第 7 章　定石から裏技，個人的こだわりまで

Cプログラム，お役立ちTips集

舘 伸幸，酒井 郁子，武子 秀樹

ここでは，C言語によるプログラミングの際に役に立つ豆知識を紹介する．ただし，C言語はさまざまな環境のもとで使われているので，ここに掲載した全項目が必ずしも全環境で有効であるというわけではない．「この項目は使用できるメモリ容量が少ない場合に有効」，「この項目はコプロセッサが存在しない場合に有効」など，それぞれ有効となる前提は異なる．解説をじっくり読んで，自分の環境ではどれが「使える技」なのかを見極めていただきたい．（編集部）

1．volatileを知っておこう

例えば，**リスト1-1**のようなプログラムは必ず正常に動作するでしょうか？

このプログラムは，①で変数iFlagをクリアし，②でタイマ機能を開始する関数voStartTimer()を呼んでいます．一定時間経過すると，タイマ割り込みが発生し，その割り込みハンドラ・ルーチンで変数iFlagに1を代入するようにしてあります．そのことを前提に，③のループではiFlagが1になるまで，つまりタイマが一定時間経過するまで待ち続けています．

コンパイラは，iFlagが割り込み先で書き変わる変数とは知りません．なので，最適化処理で③は単なる無限ループと同じと解釈し，while文の条件などを消してしまうことがあります．このことを知らずにデバッグを始めると，いつまでたってもvoWaitTime()が終了せず，タイマの設定や割り込みの設定をさんざん見直して悩むことになります．

● 困った「最適化」を防ぐvolatile

これを防ぐには，変数iFlagにvolatileという修飾子を付けて，コンパイラにwhile文の条件やiFlagを消してしまわないように伝えます．

```
volatile int iFlag;
```

と宣言するだけです．

なお，割り込みによって書き変わる変数に限らず，以下のようなものにもvolatileを付ける必要があります．

リスト1-1　一定の時間が経過するまで待つプログラム

```
int iFlag;

void voWaitTime()
{
    iFlag = 0;              ・・・①
    voStartTimer();         ・・・②

    while( iFlag == 0 ){    ・・・③
        ;
    }

    return;
}
```

- OS上のプログラムにおける共有変数
  （別のプロセスによって書き変わる）
- DMAの転送先
- マイコンの内蔵ハードウェアのレジスタ
  （ハードウェアの動作や外部入力で値が変わる）

たち・のぶゆき　NECマイクロシステム（株）

2．constを活用しよう

パソコンなどの環境では，constという修飾子は単に書き換え不可能を明示し，信頼性向上を目的に使用します．一方，マイコン環境におけるプログラムでは，constの有無で最終的に生成されるプログラム・コードが大きく変わります．

● 初期値はROMに，変数領域はRAMに

図2-1はマイコンのメモリ・マップです．大きくROM（Read Only Memory；プログラムからは書き換えできな

KEYWORD ── volatile，割り込みハンドラ，const，初期化，スタートアップ・ルーチン，関数ポインタ，シフト演算，リング・バッファ，ビット演算，ビット・フィールド，リード・モディファイ・ライト，MISRA-C，3項演算子

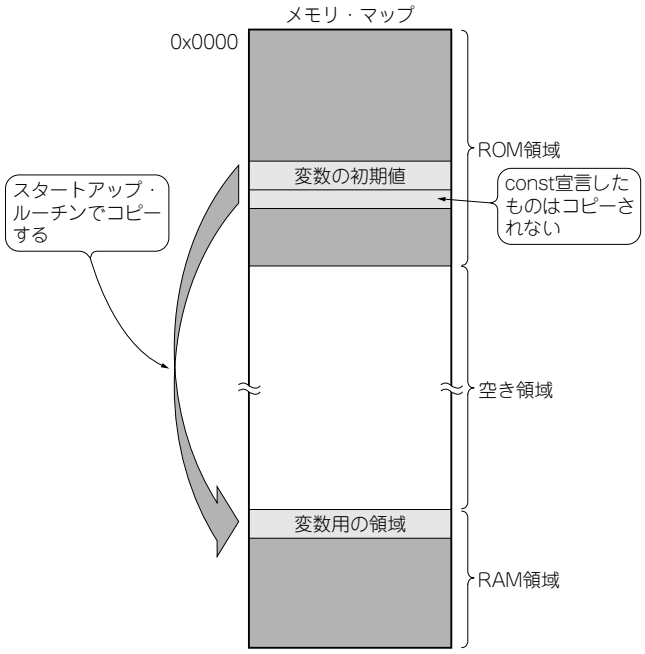

図2-1　マイコンのメモリ・マップ例

プログラムの実行コードや変数の初期値はROMに，変数用の領域はRAMに確保される．

い）とRAM（Random Access Memory；プログラムから読み書きできる）に分かれます．プログラムの実行コードはROMに格納します．初期値を持った静的変数（グローバル変数も含む）は値をROMに格納し，RAMにはその変数用の領域を確保します．

これは，初期値が必要なので最初はROM上にあるものの，変数なのでプログラムから書き換えできないと困るため，起動時にスタートアップ・ルーチンでROMからRAMへコピーして使う仕組みになっています．例えば，

```
short sTemp = 0x1234;
```

と静的変数を宣言した場合，ROMのどこかに0x1234を格納しておき，起動時，RAMのsTempに相当する場所にそれをコピーします．つまり，メモリ空間の中に同じものが二つ存在していることになります．

それでは，以下のように，初期値を持たせずにプログラムで初期化したらどうでしょうか．

```
short sTemp;
func()
{
    sTemp = 0x1234;
}
```

この場合も，結局0x1234という値はプログラムの一部としてROMの中に入っているので，メモリ効率という意味では初期値がある場合となんら変わりません．

プログラム中で書き換えが必要な変数であればやむを得ませんが，例えば正弦波形のための$\sin\theta$値の参照テーブル[注2-1]や状態遷移テーブルなど，書き換える必要のないものや書き換わっては困る変数もあります．このようなものにはconstを付けて，ROM領域のみに格納するようにします．こうすることにより，無駄にRAM領域を消費しないばかりか，書き換わってしまう危険も回避できます．

たち・のぶゆき　NECマイクロシステム（株）

3．初期化のタイミングを「見える化」する

グローバル変数やstatic変数などの静的な変数に初期値を持たせたいことがあります．C言語の文法上は，

```
int iVariable = INITVALUE;
```

と，宣言時に代入文を記述することで実現できます．ところで，この「代入」は，いつ行われているのでしょうか？

main()が動き始めたらいつの間にか環境として整っている…というのは，それ以前に誰かがやってくれているからにほかなりません．その「誰か」が，スタートアップ・ルーチンです．スタートアップ・ルーチンは，組み込みではリセット後に，パソコン・アプリケーションではアプリケーションの起動後に，CPUのレジスタ値などプログラムが動作するための環境設定を行ってmain()関数を呼び出すプログラムです．静的変数の初期値も，ここで設定するようになっています．

● スタートアップ・ルーチンの落とし穴

スタートアップ・ルーチンが動作するのは，プログラム起動時の1回だけです．リセットまたは再起動をかけない限り，プログラム中から呼び出すことはできません．この変数初期化の仕組みが，特にエラー発生時などの異常系処理において，思わぬ不具合につながることがあります．

注2-1：一般に組み込み用のマイコンでは浮動小数点を計算することは大きな処理負担となるため，あらかじめ値をテーブル（配列）に入れて，実行中はそれを参照する方法がとられている．