コンピュータ工学 講義プリント(12 月 11 日)

今回は、ローテイト命令を用いて、前回よりも高度な LED の制御を行う。

・光が流れるプログラム、片道バージョン(教科書 P.119 参照)

0.5 秒ごとに、教科書 P.119 の図 5.23 の様に、LED の点灯パターンが変化するプログラムを作成する事を考える。この様にすれば、光っている点が、徐々に右に動いているように見え、右端まで移動したら、また左端に光が移動するように見える。

この様な点灯パターンを実現するには、PORTB レジスタに 80H、40H、20H、10H、08H、04H、02H、01H、80H、40H、20H、・・・と順に、0.5 秒ごとに書き込めばよい。

この様な点灯パターンを実現する方法は色々考えられるが、最も効率が良い実現方法は、おそらくローテイト命令を利用する方法である。

・ローテイト命令(教科書 P.81～P.82 参照)

PIC16F84A には、RRF 命令(教科書 P.81 参照)と RLF 命令(教科書 P.82 参照)の 2 種類のローテイト命令がある。

RRF 命令は、図 1 に示す様に、C フラグをファイルレジスタの最上位ビットの左に配置し、C フラグおよびファイルレジスタの各ビットを一つずつ、右方向に回転(ローテイト)する命令である。(ファイルレジスタの最下位ビットは C フラグに入る) そして、第 2 オペランドの d が 0 なら結果は C フラグと W レジスタに、d が 1 なら結果は C フラグと元のファイルレジスタに格納される。

書式：RRF f,d

オペランド：f はファイルレジスタのアドレス (0 ≦ f ≦ 127)
：d は格納先 (d=0 の場合 W レジスタ、d=1 の場合 f で指定されるファイルレジスタ)

動作

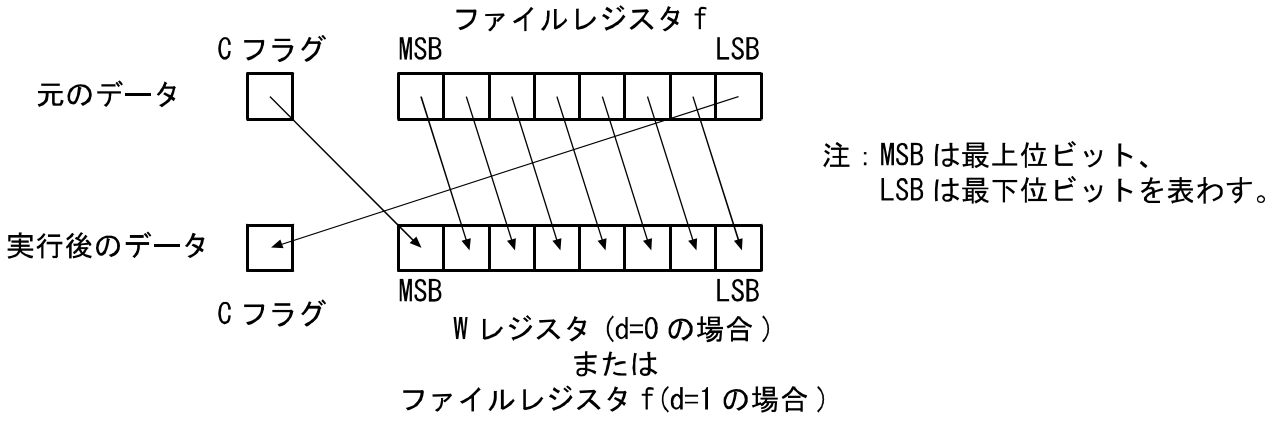
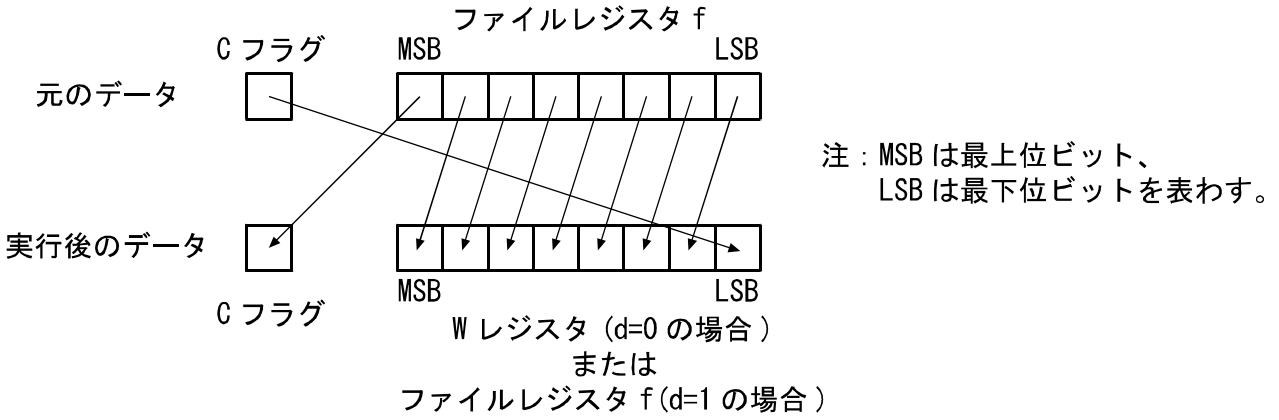


図 1、RRF 命令の動作

書式：RLF f,d

オペランド：f はファイルレジスタのアドレス (0 ≦ f ≦ 127)
　　：d は格納先 (d=0 の場合 W レジスタ、d=1 の場合 f で指定されるファイルレジスタ)

動作



C フラグ以外は一つ左に移動し、C フラグは LSB に移動する (左ローテイト)

図 2、RLF 命令の動作

また RLF 命令は、図２示す様に、RRF 命令とは逆に、各ビットを一つずつ、左方向に回転(ローテイト)する命令である。

8 ビット変数のアドレスを A とし、C フラグに 0 を代入してから RLF A,1 を実行すると、変数の内容が、左に 1 ビットシフトし、最下位ビット(LSB)には 0 が入る。これは、変数の値を 2 倍したことに相当する。RLF A,1 の結果、C フラグが 1 になれば、2 倍した際にオーバーフローが起こった事を意味する。(リスト 1 参照)

リスト 1、8 ビット変数を 2 倍するプログラム

```
BCF     STATUS,C        ; C フラグをクリア
RLF     A,1             ; A を左ローテイト
; この時点で 8 ビット変数 A は 2 倍され、オーバーフローが起こっていれば、C フラグが 1 になっている
```

また、16 ビット変数 A の下位バイトのアドレスを AL、上位バイトのアドレスを AH とした場合、まず C フラグを 0 にした上で、先に下位バイトを左ローテイトし、次に上位バイトを左ローテイトすると、16 ビット変数 A の値を 2 倍する事ができる。(リスト 2 参照)

リスト 2、16 ビット変数を 2 倍するプログラム

```
BCF     STATUS,C        ; C フラグをクリア
RLF     AL,1            ; 下位バイトを左ローテイト
RLF     AH,1            ; 上位バイトを左ローテイト
                        ;  (AL の MSB は C フラグ経由で AH の LSB に転送される)
; この時点で 16 ビット変数 A は 2 倍され、オーバーフローが起こっていれば、C フラグが 1 になっている
```

リスト 1 やリスト 2 の例で分かるように、RLF 命令は、変数の値を 2 倍する時に使われる命令である。
一方で RRF 命令は、変数の値を 1/2 にする時に使われる命令である。
8 ビット変数のアドレスを A とし、C フラグを 0 にした上で、RRF A,1 を実行すると、変数の内容が右に 1 ビットシフトし、最上位ビットには 0 が入る。これは変数を 1/2 にし、小数点以下の端数を切り捨てる操作に相当する。端数が出た場合は C フラグが 1 になる。(リスト 3 参照)

リスト 3、8 ビット変数を 1/2 にするプログラム

```
BCF     STATUS,C        ; C フラグをクリア
RRF     A,1             ; A を右ローテイト
; この時点で 8 ビット変数 A は 1/2 になり、小数点以下の端数があれば、C フラグが 1 になっている
```

また、16 ビット変数 A の下位バイトのアドレスを AL、上位バイトのアドレスを AH とした場合、まず C フラグを 0 にした上で、先に上位バイトを右ローテイトし、次に下位バイトを右ローテイトすると、16 ビット変数 A の値を 1/2 にして、小数点以下の端数を切り捨てする事ができる。(リスト 4 参照)

リスト 4、16 ビット変数を 1/2 にするプログラム

```
BCF     STATUS,C        ; C フラグをクリア
RRF     AH,1            ; 上位バイトを右ローテイト
RRF     AL,1            ; 下位バイトを右ローテイト
                        ;  (AH の LSB は C フラグ経由で AL の MSB に転送される)
; この時点で 16 ビット変数 A は 1/2 になり、小数点以下の端数があれば、C フラグが 1 になっている
```

・光が流れるプログラム、片道バージョンの RRF 命令による実装(教科書 P.120 参照)
C フラグに 0 を代入し、PORTB レジスタに 80H を代入した状態で、RRF PORTB,1 を何度も実行すると、C フラグと PROTB レジスタの内容は、表 1 の様に変化してゆく。

表 1、RRF 命令の実行回数と C フラグおよび PORTB レジスタの内容の変化

RRF 命令の実行回数	C フラグ	PORTB	RRF 命令の実行回数	C フラグ	PORTB
0	0	10000000B (80H)	5	0	00000100B (04H)
1	0	01000000B (40H)	6	0	00000010B (02H)
2	0	00100000B (20H)	7	0	00000001B (01H)
3	0	00010000B (10H)	8	1	00000000B (00H)
4	0	00001000B (08H)	9	0	10000000B (80H)

この様に PORTB の内容は、80H、40H、20H、10H、08H、04H、02H、01H、**00H、**80H、40H、20H、･･･と変化していく事がわかる。ほぼ目的の変化の仕方をしているが、01H の次は 80H になっているのではなく、一旦 00H を経由する事に注意が必要である。
教科書 P.120 では、図 5.24 のフローチャートに示すように、タイマ 3 (0.5 秒のウェイトを与えるサブルーチン)の呼び出しと右ローテイト(RRF 命令)を交互に行えば、教科書 P.119 の図 5.23 の様な、LED の点

灯パターンが得られると説明されている。しかしながら、表 1 を見れば分かるように、この方法では LED が全て消灯する(PORTB が 00H になる)瞬間が生じるようになり、目的の点灯パターンは得られない。

講義では、実際に教科書 P.120 のリスト 5.5 のプログラムを実行するデモを行うので、LED が全て消える瞬間があるのを、各自で確認せよ。

また、教科書では特に説明されていないが、リスト 5.5 のタイマルーチン(TIMER1～3)が、C フラグに影響するような命令を一切含んでいない事に注意が必要である。タイマルーチンが C フラグを書き換えてしまうと、PORTB の変化が表 1 には従わなくなるため、LED の発光パターンが崩れてしまう。もし C フラグに影響する命令を使ってタイマルーチンを作成するなら、タイマルーチンの実行の前に C フラグを退避し、実行後に C フラグを復帰する必要がある。

さて、教科書のリスト 5.5 では目的の点灯パターンが得られない事が分かったので、一部を書き換えて、ちゃんと動作するようにする事を考える。リスト 5.5 の初期化ルーチン(BCF STATUS,C 以前)と、タイマルーチン(TIMER1 MOVLW D'62'以降)には問題がないので、メインルーチン(MOVLW LEDD から GOTO REPEAT まで)を書き換えることにする。

教科書のリスト 5.5 のメインルーチンを抜き出して書いたのが、次のリスト 5 である。

リスト 5、教科書のリスト 5.5 のメインルーチン

```
        MOVLW   LEDD            ; 点灯データを W レジスタにセット
        MOVWF   PORTB           ; 点灯データをポート B に出力
REPEAT  CALL    TIMER3          ; 0.5 秒タイマの呼び出し
        RRF     PORTB,1         ; ポート B を 1 ビット右にローテイト
        GOTO    REPEAT
```

問題になっているのは、PORTB が 00H になる瞬間があることである。表 1 をみると、PORTB が 00H になっているときは C フラグが 1 になっており、PORTB が 00H 以外の時は C フラグが 0 になっている事が分かる。これを利用して、RRF 命令の実行後に C フラグが 1 になっている場合に、もう一度 RRF 命令を実行すれば、問題を回避できる。

なお、RRF 命令は Z フラグには影響しないので、Z フラグで PORTB が 00H になっているかどうかの判断はできないことに注意が必要である。

この様な考え方でメインルーチンを書き直すと、次のリスト 6 の様になる。

リスト 6、リスト 1 を書き直し、LED が全て消灯する不具合を修正したもの

```
        MOVLW   LEDD            ; 点灯データを W レジスタにセット
        MOVWF   PORTB           ; 点灯データをポート B に出力
REPEAT  CALL    TIMER3          ; 0.5 秒タイマの呼び出し
        RRF     PORTB,1         ; ポート B を 1 ビット右にローテイト
        BTFSC   STATUS,C        ; C フラグが 0 ならば、次の命令をスキップ
        RRF     PORTB,1         ; C フラグが 1 ならば、もう一度、右にローテイト
        GOTO    REPEAT
```

講義では、メインルーチンをリスト 6 に書き換えたプログラムを実行するデモを行うので、LED が全て

消えてしまう不具合が修正されている事を、各自で確認せよ。

なお、厳密に言えば、リスト 6 のプログラムでも、PORTB が 00H になって LED が全て消える瞬間が 2 実行サイクル(0.8μs)だけ発生するが、人間の知覚できる時間ではないので、問題にはならない。

PORTB が 00H になる瞬間が発生するのを厳密に回避したいなら、PORTB の内容を直接 RRF 命令で右方向にローテイトするのではなく、PORTB に書き込む値を保持する変数を作り、その変数を RRF 命令で操作してから、PORTB に転送すればよい。

なお、今回は、RRF 命令を用いて、光の点が右に流れるプログラムを作ったが、RLF 命令を用いると、同様の方法で、光の点が左に流れるプログラムを作る事もできる。

・光が流れるプログラム、往復バージョン(教科書 P.121 参照)

今度は、LED の点灯が、教科書 P.121 の図 5.25 に示すように、往復して流れるように見えるプログラムを作成する。このためには、PORTB の内容が、80H、40H、20H、10H、08H、04H、02H、01H、02H、04H、08H、10H、20H、40H、80H、40H、20H、･･･という順番で変化すればよい。

光が右に流れている間は、RRF 命令で、点灯する LED を右にずらしていけばよい。また、光が左に流れている間は、RLF 命令で、点灯する LED を左にずらしていけばよい。しかし問題は、光の流れる方向が変わる時の処理である。

まず C フラグを 0、PORTB を 80H に初期化して、RRF 命令で LED の点灯位置を右にずらしていく事を考える。この場合、表 1 の様に PORTB の内容が変化してゆく。

RRF 命令を 7 回実行して PROTB が 01H になるところまでは良いが、ここでもう 1 度 RRF 命令を実行すると PORTB が 00H になってしまう。

この時に C フラグが 1 になる事を利用して、BTFSS STATUS,C で、RRF 命令を実行するループから抜け、その後に RLF 命令を 2 回実行すると、PORTB は、目的の 02H となる。

以後は、RLF 命令を繰り返せば、PORTB の内容は 04H、08H、10H、20H、40H、80H と変化し、LED の点灯位置は左にずれていく。

しかしここでもう一度 RLF 命令を実行すると、PORTB は 00H となってしまう。

この時に C フラグが 1 になる事を利用して、BTFSS STATUS,C で、RLF 命令を実行するループから抜け、その後に RRF 命令を 2 回実行すると、PORTB は、目的の 40H となる。(教科書 P.121 図 5.26 参照)

以後は、RRF 命令を繰り返すループに戻ればよい。

この考え方をフローチャートにすると、教科書 P.122 の図 5.27 となり、それをプログラムにしたのが、同じページのリスト 5.6 である。

講義では、リスト 5.6 のプログラムを実行するデモを行うので、教科書 P.121 の図 5.25 の通りの点灯パターンが得られている事を、各自で確認せよ。