

シングルチップマイコンソフトの再利用指向設計の一考察

4L-4

簡易スケジューラの提案

西村 朋子, 平尾 繁晴, 三原 幸博
株式会社 東芝 システム・ソフトウェア技術研究所

1. はじめに

最近の半導体技術の急速な進歩により、マイクロコンピュータ(以下、マイコン)組み込み製品が急増している。それに伴い、マイコンソフトの需要も急増しており、ソフトウェア(以下、ソフト)の再利用などによる生産性の向上が必要となってきた¹⁾。本論では、シングルチップマイコンソフトを再利用しやすい形で設計する手法について検討する。

2. シングルチップマイコンソフト開発の問題点

再利用を効率良く行うには、設計段階から再利用のしやすい構造にする必要がある。我々は、一般のアプリケーションソフト開発で、モジュール設計記述法TFPを用いて再利用指向の設計を行っている²⁾。しかし、シングルチップマイコンソフトの分野では、現状では、そういう設計方法はあまりとられていない。それは、次のようなマイコンソフト特有の問題が原因となっている。

- ・コスト抑制のため、ROM容量(プログラムサイズ)が制限されるので、構造化などによる冗長性が許されない。
- さらに、シングルチップの場合は、次のような問題点がある。
- ・汎用計算機などに比べ製品自体の価格が安いので、さらに厳しいコスト低減が要求される。(ROM容量, RAM容量, 処理時間, ポート数, 開発工数など)
- (例) 東芝製4bitマイコンTMP47C200Aは,
ROM=2048[*8bit] (1000~2000命令)
RAM= 128[*4bit]
- ・ソフトとハードウェア環境が相互依存しているため、製品の機種変更にもなって多様化していくハードウェア環境のもとでは再利用しにくい。

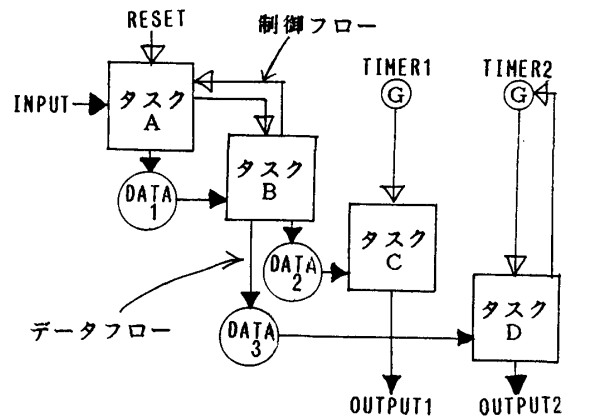


図1 割り込みで直接タスクを起動する例

再利用を考慮しないで設計を行うと、機能を1つ追加(または修正)するだけで、プログラム全体を書き直さなければならない場合もあり、逆にコスト増加を招く恐れがある。

ここでは、再利用しやすい設計方法として、タスクの並行処理の実現方法としてスケジューラを取り上げ、シングルチップマイコンソフトにスケジューラを導入する場合の問題点、特に上記のコスト問題を解決するために「簡易スケジューラ」を提案する。さらに、電話のマイコンソフトを例に、簡易スケジューラを用いた設計方法とその実現可能性を検討する。

3. タスク分割とスケジューラ

3.1 タスク分割

まず、ソフト開発の基本設計段階で、いくつかの並行処理単位(タスクと呼ぶ)に分割する。タスク相互の関係はデータの共有のみとする。これにより、大規模なソフトを論理的に扱いやすい大きさにブレイクダウンできる。

3.2 タスク並行処理の実現

一つのCPUで複数のタスクを並行処理するためには、タスクを切り替えて実行しなければならない。シングルチップマイコンソフトでは、マイコンに実装されているハードウェア割り込み(外部割り込み・タイマ割り込みなど)で直接タスクを起動する手法が、しばしばとられる。

しかし、この場合、マイコンの機種によってハードウェア割り込みの種類・個数・優先度・(タイマ割り込みの)周期などが違うので、マイコンの機種を変更したときの再利用性が低下する。また、タスク内部に「処理」と「タスク切り替え」が入り混じってしまうので、構造が煩雑になる。

そこで、「スケジューラ」を取り入れ、タスク切り替えの機能を集中させて、一元管理することを考える。また、割り込み

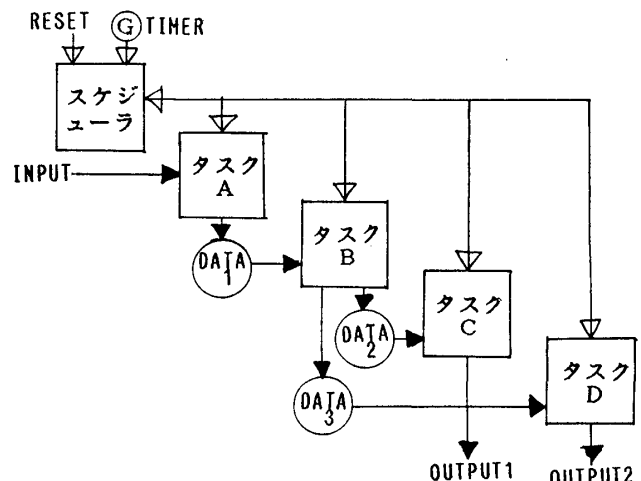


図2 簡易スケジューラを導入した例

処理もスケジューラが行い、スケジューラがタスクを起動するものとする。

3.3 スケジューラ導入の利点

ソフト開発にスケジューラを導入した場合、次のような利点がある。

- ・タスクはスケジューラとの制御関係だけを考えればよく、タスク同士の直接の制御依存はないので、設計が単純化される。
 - ・タスク内部の設計が他のタスクに影響を与えないようにタスクを切分けることが可能となり、タスク単位で再利用できる。
- 電話マイコンソフトの設計について、ハードウェア割り込みを直接用いた例と、スケジューラを用いて設計した例を図1、図2に示す。

4. 簡易スケジューラ

4.1 簡易スケジューラの提案

シングルチップのマイコンソフトの場合は、ROM・RAM・処理時間などの制約が厳しいために、現状ではスケジューラ的なものはほとんど導入されていない。

そこで、適用する問題に必要な機能を全て削除した「簡易スケジューラ」を提案する。問題に即して機能を縮小することにより、スケジューラのROM・RAM・処理時間などのオーバーヘッドをできるだけ少なくし、スケジューラの実現を図る。

4.2 スケジューラの機能

スケジューラの機能として、以下に例を挙げる。

1. タスク数……固定/可変
2. 優先度……無し/有り (固定/可変)
3. タスク起動の方法
 - ……時間 (固定/可変) / 外部割り込み
 - / イベント発生 / 他タスクからの起動要求
 - または、これらの混合 (固定/可変)
4. クロック・タイマ管理
5. リセット処理, 外部割り込み処理

簡易スケジューラを設計する場合、これらの中から取捨選択し、また、新たに必要な機能を追加するなどして、スケジューラのオーバーヘッドを最小限にするように構成する。

例として、図2の電話マイコンソフトの基本設計方法を、以下に述べる。

4.3 簡易スケジューラを用いた基本設計の例

まず、ソフト全体をタスクに分割し、タスク間データを定義する。タスクの数をできるだけ少なくしておいた方が、タスク切り替えの回数も減り、スタック領域の分割も1タスク分の割

タスク	起動方法	起動周期	その他
A	時間	8	—
B	特になし	—	—
C	時間	16~128	最優先
D	時間	32	—

表1 各タスクに関するデータ

当が多くなる。この例では、入出力データの種類とその時間的制約から、A, B, C, Dの4つに分割する(表1)。

次に、以下の手順で簡易スケジューラを設計する。

- [STEP1] タスク数 (=4) は固定とする。
- [STEP2] タスクCは出力波形の厳格な規定があるため、最優先とする。タスクBはDATA1によるイベント起動でもよいが、スケジューラの単純化のために、時間起動で優先度を最低とする。タスクAとDでは、Aの方が起動周期が短いので、Aを優先させる。従って、優先度=C, A, D, B (固定)
- [STEP3] 起動方法は時間のみとする。
タスクCの起動周期が可変なので、可変とする。各タスクは1つずつカウンタを持ち、次に起動されるまでの時間を指定できる。
- [STEP4] スケジューラは基本的には、タイマ割り込みで動作するものとする。割り込み周期は、短いとタスクの処理よりスケジューラの処理の方が多くなってしまい、タスクの起動周期より長いとタスクの応答性が悪くなる。従って、マイコンのタイマと各タスクの起動周期から、できるだけ長くなるように決定する。(ここでは8とする)
- [STEP5] スケジューラの機能を定義する。
(1) タスクがスリープ要求を出した時、スリープさせる。
(2) 一定時間毎にタスクのカウンタをカウントする。要求された時間分のカウントが終了したら、起動待ちとする。
(3) (1), (2)の後、起動待ち状態のタスクのうち、優先度が最高のものを起動する。

[STEP6] リセット処理・外部割り込み処理を定義する。

4.4 簡易スケジューラの実現可能性の検討

上記の設計によるメモリ消費量の増加を試算したところ、次のような結果が得られた。

ROM……1.07倍

RAM……1.07倍

また、簡易スケジューラの処理時間は、タイマ割り込み周期の15.8%であった。

この結果より、この程度のROM・RAM・処理時間の余裕があれば、この程度の簡易スケジューラは実現可能であることが実証された。

5. おわりに

シングルチップマイコンソフトに簡易スケジューラを導入することにより、タスク単位での再利用・修正・追加が容易になり、開発工数が減少すると考えられる。

ただし、簡易スケジューラは問題に依存したものになってしまうため、適用する問題に即して簡易スケジューラを簡単に作り直せるか、その場合タスクの再利用性を保てるかといったことが、今後の課題である。

[参考文献]

- 1) 山岡他; "マイコンソフトウェア部品化・再利用指向のドキュメンテーション", 情報処理学会第35回全国大会, 1987, 9
- 2) 大筆他; "IMAPシステム(1)~(10)", 情報処理学会第31回全国大会, 1985, 9