ARMマイコン No.2

あれこれやりたくなったなら…
機能ごと小分けで
楽々プログラミング!

Cortex-M3用
マルチタスク・ミニOSの
全ソース・コードを公開!

ARMでOS超入門

解説&製作 Cortex-M0マイコン基板による超並列機の試み

製品情報 各社ARM/Cortexマイコン情報

- マルチタスクOSとは?
- タスク間通信
- スケジューラとステート遷移
- セマフォを使った排他制御
- Cortex-M3の割り込み処理

CQ出版社

第1章

組み込みシステム・プログラムの
生産性を向上させる

マルチタスクOS基礎理解

桑野 雅彦
Masahiko Kuwano

マルチタスクは難しい？？？

少しマイコンに親しんでくると，マルチタスクという言葉を耳にすることも増えてくるのではないかと思います．

マルチタスクOS，すなわち複数(マルチ)のタスク(プログラム)を並行して動作させるOS(オペレーティング・システム)は，現在ではごく一般的なものになっています．

PCなどで動作するWindowsやMac OS，Linuxなどは言うに及ばず，小規模な組み込み系でもITRON/μITRONなど，さまざまな機器でマルチタスクOSが利用されています．

しかし，マルチタスクと聞くと，どうしても非常に大規模でやっかいなもの，難しいものとまだまだ考えがちです．

● Linuxの場合

マルチタスクOSとして広く普及しているオープン・ソースのLinuxにしても，ソース・コードのファイル一覧だけでも目がくらむほどの分量で，よほど必要に迫られていなければ，とうてい，ソース・コードを読もうという気にはなれないでしょう．LEDを一つ点滅させるにも学ばなくてはならないことがたくさんあり，面倒さという壁が便利さを覆い隠しているような感じがするのではないかと思います．

● TOPPERS/JSPカーネルの場合

小規模な組み込み用として広く普及しているμITRON仕様(注1)に準拠したOSの一つとして，オープン・ソースのTOPPERS/JSPカーネル(http://www.toppers.jp/jsp-kernel.html)というものがありますが，カーネル部分のソース・コードのサイズだけで200Kバイトほどあります．

OSとしてはかなりコンパクトなものですが，それでも，main()からの処理と割り込みルーチンを利用する程度のものに比べれば，規模としてはそれほど小さくは見えません．また，マルチタスクOS独特の用語や考え方，仕組みなどもあって，複雑でとっつきにくく，ハードルの高いものに見えるかもしれません．

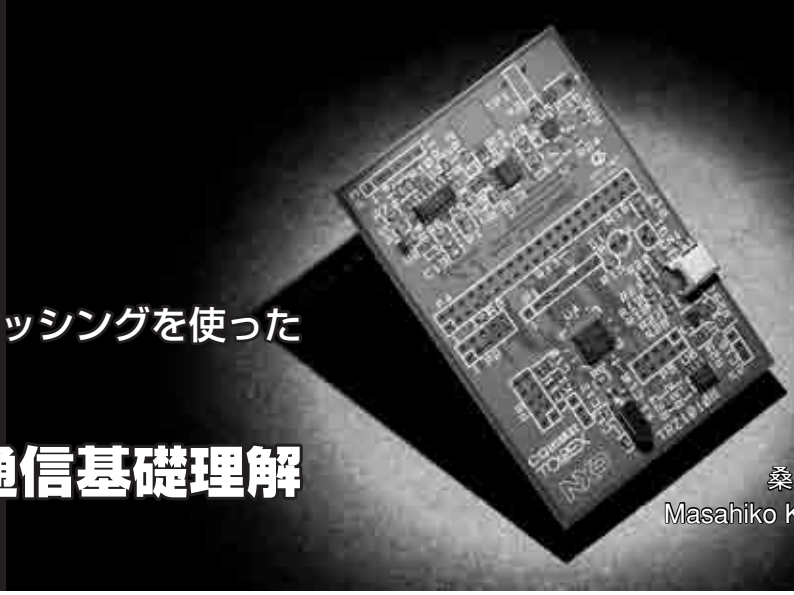
● コンパクトなマルチタスクOSを作る

そこで，この特集では，NXPセミコンダクターズの32ビット・ワンチップ・マイコンLPC1343(CPUコアはCortex-M3)を使ったボードをターゲットとして，μITRONよりもはるかにシンプルなマルチタスクOSを作成してみました．

ソース・コードは，IDEが自動生成するものやライブラリを除くとmain.cの1本だけで，OSとAPI(ライブラリ関数)部分を含めても750行程度という，ごく単純なものです．しかし，マルチタスクの感じをつかむことはできるのではないかと思います．

注1：Linuxなどと異なり，μITRONはあくまでも「仕様書」，すなわちOSとしての機能を規定しているだけで，具体的なソース・コードの作成は利用者に任されている．

第2章



メッセージ・パッシングを使った

タスク間通信基礎理解

桑野 雅彦
Masahiko Kuwano

タスクを分割して連携動作をさせようとしたときに考えなくてはならないのが，タスク間でのデータや要求のやり取りの方法です．このようなやり取りを「タスク間通信」と呼びます．

「通信」とはいっても，実行しているCPU自体は1個ですから，実際には自分で送信して自分で受信しているに過ぎませんが，あたかも別のCPUとのやり取りであるかのように扱うことで見通しがよくなります．

共有メモリ法によるタスク間通信

タスク間が1対1の関係の場合（第1章-図3のタスクCのような場合），固定したデータ領域（一つのファイルにまとめるならば，グローバル変数のようなもの）をタスク同士の間で共有するのが簡単でしょう．

図1(a)の1対1の事例がこれにあたります．シングル・タスクでプログラムを組んだときに，割り込み処理の中でフラグを立てて，タスク側（メイン・ルーチン側）でそれをチェックして動作させてみたり，逆にタスク側でフラグを立てておいて，割り込みが入ったときにそれをチェックして動くといった手法はよく利用されています．

ちょうど，それと同じように，タスク同士の間でもフラグを立てるなどして，やり取りをすればよいという考え方です．

一つのメモリ領域を複数のタスクで共有することから，これを仮に共有メモリ法と名付けておきましょう．

共有メモリ法の欠点

● タスクが増えると面倒になる

共有メモリ法は非常にシンプルでトラブルも少ない方法ですが，規模が大きくなり，いくつものタスク同士でメッセージをやり取りし始めると次第に面倒になってきます．

図1では，1対NやN対Nの図として描かれていますが，複数のタスクとのやり取りが発生すると，送る側，受ける側ともにだんだんと面倒になってくることは容易に想像できるでしょう．

送る側は，毎回送る先との共有メモリがどれであるのかによって処理を変えなくてはなりません．受け取る側はそれぞれのタスクごとに用意された共有メモリ領域をチェックしなくてはなりません．せっかくタスクごとに分割したのに，呼ぶ側のタスクが変更されるたびに呼ばれる側のタスクにも手を入れなくてはならないというのでは面倒です．

● 要求先の処理に時間がかかると待ち時間が増える

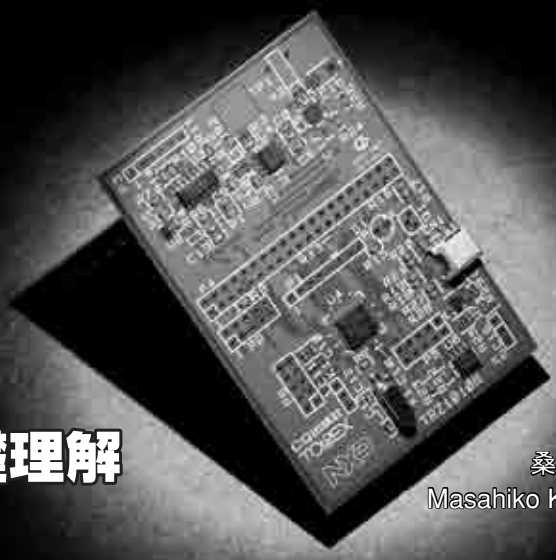
また，シリアル・ポートに定期的にデータを出力する場合のように，要求を受けた側の処理に時間がかかり複数回の送信要求が間欠的に発行される場合を考えてみましょう．

このとき，**図2**(a)のように要求をSIO操作タスク

第3章

割り込み処理とスタックのすり替えがポイント

タスク切り替え基礎理解



桑野 雅彦
Masahiko Kuwano

マルチタスクでは，それぞれのタスクはシングル・タスクと同じように記述されていても，実際には複数のタスクを少しずつ切り替えて処理していきます．これは，イメージとしては理解できても，実際に動作しているのを見るととても不思議な感じがするかもしれません．

特に，これまでシングル・タスクだけでプログラムを書いていた方にとっては，勝手に中断した処理が何事もなかったかのように再開されることが，とても不思議に感じられるかもしれません．

本章では，マルチタスクOSの最もコアとなる部分である，タスク切り替え(タスク・スイッチング)の実現方法の基本的な部分について見ていきます．

マルチタスクの基本はシンプル

難しそうに思えるタスク・スイッチングですが，実はタスク・スイッチングの基本的な考え方自体はとてもシンプルです．ソース・コードにしても，本当に核となる部分を取り出してくれば，これだけ？と思えるほど小さいプログラムに過ぎません．

マルチタスクOSが大きくなっているのは，マルチタスクの核となる部分の外側に，さまざまな用途に対応するための機能がいろいろと追加されているからなのです．

マルチタスクの考え方のヒントは，割り込み処理にあります．

図1のように，割り込みが発生すると今まで実行していたプログラムが中断され，割り込み処理が行われた後，何事もなかったかのように元の処理が再開されます．普段あたりまえのように利用している割り込み処理ですが，このとき裏では何が起きているのでしょうか．

割り込み処理と復帰

割り込みが発生したときの動作は，細かい部分はCPUの種類によっても異なりますが，基本的な動作はほとんど同じです．

図2に，割り込みの発生から終了までの動作を簡単に示してみました．

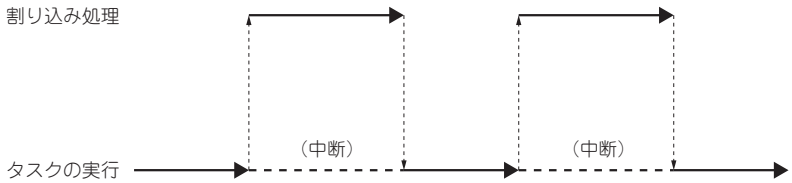


図1　割り込み処理

トランジスタ技術2011年4月号増刊
「2枚入り! 組み合わせ自在! 超小型ARMマイコン基板」を活用

Cortex-M0マイコンを使った超並列マシンの試み

岡田 好一
Yoshikazu Okada

2011年4月に発行された「トランジスタ技術2011年4月号増刊，2枚入り! 組み合わせ自在! 超小型ARMマイコン基板」(編注1)に付属するMB(MCU Board)基板ではお楽しみでしょうか.

本稿では，そのMARYシステムを応用した，多数のCPUを通信で連携させる，超並列(massively parallel)マシンの試みを取り上げます.

最初に，通信系を便利に使うための基本ソフトウェアの増強を行います．新開発になるので，“active MARY”と名付けました.

その次に，active MARYをどのように使うかを学習するために，複数のCPU連携による，ちょっと風変わりなミュージック・シンセサイザを作ります.

最後に，記憶容量の拡大と通信の高速化を狙ったハードウェアの追加計画，“hyper MARY”について説明します.

事例紹介では多数の基板を使うので，追試する場合は，増刊のp.27で紹介されているLPCXpressoによるデバッガを用意されることをお勧めします．本稿の説明は，LPC-LINKを接続してデバッグができる程度の技能を持った方を想定しています.

MARYシステムの概要

増刊号に付属する2枚のMB基板には，縦横4方向に連結するためのコネクタが付いていて，互いに連絡しながら計算処理を分散させることができます．そのため，MCU(Micro Control Unit = マイコン) Arrayにちなんで，全体がMARYシステムと呼ばれています.

MARYシステムは，USBケーブルがあれば，とりあえずWindows PCと接続するだけで開発ができます．OLED(Organic LED：有機LED)表示装置などの，楽しい拡張基板が5種類用意されていて，コントロール用のAPI(Application Program Interface)と例示のためのプログラムが用意されていますから，すぐにマイコンの動作を試すことができます(拡張基板を試すなら，ピン・ヘッダのはんだ付けが必要).

● MB基板の概要

MB基板は，一辺が34mmの小さな基板に，NXPセミコンダクターズのARM Cortex-M0マイコンLPC1114(33ピン・タイプ)を搭載した，試作用基板です(**写真1**)．USBインターフェースが付いていて，Windows PCとの通信に使えます.

裏面には，デバッグ用として3原色のLEDが1個付いています．そして，何よりの特徴が，東西南北の4方向に備えられた，MB基板同士を連結するコネクタです．接続用ケーブルは，マルツパーツ館から市販されています．このケーブルの自作は面倒なので，必要な分だけ購入されることをお勧めします.

拡張基板を接続するためには，ピン・ヘッダをはん

編注1：現在は，増刊(雑誌)から書籍(ISBN978-4-7898-4829-9)に移行し販売中です.

hyper MARY ── ハードウェアの追加による，通信速度と記憶容量の向上

どれくらいのハードウェアを追加したらどこまで良くなるのかが次の段階です．

通信ラインで高速に使えそうなのは，SPIユニットです．一方をマスタにし，他方をスレーブにすれば，クロック周波数12MHzなどで通信できそうです．問題は，MB基板のLPC1114ではSPIが1ユニットしかないので，1対1ではなく，多数接続するときにどうするかです．

メモリの拡張は，OB基板の表示能力を考慮すると，ぜひ実現したいところです．

● メモリの増設

ところで，トランジスタ技術2011年3月号のARMマイコン特集で，主記憶を増やしたいときにSPI接続のメモリが使えるという文章がありました(p.78)．調べてみると，SPI接続で256Kビットのシリアル RAM 23K256(マイクロチップ・テクノロジー)が簡単に手に入りそうです．手ごろな価格なので，MB基板一つ一つに拡張基板を設け，分散してRAMを増やすことを考えました．通信は12MHzのSPIが使えるので，初期の8ビット・パソコン程度の能力なら簡単に実現できそうです．

通信回線とシリアルSRAMを切り替えながら使おうと回路を考えていたところ，シリアルSRAMをCPUの中間に置いて，通信路として使えるのではないかと思いつきました．隣のCPUと通信したところで，結局はデータはどこかのメモリに入るはずです．

● hyper MARYの構成

最初に思いついた回路は微妙に非対称だったのですが，次に思いついたのは，ブロック図としては極めてシンプルです(**図8**)．

一つのCPU当たりメモリが二つになってしまいますが，シリアルSRAMは小型で経済的なので，思いきって奮発することにしました．1Mビットのシリアル EEPROM 25AA1024(マイクロチップ・テクノロジー)もすぐに手に入ります．接続は簡単なので，拡張基板上に用意しておくことにしました(**写真6**，**写真7**)．

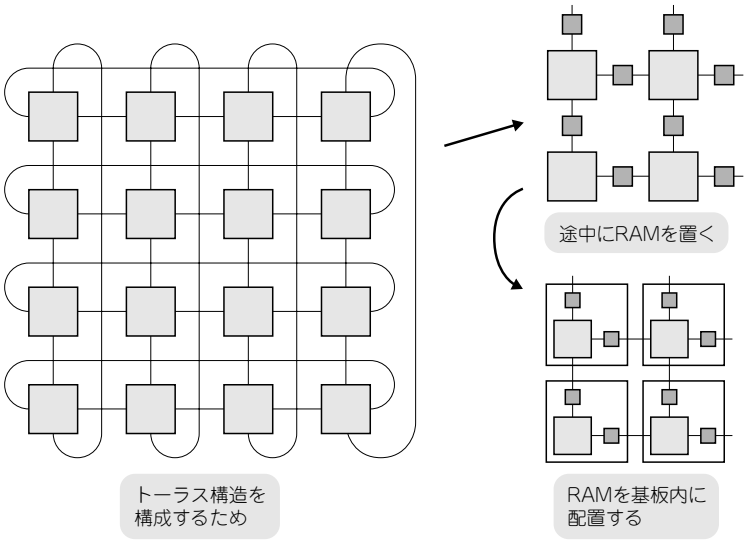


図8　hyper MARYの構成図

CQ出版社

見本

ISBN978-4-7898-3595-4

C3055 ¥2600E

CQ出版社

定価：本体2,600円（税別）

9784789835954

1923055026001

ARMマイコン
No.2