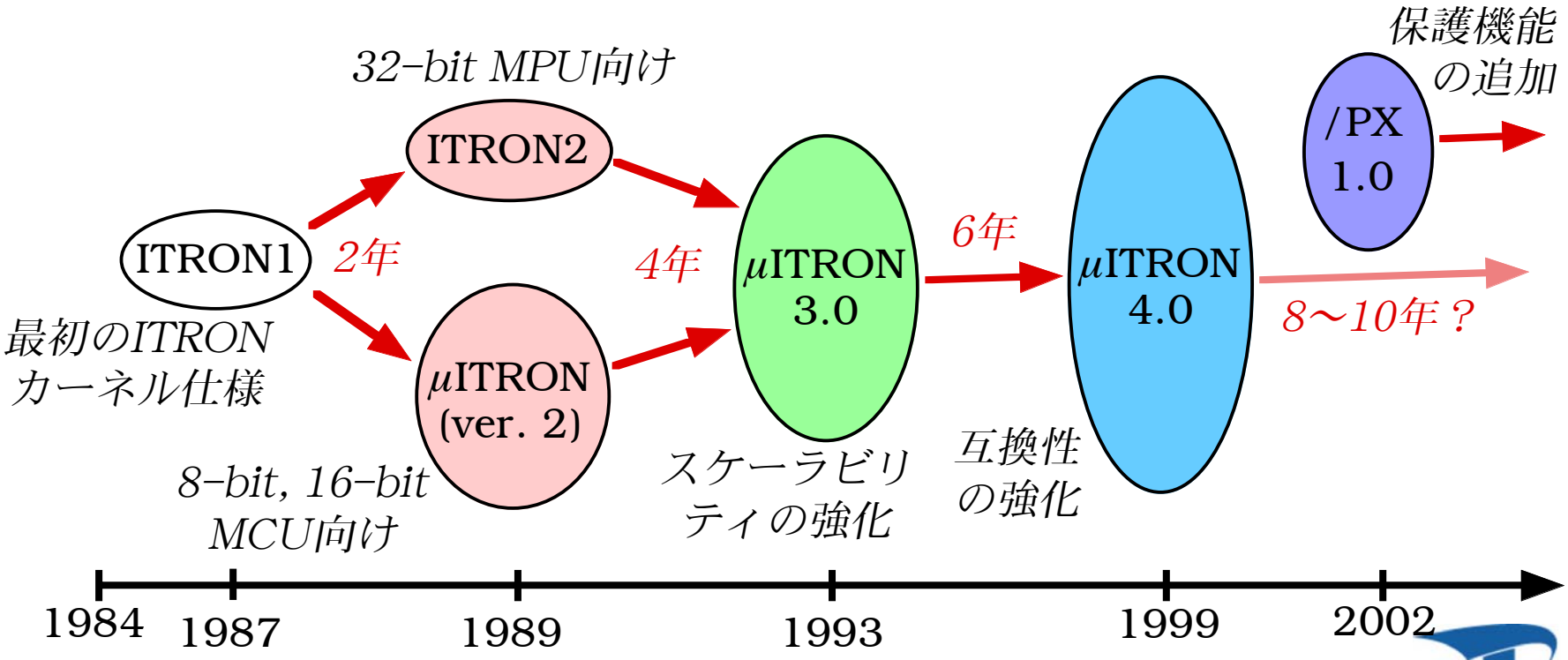
- 先へ進むために足場を固めることが必要（大きい建物を建てるには，しっかりとした基礎が必要）

自動車制御システム向けプラットフォームの開発

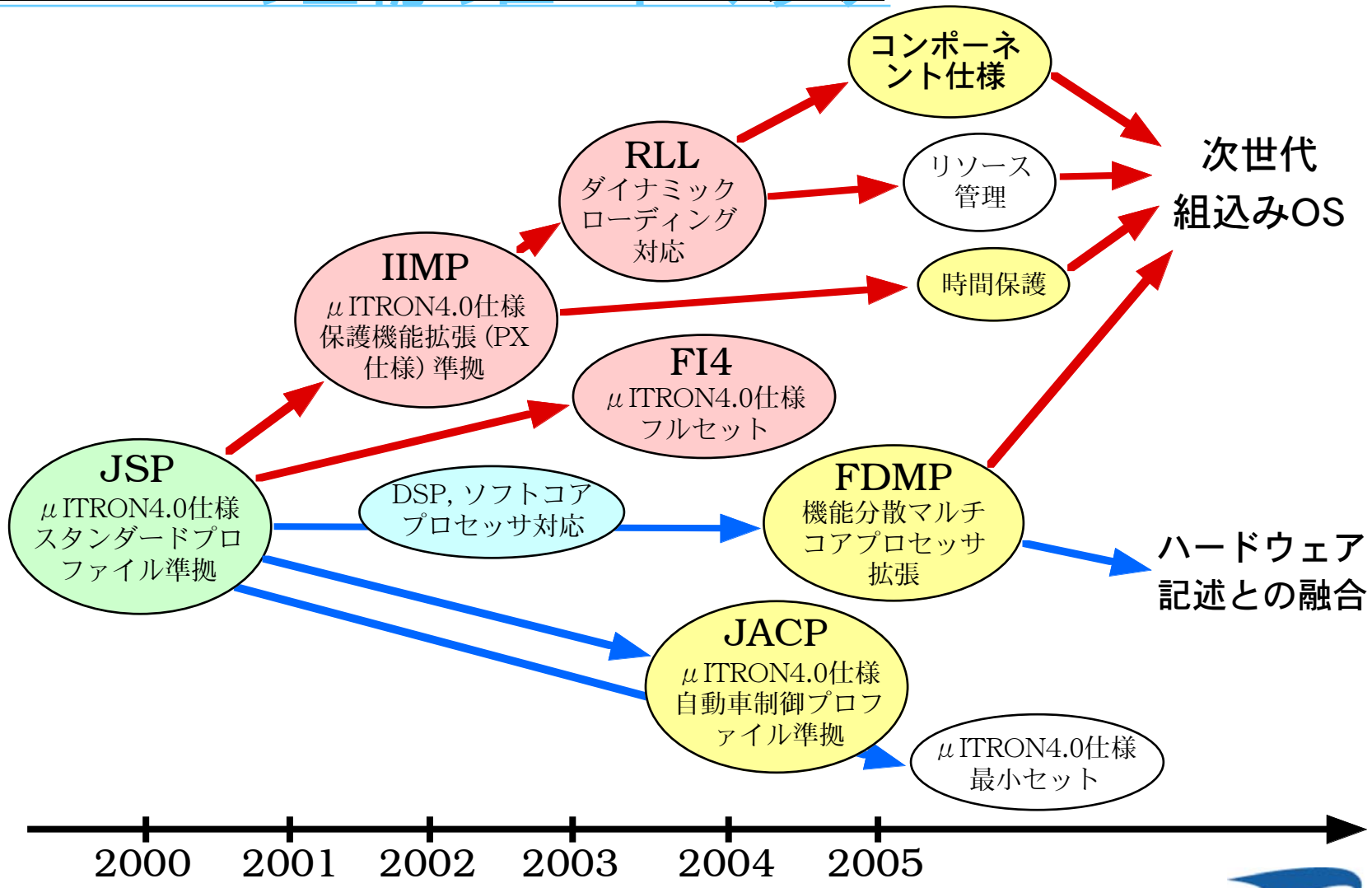
- 日本では，自動車向けの組込みシステム技術が，組込みシステム技術全体のドライビングフォースに
- プラットフォーム開発の成果をOSにも反映

ITRONカーネル仕様の歴史

- μITRON4.0仕様が公表されてから，すでに8年が経過
- μITRON4.0仕様は，現世代のリアルタイムOS技術の範囲では完成度の高い仕様になっているとは言え，組込みシステムの発展により不十分に



TOPPERSの当初のロードマップ



TOPPERS新世代カーネル仕様の必要性

組込みシステムにおける要求の変化

- システム/ソフトウェアの大規模化・複雑化
- これまで以上に高い信頼性・安全性
- 小さい消費エネルギーで高い性能
- 適用範囲の拡大（より小規模なシステムへも）

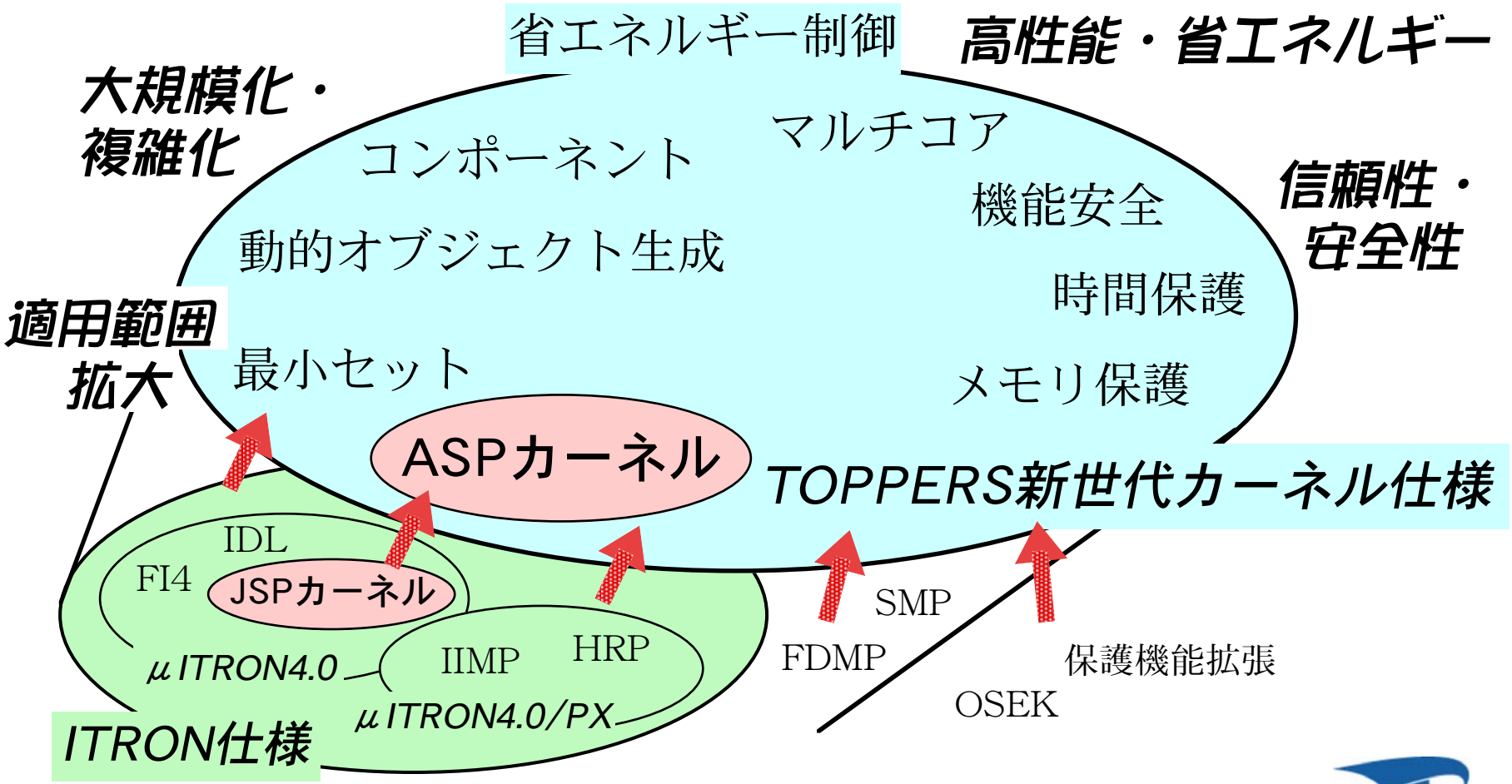
μITRON4.0仕様以降の技術開発成果の取込み（上記と対応）

- マルチコアプロセッサ対応
- 保護機能（メモリ保護, 時間保護）
- 機能安全対応

μITRON4.0仕様で完成度が低かった箇所の改良

- システムコンフィギュレーション手順など

TOPPERS新世代カーネル仕様の位置付け ～ ITRON仕様からの進化



TOPPERS

TOPPERS/ASPカーネルの概要

位置付け

- TOPPERS新世代カーネルの基盤（出発点）となるリアルタイムカーネル

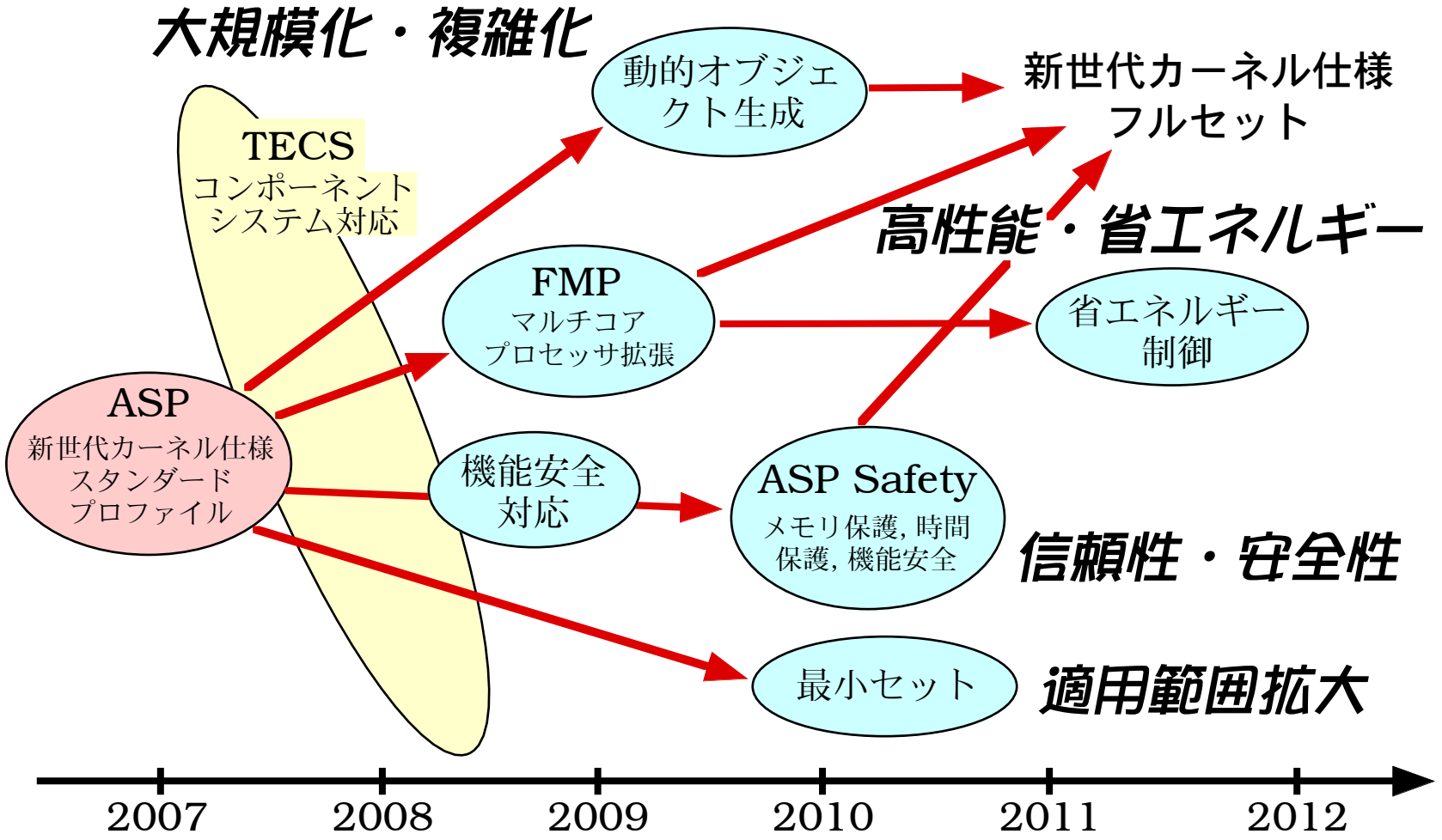
仕様の概要

- TOPPERS/JSPカーネルに対して，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標準割込み処理モデル」によりプロセッサによらず標準化
- システムコンフィギュレーションの仕組みの見直し
- TOPPERS組込みコンポーネントシステムへの対応（検討中）
- 信頼性・安全性の向上については細かな改良の積み重ね
- いくつかの独自の機能拡張

TOPPERS新世代カーネルロードマップ



※ ASP以外のカーネルの名前は仮称

マルチコアプロセッサ対応

TOPPERS/FMPカーネル（仮称）

- 対称型（SMP）またはそれに近いマルチコアプロセッサに対応した，リアルタイム性と柔軟性を両立させるリアルタイムカーネル
- TOPPERS/FDMPカーネルの仕様をベースに，タスクをマイグレーションさせるAPIなどを追加
- ASPカーネルをマルチコア拡張する形で実装

開発計画

- 名古屋大学 組込みシステム研究センターにおいて，2007年度下期～2008年度にかけて開発する計画（すでに開発が始まっている）
- 複数の半導体メーカの5種類程度のターゲットプロセッサを当初からサポートする予定

機能安全対応

開発内容

- 機能安全規格の認証を取れるレベルのリアルタイムカーネル
- リアルタイムカーネルに対する安全要求分析の結果，新しい機能の必要性が出てくる可能性
- 設計ドキュメントの整備や，検証の実施も重要な課題

開発計画

- 経済産業省の平成18年度戦略的基盤技術高度化支援事業の採択テーマとして，(株)ヴィッツを中心に，2006年12月より3年間のプロジェクトで開発中
- IEC 61508のSIL 3の認証が取れるレベルのリアルタイムカーネルと車載ネットワークミドルウェアを開発
- ASPカーネルをベースに

コンポーネント対応

開発内容

- カーネルをTECS (TOPPERS組込みコンポーネントシステム) から呼び出せるようにする
- カーネルの周辺モジュール (システムログ機能, シリアルドライバ等) をTECS対応にする

開発計画

- コンポーネント仕様WGの中で，ASPカーネルをTECS対応にする開発が進行中

その他の拡張の内容

メモリ保護，時間保護

- メモリ保護と時間保護のための機能を追加する
- TOPPERS/OSEKシリーズに対する保護機能拡張の研究開発が進行中であり，その開発成果を取り込む

動的オブジェクト生成

- カーネルオブジェクトの動的な生成・削除機能をサポートする

最小セット

- リアルタイムカーネルを使いはじめるユーザのためのASPカーネルの下位互換のリアルタイムカーネル
- 待ち状態をサポートせず，すべてのタスクおよび割込みハンドラが，同一のスタックを使って実行する

TOPPERS新世代カーネル仕様の設計方針

(1) μITRON4.0仕様をベースに拡張・改良を加える
- 多くの実績があるμITRON4.0仕様をベースに
- μITRON4.0仕様の不十分な点は積極的に拡張・改良

(2) ソフトウェアの再利用性を重視する
- ソフトウェアの再利用性向上のために，少々のオーバヘッドがあっても，ターゲット依存項目を減らす

(3) 高信頼・安全なシステム構築を支援する
- アプリケーションに誤用されにくい仕様とする
- 妥当なオーバヘッドで救済できる誤用は救済する

(4) アプリケーション構築に必要な機能は積極的に取り込む
- 多くのアプリケーションに共通に必要な機能を実装
- ただし，(1)～(3)の方針を満たすことが前提

TOPPERS/ASPカーネルの仕様設計方針

- 枯れた技術で実装できる範囲の仕様に留める
  - 完成度の高いリアルタイムカーネルを実現するため
  - 保護機能やマルチプロセッサ対応は，ASPカーネルには取り込まず，ASPカーネルを拡張する形で実現
- 主な適用対象
  - 高い信頼性・安全性・リアルタイム性を要求される組込みシステム
  - プログラムサイズ（バイナリ）が数十KB～1MB程度
- カーネル内での動的なメモリ管理は行わない（静的OS）
  - システム稼働中のメモリ不足への対処が難しいため
  - カーネルオブジェクトは静的に生成
  - ただし，固定長メモリプール機能はサポートする

TOPPERS/ASPカーネルの仕様概要

ベースとする仕様

- μITRON4.0仕様のスタンダードプロファイルをベースとして，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
- 割込み処理機能については，「TOPPERS標準割込み処理モデル」に準拠する

拡張の内容

- スタンダードプロファイル外の機能の一部導入
  - イベントフラグの複数タスク待ち (TA_WMUL属性)
  - アラームハンドラ (CRE_ALM, [i]sta_alm, [i]stp_alm)
  - 割込みサービスルーチン (ATT_ISR)
  - 割込み管理機能 (dis_int, ena_int, chg_ipm, get_ipm)
  - オブジェクトの状態参照機能 (ref_xxx)

拡張の内容 ～ 続き

- 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機能
  - 割込み要求ラインの属性の設定 (CFG_INT) → 後で説明
  - 同期・通信オブジェクトの再初期化機能 (ini_xxx)
  - 優先度データキュー機能 (xxx_pdq) → 後で説明
  - 自タスクの拡張情報の参照 (get_inf)
  - カーネルの終了機能 (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組込みコンポーネント仕様の導入 (検討中)
  - デバイスドライバやシステムログ機能との接続

改良の内容

- μITRON4.0仕様に対する変更
  - ITRON標準データ型の見直し
  - 非タスクコンテキストからのext_tsk → 後で説明
  - 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 → 後で説明
- JSPカーネルにおける実装定義／依存事項に対する変更
  - インクルードファイルの構成の整理
  - 割込み処理／例外処理関係の型定義
  - 処理単位の実行開始／リターン時のシステム状態
  - isig_timの扱い → 後で説明
  - 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
- システムコンフィギュレーション処理の見直し

TOPPERS標準割込み処理モデル

策定の背景

- μITRON仕様では，標準化によるオーバヘッドを避けるために，割込み処理機能は弱く標準化するに留めており，プロセッサ／カーネルによる違いが大きい
- μITRON4.0仕様策定時点とはトレードオフが変化し，ソフトウェアの生産性向上のためには，若干のオーバヘッドは許される状況に

策定の目標

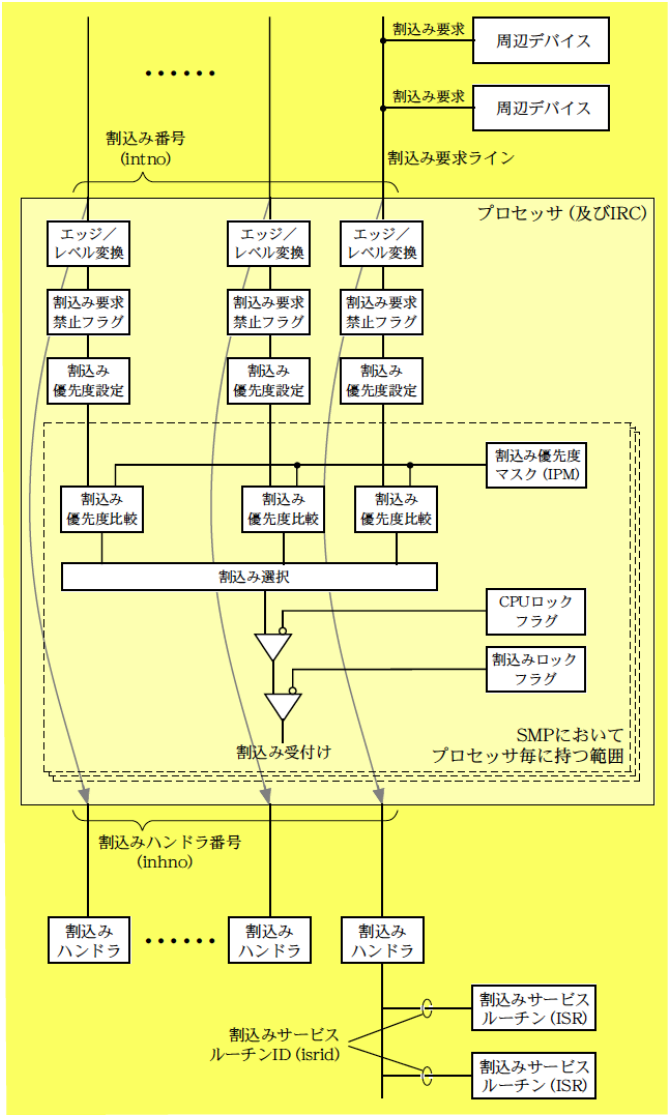
- ! プロセッサの割込みアーキテクチャの詳細を知ることなしに，割込みを用いたアプリケーション構築を可能に
- 割込み処理に関わるソフトウェアの再利用性の向上
- オーバヘッドの増加は最小限に抑える

割込み処理モデルの概念図

- 右図は，割込み処理モデルの持つすべての機能が，ハードウェアで実現されているとして描いた図
- 実際のハードウェアで不足する機能はソフトウェアで実現する

μITRON4.0仕様との大きな違い

- 割込み優先度の概念と割込み優先度マスク（IPM）を導入
- 割込み要求ラインの属性を標準化．属性を設定するためのAPIを用意
- マルチコアについても考慮



割込み優先度と割込み優先度マスク（IPM）

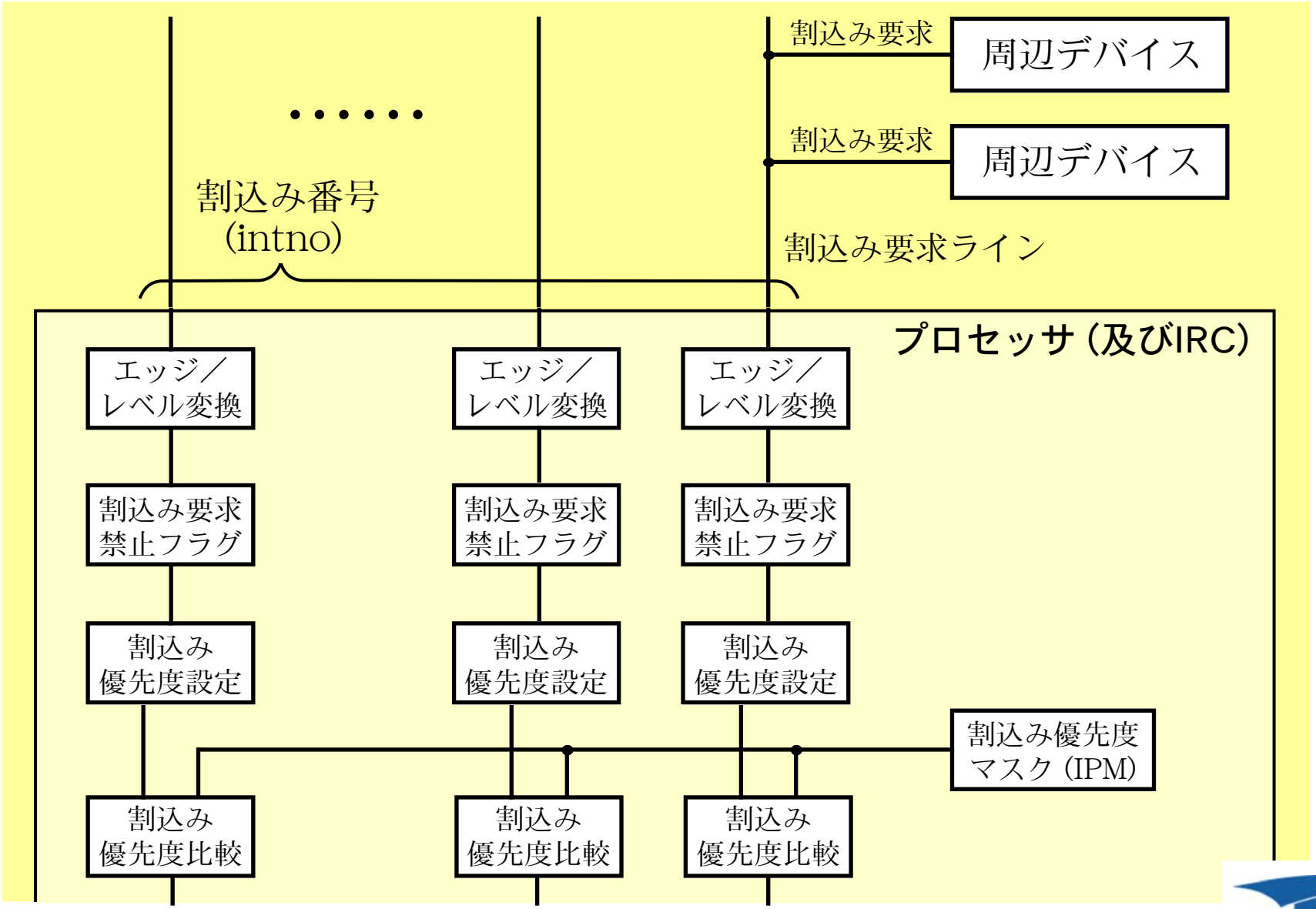
- 各割込み要求は，割込み優先度を持つ
  - 割込み優先度の段階数はターゲット依存（1段階にすると多重割込み無しとなる）
  - 割込み優先度は負の値（タスク優先度と比較可能）
- 割込み優先度を基準に割込みをマスクするための割込み優先度マスク（IPM）を持つ
  - 現在のIPMと同じかそれより低い優先度を持つ割込みはマスクされる（多重割込みの制御）
  - IPM＝TIPM_ENAALL（＝0）の時は，いずれの割込みもマスクされない
- 発生している割込み要求の中で，優先度の高い割込み要求から受け付けることは***仮定しない***
- chg_ipm, get_ipm：IPMの現在値の設定/参照

割込み要求ライン

- 周辺デバイス（割込み源）からの割込み要求を伝える信号線
  - 1つの割込み要求ラインに対して複数の割込み源が接続される場合もある
- 割込み番号（intno）で識別

割込み禁止フラグ

- 割込み要求ライン毎に，割込み要求をマスクするための割込み要求禁止フラグを持つ
- dis_int, ena_int：割込み要求禁止フラグのセット/クリア．割込み番号をパラメータに取る
  - アプリケーションが積極的に使うことは推奨しない（ポータビリティが下がるため）


割込み要求
周辺デバイス
割込み要求
周辺デバイス
・・・・・・
割込み番号
(intno)
割込み要求ライン
プロセッサ（及びIRC）
エッジ／
レベル変換
エッジ／
レベル変換
エッジ／
レベル変換
割込み要求
禁止フラグ
割込み要求
禁止フラグ
割込み要求
禁止フラグ
割込み
優先度設定
割込み
優先度設定
割込み
優先度設定
割込み優先度
マスク（IPM）
割込み
優先度比較
割込み
優先度比較
割込み
優先度比較

割込み要求ラインの属性

- 割込み要求ライン毎に次の属性を持つ
  - 割込み優先度
  - 割込み禁止フラグ
  - トリガ方式 (レベルトリガかエッジトリガか)
- CFG_INT : 割込み要求ラインの属性の設定

  CFG_INT(INTNO intno, { ATR intatr, PRI intpri })

  - 割込み属性 (intatr) として設定できる属性：

    TA_DISINT (割込み禁止フラグをセット)

    TA_LEVEL (レベルトリガ)

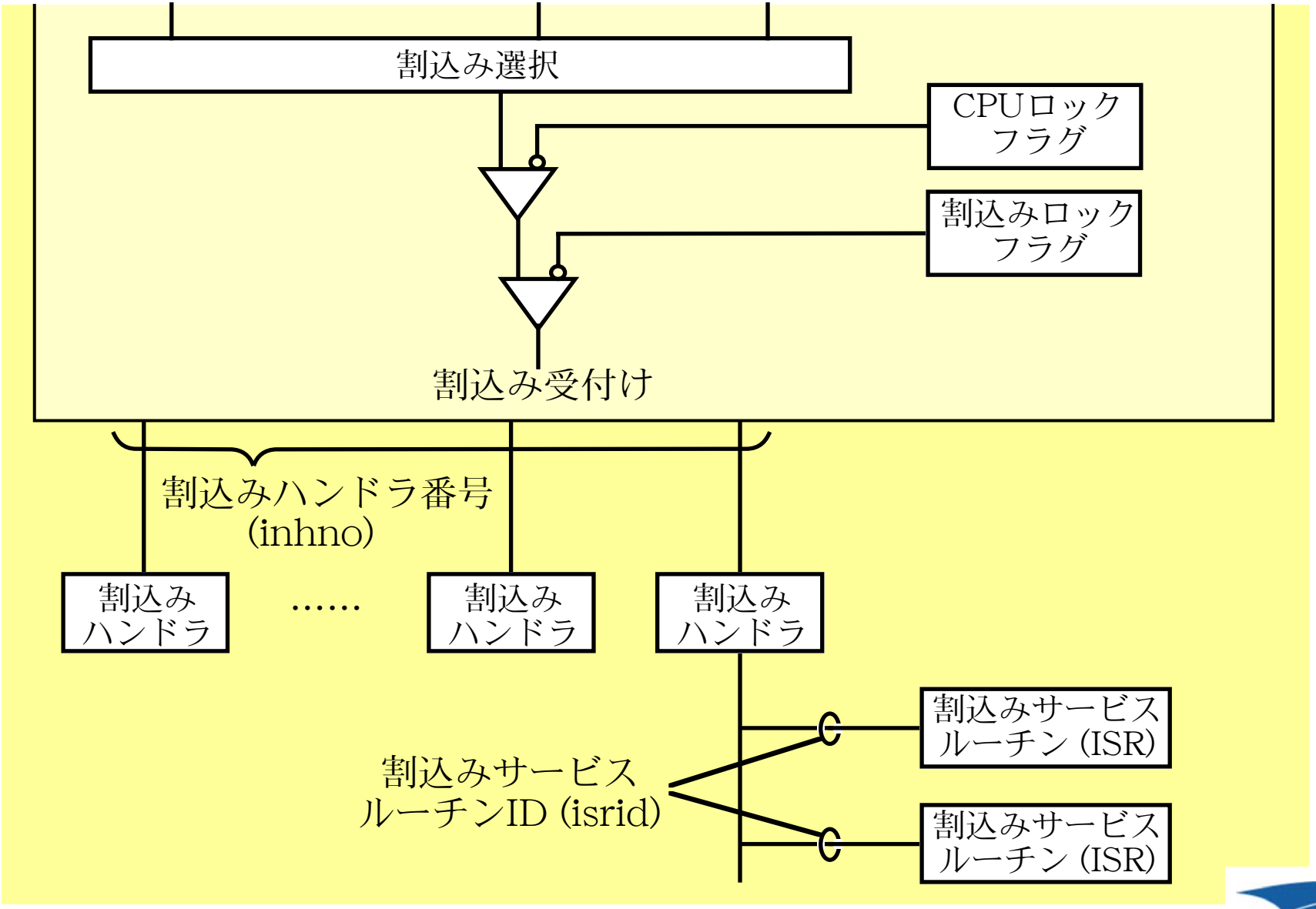
    その他のターゲット依存の属性

割込みサービスルーチン（ISR）

- プロセッサの割込みアーキテクチャに依存せず，周辺デバイス（割込み源）のみに依存して記述できる
  - ！ポータビリティが高い
- 割込み源毎に登録
  - 割込み要求ライン（割込み番号）を指定して登録
  - 1つの割込み要求ラインに対して複数登録できる
- 呼出し順序をISR優先度により制御可
- ATT_ISR：ISRの登録

```
ATT_ISR({ ATR isratr, intptr_t exinf,
                                    INTNO intno, PRI isrpri })
```

- 動的に登録/解除する場合（ASPカーネルでは未サポート）には，ID番号で識別


割込み選択
CPUロック
フラグ
割込みロック
フラグ
割込み受付け
割込みハンドラ番号
(inhno)
割込み
ハンドラ
……
割込み
ハンドラ
割込み
ハンドラ
割込みサービス
ルーチンID (isrid)
割込みサービス
ルーチン (ISR)
割込みサービス
ルーチン (ISR)

割込みハンドラ

- ISRを呼び出すルーチンで，カーネル内に持つ（コンフィギュレータが生成する）のが原則
- 特殊なケースに対応するために，ボードサポートパッケージ（BSP）等で用意することもできる
- 割込みハンドラ番号（inhno）を指定して登録
- DEF_INH：割込みハンドラの登録

```
DEF_INH(INHNO inhno, { ATR inhatr, INTHDR inthdr })
```

CPUロックフラグ

- カーネルの管理外の割込みを除くすべての割込みをマスクするフラグ

割込みロックフラグ

- マスクできない割込み（NMI）を除くすべての割込みを禁止するフラグ

ASPカーネルにおける信頼性・安全性向上

細かな改良の積み重ね

例1）ext_tskの仕様変更

- μITRON4.0仕様では，ext_tskはリターンすることのないサービスコールと規定

？*非タスクコンテキストからext_tskを呼んだらどうするか*

- μITRON4.0仕様では実装定義
- JSPカーネルでは実行を継続するよう頑張るが，結果は保証しない．安全性を重視するシステムでは，危険の可能性を検出すれば，早期にリカバリすべき場合も
- カーネルをダウンさせる方法では，アプリケーションでのリカバリの余地をなくす

➡ リカバリ方法をアプリケーションで決められるように，E_CTXエラーを返す仕様に変更

例2) 処理単位の実行開始/リターン時のシステム状態

- タスク例外処理ルーチンの例

【TOPPERS新世代カーネル】

	CPUロック/ロック解除状態	割込み優先度マスク(IPM)	ディスパッチ禁止/許可状態	タスク例外禁止/許可状態
実行開始条件	解除	任意	任意	許可
実行開始時処理	そのまま	そのまま	そのまま	禁止する
リターン前	原則解除	原則元に	原則元に	原則禁止
リターン時処理	解除する	元に戻す	元に戻す	許可する

【μITRON4.0仕様】…この表は独自に作成

実行開始条件	解除	—	任意	許可
実行開始時処理	そのまま	—	そのまま	禁止する
リターン前	解除	—	任意	任意
リターン時処理	未規定	—	そのまま	許可する

例3) カーネルの用いる管理領域の分離と優先度データキュー

- μITRON4.0仕様では，メールボックスに送信するメッセージの先頭の領域（数バイト）をカーネルが利用
- アプリケーションが誤ってこの領域を書き換えると，カーネルの中で不具合が発生する
- μITRON4.0仕様 保護機能拡張（PX仕様）では，カーネルの用いる管理領域を分離するようメールボックスの仕様を変更．ただし，元の仕様のメールボックスで発生しない送信時のメッセージフルエラーが発生する

- PX仕様のメールボックス機能の上位互換となる優先度データキュー機能（データを優先度順にキューイングするデータキュー）を新たに追加
- メールボックスは仕様変更せずに残し，使用は推奨しない

ASPカーネルの実装設計方針と前提条件

実装にあたっての基本方針

(1) ソースコードの読みやすさ，改造しやすさを重視する
- オープンソースソフトウェアの品質を向上させるために重要な特性
- システムの要求にあわせたチューニングが行いやすい

(2) 新しいターゲットシステムへのポーティングが容易な構造とする

(3) 検証が容易な構造とする

(4) 実行性能とメモリ使用量に配慮する

(5) スケーラビリティに配慮する
- 様々な規模のシステムに使えるようにする
- 使用しない機能によりメモリ使用量が増えない

1リンクモデルを採用

- システム全体（アプリケーション＋システムサービス＋カーネル）を1つのロードモジュールにリンクする
- アプリケーションからのカーネル呼出しは，サブルーチンコール（関数呼出し）で行う
  - よって，アプリケーションも特権モードで実行する

カーネルをライブラリ化

- カーネルをライブラリ化し，ライブラリリンク機構により必要なコード（サービスコール）のみをリンクする
  - 特別な設定をしなくても，不要なコードを省くことができるので，スケーラビリティの確保の観点からメリットが大きい
  - この粒度のコンフィギュレーションは，検証（テスト）に与える影響が少ない

ターゲット依存部と非依存部を明確に分離

- 多種類のターゲットシステムへのポーティングを容易にするためには必須

サービスコール実行中に割込みを許可しない

- サービスコールのほぼ全体を割込み禁止状態で実行
- 構造がシンプルで処理時間は短くなっているが，割込み応答性は犠牲になっている．多くのアプリケーションにおいて，これで差し支えないと判断
- カーネルの検証（テスト）を容易にする効果は大きい

メモリ使用量に関する方針

- メモリ使用量を減らしたいのは当然だが，性能とのトレードオフもあり，極限は追求しない
- （ROMではなく）RAMの使用量を減らすことに重点を置く
- スタックに置ける情報はできる限りスタックに置く

TOPPERS/ASPカーネルのリリース計画

これまでの経緯

- 2005年に開発を開始
- 2006年11月から会員向けに早期リリース

正式版のリリース

- 今回のプレス発表にあわせて，ASPカーネルの Release 1.0 を会員向けに配付開始する

一般向けのリリース

- 一般向けのリリースは，来年5月を計画

ASP Kernel Line Off Day
in 第11回 ESEC
2008年5月14日～16日