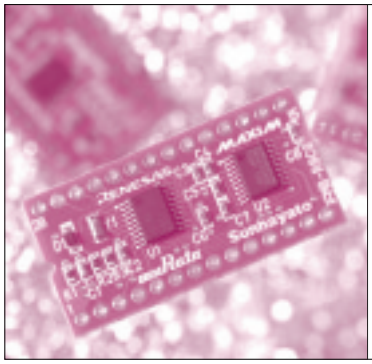

第4章　加減乗除算命令とシフト命令の動作を確認する

マイコンに演算をさせてみよう！

山本　秀樹
Hideki Yamamoto

コンピュータ(computer)は電子計算機と訳されています．マイコンもその一種なので，計算は得意であるように思えるかもしれませんが，本当のところは，計算はそんなに得意ではありません．付録マイコンでは小さい整数の四則演算を行う命令がある程度です．

この章では，演算命令によりLED点滅プログラムの点滅間隔を計算で変化させ，マイコンでどのような演算ができるのかを見てみます．

ビットを反転させてLEDを点滅させる

● 第3章のプログラムを見直して変更する

数値演算命令に入る前に，ベースとなるLED点滅プログラムの見直しを行っておきます．第3章のLED点滅プログラムをベースに，見直したプログラム・リストを**リスト1**に，その処理概要を**図1**に示します．

プログラムの基本動作は第3章のプログラムと同じですが，第3章のプログラムでは0.5 s待ち時間ループが2か所あったのを，一つにまとめています．

処理内容を簡単に説明します．まず初期化を行った後，LED_1を点灯，LED_2を消灯します．

次に0.5 s待ちを行った後，BNOT命令でLEDの点灯状況を反転します．

BNOT命令は，**図2**に示すようにオペランドを一つとり，そのオペランドでは反転するビットを指定します．この命令を実行すると，オペランドで指定したビットの‘1’と‘0’を反転します．

LEDが接続されているマイコンのポートの‘1’と‘0’を反転すると，LEDの点灯・消灯が反転します．BNOT命令を実行するたびに点灯・消灯が反転するので，二つのLEDが交互に点灯することになります．

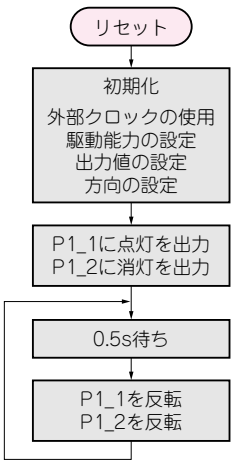

図1(1)
第3章のLED点滅プログラムをベースに変更したリスト1の処理概要
2か所あった0.5 s待ち時間ループを1か所にまとめる

● ブレークポイントを併用した動作確認

デバッガで動作を確認する場合，このプログラムのように長い間ループを回る処理があると，ステップ実行だけで動作を追いかけるのはたいへんです．第3章ではループ回数をソース・コード上で書き換えたプログラムを作りましたが，今回はブレークポイントを使って確認不要な部分をスキップしてみます．

ブレークポイントは，それを設定した箇所までプログラムが実行されると，実行が中断する機能です．実行が中断した後は，ステップ実行などデバッガの機能を自由に使えます．

リスト1のプログラムを作成し，ダウンロードしておきます．最初のBNOT命令にブレークポイントを設定するため，**図3**の位置をダブル・クリックします．●が表示されれば，その行にブレークポイントが設定されたことを意味します．設定したブレークポイントを解除するには，その箇所を再度ダブル・クリックします．

Keywords
BNOT命令，ブレークポイント，ADD命令，SUB命令，JCnd命令，フラグ・レジスタ，SUB命令，LSTファイル，メモリ・ウィンドウ，リトル・エンディアン，ビッグ・エンディアン，乗算命令，除算命令，シフト・ローテート命令

リスト1　ビットを反転させるBNOT命令を使って0.5 s待ち時間ループを一つにまとめたLED点滅プログラム

```
; トランジスタ技術 2005年4月号
; 第4章 LED点滅プログラム 見直し版

            .INCLUDE    sfr_r815.inc          ; ハードウェア定義ファイルの読込み

            ; プログラム部分
            .SECTION    PROGRAM, CODE
            .ORG        0D000h
Start:                                        ;  (B) ここから実行開始
            BSET        prc0                  ; 外部クロックに切り替える
            BSET        cm13
            BSET        cm15
            BCLR        cm05
            BCLR        cm16
            BCLR        cm17
            BCLR        cm06                                  ← 外部クロックに切り替える
            NOP
            NOP
            NOP
            NOP
            BCLR        ocd2
            BCLR        prc0                  ; 外部クロックへの切り替え完了

            MOV.B       #00000110b, drr       ; 駆動能力の設定
            MOV.B       #00000110b, p1        ; ポートに出力する初期値の設定        ← ポートの設定
            MOV.B       #00000110b, pd1       ; ポートの方向を出力に設定

            BCLR        p1_1                  ; LED1を点灯する
            BSET        p1_2                  ; LED2を消灯する                      ← LEDの点灯と消灯

Loop:
            MOV.W       #50, r1               ; 以下の10ミリ秒ループを50回まわって0.5秒待つ
Wait01:
            MOV.W       #28571, r0            ; 10ミリ秒ループ（値の求め方は第3章参照）  ← 0.5秒待つ
Wait02:
            SBJNZ.W     #1, r0, Wait02        ; 10ミリ秒ループ実行
            SBJNZ.W     #1, r1, Wait01        ; 0.5秒ループ実行

            BNOT        p1_1                  ; LED1を反転する
            BNOT        p1_2                  ; LED2を反転する                      ← LEDの状態を反転

            JMP         Loop                  ; LED点滅を繰り返す

            ; リセットベクタ部分
            .SECTION    FIXVECTOR, ROMDATA
            .ORG        0FFFCh
Reset:
            .LWORD      Start | 0FF000000h    ; (A) 実行開始箇所を指定する

            .END
```

BNOT命令にブレークポイントを設定した状態で，プログラムを実行します．すると，0.5 sほどたつと，プログラムの実行が中断して黄色い矢印がブレークポイントの行に表示されるはずです．また，LED_1は点灯し，LED_2は消灯しています．この状態では，ブレークポイントが設定された行の命令は，まだ実行されていません．

そこで，ステップ実行で最初のBNOT命令を実行してみると，LED_1が消灯します．さらに2番目のBNOT命令を実行すると，LED_2が点灯します．この

【構文】
BNOT(：format) dest
G, S(指定可能)
【オペレーション】
dest←$\overline{dest}$

図2　ビットを反転させるBNOT命令の構文と操作

図3　ブレークポイントの設定