ET2015 TOPPERSセッション

TOPPERS第3世代カーネル徹底解説

2015年11月19日

高田 広章

NPO法人 TOPPERSプロジェクト 会長
名古屋大学 未来社会創造機構 教授
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/

AGENDA

TOPPERSプロジェクトの概要

TOPPERS第3世代カーネルの概要

- TOPPERS第3世代カーネルの必要性，仕様策定方針
- TOPPERS新世代カーネルからの仕様変更
- 第3世代カーネル（ITRON系）の実装，TOPPERS/ASP3

高分解能時間管理と外部時刻同期機能

時間パーティショニング機能

その他の新規機能と仕様変更

- タスク終了要求機能
- 動的負荷分散の支援機能
- タイムイベント処理機能の見直し

おわりに

TOPPERSプロジェクトの概要

TOPPERSプロジェクトとは?

- ITRON仕様の技術開発成果を出発点として，組込みシステム構築の基盤となる各種の高品質なオープンソースソフトウェアを開発するとともに，その利用技術を提供

組込みシステム分野において，*Linux*のように広く使われるオープンソースOSの構築を目指す!

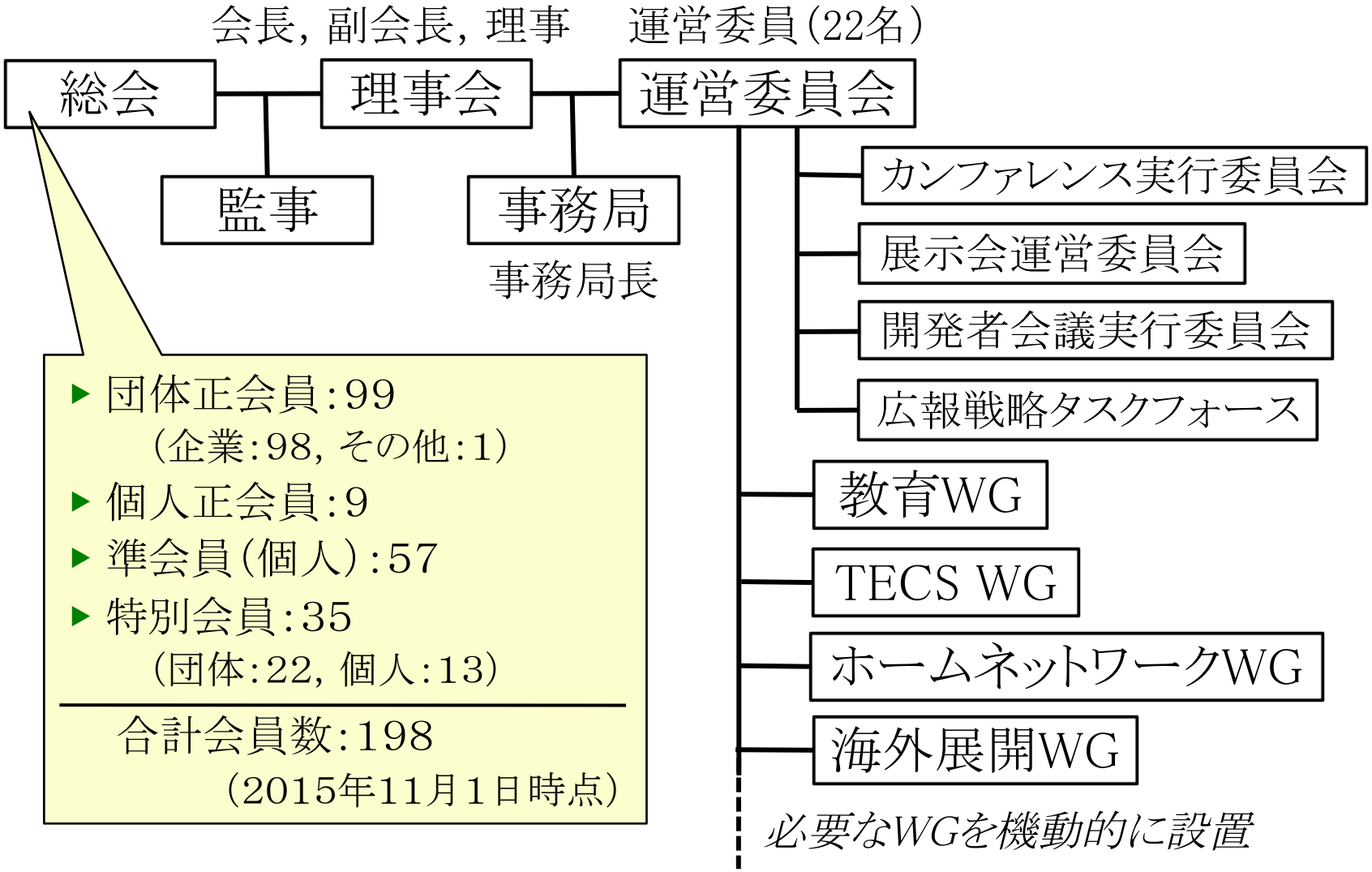
プロジェクトの狙い

- 決定版のITRON仕様OSの開発 ← ほぼ完了
- 次世代のリアルタイムOS技術の開発
- 組込みシステム開発技術と開発支援ツールの開発
- 組込みシステム技術者の育成への貢献

プロジェクトの推進主体

- 産学官の団体と個人が参加する産学官民連携プロジェクト
- 2003年9月にNPO法人として組織化

TOPPERSプロジェクトの組織と会員



主な開発成果物

一般公開しているもの

第１世代カーネル

- TOPPERS/JSPカーネル，TOPPERS/FI4カーネル
- TOPPERS/ATK1（Automotiveカーネル バージョン1）
- TOPPERS/FDMPカーネル，TOPPERS/HRPカーネル

新世代カーネル（第２世代）

- TOPPERS/ASPカーネル，TOPPERS/SSPカーネル
- TOPPERS/FMPカーネル，TOPPERS/HRP2カーネル
- TOPPERSテストスイートパッケージ（TTSP）

AUTOSAR関連

- TOPPERS/ATK2（Automotiveカーネル バージョン2）
- TOPPERS/A-COMSTACK，TOPPERS/A-WDGSTACK
- TOPPERS/A-RTEGEN

ミドルウェア

- TINET（TCP/IPプロトコルスタック），FatFs for TOPPERS
- TOPPERS/ECNL（ECHONET Lite通信ミドルウェア）
- RLL（Remote Link Loader），DLM（Dynamic Loading Manager）
- CAN/LINミドルウェアパッケージ

ツール，その他

- TECS（TOPPERS組込みコンポーネントシステム）
- SafeG（高信頼組込みシステム向けデュアルOSモニタ）
- EV3RT（LEGO Mindstorms EV3向けSPF）
- TLV（TraceLogVisualizer），TOPPERS Builder

教育コンテンツ

- 初級・中級実装セミナー教材
- 基礎1・基礎2・基礎3実装セミナー教材
- ETロボコン向けTOPPERS活用セミナー教材

開発成果物の主な利用事例

エスクード（スズキ）

スカイラインハイブリッド（日産）

IPSiO GX e3300（リコー）

H-IIB（JAXA）

提供：JAXA，イラスト：池下章裕

ASTRO-H（JAXA）
＜開発中＞

OSP-P300
（オークマ）

SoftBank
945SH
（シャープ）

UA-101（Roland）

PM-A970(エプソン)

次の10年を見据えた活動指針（2011年度に策定）

Smart Futureのための組込みシステム技術

- 組込みシステム技術を，持続可能なスマート社会の実現（Smart Future）のための重要な要素技術の1つと位置づけ，その研究開発と普及に取り組む
- それに向けての研究開発課題
  - Safety & Security
  - Ecology（高エネルギー効率）
  - Connectivity

コンソーシアムによるオープンソースソフトウェア開発

- 同じ技術に関心を持つプロジェクトメンバによりコンソーシアムを結成し，複数組織の協力によりソフトウェアを開発
- 開発したソフトウェアは，TOPPERSプロジェクトからオープンソースソフトウェアとして公開

重点的に取り組んでいるテーマ

※ SPF：ソフトウェア
プラットフォーム

次世代のリアルタイムカーネル技術

- TOPPERS第3世代カーネル（ITRON系）
- 車載システム向けRTOS（AUTOSAR OS仕様からの発展）

ソフトウェア部品化技術

- TECS（TOPPERS組込みコンポーネントシステム）

組込みシステム向けSPFと開発支援ツール

- 車載制御システム向けSPF（AUTOSAR仕様ベース）
- 仮想化技術（SafeG，A-SafeG），ホームネットワーク
- 宇宙機向けSPF（SpaceWire OS）
- 開発支援ツール（シミュレータ，可視化ツール）

技術者育成のための教材開発

- プラットフォーム技術者の育成
- ETロボコン向けSPFと教材の提供

TECS (TOPPERS組込みコンポーネントシステム)

TECSとは?

- 各種のソフトウェアモジュールを部品化し，必要な部品を組み合わせることによって大規模な組込みソフトウェ アを効率的に構築するための技術

TECS(コンポーネント技術)を用いる利点

- 大規模な組込みソフトウェアの見える化
- ソフトウェア部品の流通性・再利用性の向上
- 分散フレームワークによる分散システムの開発効率化

TECSの特徴とアプローチ

- コンポーネント間の結合を静的にし，最適化を可能に
- すべてのソフトウェアをコンポーネントとして扱える
- 遠隔呼出し(RPC)のためのコンポーネントをツールにより生成

TOPPERS第3世代カーネルの概要

TOPPERS第3世代カーネルの必要性

求められている/求められつつある技術・機能

- 機能安全からの要求に応えられるパーティショニング
- ティックレスの高分解能時間管理と外部時刻同期
- マルチコアにおける動的ロードバランシング
- メニーコアプロセッサへの対応 … 今後の課題

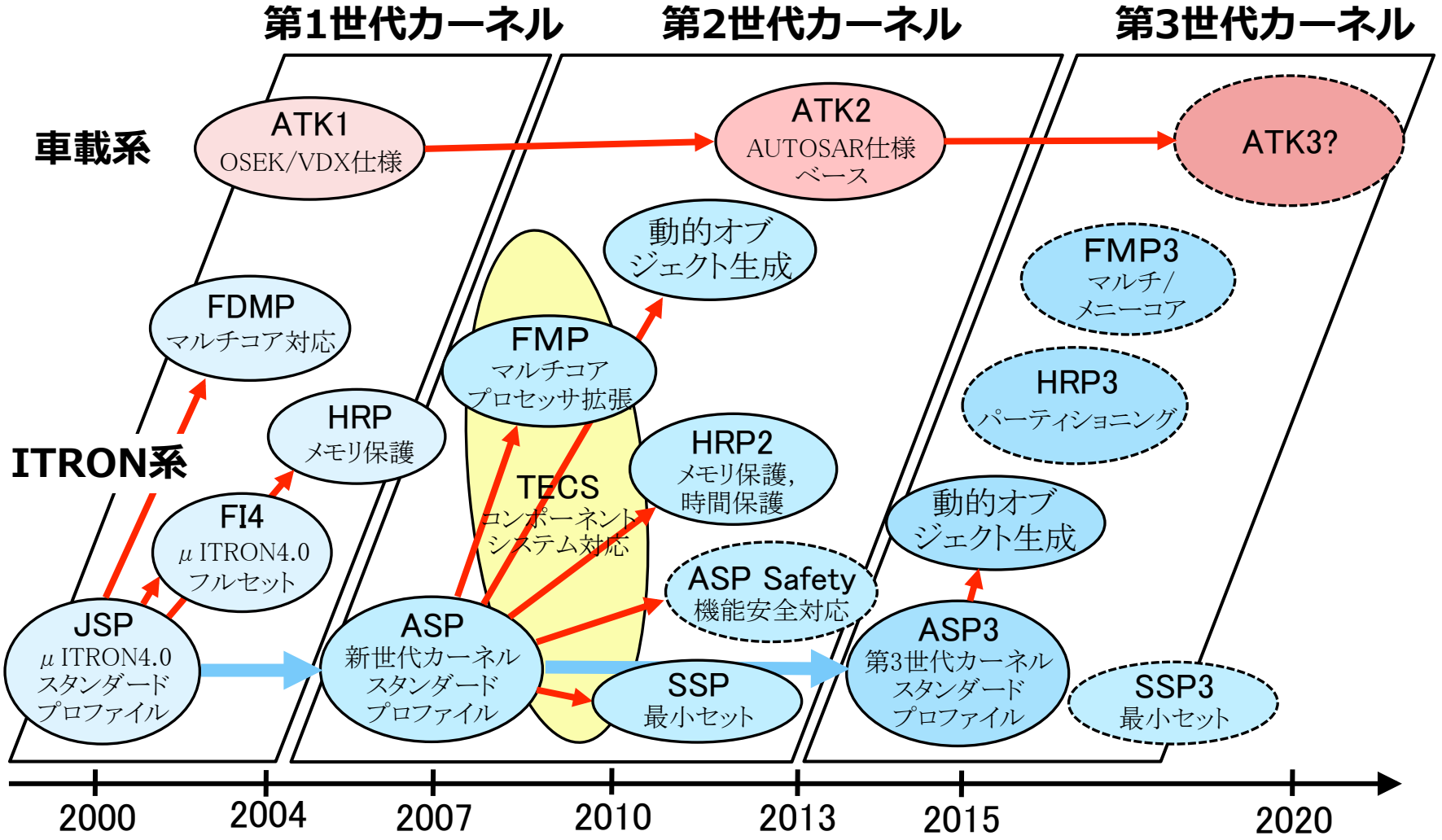
一方，廃止すべきと考えられる機能もある

- タスク例外処理機能，メールボックス

TOPPERS第3世代カーネルへ

- 現状のリアルタイムカーネル（第2世代）の次の世代と位置付けた方が，大胆な仕様変更が可能
- 第3世代においても，2系列（ITRON系，車載系）のリアルタイムカーネル開発は，引き続き維持していく

TOPPERSカーネル開発ロードマップ



第3世代カーネル（ITRON系）の仕様策定方針

基本的な設計方針 … 新世代カーネル仕様を踏襲

- μITRON4.0仕様をベースに拡張・改良を加える
- ソフトウェアの再利用性を重視する
- 高信頼・安全なシステム構築を支援する
- アプリケーションシステム構築に必要な機能は積極的に取り込む

特に留意した設計方針（これまでも暗黙に存在）

- システム／アプリケーションによって要求が異なる機能は，ミドルウェア等で実現することとし，カーネルにはその実現に必要な最低限の機能を導入する
  - ポリシーとメカニズムの分離の考え方
- 実装方法を合わせて検討し，オーバヘッドの大きい仕様を避ける

TOPPERS新世代カーネルからの仕様変更

（1）時間パーティショニング機能の導入

- 機能安全におけるソフトウェアパーティショニングの実現を容易にするための機能

（2）ティックレスの高分解能時間管理と外部時刻同期

- カーネルが管理する時間の分解能をミリ秒からマイクロ秒に変更
  - タイムティックを使わない実装に変更
- 外部時刻との同期のための機能を導入

（3）マルチコアにおける動的負荷分散機能への対応

- マルチコアプロセッサにおけるロードバランシングを実現するために以下の機能を導入
  - タスクとスケジューリング状態の参照機能
  - サブ優先度機能

(4)仕様のスリム化・シンプル化(主なもの)

- タスク例外処理機能の廃止とタスク終了要求機能の導入
  - 必要性が低いにもかかわらず，ターゲット依存部(特に保護機能対応の場合)の実装負担が大きいタスク例外処理機能と待ち禁止状態を廃止
- 非タスクコンテキスト専用のサービスコールの廃止
- メールボックス機能の廃止
  - 保護機能対応では不適切な機能(サポートしていない)

(5)ミューテックスの標準機能化

(6)その他の改良(主なもの)

- タイムイベント処理機能の見直し
- 起動要求をキューイングしないタスクの追加
- 保護ドメインに対するアクセス許可ベクタの新設
- 保護ドメイン毎の標準メモリリージョン

第3世代カーネル (ITRON系) の機能リスト

- タスク管理機能
- タスク付属同期機能
- タスク終了機能　　タスク例外処理機能は廃止
- 同期・通信機能　　メールボックスは廃止
  - セマフォ
  - イベントフラグ
  - データキュー
  - 優先度データキュー
  - ミューテックス
  - メッセージバッファ
  - スピンロック
- メモリプール管理機能
  - 固定長メモリプール

- 時間管理機能
  - システム時刻管理
  - 周期通知　　周期ハンドラを拡張
  - アラーム通知　　アラームハンドラを拡張
  - オーバランハンドラ
- システム状態管理機能
- メモリオブジェクト管理機能
- 割込み管理機能
  - 割込みサービスルーチン
- CPU例外管理機能
- 拡張サービスコール管理機能
- 保護ドメイン管理機能
- システム構成管理機能

第3世代カーネル（ITRON系）の実装

! カーネルのバージョン番号を「3」に統一

TOPPERS/ASP3カーネル

- TOPPERS第3世代カーネルの出発点

TOPPERS/HRP3カーネル

- ASP3カーネルに以下の機能を追加し，さらに高い信頼性・安全性を要求される組込みシステムに対応
  - メモリ保護機能，オブジェクトアクセス保護機能
  - 時間パーティショニング機能
  - 拡張サービスコール機能

TOPPERS/FMP3カーネル

- ASP3カーネルをマルチ/メニーコアプロセッサ向けに拡張

TOPPERS/SSP3カーネル

- ASP3カーネルをベースに可能な限り機能を絞り込む

TOPPERS/ASP3カーネルの概要

基本パッケージでサポート

- タスク終了要求機能
- ティックレスの高分解能時間管理，システム時刻の調整機能，システム時刻の参照／設定機能
- ミューテックス機能 … 基本機能に格上げ

拡張パッケージでサポート

- ドリフト調整機能
- メッセージバッファ機能
- オーバランハンドラ機能
- タスク優先度拡張
- 制約タスク
- サブ優先度機能
- 動的生成機能

内部構造の見直し（主なもの）

- ティックレスの高分解能時間管理の実装
- ディスパッチャ内のアイドル処理の実装方法を変更
- タスク例外処理機能の廃止に関連して，システム状態の持ち方を整理
- p_schedtskの意味を「実行すべきタスク」に変更
- kernel.tfの分割

開発状況

- ターゲット非依存部のコーディングは完成
- 新規機能の一通りのテストも完了
- いくつかのプロセッサのターゲット依存部もほぼ完成
  - ARM，ARM-M

未完成の開発項目

- TECSの活用
  - システムサービス(システムログ機能など)とデバイスドライバのコンフィギュレーションにTECSを活用
  - カーネルを利用するだけであれば，TECSを勉強/利用する必要はない(配付パッケージには，TECSジェネレータの出力ファイルを含める)
- Ruby版コンフィギュレータの適用

リリース計画

- TOPPERS会員には，β版(Release 3.B.0)を早期リリース中
  - 最新のバージョンは，subversionからアクセス可能
- 今年末(もしかしたら，年明けになるかも…)に一般公開の予定

高分解能時間管理と外部時刻同期機能

考慮する要求事項

高分解能の時間管理と絶対時刻の表現

- マイクロ秒単位で実行を制御できる高分解能の時間管理機能を持つこと
- 相対時間として，少なくとも1時間強まで指定できること
- 64ビットで表現される絶対時刻を管理できること

ティックレスの時間管理

- 処理すべきタイムイベントがない時に，タイマ割込みがかからないようにすること（消費エネルギー削減のため）

外部時刻同期

- 外部から供給される時刻に，内部の時刻を同期させるために必要な機能を持つこと

カレンダ機能との統合 … *未検討*

- カレンダ機能を容易に扱うために必要な機能を持つこと

外部時刻同期とその方法

同期対象となる外部時刻

- ネットワークの時間（FlexRay，SpaceWire）
- GPSなどから来る絶対時刻
- カレンダチップから読んだ絶対時刻

ミドルウェア（またはアプリケーション）とカーネルの役割分担

- 外部時刻の取得 … ミドルウェアで実現
- 外部時刻のジッターの吸収 … ミドルウェアの役割
- システム時刻の調整 … カーネルでサポート

システム時刻の2つの調整方法をサポート

- 時間区間の一部（通常は，システム周期の最後のアイドル時間）のみを伸び縮みさせる方法
- ある時間区間全体を一様に伸び縮みさせる方法

外部時刻同期の実現例(1) – 時間区間の一部を伸縮

- FlexRayに接続されたノードを考える
- FlexRayの特徴は次の通り
  - ネットワークが周期動作している
  - ネットワークの周期末にアイドル時間(NIT)があり，それを伸び縮みさせることで時刻同期を行う
  - 周期末(＝次の周期の始まり)に割込みが要求される
- 周期末の割込み処理で，カーネルが管理するシステム時刻を読み，想定時刻との時間差を求める
- 求めた時間差に対して，割込み処理遅延等によるジッタの補正を行う
  - 例えば，過去に求めた時間差との移動平均を取る方法が考えられる
- 補正後の時間差の分だけ，システム時刻を即座に調整する(進める or 戻す)

外部時刻同期の実現例(2) – 時間区間全体を一様に伸縮

- 定期的に，時刻サーバに対して，現在時刻を問い合わせる場合を考える(これにより得られる時刻を，外部時刻と呼ぶ)
- 外部時刻が得られると，カーネルが管理するシステム時刻を読み，通信遅延等によるジッタの補正を行って，外部時刻とシステム時刻の時間差を求める
- 時刻サーバに対して次に問い合わせるまでの間に，求めた時間差の分だけシステム時刻を補正するように，システム時刻の進み方を調整する(早くする or 遅くする)
- システム時刻を進めているクロックの発振周波数の誤差が求まっている場合には，上で求めた時間差の分に，発振周波数の誤差分も加えて，システム時刻の進み方を決定する

外部時刻同期のためのカーネル機能

システム時刻の調整（adj_tim）

- システム時刻を即座に進める／戻す機能
  - 時間区間の一部のみを伸び縮みさせる場合に使用
- タイムイベント発生の（リアルな）時刻が変化する

ドリフトの調整とドリフト量の設定（set_dft）

- システム時刻の進み方を早くする／遅くする機能
  - 時間区間全体を一様に伸び縮みさせる場合に使用
- カーネルにドリフトを調整する機能を持たせ，ドリフト量を（動的に）設定する機能を用意することで実現
  - ドリフト量はppm単位で指定

システム時刻の参照／設定（get_tim／set_tim）

- 64ビットのシステム時刻の参照／設定
- タイムイベント発生の（リアルな）時刻は変化させない

ティックレスの時間管理の基本

タイマの使い方の基本

- 先頭のタイムイベント(発生を待っているタイムイベントの中で，最も早く発生するタイムイベント)の発生時刻に割込みを発生させるように，タイマを設定するのが基本

タイムイベントヒープ(実装上の概念，仕様には登場しない)

- 発生を待っているタイムイベント(の集合)を管理するためのデータ構造
- タイムイベントを効率的に処理するためには，次の性質を持つデータ構造を用いることが必要
  - 先頭のタイムイベントが高速に取り出せること
  - タイムイベントの登録・削除が効率よく行えること
- JSPカーネル以来，ヒープ構造を採用している

時刻を表すデータ型

システム時刻(SYSTIM)

- 64ビット符号無し整数．単位はマイクロ秒
  - 64ビット整数が扱えないターゲットでは32ビットとする

相対時間(RELTIM)

- 32ビット符号無し整数．単位はマイクロ秒
- 32ビット符号無し整数で表現できる最長時間は，約4294秒(約71.5分)になる
  - 1時間強まで指定できるという要求を満たす
- 効率的な実装を可能とするために，指定できる最大値に制限を設ける … *次に議論*

タイムアウト指定(TMO)

- 32ビット符号無し整数
- 相対時間またはTMO_POLまたはTMO_FEVR

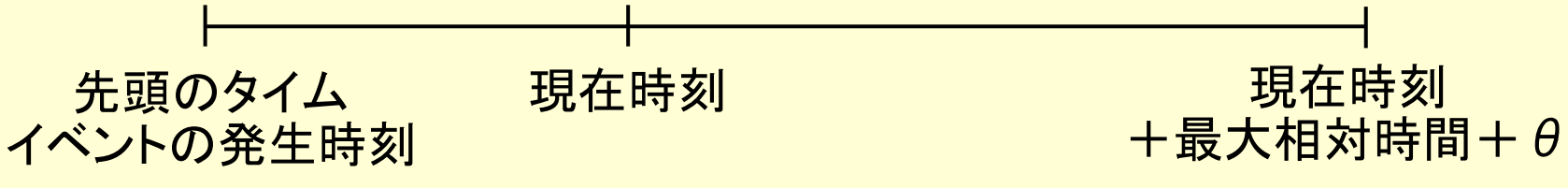
相対時間に指定できる最大値

イベント時刻（実装上の概念，仕様には登場しない）

- オーバヘッドを小さくするために，タイムイベントヒープ中のタイムイベントの発生時刻は，32ビットで管理したい
- 32ビットで管理するタイムイベントの発生時刻を，イベント時刻と呼ぶことにする

イベント時刻の範囲

- タイムイベントヒープ中には，現在時刻よりも以前の発生時刻を持つタイムイベントが登録されている場合がある（1）
- 現在時刻に最大相対時間を加えた時間よりも後の発生時刻を持つタイムイベントが登録されている場合がある（2）
- 下の図の全範囲が32ビットで表現できることが必要



(1)の要因

- タイムイベント処理の遅れ
  - 大幅に遅れる原因はアプリケーション(タイマ割込みが長時間マスクされていた場合，プロセッサが処理できる以上の頻度でタイムイベントが発生した場合など)
  - この状況を防ぐのはアプリケーションの責任とする(具体的には，タイムイベントの処理が2秒以上遅れる状況では，カーネルの動作は保証されないものとする)
- システム時刻の調整によりシステム時刻を進めた
  - adj_timによりシステム時刻を進められる幅を制限する

(2)の要因

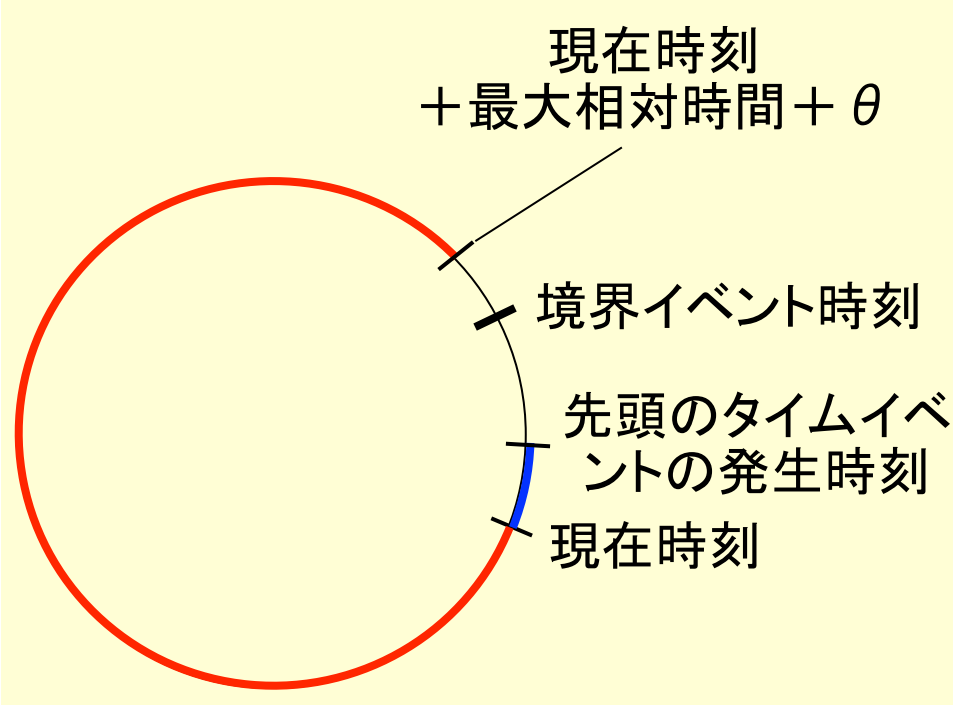
- 相対時間を長い方に丸める
  - 丸めにより長くなる最大時間はタイマの刻み幅(高分解能タイマを想定しており，1ミリ秒を超えることはない)
- システム時刻の調整によりシステム時刻を戻した
  - adj_timによりシステム時刻を戻せる幅を制限する

相対時間に指定できる最大時間の決定

- (1)と(2)の要因を合計しても，数秒程度
- 相対時間（単位はマイクロ秒）に指定できる最大時間を，4,000,000,000（＝4000秒，66分40秒）とする
  - 32ビットで表現できる最大時間との間に，十分なマージン（約294秒）がある
  - キリのよい数字にした

イベント時刻の比較（実装の話し）

- 現在のイベント時刻の200秒前を境界イベント時刻とし，それを基準として比較する



システム時刻の調整

システム時刻の調整（adj_tim）

ER ercd = adj_tim(int_t adjtim)

- システム時刻の現在値に，adjtimで指定した時間（単位はマイクロ秒）を加える
  - adjtimが正の値の時はシステム時刻は進み，adjtimが負の時はシステム時刻は戻ることになる
  - adjtimに指定できるのは，±1秒の範囲内（数値でいうと，–1,000,000～1,000,000の範囲内）
- タイムイベントが発生するまでの相対時間も調整する
  - 例えば，タイムイベントを5ミリ秒後に発生するように設定した後に，adj_timによりシステム時刻を1ミリ秒進めると，タイムイベントは最初に設定した時刻の4ミリ秒後に発生

システム時刻の調整(adj_tim)– 続き

- adj_timを連続して呼び出した場合，未処理の調整時間の累積値に対しても，制限を設ける必要がある
  - 時間を進める場合には，1秒以上過去の発生時刻を持つタイムイベントが処理されずに残っている場合にエラーとする
  - 時間を戻す場合には，現在のシステム時刻が，システム時刻が最も進んでいた時のシステム時刻より1秒以上戻っている場合にエラーとする
- adj_timによってシステム時刻を戻した場合でも，get_timで参照するシステム時刻は戻らないように調整する
  - システム時刻によりイベントの前後関係を判断できなくなるため
  - 具体的には，get_timは，システム時刻が最も進んでいた時のシステム時刻を返すものとする

使用上の注意

- adj_timによりシステム時刻を進めた場合，多くのタイムイベントが一斉に発生する可能性
  - 例えば，1ミリ秒周期の周期ハンドラが動作している時に，adj_timによりシステム時刻を1秒進めると，周期ハンドラは1000回呼び出される
  - 基本的には，周期ハンドラの周期よりも大きく時間を調整するような使い方は想定していない
- システム時刻を大きく進めたことにより発生する過負荷（その結果，タイムイベント処理が遅れる）への対処は，ユーザの責任
  - タイムイベントの処理が2秒以上遅れる状況では，カーネルの動作は保証されないものとしている（前述の通り）

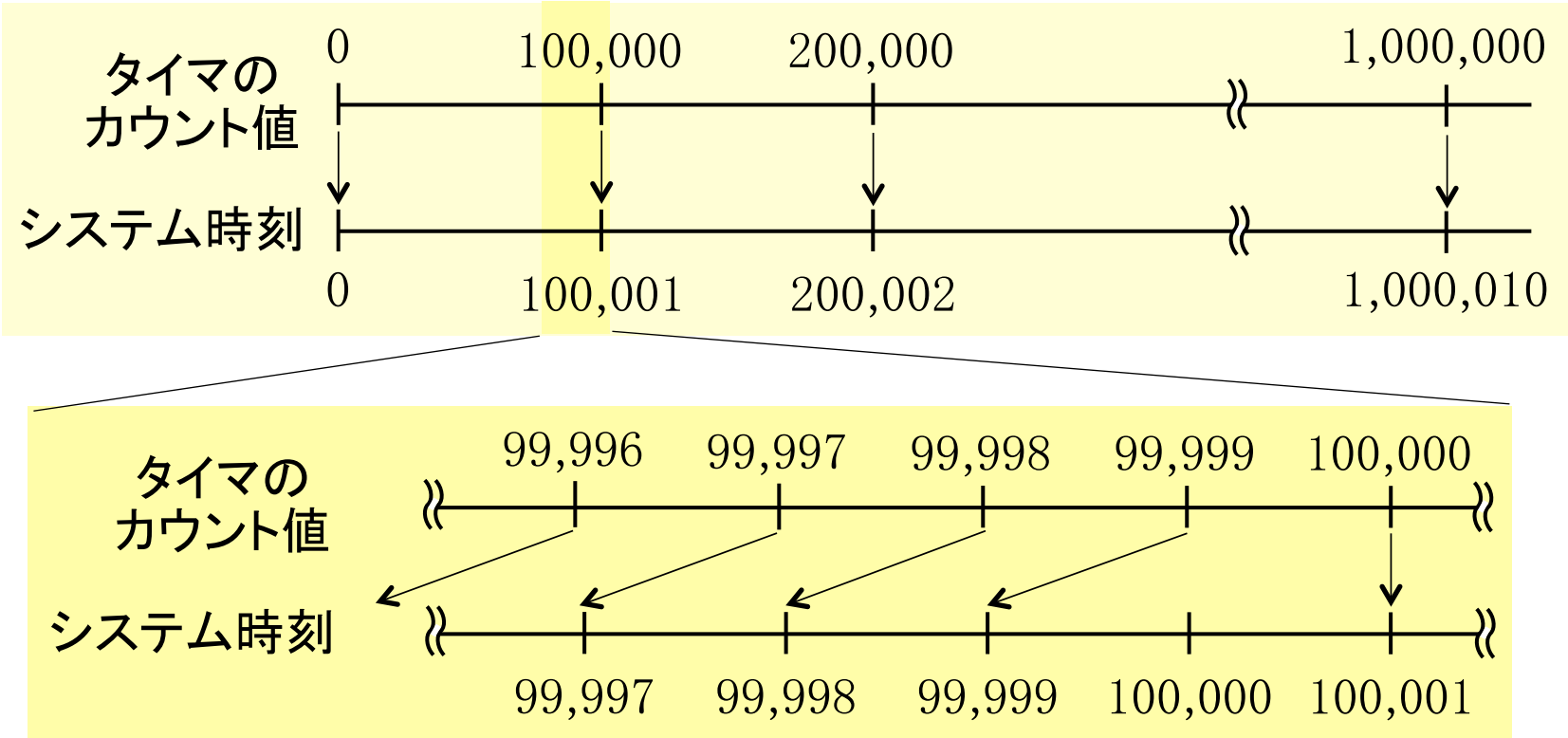
ドリフトの調整とドリフト量の設定

ドリフトの調整機能とドリフト量

- カーネルは，タイマ（ハードウェア）の進みに対するシステム時刻の進みを調整する機能を持つ
  - ±10%の範囲で調整できる
- ドリフト量は，タイマの進みに対して，システム時刻をどの程度早く（負の値の場合には，遅く）進めるかを指定するパラメータ．単位はppm
  - 例えば，ドリフト量を+10ppmとすると，タイマが1秒（＝1,000,000μ秒）進む間に，システム時刻は1,000,010μ秒進む
  - 逆に，ドリフト量を–10ppmとすると，タイマが1秒（＝1,000,000μ秒）進む間に，システム時刻は999,990μ秒進む

タイマとシステム時刻の対応例(ドリフト量:+10ppm)

- タイマのカウント値から，システム時刻(を早い方に丸めた値)を求める場合の対応付け



ドリフト量の設定（set_dft）

ER ercd = set_dft(int_t drift)

- ドリフト量（単位はppm）を，driftで指定した値に設定する
- driftに指定できるのは，±10%の範囲内（数値でいうと，–100,000～100,000の範囲内）
- 設定したドリフト量は，再度設定するまで有効

システム時刻の設定と参照

システム時刻の設定（set_tim）

ER ercd = set_tim(SYSTIM systim)

- システム時刻の現在値（64ビットの絶対時刻）を，systimで指定した時刻に設定する
- タイムイベントが発生するまでの相対時間は変化しない

システム時刻の参照（get_tim）

ER ercd = get_tim(SYSTIM *p_systim)

- システム時刻の現在値（64ビットの絶対時刻）を参照し，p_systimが指すメモリ領域に返す
- adj_timによってシステム時刻を戻した場合，get_timは，システム時刻が最も進んでいた時のシステム時刻を返す

時間パーティショニング機能

時間パーティショニング機能の概要

対応する要求

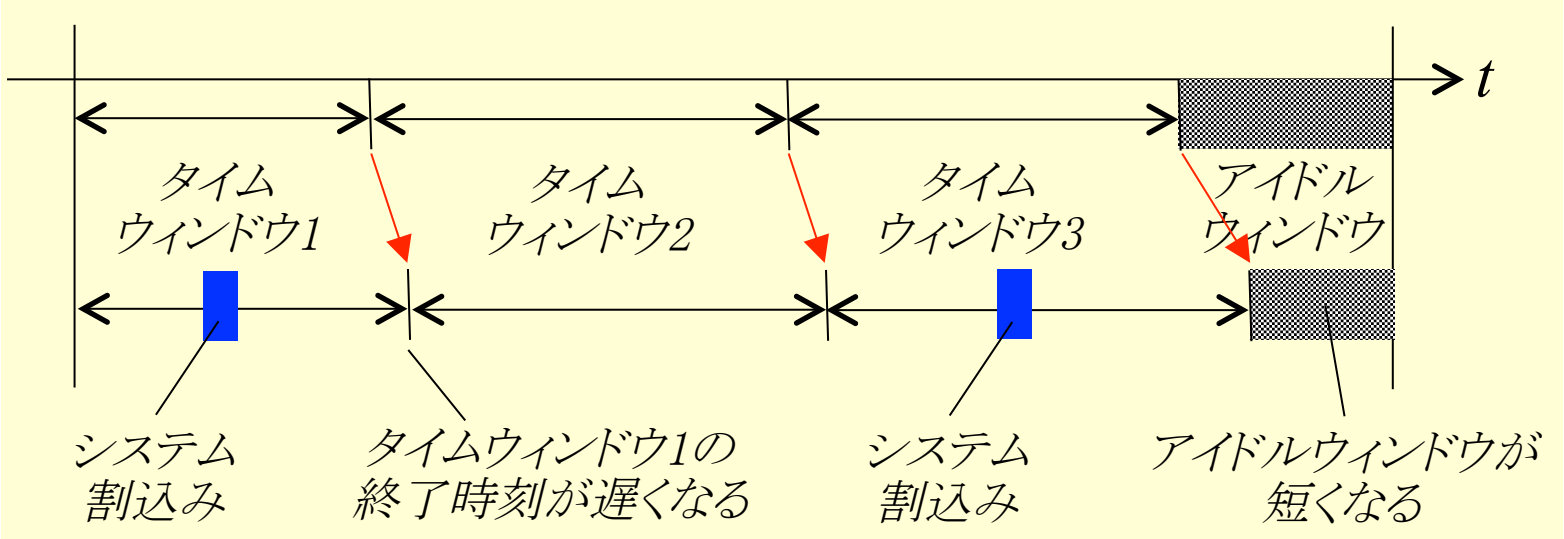
- 時間分割により共有されるプロセッサ（およびその一部とみなされる資源）のパーティショニング

基本的なアプローチ

- TOPPERS時間パーティショニングスキーム（次のスライドで説明）に基づく
- 保護ドメイン（厳密には，スケジューリングドメイン）単位でパーティショニングを行う
- タイムウィンドウを割り当てられていない保護ドメインは，アイドル時間（タイムウィンドウ内の空き時間，アイドルウィンドウ内の空き時間）に実行する

TOPPERS時間パーティショニングスキーム

- 各パーティションは，システム周期内で各パーティションを実行するタイムウィンドウを決める方式（航空機向け規格であるARINC 653で採用）をベースとして，システム割込み（タイムウィンドウによらずに受け付けられる割込み）を許すように拡張した方式でスケジュール
- パーティション内で複数のタスクを実行する場合には，従来のOSと同じ方式でスケジュール（階層型スケジュール）



時間パーティショニングに関する主な概念

システム周期（system cycle）

- 保護ドメインを繰り返し実行する基本的な周期

システム動作モード（system operating mode）

- 保護ドメインをどのように繰り返し実行するかを決定するモード
- ID番号（システム動作モードID）によって識別
- システム周期停止モード（TSOM_STP）では，保護ドメインを繰り返し実行するのを停止

タイムウィンドウ（time window）

- システム周期内の連続した時間区間
- タイムウィンドウは，システム動作モード毎に登録

アイドルウィンドウ（idle window）

- システム周期内で，タイムウィンドウに含まれない時間区間

タイムウィンドウの割当て

- タイムウィンドウは，1つのユーザドメインに割り当てる
  - 1つのユーザドメインに，任意の数のタイムウィンドウを割り当てることができる

アイドルドメイン（idle domain）

- どのシステム動作モードにおいてもタイムウィンドウを割り当てられていないユーザドメインを1つにまとめたもの
  - 該当するユーザドメインが1つ以上ある場合，スケジューリング上は，それらのユーザドメインをまとめて1つのユーザドメインであるかのように扱う
- あるオブジェクトが「アイドルドメインに属する」といった場合には，タイムウィンドウを割り当てられていないユーザドメインのいずれかに属することを意味する

時間パーティショニング時のスケジューリング規則

タイムウィンドウのスケジューリング

- プロセッサは，システム周期毎に，現在のシステム動作モードに対して登録されたタイムウィンドウを順に実行する
- システム周期が終わると，システム動作モードを，次のシステム周期での遷移先システム動作モードに切り換える

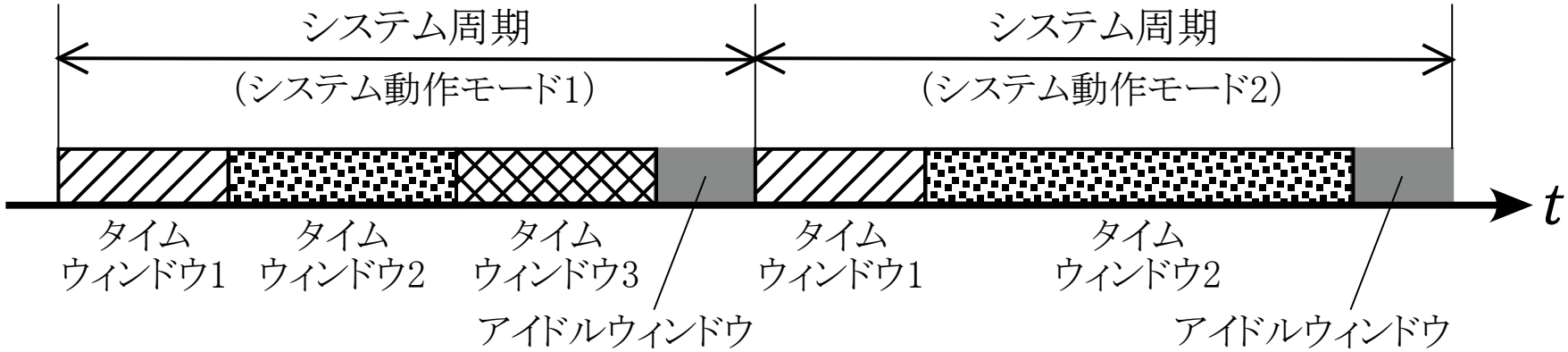


図2-4. システム周期毎のタイムウィンドウの実行

タイムウィンドウ内で実行するドメイン

- そのタイムウィンドウを割り当てられたユーザドメインに属するタスクを，優先度順に実行
- カーネルドメインに属する処理単位（タスク，割込みハンドラ等）は，すべてのタイムウィンドウで実行される
- 実行できる処理単位がない場合には，アイドルドメインに属するタスクを実行

タイムウィンドウの切換え

- カーネルドメインに属する処理単位が使用したプロセッサ時間の分，タイムウィンドウの切換えを遅延させる
  - 割込みハンドラとCPU例外ハンドラは，カーネルドメインにしか属することができない．つまり，すべての割込みは，システム割込みの扱いになる
  - カーネルドメインに属するタスクの処理時間も，タイムウィンドウに含まない扱いになる

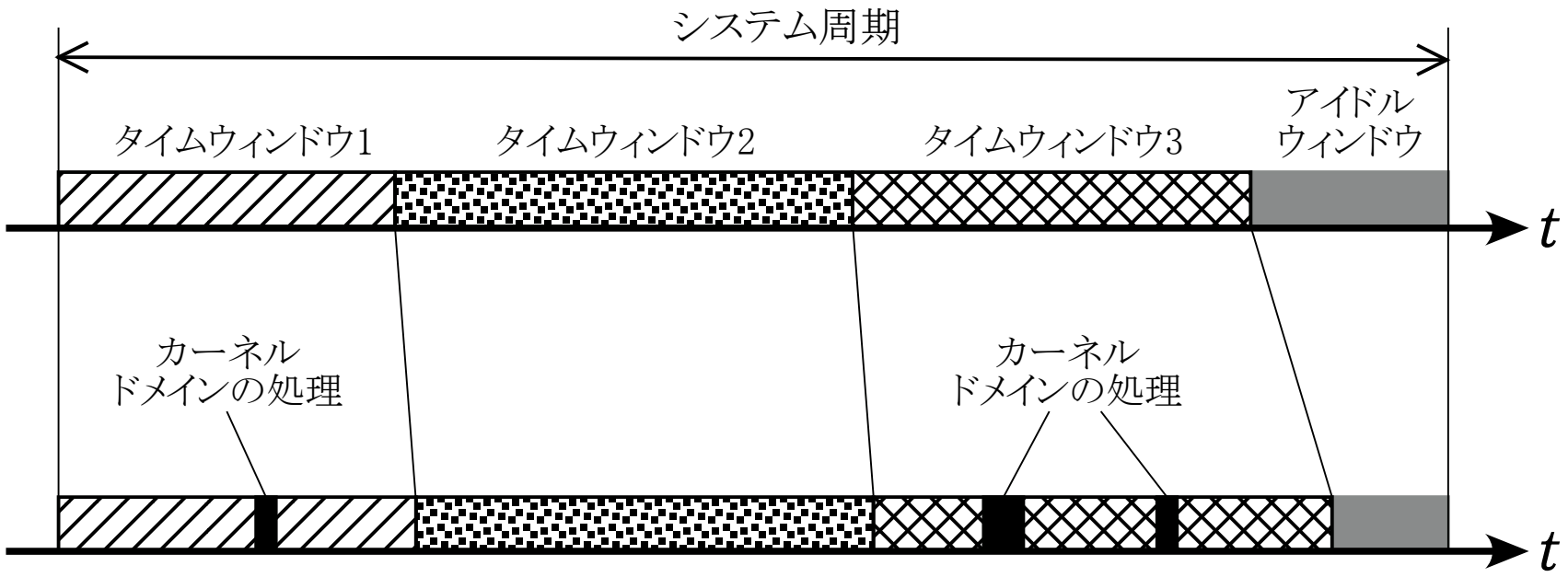


図2-5. タイムウィンドウの切換えタイミング

アイドルウィンドウの扱い

- すべてのタイムウィンドウの実行が終わると，アイドルウィンドウに
- アイドルウィンドウでは，アイドルドメインに属するタスク(とカーネルドメインに属する処理単位)を優先度順に実行

システム周期オーバラン例外

- システム周期の終了時刻に，タイムウィンドウが実行中であった場合（アイドルウィンドウでなかった場合）に発生する（エミュレートされた，カーネル管理外の）CPU例外
  - システム周期内で，カーネルとカーネルドメインに属する処理単位が使用するプロセッサ時間を考慮し，十分な長さのアイドルウィンドウを確保するのは，ユーザの責任

システム周期停止モード

- システム周期停止モードでは，システム周期の切換えも，タイムウィンドウの実行も行われず，カーネルドメインに属する処理単位のみが実行される
- 機器が待機している時に使うことを想定

注意事項

- 非タスクコンテキストの実行中，CPUロック状態，割込み優先度マスクが全解除でない状態では，システム周期の切換えとタイムウィンドウの切換えは保留される
- システム動作モードAにおいてはタイムウィンドウが割り当てられており，システム動作モードBにおいては割り当てられていないユーザドメインは，アイドルドメインにはまとめられない．そのため，システム動作モードBでは，そのユーザドメインは全く実行されない

時間パーティショニングに関するAPI

システム周期の設定（DEF_SCY）

DEF_SCY({ RELTIM scyctim })

- システム周期を scyctim に設定
- システム周期を設定しない場合，時間のパーティショニングを行わない

システム動作モードの生成（CRE_SOM）

CRE_SOM(ID somid, { ATR somatr, ID nxtsom })

- IDが somid，システム動作モード属性が somatr（初期モードかどうかを指定），遷移先システム動作モードが nxtsom であるシステム動作モードを生成
- nxtsom の指定を省略した場合，遷移先システム動作モードは，生成したシステム動作モード自身に設定される

タイムウィンドウの登録(ATT_TWD)

ATT_TWD({ ID domid, ID somid, int_t twdord,
PRCTIM twdlen })

- 下のパラメータで指定したタイムウィンドウの登録情報に従って，タイムウィンドウを登録
  - domid：割り当てるユーザドメインのID
  - somid：登録対象のシステム動作モード(名称でしか指定できない)
  - twdord：システム周期内での順序(タイムウィンドウは，twdordの小さい順に並べられる)
  - twdlen：タイムウィンドウの長さ(単位はμ秒)
- PRCTIM型は，プロセッサ時間を表すデータ型．従来仕様のオーバランハンドラのためのOVRTIM型は廃止し，PRCTIM型に統合

システム動作モードの変更(chg_som)

ER ercd = chg_som(ID somid)

- 次のシステム周期での遷移先システム動作モードを，somidで指定したシステム動作モードに設定
  - 再度chg_somを呼び出さない限りは，現在のシステム周期の終了後に，somidで指定したシステム動作モードに切り換わる
- ただし，現在のシステム動作モードがシステム周期停止モード(TSOM_STP)の場合には，somidで指定したシステム動作モードに即座に切り換わる
- somidにTSOM_STPを指定した場合，現在のシステム周期の終了後(現在のシステム動作モードがシステム周期停止モードの場合は，即座)に，システム周期停止モードに切り換わる

ユーザドメインに属するタイムイベント処理

タイムイベント処理機能の見直し

- ユーザドメインにおいても，安全に使用できる周期処理／アラーム処理のための機能として，周期通知／アラーム通知機能（タイムイベント通知と総称）を設けた … 後で説明
  - 従来の仕様の周期ハンドラ／アラームハンドラは，カーネルドメイン（特権モード）で実行されるため，ユーザドメインで自由に使うことができない

ユーザドメインに属するタイムイベント通知の処理タイミング

- ユーザドメインに属するタイムイベント通知は，そのユーザドメインに割り当てられたタイムウィンドウへの切換え直後に処理される
- アイドルドメインに属するタイムイベント通知は，アイドルウィンドウへの切換え直後に処理される

その他の新規機能と仕様変更

タスク終了要求機能

対応する要求

- アプリケーション(パーティション)の安全な停止
  - アプリケーション間のパーティショニングを行い，アプリケーション単位の終了／再起動を行う場合に，アプリケーションを安全に停止させることができること

機能の必要性

- アプリケーションを構成するタスクを，他のアプリケーションとの共有リソースをアクセス中に強制終了させると，他のアプリケーションに悪影響が及ぶ
  - 共有リソースには，拡張サービスコール中でアクセスするのが原則
- そこで，タスクが強制終了すると不都合な状況では，タスクの終了を遅延させ，タスクを安全に停止させることを可能にする必要がある

機能概要

- タスクに対して終了要求を行うと，対象タスクの終了が遅延されている状態（以下のいずれか）でない限り，対象タスクは即座に終了する
  - CPUロック状態の間
  - 割込み優先度マスク全解除状態でない間
  - ディスパッチ禁止状態の間
  - タスク終了禁止状態の間
- 対象タスクが終了禁止状態であれば，タスク終了要求フラグをセットする．また，待ち状態を解除し（E_RASTERが返る），強制待ちから再開する
- タスク終了要求フラグがセットされている状態では，タスクを広義の待ち状態にする可能性のあるサービスコールは呼び出せない（E_RASTERエラーとなる）

追加するサービスコール

- ras_ter：タスクの終了要求
  - 指定したタスクに終了要求を行う
  - 対象タスクが待ち状態であれば待ち状態を解除，強制待ち状態であれば強制待ちから再開する
  - マルチプロセッサ対応では，自タスクと同じプロセッサに割り付けられているタスクに対してのみ発行可能
- dis_ter：タスク終了の禁止
- ena_ter：タスク終了の許可
- sns_ter：タスク終了禁止状態の参照

ref_tskで参照できる情報の追加

- ref_tskで参照できる情報に以下を追加
  - raster：タスク終了要求状態
  - dister：タスク終了禁止状態

タスクの終了禁止の制限

- ユーザアプリケーションがタスク終了を禁止することができると，そのアプリケーションの終了・再起動ができない状況があるため，それを制限する機能が必要
  - ただし，この状況は，他のアプリケーションへ悪影響を及ぼしているわけではないため，許容可能な場合も
- 明示的に許可しない限り，ユーザタスクが，dis_dsp, ena_dsp, dis_ter, ena_terを呼び出すことを禁止

代わりに廃止する機能

- タスク例外処理機能
  - DEF_TEX, ras_tex, dis_tex, ena_tex, sns_tex, ref_tex, xsns_xpn
- 待ち禁止状態に関する機能
  - dis_wai, ena_wai
- タスク例外処理マスク状態

動的負荷分散の支援機能

対応する要求

- マルチコアプロセッサにおいて，動的負荷分散を行うミドルウェアを実現するために必要な機能を持つこと
  - 適切な動的負荷分散の方法は，アプリケーションによって異なるため，動的負荷分散はミドルウェアで実現することとし，カーネルではその実現に必要な最低限の機能を提供
  - 少なくともタスク数平準化方式と仕事量平準化方式を実現できるようにする

参考文献

[1] 石田利永子，本田晋也，高田広章，福井昭也，小川敏行，田原康宏：マルチストリーミング処理のためのマルチプロセッサ向けロードバランス機構，電子情報通信学会論文誌，vol. J96-D，no. 1，pp. 168-182，2013年1月．

タスクおよびスケジューリング状態の参照

- get_tst：タスク状態の参照
  - タスク状態を参照する（ref_tskと異なり，タスク状態のみを参照するため，オーバヘッドが小さい）
- get_nth／mget_nth：指定した優先順位のタスクIDの参照
  - 指定した優先度を持つ実行できる状態のタスクの中で，指定した優先順位のタスクを参照する
- get_lod／mget_lod：実行できるタスクの数の参照
  - 指定した優先度を持つ実行できる状態のタスクの数を参照する

サブ優先度機能

- タスクにサブ優先度の情報を持たせ，サブ優先度を設定するサービスコール（chg_spr）を追加
  - サブ優先度はuint_t型で表現し，uint_t型で表現できるすべての値を使うことができる
- ある優先度をサブ優先度を使ってスケジュールするものと（静的APIにより）指定すると，その優先度の中では，高いサブ優先度を持つタスクから順に実行
  - サブ優先度も同じであった場合にはFCFS順
- サブ優先度は，タスクの生成時に最も低い優先度（＝UINT_MAX）に初期化し，タスクが休止状態の間も有効

追加するAPI

- ENA_SPR：サブ優先度を使用するタスク優先度の設定
- chg_spr：タスクのサブ優先度の変更

タイムイベント処理機能の見直し

対応する要求

- ユーザドメインにおいても，安全に使用できる周期処理／アラーム処理のための機能を用意したい
  - 現行仕様の周期ハンドラ／アラームハンドラは，カーネルドメイン（特権モード）で実行されるため，ユーザドメインで自由に使うことができない

機能概要

- ユーザドメインの周期処理／アラーム処理では，あらかじめ定められた安全なイベント通知処理（タスクの起動，起床，セマフォの返却など）のみを行えるようにする
  - 周期通知／アラーム通知機能と呼ぶことに
- 保護機能（保護ドメイン）のないカーネルにも，下位互換性のために導入する

API仕様

周期通知を登録する静的APIの例

CRE_CYC(USER_CYC1, { TA_STA,
{ TNFY_ACTTSK, TASK_1 }, 5000U, 5000U);

- 5ミリ秒周期で，TASK_1を起動する

CRE_CYC(USER_CYC2, { TA_STA,
{ TNFY_WUPTSK|TENTF_SETFLG, TASK_2,
FLG_1, 0x0001U }, 5000U, 5000U);

- 5ミリ秒周期でTASK_2を起床し，それがエラーになった場合には，FLG_1の最下位ビット（0x0001）をセットする

CRE_CYC(KERNEL_CYC1, { TA_STA,
{ TNFY_HANDLER, exinf, cyclic_handler1 },
5000U, 5000U);

- 5ミリ秒周期で，cyclic_handler1を呼び出す（カーネルドメインでのみ使用可能）

その他の仕様のスリム化・シンプル化

非タスクコンテキスト専用のサービスコールの廃止

- サービスコール名称が“i”で始まる非タスクコンテキスト専用のサービスコールは廃止
- 非タスクコンテキストからも，act_tskやsig_sem等を呼ぶ（呼べるサービスコールが増えるわけではない）

ミューテックス機能の仕様変更

- ミューテックスの解放順序をLIFO順に制限

メッセージバッファ機能の仕様変更

- メッセージバッファの送信待ち行列をタスクの優先度順にする機能を廃止

割込み優先度マスクの操作

- 割込みサービスルーチンとタイムイベントハンドラからのリターン時に，割込み優先度マスクは「そのまま」に

その他の仕様変更

保護ドメインに対するアクセス許可ベクタの新設

- 保護ドメインに対するアクセス許可ベクタを新設
- システム状態に対するアクセス許可ベクタで制御していた一部のサービスコール呼出しを，保護ドメインに対するアクセス許可ベクタで制御するようにした

起動要求をキューイングしないタスクの追加

- 起動要求をキューイングしないタスクを表すタスク属性とTA_NOACTQUEを追加

性能評価用システム時刻の概念の変更

- 性能評価用システム時刻を参照するサービスコール（get_utm）を廃止
- 性能評価に用いるために，高分解能タイマを参照するサービスコール（fch_hrt）を追加

割込みサービスルーチン(ISR)IDの付与

- ISRは必ずIDを持つことに
- ATT_ISRは廃止し，CRE_ISRを標準でサポートする

保護ドメイン毎の標準メモリリージョン

- 保護ドメイン毎に標準メモリリージョンを定義できるように
  - 保護ドメイン毎に，メモリ配置(コードやデータを何番地から置くか)をユーザが制御できるように

標準ROMリージョンの属性

- 標準ROMリージョンは，TA_NOWRITE属性でなければならないという制約を外した

おわりに

今後の予定・計画

TOPPERS/ASP3カーネル

- Release 3.0.0を今年末（または年明け）に一般公開の予定
- 各種のプロセッサへのポーティングを進める

TOPPERS/HRP3カーネル

- かなり実装が進んでいる
  - 時間パーティショニング機能はほぼ完成
- TOPPERS会員は，最新のバージョンをsubversionからアクセス可能

その他のカーネル

- TOPPERS/FMP3カーネル，TOPPERS/SSP3カーネルの実装は今後

課題

- テストスイート（TTSP3）の開発