RTOS を用いた

リアルタイムシステムの開発手法

14th Nov. 1997

高田 広章
ITRON専門委員会／東京大学 理学部

hiro@is.s.u-tokyo.ac.jp
http://tron.um.u-tokyo.ac.jp/~hiro/

はじめに

リアルタイムシステムとは?

… いくつかの定義がある

- 処理結果の正しさが、出力される結果値の正しさに加えて、結果を出す時刻にも依存するようなシステム

- 多くの組み込みシステムはリアルタイムシステム
- 一部のタスクのみにリアルタイム性が求められる場合も

! 単に高速なシステムをリアルタイムシステムと呼ぶわけではない

リアルタイムシステムの設計手法

- システム設計時からリアルタイム性を考慮すべき

この講演の目的

RTOS を使ったシステム設計における基本課題に1つの解

- どのようにタスク分割するか？
- どのように優先度を付けるか？

リアルタイムスケジューリング理論からのアプローチ

タスク分割を行う理由

- モジュール化のための分割 … 以下では扱わない

  → *生産性, 保守性, 信頼性の向上*

  - 異なる機能単位は別タスクへ
  - 開発者 (グループ) 毎にタスクを分ける
- リアルタイム性向上のための分割

前提知識

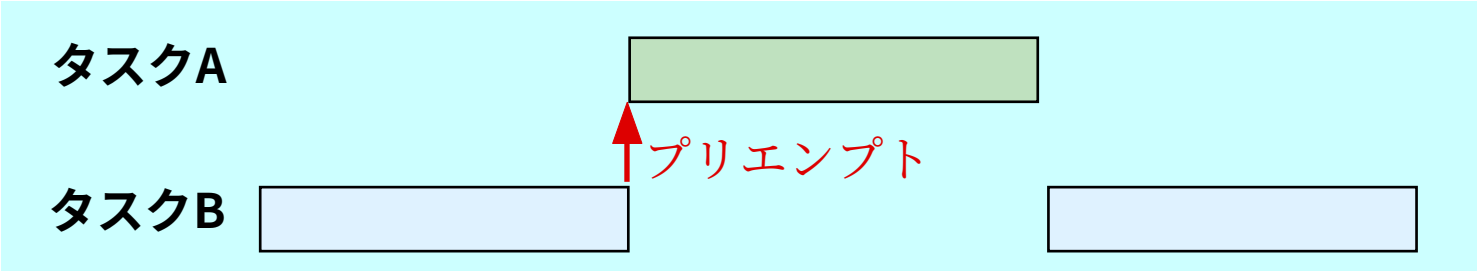
RTOS のスケジューリング方式

- 優先度ベーススケジューリング
  - 最も優先度の高いタスクが実行される
  - 優先度の高いタスクが実行できなくなるまで、優先度の低いタスクは実行されない
  - 同優先度間では FCFS (First Come First Served)
- プリエンプティブスケジューリング
  - 優先度の高いタスクが実行可能になると、優先度の低いタスクが実行途中であっても、タスク切替が起こる
  - 実行可能になるきっかけは割り込み

! TSS のためのスケジューリングとは大きく異なる

タスク分割によるリアルタイム性の向上

- （例）タスクA … 処理時間：短，デッドライン：短
  タスクB … 処理時間：長，デッドライン：長



RTOS なしで実現するためには…

- タスクB の中で定期的にタスクA の実行が必要かをチェックする必要
  - ➡ 応答性が低下，チェックのオーバヘッドが増える
  - ➡ タスクB を改造するチェック箇所の見直し

➡ リアルタイム性を持ったプログラムの保守性が向上

結論から言うと…

どうやって判定するか？

優先度割付

- すべてのタスクが時間制約を守って実行できるなら
  急ぐタスクに高い優先度
- 時間制約を守れないタスクが出てくる場合には
  重要なタスクに高い優先度

タスク分割

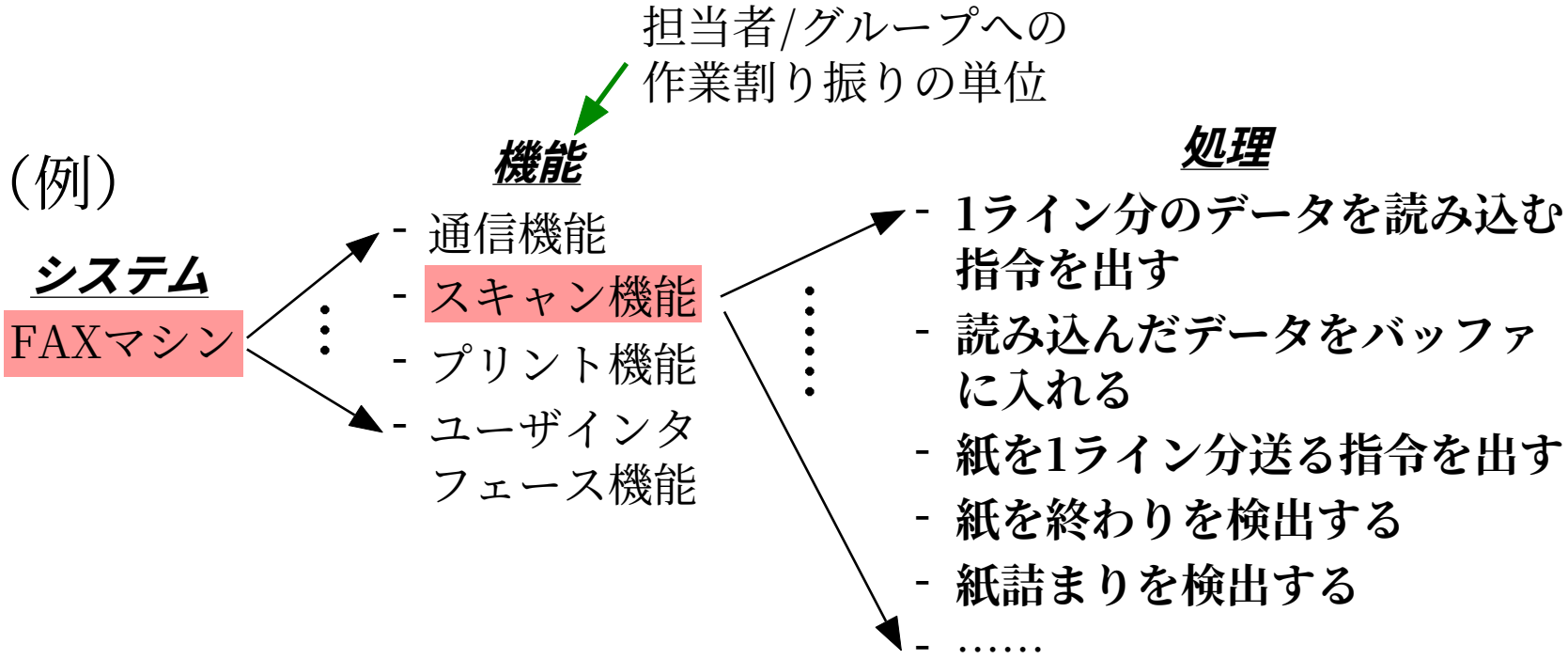
- 上記の優先度割付を可能にするようにタスクを分割する
  - 急ぐ度合いが違う処理は別タスクに
  - 重要度合いが違う処理は別タスクに

具体的にどうすれば良いか？

システム分析

システム → 機能 → 処理

- ここでは、システムが機能を経て、個々の処理 (関数程度の大きさ, タスクより小さい) に分割されたところから考え始める



処理の性質の洗い出し

- 起動タイミング/条件
  - その処理をいつ実行する必要があるか？
  - 起動要因

    外部イベント，時間，他の処理
  - 最大起動頻度
- 最大実行時間

この2つの積がシステムにかける最大負荷

- 時間制約とその厳しさ
- 処理の重要性
  - 処理をさぼることは可能か？
- 排他制御条件

時間制約とは?

＝ 処理の実行完了時刻に関する制約

時間制約のタイプ

- ある時刻までに実行完了しなければならない ➡ 本質的
- ある時刻以降に実行完了しなければならない
  ➡「ある時刻」に実行開始すればよい
- ある時間区間内に実行完了しなければならない
  ➡ 上の2つの組み合わせ

出所に着目した分類

(a) アプリケーションシステムの要請からくる本質的な時間制約

(b) 用いるハードウェア構成の制限からくる時間制約

(c) 用いるソフトウェア部品の制限からくる時間制約

体系的に洗い出すには?

(A) 入出力の時間的関係に対する制約
(B) 入力間の時間的関係に対する制約
(C) 出力間の時間的関係に対する制約

時間制約の厳しさ

- ハードリアルタイム
- ファームリアルタイム
- ソフトリアルタイム
- 非リアルタイム

時間駆動型の処理に対する時間制約

! この場合、本質は制御の精度

タスク分割の基本

同じ性質を持つ処理は同一のタスクへ

除: 最大実行時間，排他制御条件

- 同一の起動タイミング/条件
- 同一の時間制約とその厳しさ
- 同一の処理の重要性

! 分割しすぎはオーバヘッドの増大を招く

- 性質が少し違うだけの処理は、厳しい方の性質にあわせるなどの方法で、同一タスクに入れることが可能

優先度割付の基本

急ぐタスク程、高い優先度

スケジュール可能性の解析

- すべての処理のすべての性質がわかっていれば、時間制約を満たせるか（スケジュール可能性）を数学的に検証できる

過負荷対策

- すべての処理を時間制約を満たすには、プロセッサの能力が足りない場合

- プロセッサを高速のものにする
- バッファを使って負荷を均す
- いずれかの処理をさぼる (QOS制御)

負荷を均す

【例】時間区間 [*t1*, *t2*] に実行完了すべき処理

- 時刻 *t1* に実行開始すると、*t2* までに実行を完了できない可能性

  プロセッサに対する負荷が [t1, t2] に集中したため、処理能力をオーバーした

- 処理を計算部分と結果を出力する部分に分割

  [*t1*, *t2*] には出力処理のみ行えばよい

  バッファを使って負荷を均すことができた

QOS制御

- 重要でない処理, さぼっても良い処理の優先度を下げる

重要なタスク程、高い優先度

その他の話題

- 排他制御
  優先度逆転の問題に注意が必要
- バッファの実現
  RTOS のメイルボックス機能を使うのが手軽
- 他の処理による起動
  RTOSのタスク間同期・通信機能を使う
- モードの導入
  システムの状態によって、動作するタスクの種類、タスクの振舞が大きく変化する場合
- 割り込みハンドラ

おわりに

- RTOS … 拡張性・保守性の高いリアルタイムシステムを実現する上で重要

- RTOS の利点を活かす設計が必要
  - タスク分割
  - 優先度割付

ITRONプロジェクトでの取り組み

- ITRONハードリアルタイムサポート研究会
- リアルタイムスケジューリング理論に基づいたアプリケーション構築ガイドラインの作成 (ソフトウェア部品を重視)
- それをサポートするためのカーネル仕様の検討