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2回 ネットワーク経由でWAVファイルを送信して音楽を再生しよう

中本 伸一

今回は，より実用的なユーザ・ドライバの作成方法を説明する．すなわち，パソコンなどからネットワーク経由でWAVファイルを転送して，それをColdFireマイコン基板で受信し，さらにPWMコントローラを使ってD-A変換を行い，音楽を再生するシステムを構築する．（編集部）

実用的なユーザ・ドライバの製作例

● 音声再生システムの製作

より実用的なドライバの例として，CPUに内蔵されているデバイスを操作するユーザ・ドライバを作成してみましょう．題材として，PWM（Pulse Width Modulation）モジュールとDTIM（DMA Timer Module）の二つを連携させて，基板から音声を鳴らしてみます．

PWMモジュールは，一定間隔でユーザが指定した幅の信号を出力するモジュールで，D-A変換器として利用できます．DTIMモジュールは設定した間隔で割り込みをかけるモジュールで，一定間隔でデータを更新する場合などに用いられます．

今回は，8kHzで動作するDTIMモジュールでタイミングをとりながら29.3kHz周期でパルス信号を出力するPWMモジュールに音声データを書き込むという動作を行います．

ここでは，SilentCにはネットワークのインターフェース部分を，ユーザ・ドライバにはSilentCで受け取ったデータをD-A出力を利用して音声として出力する部分を担当させます．パソコンからColdFireマイコン基板に対して音声データをUDPで送り出します．

パケットを受け取ったSilentCは，直ちに受け取ったデータをユーザ・ドライバに渡します．ユーザ・ドライバは，渡されたデータを内部のバッファに蓄えておき，DTIMモジュールからの割り込みに同期して音声データをPWMモジュールに渡します．PWMモジュールは，渡された音声データをD-A変換してアナログ信号として出力するという流れになります．

音声データのフォーマットは，圧縮しないWAVフォーマットを使います．サンプリング周波数8kHz，量子化数16ビット・モノラルの場合，データの転送量は1秒当たり16Kバイトになります．通信速度10MbpsのEthernetで接続しているのであれば，UDPで1秒当たり300Kバイト程度は問題なく転送が可能なので，今回は生のWAVデータを転送することにします．

● PWMモジュールからアナログ信号を取り出すハードウェア

PWMモジュールからの出力をアナログ信号に変換するには，簡単な外付け回路が必要です．**図1**の回路をブレッド・ボードなどを利用してColdFire基板と接続します．1kΩと0.004 μFのコンデンサだけという極めてシンプルな回路ですが，これだけでカットオフ周波数が約4kHzのローパス・フィルタとして動作します．後述するように，

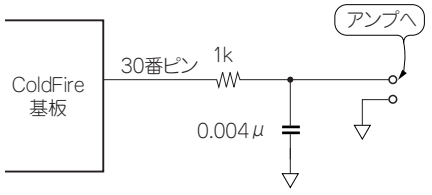

図1　PWM出力用フィルタ回路

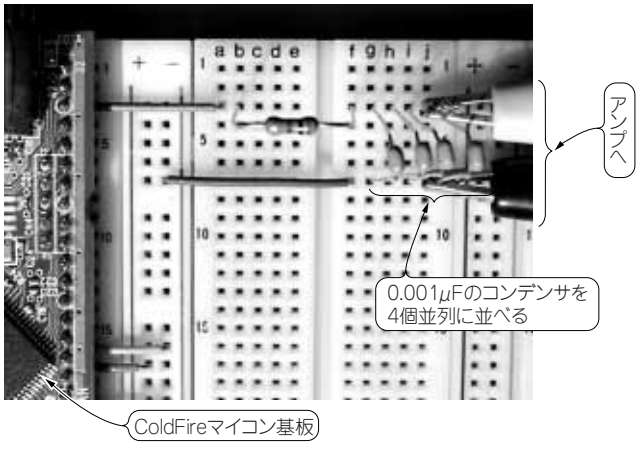

写真1　図1の回路をブレッド・ボードで試作

KEYWORD —— SilentC，ユーザ・ドライバ，PWM，D-A変換，ローパス・フィルタ，WAVファイル

リスト1　UserDriver.s

```
    .global   start
    .extern   _Init
    .extern   _Size
    .extern   _Voice
    .extern   _Clean

    .text

start:
    .long     _Init
    .long     _Size
    .long     _Voice
    .long     _Clean
```

SilentCとリンクする関数

PWMの繰り返し周波数は29.3kHzなので，ローパス・フィルタを通じて音声帯域のみを取り出すためには不可欠な回路です．0.004μFのコンデンサは入手しにくいので，パスコンとして最も広く利用されている0.001μFのコンデンサ（104と書かれている）を，**写真1**のように四つ並列に接続してください．ブレッド・ボードなら挿すだけなので，とても簡単に作れます．

次に，スピーカ付きのオーディオ・アンプを用意してローパス・フィルタの出力に接続しておきます．これでハードウェアの準備は完了です．

● 音声出力ユーザ・ドライバの解説

本誌のダウンロード・ページ（http://www.cqpub.co.jp/interface/）からダウンロードしたアーカイブを解凍すると生成されるフォルダにSoundDriverというCodeWarriorプロジェクトがあります．この中のSoundDriver.mcpをダブルクリックしてCodeWarriorプロジェクトを開きます．ソース・ファイルは，第1回（2008年12月号，pp.170-176）でも解説したUserDriver.sとMain.c，MoonLibrary.sの三つです．

UserDriver.s（**リスト1**）を見るとわかりますが，SilentCとリンクするのはInitとSize，Voice，Cleanの四つの関数です．Initでバッファ確保を含む初期化を行い，SizeでUDPで受け取ったデータ・サイズを設定し，Voiceで実際のデータをドライバに渡します．Cleanは，確保したバッファを開放してデバイスの割り込みを止めるという後始末を担当します．

● ユーザ・ドライバ本体の動作

次に，メインとなるユーザ・ドライバ本体Main.cを**リスト2**に示します．ソースには全行コメントを付けているので読みやすいと思います．まず，Initではから受け取った音声データをキャッシュしておくバッファ領域を確保しています．

リスト3　SilentC側プログラムVmain

```
main(){int i;char *b,soc;
soc=CreateSocket(0);Bind(soc,5000,1);UserDriver(0,2400);
for(;;){i=RecvFrom(soc,1);if(i<0)continue;UserDriver(1,i);
 b=GetReceiveBuffer(soc,1);UserDriver(2,b);MemoryFree(b);
if(i<800)break;}UserDriver(3,0);}
```

バッファは2400バイトです．ネットワークから送られるデータはリアルタイム性に乏しいので，受け取ったデータを一度キャッシュしておき，DTIM（タイマ）割り込みで一定間隔で取り出してPWMモジュールに転送します．今回の事例では，音声のサンプリング周波数を8kHzに設定しているので，2400バイトだと150ms分の音声データをキャッシュできます．ColdFireマイコン基板に送られるUDPデータが，このキャッシュ時間以上途切れると音声も途切れることになります．

次に，PWMモジュールを設定します．PWMモジュールに与えられるクロックが30MHzであり，今回の事例では10ビットの解像度でD-A変換出力をするので，PWMの解像度は1024ビットに設定します．PWMの繰り返し周期は，30MHz/1024=29.3kHzということになります．PWMモジュールは，この設定で29.3kHzの周期で10ビットのPWMを繰り返します．

DTIMモジュールには，60MHzの内部クロックが与えられています．そのためカウンタ値を7500に設定すると，8kHzで割り込みがかかります．割り込みベクタをセットして割り込みレベルと優先順位を設定し，割り込みマスク・ビットをクリアしてDTIMモジュールの割り込み発生フラグを立てると，設定は終了です．

SilentCがUDPでデータを受け取ると，Voiceを呼び出してWAVデータをリング・バッファに格納します．DTIMモジュールからの割り込みで呼び出されるTimerHandler関数がリング・バッファからデータを取り出してPWMモジュールにセットする動作を繰り返します．

SilentC側のプログラムは極めてシンプルです．アーカイブ内のフォルダSilentCの中のVmain（**リスト3**）を参照してください．この短いプログラムはUDPで5000番ポートに届くデータを待ち続け，データが到着したらユーザ・ドライバに渡すという動作をしています．800バイト以下のデータが送られてきた場合には，ループを抜けてプログラムを終了します．