第 6 回 セマフォによる排他制御

こんにちは。
前回までは OS の機能を使わないで並行処理を行うプログラム間で排他制御を行う手段に関して述べてきました。今回は RTOS の提供する、排他制御のための最も基本的な仕組みであるセマフォに関して説明します。ただしセマフォの基本的な使い方などは説明していませんので、マニュアルなどで確認してください。

ここで少し復習しておきます。まず組込みシステムとは現実の様々なイベントに対応して、さまざまな処理を実行しなければなりません。そのためにはひとつのプログラムの中で、さまざまなイベントに対応した処理を行うよりも、個々のイベントに対応したプログラムを個別に作っておき、イベントの発生に応じてそのプログラムを実行したほうが、呼び出す処理と呼び出される処理が分離され、さらに単独のイベントに対する処理の作成に専念できるためプログラムも短くなり、品質も向上します。
もちろん処理に優先度をつけリアルタイム性を満足させたり、ある処理の実行中にそれを一時停止しておき他の処理を実行させることも並行処理の実行による利点となります。
並行処理実現の具体的手段（これもインスタンスと呼んでもよいのかもしれません）としては、割り込み処理(ISR)と OS の機能であるタスクがあります（OS が並行処理を実現するためにも割り込みを利用するという意味では、すべて割り込みなのですが・・・）。ISR とタスクは、ある意味並行処理の二重構造なのですが、その話をすると長くなるのでここではそういうものだということで話を進めます。

RTOS を使用している組込みシステムでは、ほとんどの並行処理はタスクで実現しています。並行処理をタスクで実現するほうが、RTOS の提供する様々な機能を使用することが出来るし、デバッグにも RTOS の提供する様々な機能を利用できるからです。
排他制御のために RTOS が提供している、最も基本的な機能が今回お話しするセマフォです。セマフォは排他制御だけでなく、同期にも使用しますが、ここでは排他制御に話を絞ります。
表 1 は第 4 回にも載せたタスクと ISR の比較の排他制御に関連する部分のみの抜粋です。

表 1 タスクと割り込み処理の比較

並行性の実現手段	タスク	ISR
排他制御の手段	セマフォが代表 他にも RTOS の提供する多くの手段がある	割り込み禁止および許可
排他制御の	関連するタスクのみに限られる	システム全体、全 ISR に及ぶ優先度の高い

（悪）影響	関連しない優先度の高いタスクには影響を与えない 優先度逆転などによるデッドロックに注意	ISR も停止させてしまう
特徴	多項目に及ぶが使い方を誤ってもシステム全体に影響を及ぼすことはまれ ただし優先度の高いタスクで使用する場合は要注意 またデッドロックにも要注意	比較的単純に使えるが、使い方を誤るとシステム全体の性能に影響を及ぼすため危険 使い方を誤ると最悪システムダウンを招く

例によってセマフォの一般的な仕組みを示します。まず図 1 はセマフォのデータ構造を、リスト 1 は RTOS の内部的なセマフォの処理の概要を示します。

図 1 セマフォのデータ構造

```
semTake(セマフォID…)
{
  if ( S >= 1 ) {
    S = S - 1;
    semTake()を発行したタスクはそのまま実行を続ける;
  }
  else {
    実行中のタスク(semTake()を実行したタスク)は
    そのセマフォに対する待ち行列に入れる;
    つまりセマフォ開放待ち状態になりCPUの使用権を放棄する;
      ⇒ ディスパッチャ経由で他のタスクが実行を開始する;
  }
}

semGive(セマフォID…);
{
  if(セマフォの待ち行列が空でない) {
    待ち行列の先頭タスクを実行可能状態に変更してレディキューに繋ぎかえる;
      ⇒ semGive()を実行したタスクも含めてディスパッチャ経由で
        再スケジューリングされる;
    }
  else {
    S = S + 1;
    semGive()を発行したタスクはそのまま実行を続ける;
  }
}
```

リスト 1 セマフォの内部動作

セマフォのデータ構造はセマフォの値(0 以上の整数値 S)と、そのセマフォが解放されるのを待っているタスクの待ち行列、そのセマフォを所有しているタスクの情報、待ち行列が FIFO 順なのか、優先度なのかの情報など、優先度継承セマフォなどセマフォの種類によっては必要なその他の情報などから構成されています。
またリストはシングル CPU の場合です。話を簡単にするため以後はシングル CPU のみを対象とします。なお特に特定の RTOS の内部構造を示したものではありません。
セマフォ操作中、タスクの待ち行列を操作するような処理は割り込み禁止にして、排他制御を実現していますが、その部分は省略しています。
semGive が ISR から呼び出された場合、semGive の大部分は ISR のコンテキストで実行されますが、⇒以降の処理は ISR の中では実行されず、ISR を終了して RTOS に制御が移った時に実行されます。RTOS の種類にもよりますが、セマフォの解放処理を ISR から呼び出せるとしても、ISR の中で実行されるのはシステムレディキューへのタスクの登録までで、セマフォを獲得できたことによって実行可能になったタスクが実際に実

行されるのは ISR が終了して RTOS に制御が移り、さらに RTOS がタスクを起動するタイミングになってからです。もちろん、その場合も実行はタスク優先度に応じてです。
タスクからセマフォ獲得要求(semTake)を発行してそのままセマフォを獲得できた場合、セマフォ解放操作(semGive)をして待っているタスクがいない場合は、そのままそのタスクが実行を続け、タスクの再スケジューリングは発生しないことに注意してください。

説明の順番が逆になってしまいましたが、今説明したように、通常はタスクからしか呼び出せないセマフォですが VxWorks では semGive()は ISR からも呼び出すことが出来ます。
リストからもわかるように semGive()ではセマフォの解放待ち状態になることはないからです。もちろん ISR からの semGive()の呼び出しではその時点ですぐにタスクが起動されるわけではありません。
semGive()を ISR から呼び出す場合は semTake()で待っているタスクとの同期などに使用する時などです。

VxWorks のセマフォとしては排他制御セマフォ、バイナリセマフォ、カウンティングセマフォがあります。
排他制御セマフォは、いうなればタスク間の排他制御専用のセマフォで、機能的にはバイナリセマフォなのですが、それに優先度継承機能や、タスク削除遅延機能、セマフォを所有しているタスクは同じセマフォに対して複数回の semTake()を発行できる再帰的アクセス機能などを追加したものです。なお排他制御セマフォに対しては ISR からは semGive()であっても呼び出すことはできません。
バイナリセマフォやカウンティングセマフォは排他制御にも同期にも使用できるセマフォです。
RTOS の中にはバイナリセマフォというものはなく、単に最大値が 1 のカウンティングセマフォの扱いとなっているものもありますが VxWorks では区別しています。
それは VxWorks においては、最も使用頻度の高いセマフォはアセンブラで作られていて、速度が最適化されているからです。一方カウンティングセマフォのほうは C 言語で作られていて移植性を確保しています。これは RTOS メーカのやり方として、ユーザーメリットとメーカーメリットを両立させる、なかなか良い選択だと思います。

生産者と消費者の問題、リーダーライター問題、食事する哲学者の問題など古典的排他制御の問題を含む各種排他制御の問題は、個人的には興味があり、書いてみたいのですが今回の趣旨を外れるのと、分量の関係からまた別の機会に回します。

その代わり、最後にまた一つ問題を出しておきますので、興味のある人は是非考えてみてください。
今回書いたようにカウンティングセマフォでバイナリセマフォを実現することは簡単です。では逆はどうでしょうか？
バイナリセマフォしか提供していない RTOS のもとでタスクレベルでカウンティングセマフォを実現する手段（プログラム）を考えてみてください。
割り込み禁止／許可も使用しないこととします。

次回はデバイスドライバに関して述べたいと思います。