オリジナルOS「MicrOS」の設計と実装

第2回 MicrOSのシステム・コール

田口 信夫

本誌2007年5月号に付属したV850マイコン基板で動作するオリジナルOS「MicrOS（マイクロス）」が登場した．今回はMicrOSのシステム・コールについて解説を行う．なお，本OSのソース・コードは本誌のWebサイト（http://www.cqpub.co.jp/interface/）からダウンロードできる．（編集部）

今回は，MicrOSを制御OSとして使用する組み込みシステムを作成するときに，アプリケーション・ソフトウェアの側で利用できる機能について説明します．

MicrOSは，OSとしては定義している言葉が少なく，処理構造も単純です．ソース・コードを公開しているので，MicrOSをブラックボックスのOSとして扱わず，できるだけ処理している内容を理解して使ってください．OSに対する機能追加やサービス方法の変更といったアプリケーション固有のカスタマイズも比較的簡単です．

カスタマイズを行うには，MicrOSのソースを直接変更します．このような手段を取ることによって，より効率的なアプリケーション・システムの制御が行えます．MicrOSはアプリケーション・システムを効率的に制御できるように開発したOSです．このような目的があるからこそ処理構造を単純にし，できるだけ少ない概念でマルチプログラミング環境を提供するように構成しています．

MicrOSの制御の概要と主な用語を**表1**に示します．

1．MicrOSの導入

● アプリケーションごとに修正が必要なファイル

MicrOSは構造が単純で小さなOSなので，機能そのものの変更も可能です．処理内容を変更せずに利用する場合の導入方法を説明します．

MicrOSをアプリケーション・システムに組み込む前に行わなければならない作業があります．それは，次のソース・コードまたは関数の修正作業です．

- MicrOS.h，MicrOSasm.inc
- aplMain関数
- タイマ処理関数
- MicrOScmd.c

aplMain関数とタイマ処理関数については，そのテンプレートがMicrOS.cの先頭にあります．これをアプリケーションのソースにコピーして使用します．最小限MicrOS.hとMicrOSasm.hのカスタマイズが必要になります．

aplMain関数はMicrOSがマルチプログラミング制御に入る前に1回だけ呼ばれる関数です．タスク登録はaplMain関数の中で__task関数を使って行います．この関数以外の部分で__task関数をコールしてはいけません（MicrOSはこのチェックを行わない．この関数以外の部分でタスク関数をコールしたとき，メモリ・アロケーションやタスクの動作に異常が発生し，それがシステム全体に伝播して，アプリケーション・システムは破滅的な動作を行う）．

● タイマをこまめに停止して動作する

タイマ処理関数はハードウェアの違いを吸収するための関数です．MicrOSのインターバル・タイマの制御は，省電力を図るためにできるだけタイマ割り込みを少なくしています．また，インターバル・タイマを使わないときはタイマをストップさせるというのが基本です．このような制御を行うためには，ハードウェアによって途中の経過時間を読み取る機能がサポートされていなければなりません．このような機能を持たないハードウェアであった場合には，定周期割り込みを発生させ，その割り込みをカウントする方法をとります．このときも，インターバル・タイマを使用しないならハードウェア・タイマをストップさせます．MicrOSのタイマ制御にはこの二つの処理がテンプレートとして組み込まれています．V850のタイマには，途中のタイマ・カウントを読み取る機能を持ったインターバル・タイマとカウントを読み取れないタイマがあり，2種類のタイマによる両方の機能がテンプレートとして組み込まれています．このどちらを使用するかは，MicrOS.hの中に

KEYWORD ── MicrOS，リアルタイムOS，マルチプログラミング，マルチタスク，システム・コール，コールバック関数

表1　MicrOSの制御の概要と主な用語

マルチプログラミング方式			マルチタスク制御．タスク単位に制御ブロックTCB（タスク・コントロール・ブロック・スタック領域内包）
制御するタスク数			制限なし，システム・コントロール・ブロックのサイズに依存する
タスクの種類			1種類．ユーザ・モードによる走行
タスクの登録			アプリケーション・システム・イニシャライズ（aplMain：マルチタスク制御以前）でのみ．__task関数で登録
タスクのID			TCBのアドレス
タスクの優先度レベル数			最大256レベル（デフォルト：16レベル・コンパイル時）
同一優先度レベル			ラウンドロビン/FCFS選択可能（コンパイル時）
プライオリティの変更			自タスクによるシステム・コールのみ．__chgpri関数
タスクの状態			レディ・ラン・ウェイト（終了はなし）．初期状態（タスク登録時）はレディ状態
タスク・スイッチの起因			内部割り込み（ウェイト関数，アクティブ関数），外部割り込み． 実行中のタスクがあるときは優先度（プライオリティ）の高いタスクがウェイトを解かれたときのみ
タスク・スイッチの禁止・解除			システム・コール，ウェイト系システム・コールの発行で強制解除
割り込み制御			——
	内部割り込み		Trap　0x00のみ．タスクがウェイトするとき，および自分より優先度の高いタスクをアクティブにしたときのみ使用している
		処理用スタック	TCBを使用（ワーキング・レジスタはセーブしない）
	外部割り込み		インターバル・タイマ，UARTのみ組み込まれている（いずれもアプリケーションによる追加・修正要）
		その他の外部割り込み	アプリケーションによる組み込み（V850-MicrOSはCA850の機能を利用）
		多重割り込み制御	可（コンパイル時）．ただし，V850-MicrOSは不可
		処理用スタック	システム・コントロール・ブロック内．多重割り込み制御使用時はアプリケーションによる多重度を考慮した調整が必要．ただし，V850-MicrOSはタスクのスタック領域を利用する．サイズは計算不可能
主な構造体			システム・コールで使われる構造体．サービスによって少しずつ構造が変わる
	TCB		タスク・コントロール・ブロック．マルチタスク制御のための構造体．MicrOSの諸サービスの客体としても機能する
	SCB		サービス・コントロール・ブロック．サービスの主体．SCBのIDはSCBのアドレス．以下は代表的なSCB
		_MALCCB	メモリ・アロケーション．MicrOS内のみ
		_MBLKCB	メモリ・ブロック
		_MBXCB	メールボックス
		_SEMCB	セマフォ
		_EVTCB	イベント・フラグ
		_TMRCB	タイマ制御．MicrOS内のみ
	RQB		リクエスト・ブロック．サービスの客体．RQBのIDはRQBのアドレス．TCBもRQBの一種．以下は代表的なRQB
		_MBXRQB	メールボックス
		_TMRRQB	タイマ制御
システム・コントロール・ブロック			MicrOSが所有する制御ブロック．以下に示すデータから構成される ●実行中タスクのTCBアドレス ●_TMRCB ●_MALCCB ●タスク・スイッチのフラグ ●タスク・スイッチ禁止・解除のカウンタ ●優先度制御テーブル（レディ・タスク・プール） ●外部割り込み処理・アイドル時のスタック ●TCB生成エリア ●メモリ・アロケーション・エリア
システム・コール			システム・コールは次の3種類に集約できる．ウェイトが必要なシステム・コールはウェイト系とアクティブ系で対になっており，それぞれ対応する関数がある．MicrOSの中ではSCBとRQB/TCBの組み合わせで処理される
	ウェイト系		__waitで代表されるシステム・コール．システム・コールを発したタスクはウェイト状態になる可能性がある．ウェイトするときは__waitを使う．タスク以外での使用は不可．
	アクティブ系		__activeで代表されるシステム・コール．ほかのタスクをアクティブにしてタスク・スイッチを行うこともある．タスク・スイッチが行われるときはレディ状態のまま
	その他		ウェイト系，アクティブ系以外のシステム・コール．タスク・スイッチが行われることなくコールされたタスクに戻る

コンパイル時とあるのはコンパイルによって変更可能な機能．