ITRONサブプロジェクトの最新状況

― μITRON4.0仕様を中心に ―

1999年3月11日

高田 広章

豊橋技術科学大学／ITRON専門委員会

hiro@ertl.ics.tut.ac.jp

ITRONホームページ

http://www.itron.gr.jp/

ITRONサブプロジェクト ー 第2フェーズ

- リアルタイムカーネル仕様の標準化から
  周辺仕様まで含めた標準化へ　（1996年頃～）

背景：組込みソフトウェアの大規模化・複合化

ソフトウェア部品 (ソフトウェアIP, 実行時ソフトウェア)

- ソフトウェア部品の流通を促す**前提条件の整備**
- ソフトウェア部品の**インタフェースの標準化**

開発環境・言語

- カーネルとデバッグ環境間のインタフェースの標準化
- C言語以外のプログラミング言語バインディング

応用分野に特化した標準化

- 応用分野に固有の要求の分析とそれへの対応

ソフトウェア部品流通の前提条件の整備

ソフトウェアの移植性の重視

! 「弱い標準化」により移植性が阻害されているという指摘

- スタンダードプロファイルの考え方を導入
  - → μITRON4.0仕様　近日中に公開
    - ソフトウェア部品向けの機能も導入

リアルタイム性の保証

? 市販のソフトウェア部品を用いた時に、どのようにしてソフトウェア部品とアプリケーションの持つリアルタイム制約を保証するか？

リアルタイム制約を持ったソフトウェア部品の例:
ソフトウェアモデム, 音声圧縮/解凍, MPEG

- ハードリアルタイム性を持ったシステムの構築手法
  - → アプリケーション設計ガイドライン　検討中

ソフトウェア部品インタフェースの標準化

- 分野毎に標準化を行う必要性
  - → 重要と考えられる分野から着手
- TCP/IPプロトコルスタックのAPI
  - → ITRON TCP/IP API 仕様　1998年5月に公開
- Java 実行環境とのインタフェースの標準化
  - → JTRON2.0仕様　1998年10月に公開
- デバイスドライバの標準化
  - … 重要性の高い分野であるが，デバイス毎の違いが大きく，性能を落さずに標準化する事は困難
  - → デバイスドライバ設計ガイドライン　検討中
- 他の標準化候補

  ファイルシステムAPI，MPEG解凍，IEEE 1394　などなど

ITRON TCP/IP API 仕様

TCP/IPプロトコルスタックのAPI

- ソケットインタフェースが主流だが，組込みシステム(特に小規模なもの) には不向きとの指摘
  - プロトコルスタック内で動的なメモリ管理が必須
  - RTOSのタスクモデルとUNIXプロセスモデルの違い

↓ 標準化の必要性

- Embedded TCP/IP **技術委員会**

ITRON TCP/IP API の設計方針

- ソケットインタフェースをベースに問題点を解決．ライブラリを載せてソケットインタフェースを実現可に
- 静的設定を活用
- ITRONと整合させるが，他のRTOSにも載るよう配慮

ソケットインタフェースとの違い

- TCP の API と UDP の API を別々に定義
- ソケットのアブストラクションに代えて，端点 (end point) を採用．TCP接続要求を待ち受ける端点と，TCP接続の端点とは別オブジェクトと扱う．端点は，種類毎にシステム内でユニークなID番号で識別．接続要求を待ち受けるAPIには両方の端点を渡す
- TCP 用の省コピーAPI を用意

送信API	受信API
get_buf	rcv_buf
snd_buf	rel_buf

- タイムアウトとノンブロッキングコール (完了はコールバックで通知) をサポート
- UDPパケットの受信はコールバックで処理する

JTRON2.0仕様

Java と RTOS のハイブリッドアーキテクチャ

- RTOS上に Javaの実行系を実現
- RTOSと Java の双方の利点を活用

リアルタイム性の必要な処理 … RTOS上で実現
ダウンロードするソフトウェア，GUI … Javaで実現

↓

RTOS上のプログラムと Java のプログラムの通信方法は？

↓ 標準化の必要性

- Java Technology on ITRON-specification OS **技術委員会**

ITRON上のプログラムと Java のプログラムの間の通信インタフェースを標準化

→ JTRON2.0仕様

RTOSーJava インタフェースのアプローチ

タイプ1) Java側からRTOS側にアクセス

- RTOSの資源を操作するオブジェクトをJava側に用意
  JTRON1.0仕様

タイプ2) RTOS側から Java側にアクセス

- Javaオブジェクトを共有メモリ的に用いる
  共有オブジェクトアプローチ

! JNI (Java Native Interface) を使う方法もある

- Javaに本来定義された方法だが，オーバヘッド大

タイプ3) 粗結合アプローチ

- RTOS側とJava側はストリームで通信
- Javaプログラムからは，RTOSプログラムがネットワークの向う側にあるように見える

ソフトウェア部品の整備

JCGプロジェクト

- μITRON仕様カーネル上で動作するソフトウェア部品の整備を図る
  - → 情報家電のための分散プラットフォーム
    - JTRON2.0仕様（Java VM は既製品を利用）
    - 組込みシステム用の CORBA
    - 組込みシステム用の GUI パッケージ
  - → 完成したらフリーソフトウェアとして公開
    （1999年度中に完成の予定）
- 情報処理振興事業協会（IPA）の「次世代デジタル応用基盤技術開発事業」の採択テーマの1つ

デバッグ環境とのインタフェースの標準化

現状の問題点

- ソフトウェアデバッグ環境（デバッガ, ICE, ロジアナなど）はそれぞれのμITRON仕様カーネルに個別に対応することが必要

デバッギングインタフェースの標準化

- μITRON仕様のリアルタイムカーネルとデバッグ環境の**RTOSサポート機能**とのインタフェースを標準化

1999年2月に検討活動を開始

標準化の利点

- デバッグ環境が，多種のμITRON仕様カーネルへ対応することが容易に
- カーネル側は，開発環境を充実させることが容易に

標準化の目標

- カーネルのオーバヘッドが最小限
- 異なるデバッグ環境とのインタフェースをできる限り共通化
- 他のRTOSやソフトウェア部品にも適用可能

アプローチ

- デバッグ環境側からの**処理要求**を標準化
  例) あるタスク/オブジェクトの状態を読み出す
- デバッグ環境における処理の**プリミティブ**を標準化
  例) 指定した番地のメモリの内容を読み出す
- あるカーネルで，デバッグ環境側からの処理要求を，デバッグ環境におけるプリミティブを使って処理する方法を，カーネルメーカ側が**プログラムの形**で提供

応用分野に特化した標準化

← 組込みシステムの多様性

- 自動車制御応用
  - μITRONに限らずRTOSの適用が難しかった分野
  - RTOSの必要性が高まっている
  - → 自動車分野の技術者に呼びかけて「RTOS自動車応用技術委員会」を設け、自動車制御用のリアルタイムカーネル仕様に対する要求を整理・仕様案を作成
    - → μITRON4.0仕様へ反映
  - 将来的には，OSEK COMプロトコル (自動車内通信用のプロトコル) のAPIを検討予定
- その他に取り組むべき分野

  FA，情報家電　などなど

μITRON4.0仕様 ー 策定の必要性

ソフトウェアの移植性の向上

- 組込みソフトウェアの大規模化により移植性が重視
- 移植性の向上はソフトウェア部品流通の前提条件

ソフトウェア部品向け機能の追加

- 外販することを前提としたソフトウェア部品開発

新しい要求・検討成果の反映

- リアルタイム性の保証を容易にするための機構
  - ← リアルタイム性を持ったソフトウェア部品
- よりコンパクトな実装を可能にする仕様
  - ← 自動車制御応用における要求事項の整理

半導体技術の進歩への対応

- μITRON3.0を公開してから5年以上が経過

ソフトウェアの移植性の向上

- 基本的には標準化の度合いを強くすればよい

↕ **一般には両立は困難**

- 適応化の利点（弱い標準化によって得られる）
  - 広範なスケーラビリティの実現
  - ハードウェア（プロセッサ）の特徴を活かす
  - アプリケーション毎の要求への対応

両立可能な部分は両立させる

- ライブラリリンクによる適応化を前提 → 豊富な機能
  - ⚒ 条件コンパイルによる適応化（*cf.* eCos）
- システムコールの単機能化

! スケジューラ/ディスパッチャに関わる部分は両立困難

- RISC的なリアルタイムカーネル仕様

スタンダードプロファイルの導入

コンセプト

- μITRONの適用分野の中で比較的大規模なシステムを想定し，*標準的な機能セットを「強く」標準化*
  - ← 移植性が重視されるのは比較的大規模なシステム
- 性能重視の小規模なシステムにはサブセットで対応
- 標準を越える要求のために拡張機能も定義

μITRON4.0仕様全体	… 弱い標準化
スタンダードプロファイル	… 強い標準化

言い換えると…

- 移植性を重視するソフトウェア（典型例: ソフトウェア部品）はスタンダードプロファイルの機能のみを用いる
- ソフトウェアの移植性を重視する分野向けのカーネルはスタンダードプロファイルに準拠して実装する

想定するシステムイメージ

- ハイエンド 16bit ～ 32bit プロセッサ
- カーネルサイズ 10～20KB程度（全機能を使った場合）
- *システム全体が1つのモジュールにリンク*
- *カーネルオブジェクトは静的に生成*

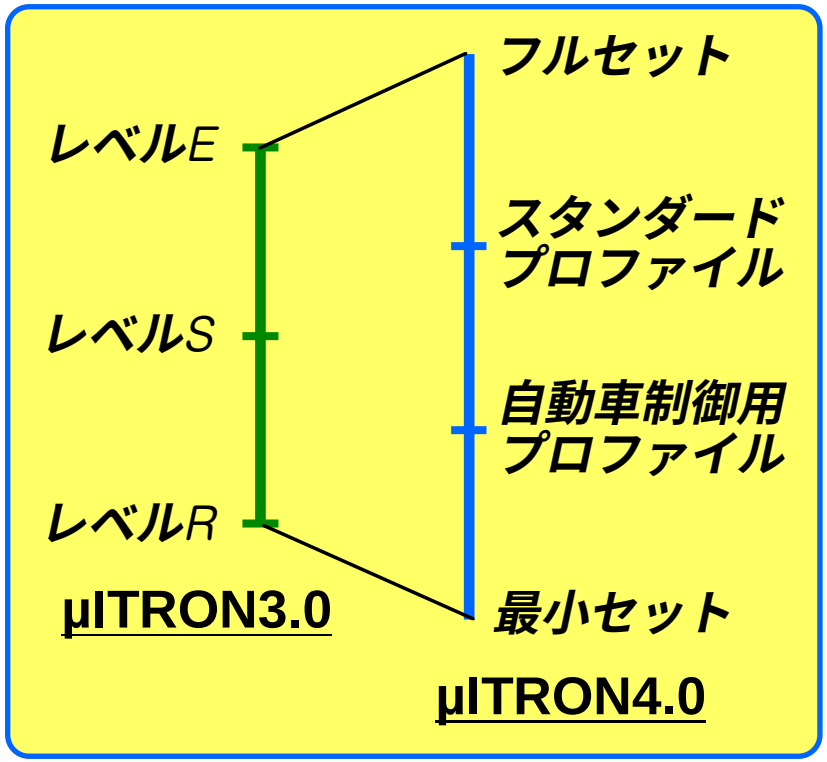
規定の概略

- サポートすべきサービスコールとその機能の規定
  - μITRON3.0仕様のレベルSのほとんど（一部仕様修正）
  - μITRON3.0仕様のレベルEの一部（一部仕様修正）
  - 新規の機能（データキュー，例外処理のための機能，システム状態参照機能など）
- 割込みハンドラから呼び出せるサービスコールの規定
- コンフィギュレーション方法の標準化（静的API）
- その他の規定についても実装依存性を削減

より広いスケーラビリティの実現

μITRON3.0仕様よりも高機能化

- データキュー
- 例外処理のための機能
- システム状態参照機能
- ID番号の自動割付け
- ハードリアルタイム対応



自動車制御用プロファイル

- 比較的小規模なシステムのためのプロファイル定義
- 制約タスクの導入

より小規模なシステムへの適応化

- 待ち状態をオプションにして，休止状態を必須に

コンフィギュレーション方法の標準化

- 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，コンフィグレーション ファイル中での記述方法を標準化

静的API

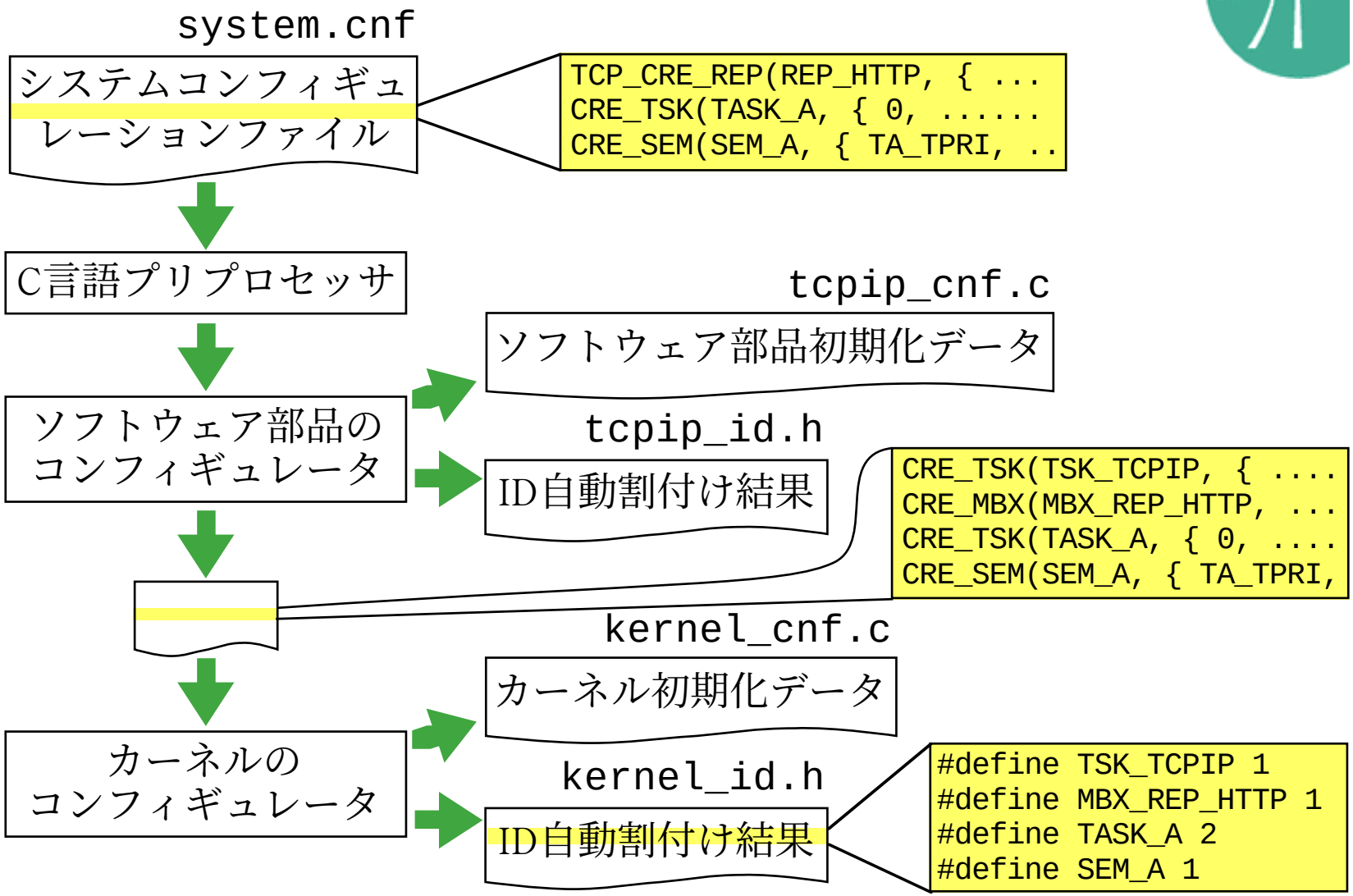
例）
```
CRE_TSK(tskid, { tskatr, exinf, task,
                 itskpri, stksz, stk });
ATT_HDR({ intatr, exinf, intsrc, inthdr });
```

- ソフトウェア部品の静的APIを混在して記述可能

利点

- ソフトウェアを別のカーネルへ移植するのが楽に
- ソフトウェア部品の組み込みが容易に
- サービスコールと静的APIを個別に覚える必要がない

コンフィギュレーション手順



データキュー

機能の概要

- 1ワードデータのメッセージ通信機構．リングバッファで実装することを想定（μITRON3.0仕様のリングバッファで実装されたメールボックスに相当）
- 自動車制御応用から強い要求，他の応用でも有用
- 使わない場合にはリンクされない

データキューのAPI

CRE_DTQ … 生成（静的API）
rcv_dtq, prcv_dtq, trcv_dtq … 受信（待ちあり）
snd_dtq, psnd_dtq, tsnd_dtq … 送信（待ちあり）
fsnd_dtq … バッファフルの時に最も古いデータを上書き
isnd_dtq, ifsnd_dtq … 割込みハンドラからも送信可能
cre_dtq, del_dtq, ref_dtq … 生成・削除・参照（拡張機能）

例外処理のための機能

例外処理のモデル

CPU例外ハンドラ ← 例外発生時にプロセッサが起動

（CPU例外の種別毎にアプリケーションで定義）

↓ 必要に応じて例外処理をタスクに任せる

タスク例外処理ルーチン

（タスク毎にアプリケーションで定義）

- タスク例外処理ルーチン

  UNIXのシグナルハンドラを軽くしたような機能

  - サービスコールにより明示的に例外処理を要求
  - タスクが次にスケジュールされる時にルーチンを呼び出す
  - タスクと同じコンテキストで実行
  - ルーチン起動時にタスク例外処理禁止状態に
  - タスク毎に１つのみで，例外要因はパラメータで渡す

例外処理のためのAPI

- CPU例外ハンドラ

  DEF_EXC … CPU例外ハンドラ定義（静的API）

- タスク例外処理機能

  DEF_TEX … タスク例外処理ルーチン定義（静的API）
  ras_tex … タスク例外処理要求
  iras_tex … 割込みハンドラからも例外処理要求可能
  dis_tex … タスク例外処理禁止
  ena_tex … タスク例外処理許可

- 例外要因は ras_tex にパラメータとして渡すと，タスク例外処理ルーチンにパラメータとして渡される．複数の ras_tex が発行された時には，例外要因を bitwise-or する
- タスク例外処理ルーチンのもう1つのパラメータとして，タスクの拡張情報が渡される

システム状態参照機能

導入の動機

- 外販するソフトウェア部品では，サービスコールを呼び出すコンテキストを限定するのは難しい
  - → 正しくないコンテキストから呼ばれたらエラーを返す
- ! μITRON3.0では，現在のコンテキストを調べる機能がオーバヘッドの大きい拡張機能でしか用意されていない

システム状態参照のためのAPI

sns_ctx … 割込みハンドラ実行中か？
sns_loc … CPUロック状態か？
sns_dsp … ディスパッチ禁止状態か？
sns_tex … タスク例外処理禁止状態か？

- どのようなコンテキストからでも呼び出し可能で，ブール値を返す

CPUロック状態の意味の変更

- ! μITRON3.0では，ディスパッチ禁止中にCPUロックした後に元に戻す方法が用意されていない
  - → ソフトウェア部品が一時的に割込み禁止した時に元に戻せない場合がある

- CPUロック状態とディスパッチ禁止状態は独立に
- CPUロック状態では，いくつか例外の除いてはサービスコールを呼び出すことができない

例）

```
{
        BOOL locked = sns_loc();
        if (!locked) loc_cpu();
        排他制御が必要な処理
        if (!locked) unl_cpu();
}
```

スタンダードプロファイル外の新機能

ID番号自動割付け

- オブジェクト生成時にID番号を自動割付けする機能

例）`tskid = acre_tsk(&pk_ctsk);`

- `tskid`：生成したタスクのID番号が返る
- `&pk_ctsk`：タスク生成情報

ハードリアルタイムサポート機能

- 優先度継承/上限プロトコルをサポートする排他制御機構
  - → POSIX リアルタイム拡張の mutex に相当
- オーバーランの検出

その他の新機能

- ポータビリティのある割込みハンドラ

μITRON3.0からのその他の変更

サービスコール名の変更

- preq_sem → pol_sem
- snd_msg → snd_mbx，rcv_msg → rcv_mbx
  get_blf → get_mpf，rel_blf → rel_mpf
  get_blk → get_mpl，rel_blk → rel_mpl
- get_ver → ref_ver，ref_iXX → get_iXX

タスクの起動

- 起動要求のキューイングできる act_tsk を標準に
- 起動コードの渡せる sta_tsk は拡張機能に

タスク/ハンドラの記述方法

- C言語からの ret_int は廃止．関数からリターンで割込みハンドラからリターン（他のハンドラも同様）
- タスクのメイン関数からのリターンでもタスク終了

タイムイベントハンドラ

- 周期ハンドラ
  - CRE_CYC … 定義（静的API），周期・位相を与える
  - sta_cyc … 周期ハンドラの開始
  - stp_cyc … 周期ハンドラの停止
- アラームハンドラ
  - CRE_ALM … 定義（静的API），時間は与えない
  - sta_alm … アラームハンドラの開始，時間を与える
  - stp_alm … アラームハンドラの停止

その他

- 自タスクに対する sus_tsk を可能に
- タスク優先度の最低段数は16段階
- イベントフラグにもタスク優先度待ちをサポート
- ランデブ呼出し待ちのタイムアウトの意味の変更　など

仕様の厳密化・実装依存性の削減

- メールボックスを線形リストによる実装に限定
- 相対時間指定の意味を厳格化
- 検出を省略できるエラーコードについて明確に規定
- スケジューリング規則をより強く規定

スタンダードプロファイルにおける実装依存性の削除

- sus_tsk のネスト機能はスタンダードプロファイル外
- セマフォ等のタスク優先度順待ちをサポート
- メールボックスのメール優先度順キューイングをサポート
- セマフォカウントの初期値と最大値が指定可能
- イベントフラグの複数タスク待ちはスタンダード外
- 各データ型の最低ビット数を規定

などなど

システムの初期化手順

- システム初期化の流れ

  ハードウェア依存の初期化処理 ← リセット

  （ユーザが作成，カーネルの管理外）

  ↓

  カーネルの初期化

  ↓

  静的APIの処理，*初期化ルーチンの実行*

  （ユーザが作成，静的APIで定義）

  ↓

  カーネル起動，タスクの実行が開始

  割込みが許可，システムクロックが動作開始

- 初期化ルーチンは，CPUロック状態のタスクと同等のコンテキストで実行
- 静的APIでのエラーの処理は実装依存

自動車制御用プロファイルの概要

スタンダードプロファイルから削減する機能

- タイムアウト付きのサービスコール
- 強制待ち状態（sus_tsk, rsm_tsk）
- タスク例外処理機能
- メールボックス，固定長メモリプール
- rot_rdq, snd_dtq, set_tim, get_tim, dly_tsk

制約タスク → 使用メモリ節約のため

- 待ち状態に入れないタスク
- 複数の制約タスクでスタックエリアを共有
- 待ち状態に入ろうとした時にエラーが報告されることに依存しなければ，通常のタスクにしてもそのまま動作

! この意味ではスタンダードプロファイルの下位互換

活動の枠組み

- (社)トロン協会 ITRON専門委員会
  - ITRON企画・広報WG
  - JCGプロジェクト実行委員会
  - ITRON専門委員会 米国支部 (1999年度より活動開始)
- μITRON4.0仕様研究会　オープンな研究会
  - カーネル仕様WG
  - デバッギングインタフェース仕様WG
  - アプリケーション設計ガイドラインWG
  - デバイスドライバ設計ガイドラインWG
- Embedded TCP/IP 技術委員会
- Java Technology on ITRON-Specification OS 技術委員会
- RTOS自動車応用技術研究会

将来展望

- 組込みシステム開発のオープン化の流れ

  既製の技術を安心して使うには標準化が不可欠

  ! μITRON4.0仕様はオープン化の流れに対応する基礎
- これをベースとして要求の高い課題にタイムリーに取り組むことが必要

  → *time-to-market の短い標準化*

宣伝: ITRONオープンセミナー '99

- 6月30日(水) ～ 7月1日(木) 東京にて
- 関連製品の紹介, μITRON4.0仕様の詳細解説など

ITRONホームページ

http://www.itron.gr.jp/