第 3 回 割り込みと RTOS

こんにちは。今回はソフトウェアによる排他制御の話をする予定でしたが、その前にまず RTOS と割り込みの関係をお話ししたほうがよいと思い予定を変更します。

OS はタスクを並行に動作させる機能を提供しますが、どのようにして実行させるタスクを決定しているのでしょうか？研究ベースではない、実際に製品に使われているほとんどの RTOS は、タスクをどのような順番で走らせるかを決めるスケジューリング方式として、優先度ベースのプリエンプティスケジューリング方式を採用しています。優先度はシステムを動作させる前に決めておく方式で、静的スケジューリングとも呼ばれています。
図 1 は VxWorks の一世代前の統合開発環境である Tornado の WIND VIEW というアドオンツールですが、タスクの動作を視覚的に見ることができます。
図では優先度が低い tTaksA タスク実行中に優先度が中の tTaskB タスクが実行可能になったため、それに制御が移り、さらに優先度が高い tTaskC に制御が移っていき、それぞれのタスクの終了によりそれぞれ一時停止刺されたタスクが再開する様子を示しています。優先度は処理の重要度に応じてあらかじめタスクに与えておきます。また処理に応じて優先度を実行中に変更することも可能です。

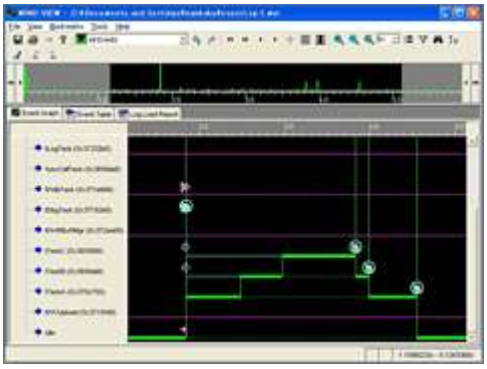

図 1 VxWorks の統合開発環境 Tornado のツールの一つである WIND VIEW

では実行しているタスクが RTOS によって実行を切り替えられるとしても、実行中のタスクから RTOS に制御が切り替わらなければこれは実現できないことになります。このタスクの実行から RTOS の実行への制御の切り替えはどのような仕組みで実現しているのでしょうか？
実はこれも割り込みを使用しています。

前回、CPU ひいては RTOS が現実世界の変化を知るのには 3 種類のやり方があるという話をしました。その中では割り込みのみがあるプログラムが動いている最中に、強制的にほかのプログラムに切り替えることが出

来ます。アプリケーションプログラムを実行する主体である、タスク実行中にも、実は図 2 に示すように定期的に外部割り込みをいれて、それにより定期的に RTOS を起動しています。

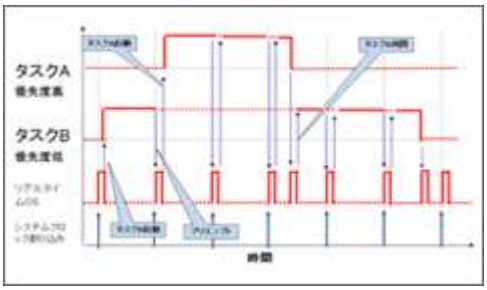

図 2 RTOS とシステムクロックの関係

OS を実装するうえで、システムクロックと呼ばれる定期的な割り込みが必要です。図に示すように、タスク実行中もシステムクロック割り込みにより絶えず OS に制御が移っています。OS に制御が移る（言い換えると OS が呼び出される）タイミングは、外部割り込みを終了するときや SVC（スーパーバイザーコール、OS がアプリケーションに提供するサービス、システムコール、サービスコールとも呼ぶ）を呼び出した場合などがありますが、これらはシステム構成や、システムの実行状態によっては長期間呼び出されないことも考えられます。そこで OS が呼び出される最長の（最低限の）周期を作り出すものが、OS のための割り込み、システムクロック（システムティックとも呼ばれています）で、OS はこれをベースに内部の時間管理を行います。これは Unix や Windows だと通常 1 秒間に 60 回程度、RTOS で 1 秒間に数百回程度周期的に CPU に入れるタイマー割り込みです。スリープなど時間指定の待ちはこのシステムクロックを基準に実行されるため、待ち時間はシステムクロックの整数倍になります。
インダストリアル分野の NC 装置などでは、もっと早い周期でサーボアンプにデータを送らなければならないため、最優先の周期割り込みはサーボ通信割り込みで、システムクロック割り込みは重要ではなく、システムクロック割り込みの優先度もそれほど高く設定していませんでした。
しかしシステムクロック割り込みをシステム全体の動きの基準として用いる場合は、システムクロック割り込みの優先度もそれなりに高く設定すべきでしょう。RTOS は CPU のドライバの側面もあるという話は聞かれたことはあるとは思いますが、このように RTOS 自身もいわばシステムクロック割り込みハンドラとしての側面も持っていることも知っておいてください。
ところで、VxWorks には標準では周期起動割り込みの仕組みは提供されていませんが、これは定期割り込みを発行できるタイマーハードウェアを利用すれば簡単に実現できるので、特に不便に思ったことはありません。逆に周期起動が提供されている場合は、それを利用する複数の周期起動ハンドラ間の関係や、システムクロックの割り込みレベルなどが見えないため、かえって厳密な設計がしにくくなると思います。

話だけではイメージがつかみにくいと思うので、割り込みハンドラとスケジューラのフローチャートを示します。VxWorks など現役の RTOS のフローを示すのは差し障りがあると困るので、この連載の第 1 回で紹介した元祖 RTOS と呼べる mtos のものを示すことにします。（なお実は mtos はその後 C 言語で書き直されて別の RTOS として再登場しているのですが、ここではそれには触れないことにします）。

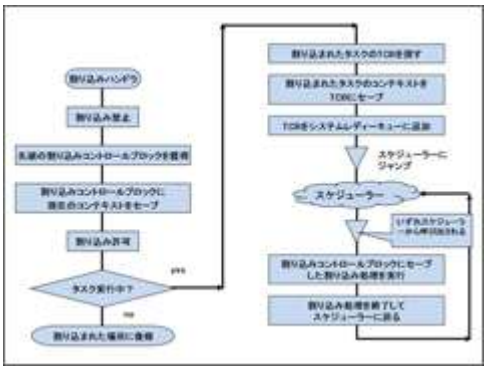

図 3 割り込みハンドラ

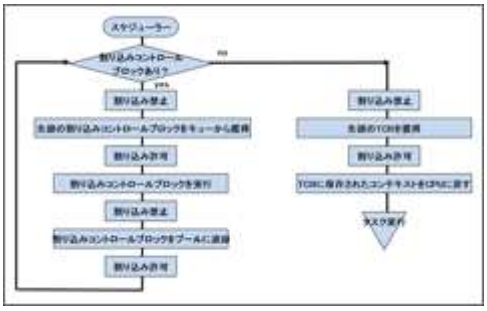

図 4 スケジューラ(ディスパッチャー）

図 3 に割り込みハンドラの図を、図 4 にスケジューラの図を示します。mtos は割り込みに関しては割り込みの要求のみキューイングしておき、その実行はスケジューラの最初にタスクを起動する前に実行するという方式で少し複雑です。これは mtos の特徴で、VxWorks とは似ている面も異なっている面もあります。
図 3、図 4 には割り込みコントロールブロックという言葉が出てきますが、これが割り込み処理部（アドレス）をキューイングするための物です。
図 3 の割り込みハンドラの構造で重要なのは、割り込みが発生した時に実行されていたのがタスクの場合はスケジューラに飛び、それ以外の場合はスケジューラを経由せず割り込まれた場所に直接戻ることです。これはほとんどの RTOS で共通の仕組みです。
この理由はなんでしょうか?
これは宿題にしましょう、考えてみてください。
図 3 から見て取れるように、RTOS のスケジューラはシステムクロック割り込みだけでなく、ほとんどのハードウェア割り込みからも呼び出されます。
次回は今回の宿題の解答とともに、並行処理とは切っても切れない関係にある排他制御に関して説明したいと思います。

第 3 回おわり

コラム : HW のシステムクロックと OS のシステムクロック

システムクロックとは、説明したように OS の基準クロックとして使われるものの呼び方ですが、他に CPU の動作の基準として使われるものもやはりシステムクロックと呼ばれる場合があります。この CPU のシステ

ムクロックとは、例えば Core i7 のクロックが 2GHz とか 3GHz とよばれるもののことです。組込みシステムに使われる CPU は必ずしもそれほど速いクロックを使うとは限りませんが、比較的性能が低いものでも 20MHz 程度はある場合が多く、OS のシステムクロックが 10ms 周期つまり 100Hz だとして、少なくとも速さは 20 万倍以上違うわけです。どちらのシステムクロックのことを言っているのかは話の流れの中で判断してください。もちろん本章では、このコラムに出てくる以外のシステムクロックはすべて OS のシステムクロックのことです。
なお上記の紛らわしさを避けるため、OS に供給するシステムクロックはシステムティックと呼ぶことも多いです。またタイムティックとよぶ RTOS もあります。