新連載

# ColdFireマイコン基板搭載 Cインタプリタ SilentC活用テクニック

関連データ

## 第1回 Cコンパイラによるユーザ・ドライバの作成と呼び出し方法

中本 伸一

本誌2008年9月号付属ColdFireマイコン基板には，CインタプリタSilentCが書き込み済みのため，簡単にプログラムを実行できる．しかしSilentCには，割り込みなど細かな処理ができなかったり，インタプリタであるがゆえにプログラムの実行速度が遅いといったデメリットがある．本連載では，これらSilentCの欠点を補うための各種テクニックについて解説する．（編集部）

本誌2008年9月号付属ColdFireマイコン基板に搭載されているCインタプリタSilentCは，ユーザがネットワーク・プログラムを簡単に作成できるという特徴を持っています．しかし，ColdFireに搭載されている各種コントローラを細かく制御するには，どうしてもレジスタに直接アクセスしたくなります．

今回はSilentCに，自分で作成したユーザ・ドライバを組み込む方法を解説します．「SilentCにこんな機能があれば…」，「プログラムの実行速度がもう少し上がれば…」と感じている読者は，ぜひ参考にしていただきたいと思います．

## 1．SilentCのユーザ・ドライバ仕様

### ● SilentCのメリットとデメリット

SilentCは，TCP/IPスタックを利用できるC言語ライクなインタプリタ言語です．telnetあるいはシリアル経由でSilentCを操作してユーザ・プログラムを簡単に開発できます．従来の開発環境では，ソースを修正した後にコンパイルとリンクを行って実行イメージを作成し，フラッシュROM上の古いプログラムを消去した後に，新たにプログラムし直して，やっと実行を開始するという手順が必要でした．

しかしSilentCでは，コンソールからプログラムを書き換えてrunコマンドで即座に実行できます．このように，SilentCは簡単な操作が大きなメリットですが，その便利さの影で実行速度が遅くなるというデメリットもあります．

通常，ネットワークを利用するアプリケーションを構築する際には，実行速度はあまり問題になりません．ネットワークのアクセスにはリアルタイム性を必要としない場合が多く，特にほかのサーバと交信するアプリケーションはサーバの返答を待ってから次に進まなければならないので，クライアント側の処理がいくら早くてもあまりメリットが得られないためです．

しかし，MCF52233に内蔵されている各種内蔵デバイスをコントロールする際には，ある程度高速なスピードが要求されます．また，内蔵デバイスをコントロールするためには割り込み処理が不可欠です．残念ながらSilentCでは，割り込み処理のハンドラを記述できません．つまりSilentCだけでは，内蔵デバイスを有効に活用するのはかなり難しいのです．

### ● ユーザ・ドライバとは

こうした弱点を解決するために，SilentCではユーザが自分で作成したネイティブなプログラムを呼び出すための仕組みが最初から実装されています．この拡張部分のプログラムを「ユーザ・ドライバ」と呼びます．これをSilentCから呼び出すには，`UserDriver()`関数を使用します．

ユーザ・ドライバを開発してファイルとして置いておけば，リセット直後にSilentCがそれを読み込んでフラッシュROMにプログラムします．一度，フラッシュROMに書き込んでおけば，それ以降はSilentCのライブラリ関数としてユーザ・ドライバをいつでも利用できます．

こうした仕組みを利用すれば，SilentCの手軽さとユーザ・ドライバの高速性と直接性を組み合わせたアプリケーションを自由に構築できます．ユーザ・ドライバを作成できるようになると，ColdFire基板の実用性は格段にアップするでしょう．

### ● ユーザ・ドライバのサイズ

ユーザ・ドライバは`UserDriver.bin`というファイル名を持つ，サイズが4K（4096）バイトのバイナリ・ファイルです．SilentCが管理するファイル領域にこのファイル名が存在していると，SilentCは起動時にこのファイルの内容をそのままアドレス`0x00013000`～`0x00013FFF`までの


KEYWORD ―― ColdFire，MCF52233，Cインタプリタ，SilentC，ユーザ・ドライバ，ポインタ，スタック

領域にプログラムします．ファイル・サイズが4Kバイトよりも長くても短くても，上記の空間に収まるようにプログラムします．

どうしてもこのサイズに収まらないようなドライバを実行したい場合は，ドライバ本体をアドレス0x00024800～0x0002FFFFまでのフリー領域に配置した上で，その領域へジャンプするだけの処理を記述すればよいでしょう．これなら，4Kバイト以内に収めることができます．

### ● ユーザ・ドライバの構造

SilentCのユーザ・ドライバは，**図1**のような構造になっています．まず，ユーザ・ドライバの先頭にはジャンプ・テーブルが配置されます．このジャンプ・テーブルにはユーザの呼び出したいプログラムの絶対アドレスを4バイト単位で格納しておきます．次に，ユーザの作成したプログラムの本体が格納され，SilentCの内部ライブラリを呼び出すためのジャンプ・テーブルがリンクされます．

例えば，SilentCからUserDriver(0,0);のようなUserDriver関数を呼び出すと，ジャンプ・テーブルの最初のアドレスを参照してジャンプします．つまり，アドレス0x00013000から格納されている4バイトをアドレスとしてジャンプします．その際，スタックの先頭にはSilentC内の戻りアドレスが積まれています．ユーザ・ドライバの最後でアセンブラのRTS命令を実行すれば，SilentCへ制御が戻り，プログラムの実行が継続します(**図2**)．

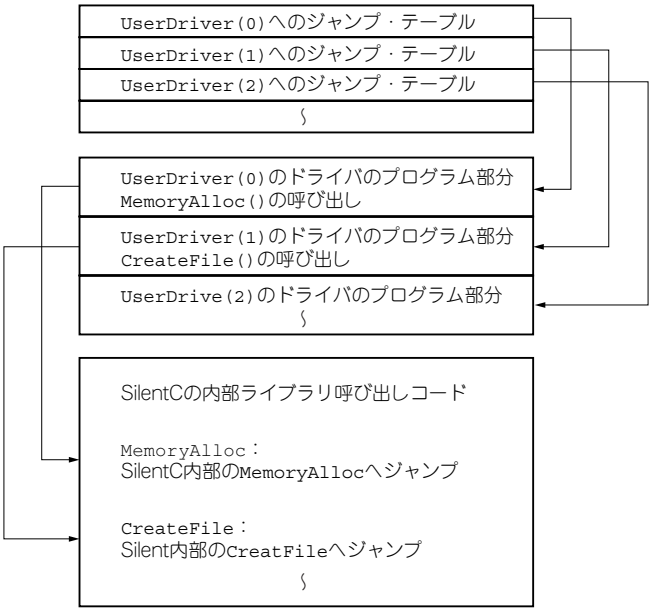


**図1　SilentCのユーザ・ドライバの構造**

UserDriver関数の2番目の引き数は32ビットの整数で，その値はユーザ・プログラムを呼び出す際にスタック上にセットされています．また，ユーザ・ドライバから戻る際，汎用レジスタD0にセットした値はUserDriver関数の戻り値としてSilentCに戻して利用できます．

### ● ユーザ・ドライバの動作

一般的なユーザ・ドライバの動作の流れを**図3**に示します．まず，制御対象デバイスの初期化やメモリの確保，割り込みベクタの設定などの初期化処理を行います．そして，SilentC上のプログラムと連携しながらそのデバイスに対して必要な制御をします．もちろん，割り込み処理もバックグラウンドで動作可能です．プログラムの実行を終了するときには，最後にクリーンアップを呼び出してデバイスを止めたり，割り込みベクタを元に戻したり，メモリを解放するという流れになります．

### ● ユーザ・ドライバが使用できるRAM空間

ユーザ・ドライバ内では，静的な変数としてRAM領域を使用する必要があります．ユーザ・ドライバが自由に利用できるRAM領域は，アドレス0x20007F80～0x20007FFFまでの128バイトです(RAM領域の一番後ろの128バイト)．

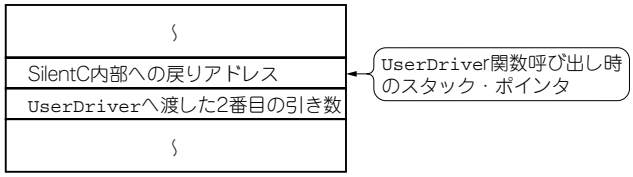


**図2　ユーザ・ドライバ呼び出し時のスタックの状態**

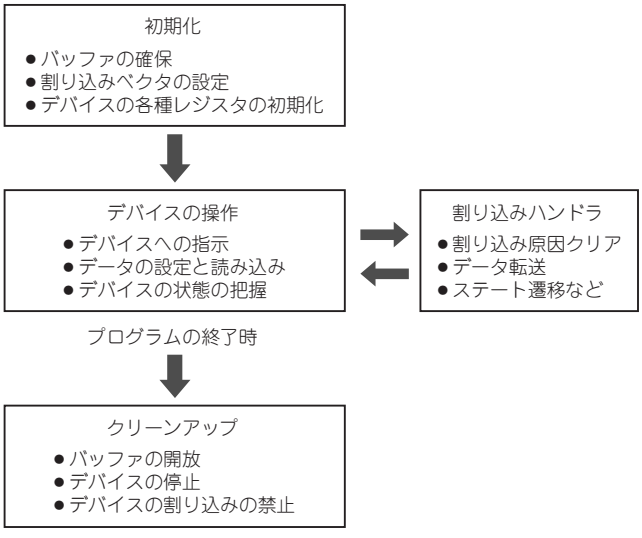


**図3　一般的なユーザ・ドライバの動作の流れ**