15 STEP

ロータリエンコーダを使おう

この STEP では、ロータリエンコーダを使ってドットマトリクス LED の点灯行を変化させるプログラムを組みます。
ロータリエンコーダは、位置、角度センサとしてロボットの関節に組み合わせるなど、様々なところで使われています。

15.1　ロータリエンコーダ

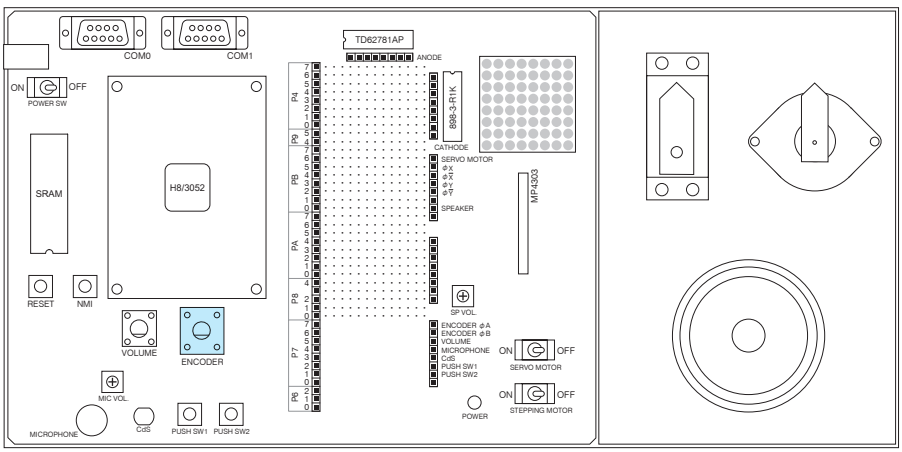


図 15-1　ロータリエンコーダ

ロータリエンコーダは入力装置ですが、ちょうどステッピングモータ逆の装置としてイメージすればいいでしょう。ステッピングモータはパルスを与えると回転しますが、ロータリエンコーダは回転させるとパルスが取り出せるのです。回転方向や回転速度でパルスが変化するのは同じです。

STEP 15 ロータリエンコーダを使おう

ロータリエンコーダを手で回すと「カタカタ」というクリック感があります。この「カタカタ」もステッピングモータと似ています。

本キットのロータリエンコーダは、1回「カタッ」と回した時、2本の信号端子φA、φBから図15-2のようなパルス信号が出力される仕組みになっています。

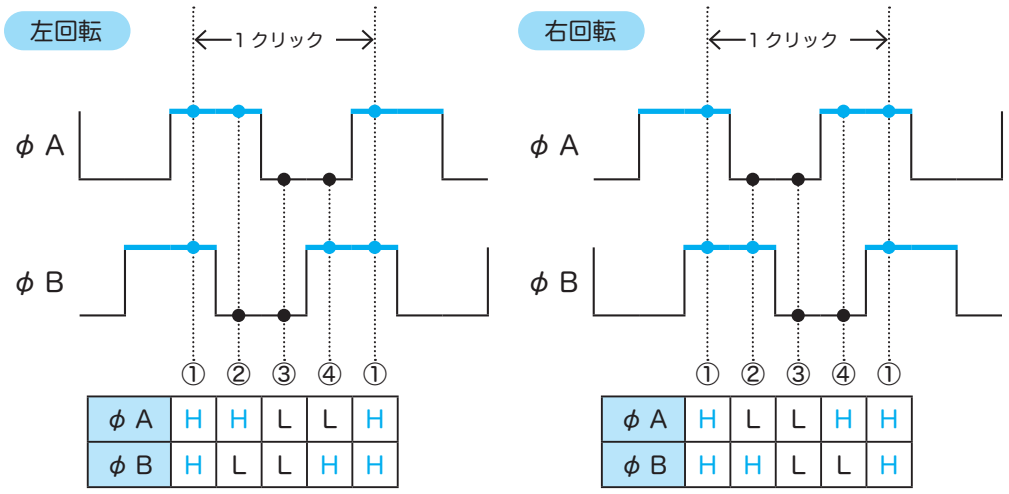
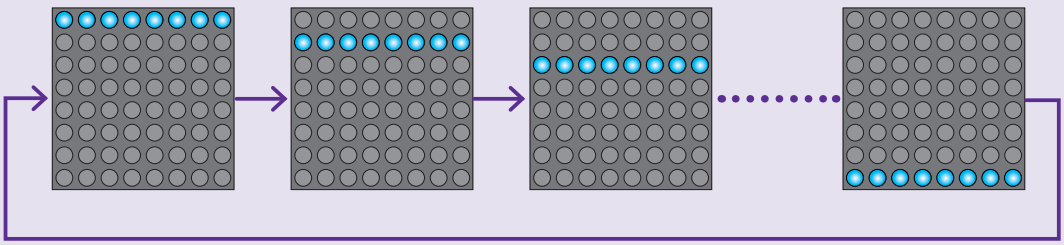


図15-2　ロータリエンコーダのパルス

1クリックでどちらの信号端子も電圧レベルがH→L→Hと変化します。つまみが落ち着いた状態ではどちらもHです。
右周りと左周りではパルスの出力パターンが異なっています。このパターンをプログラムで判断できれば回転方向を知ることができます。

課題　15-1

ロータリエンコーダを回すと、１クリックでドットマトリクス LED の点灯行が１行下がる。
８行目が点灯した後は、１行目に戻す。



ロータリエンコーダを配線しましょう。図 15-3 のように追加配線してください。

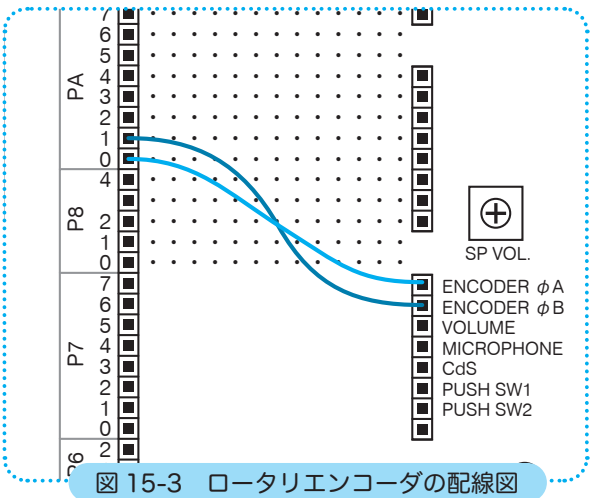


図 15-3　ロータリエンコーダの配線図

15.2 PADDR

ポート A には、STEP13 でサーボモータ、STEP14 でステッピングモータを接続して使いました。本 STEP では入力装置であるロータリエンコーダを追加します。
これまでのポート A の配線をまとめると図 15-4 になります。PA_1 と PA_0 は入力端子に設定してください。PADDR の設定値は 2 進数で 1111 1100 ですから 16 進数で 0xFC になります。

ビット：	7	6	5	4	3	2	1	0
	PA_7DDR	PA_6DDR	PA_5DDR	PA_4DDR	PA_3DDR	PA_2DDR	PA_1DDR	PA_0DDR
設定値：	1	1	1	1	1	1	0	0
使用機器：	ー	ステッピングモータ				サーボ	ロータリエンコーダ	
ソケット名：	ー	X	Y	$\overline{X}$	$\overline{Y}$	SERVO MOTOR	ϕ B	ϕ A

図 15-4 PADDR の設定

フローチャートを考えてみます。
まずは、φ A だけを見てロータリエンコーダが回転したかどうかを判断してみましょう。
回転したかどうかはφ A の立下り(H→L)と立上り(L→H)をとらえれば分かります。フローチャートでも「回転した？」の中身を示しておきます。

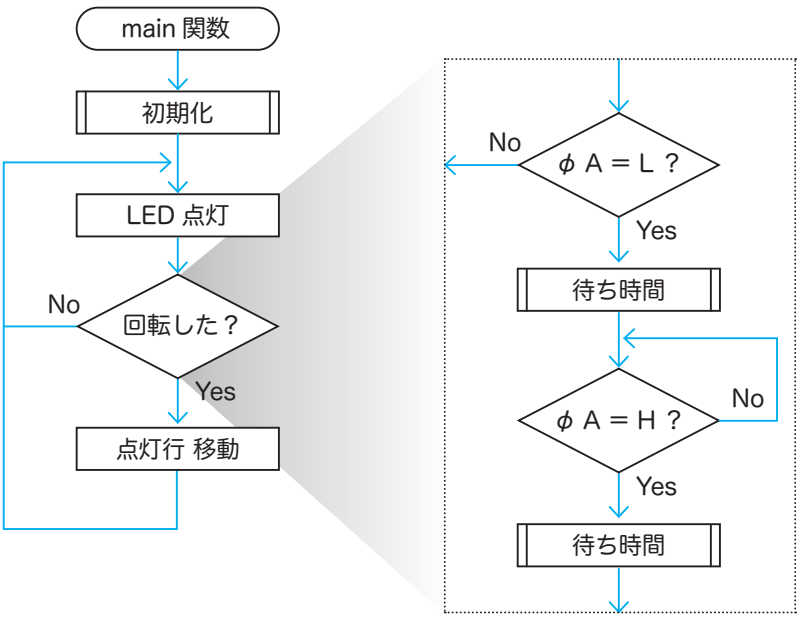
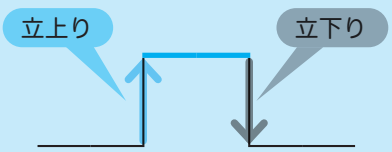


図 15-5　フローチャート

フローチャートの「待ち時間」は、チャタリング対策です。
ロータリエンコーダも SW と同じように機械的な接点があるためチャタリングを起こします。ですから、パルス A の立下りと立上りを確認した後それぞれに待ち時間を加えています。ロータリエンコーダのチャタリングは、タクトスイッチよりずっと短く 1ms ほどでいいでしょう。

立上り / 立下り

パルスなどの電気信号が L から H になることを「立上り」、H から L になることを「立下り」と言います。



プログラム例　15-1

```
36  /*
37   *  main 関数
38   */
39  int main(void)
40  {
41      int index = 0;      // 配列に使用する変数
42      initIO();           // 初期化関数の呼び出し
43      PB.DR.BYTE = 0x00; // L カソード
44
45      while (1)
46      {
47          P4.DR.BYTE = a_p4[index]; // LED 点灯
48
49          if (PA.DR.BIT.B0 == 0) // φ A が立下がると
50          {
51              wait01Ms(10); // チャタリング対策
52              while (PA.DR.BIT.B0 == 0) // φ A が立上がるまでループ
53                  ;
54              wait01Ms(10); // チャタリング対策
55
56              index++;        // 点灯行を下げる
57              if (index > 7) // index 値の補正
58                  index = 0;
59          }
60      }
61
62      return 0;
63  }
```

プログラムはこれまで学習してきたことで実現できるでしょう。
ロータリエンコーダを回すたび、ドットマトリクス LED の点灯行が変化したと思います。
しかし、どちらに回しても点灯行が下がってしまいますね。

反応が悪い場合は、チャタリング対策の待ち時間を調整してください。長すぎても反応がおかしくなるので注意してください。

15.3　ロータリエンコーダの回転方向を識別する

回転方向を識別するには、φ A とφ B のパルスのズレを利用します。もう一度、ロータリエンコーダのパルス波形を見てみましょう。

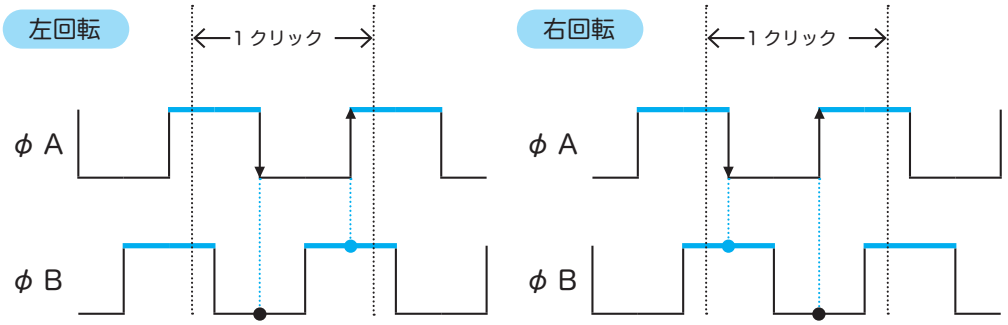


図 15-2（再掲）　ロータリエンコーダのパルス

φ A の立下り時と立上り時のφ B の電圧レベルを見れば、回転方向を識別できるようになります。違いをまとめると表 15-1 のようになります。具体的には、クリック終わりのφ A の立上り時にφ B が H なら左回転、L なら右回転と識別すればいいでしょう。

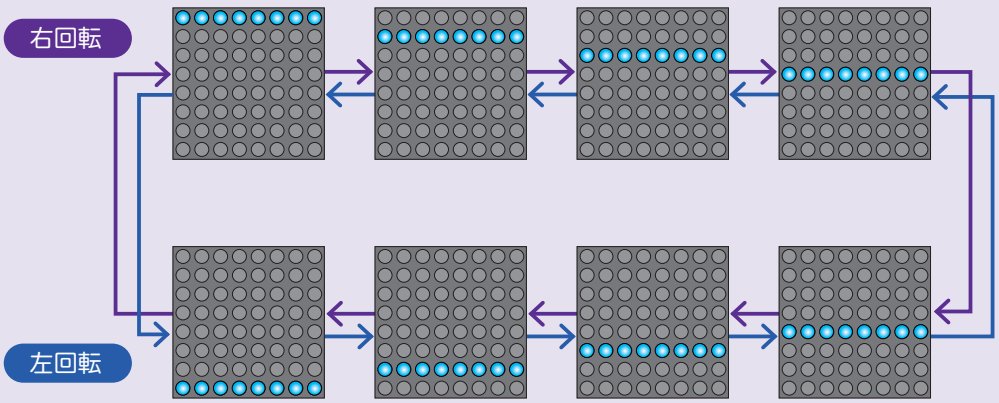
	左回転		右回転	
φ A	↓	↑	↓	↑
φ B	L	H	H	L

↑：立上り
↓：立下り

表 15-1　パルス A 立上り、立下り時のパルス B の状態

課題　15-2

ロータリエンコーダを右回転すると、点灯行が 1 行下がる。
左回転すると、点灯行が 1 行上がる。
※ 右回転の場合は 8 行目が点灯した後は 1 行目に戻り、
左回転の場合は 1 行目が点灯した後は 8 行目に戻ります。



フローチャートを考えてみます。φ A の立上り時に、φ B が H なら左回転、L なら右回転と判定すればよいだけです。図 15-6 フローチャートに追加した処理を色付けしています。

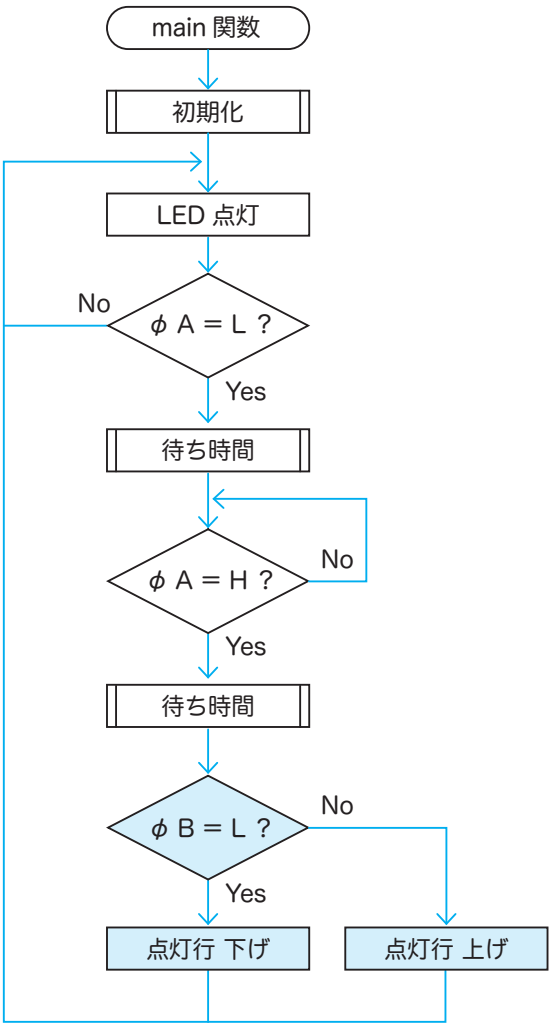


図 15-6　フローチャート

プログラム例　15-2

```
/*
 *  main 関数
 */
int main(void)
{
    int index = 0;      // 配列に使用する変数
    initIO();           // 初期化関数の呼び出し
    PB.DR.BYTE = 0x00; // L カソード

    while (1)
    {
        P4.DR.BYTE = a_p4[index]; // LED 点灯

        if (PA.DR.BIT.B0 == 0) // φ A が立下がると
        {
            wait01Ms(10); // チャタリング対策
            while (PA.DR.BIT.B0 == 0) // φ A が立上がるまでループ
                ;
            wait01Ms(10); // チャタリング対策

            if (PA.DR.BIT.B1 == 0) // 右周りなら
            {
                index++;        // 点灯行を下げる
                if (index > 7) // index 値の補正
                    index = 0;
            }
            else // 左周りなら
            {
                index--;        // 点灯行を上げる
                if (index < 0) // index 値の補正
                    index = 7;
            }
        }
    }

    return 0;
}
```

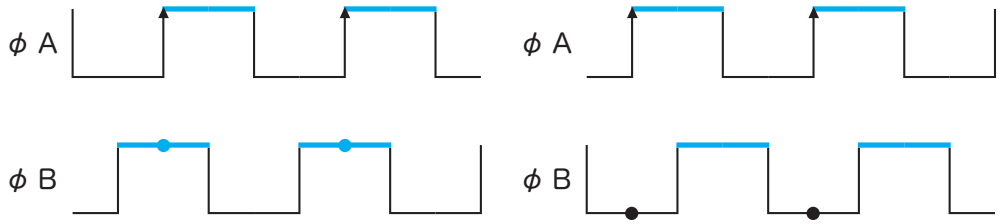
今回も新たな C 言語のコマンドはありませんから、処理の内容は理解できるでしょう。
プログラム 15-1 から追加変更した部分を示しておきます。

今回は、パルス A の立上りだけで判定しましたが、パルス A の立下りを追加したり、パルス B の立上り・立下りも追加して判定させることもできます。複数のタイミングで判定することによって機能を 2 倍、4 倍にすることができ、このことを逓倍（2 逓倍、4 逓倍）といいます。

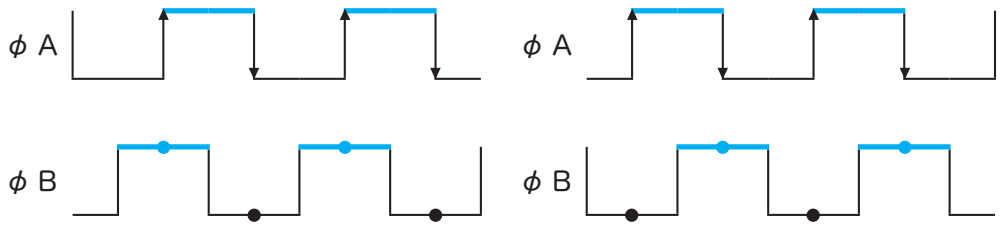
次ページに各逓倍化の方法を図示しています。実際にプログラムで組んで動作させてみるといいでしょう。2 逓倍では 1 クリックで LED 点灯行が 2 行進み、4 逓倍では 4 行進みます。

15.4　逓倍化の信号判定

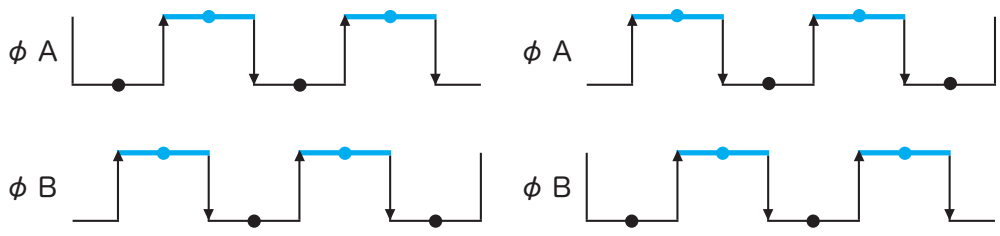
1逓倍　左回転　右回転



2逓倍　左回転　右回転



4逓倍　左回転　右回転



課題　15-3

要求仕様は「課題 15-2」と同じです。
ロータリエンコーダを右回転すると、点灯行が 1 行下がる。
　　左回転すると、点灯行が 1 行上がる。
これを「2 逓倍」で実現してみましょう。

課題　15-4

「4 逓倍」で実現してみましょう。
解答例は「テーブル」を利用したプログラムなので、少し難しいかもしれません。
実行して動作を確認してみるだけでもいいでしょう。

	左回転				右回転			
φ A	H	↓	L	↑	↓	L	↑	H
φ B	↓	L	↑	H	H	↓	L	↑

表 15-2　分解能を 4 逓倍するための判別方法