短期集中連載

CPUに依存しない開発スタイルを身に付ける

ダウンロード・データあります

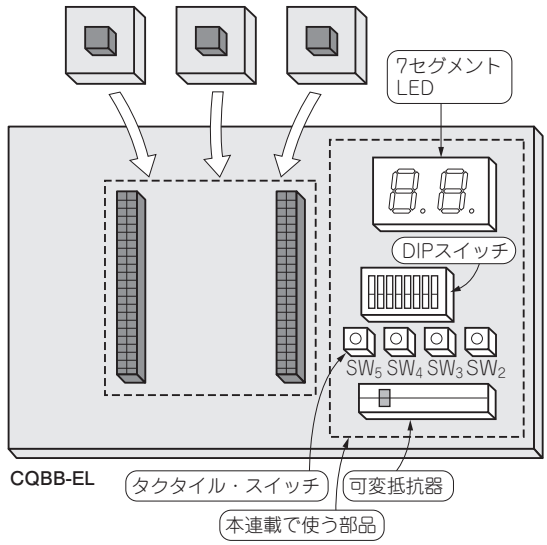
わかって書けば移植性抜群！ GCCプログラミング術

歴代の本誌付属CPUボード全対応！

第2回 CPU固有周辺機能はマクロ関数＆I/O構造体でスッキリ記述

村井 和夫

歴代の本誌付属基板を挿し換え可能
(SH-2, V850, FR60, ARM7, SH-2A, RX62N, Cortex-M3)



- 電源投入により，DIPスイッチの現在値を，7セグメントLEDの表示の初期値とする．
- タイマ割り込みにより，0.5秒ごとに，7セグメントLEDの表示を更新する．状態が「カウント停止状態」であれば値を点滅表示させ，「カウントアップ状態」であればカウントアップ表示を行う．電源投入時の初期値は「カウント停止状態」である．
- SW_2押下のエッジ割り込みにより，状態を「カウントアップ状態」または「カウント停止状態」に交互に変更する．
- SW_3押下のエッジ割り込みにより，7セグメントLEDの表示値を0にして，状態を「カウント停止状態」に変更する．
- SW_4押下のエッジ割り込みにより，DIPスイッチの現在値を7セグメントLEDの表示値として，状態を「カウント停止状態」に変更する．
- SW_5押下のエッジ割り込みにより，可変抵抗器のA-D変換結果を表示値として，状態を「カウント停止状態」に変更する．

図1 題材…主要な32ビット・マイコンすべてで(なるべく)共通で動く制御プログラム

本連載では，複数種類のCPUに対応可能な開発環境GCCを使って，CPU依存部を局所化することにより，移植性の高い組み込みソフトウェアを開発する方法を解説します．

制御プログラムを構成するファイルのうち，前回はベクタ・テーブル(vect.c/vect.s)とスタートアップ・ルーチン(crt0.s)について解説しました．実は，この二つが最もCPUに依存する部分であり，アセンブリ命令を理解しなければならない難関でした．

今回は，制御プログラムのメイン処理や割り込み処理を記述したファイル(main.c)と，インクルードしているヘッダ・ファイル(macro.h，io.h)について解説します．CPUごとに違うI/O処理は，マクロ関数としてヘッダ・ファイルmacro.hに収めて，インクルードすることにより，main.c自体はすべてのCPUに対応できるようにしてあります．

リスト1 各CPU共通！初期化とメイン関数(main.cの前半)
サンプル・プログラムを，一部，誌面向けに簡略化した．全ソース・コードは本誌ウェブ・サイトからダウンロードできる

```
//************************************************/
#include    "macro.h"
//************************************************/
volatile int out;

int main(void){
    initializePeripherals();      ← CPUや周辺I/Oの初期化を行う

    out = getDipSwitchValue();    ← DIPスイッチの値を表示値として取得する

    while(1) {
        ;
    }
}

void initializePeripherals() {
    DISABLEINTERRUPT();           ← 割り込みを禁止
    SETCPU();           // CPU固有の初期化
    SETBOARDLED();      // ボード上のLEDの初期化
    SETDIPSWITCH();     // DIPスイッチ(入力ポート)の初期化
    SET7SEGLED();       // 7セグメントLED(出力ポート)の初期化
    SETIRQ();           // IRQ割り込み・ポートの初期化
    SETAD();            // A-Dコンバータの初期化
    SETTIMER();         // タイマの初期化
    ENABLEINTERRUPT();            ← すべてのハードウェアのセットアップが終わったら割り込みを許可
}

int getDipSwitchValue() {
    return GETDIPSWITCH();
}
```

プログラム全体の処理

● 初期化＆メイン関数(main.cの前半)

メイン処理では，前回に示した仕様の通り**図1**に示す初期設定を行っています．メイン処理のプログラム