おなじみC言語でI/O制御

桑野 雅彦

シェルを使った入出力はスクリプト言語なので複雑な処理を行うのは難しく，遅延も大きくなります．

ここでは，おなじみのC言語を使ってGPIOを効率良く制御します．

GPIO制御…2種類の方法でトライする

次の二つの方法を試します（第4章の図1を参照）．

● 方法1：GPIOデバイス・ドライバを使う
…手軽で安全だけどアクセス速度は遅い

Raspberry Pi用Linux「Raspbian」にあらかじめ用意されているGPIOアクセス用のデバイス・ドライバを利用します．アプリケーションからGPIOデバイスをオープンして，リード/ライトするという形でアクセスします．

アクセス速度の面ではレジスタを直接操作する（たたく）方法には劣りますが，レジスタのアドレスなどを気にしなくてもよく，ある程度のチェックはドライバで行えます．アドレスを間違えてほかのレジスタやメモリの内容を書き換えてしまうなどといったトラブルが起こりにくいことから，安全性は高くなります．

ハードウェアの違いはドライバで吸収できるため，ハードウェアの置き換えや仮想化にも対応しやすい方法です．

● 方法2：レジスタを直接たたく
…手間がかかるがアクセス速度は速い

GPIOの制御レジスタを直接操作するというものです．アクセス速度の面では一番優れた方法ですし，複数ポートへの同時データ設定できるなど，自由度は一番でしょう．MMU（メモリ・マネジメント・ユニット）を持たない8ビットや16ビット・クラスのワンチップ・マイコンを利用してきた方にとっても一番馴染みのある方法だと思います．

ただし，RaspbianはLinuxベースであり，仮想記憶機構が動いている中で動いているアプリケーション・プログラムからのアクセスになります．アプリケーション・プログラムから見える仮想メモリ空間から，実際の物理メモリ空間（ハードウェア的なアドレス空間）へのマッピングが必要になるなど，ひと手間かかります．

方法1：デバイス・ドライバを使う

GPIOデバイス・ドライバを使うときは，基本的にはシェルからアクセスしたときと同じ考え方で，デバイス名も同じです．違うのは，シェルからのときはファイルのオープン/クローズを意識しないで済みましたが，C言語からGPIOデバイス・ドライバを使うときはGPIOデバイス名を使ってオープン/クローズするなどの処理が必要になることです．

■ プログラムを作成

サンプル・プログラムを**リスト1**（gpio_dev.c）に示します．主要な部分を順に紹介します．

● ポートの使用開始を指示する…export

シェルのときと同じように，まずexportして該当するポートに対する入出力動作を行うことを**リスト1**①のようにドライバに知らせます．そして使用するポートを通知して，クローズします．

シェル・スクリプトでアクセスしたときは使い終わるまでexportしておくので，ついopenしたままにしたくなります．しかし，ドライバ側ではopen/closeではなく，exportデバイスに対するポート番号指定が行われたことで使用可能にしているので，設定が終わったらファイルそのものはクローズしてしまいます．

C言語のアプリケーションの中からアクセスしている場合，unexportのほうは省略してもエラーにはなりませんが，リスト中ではunexportするのが作法であろうと考え

KEYWORD── C言語，GPIO，メモリ空間，malloc，mmap