# 13 STEP

## サーボモータを使おう

この STEP では、サーボモータを回転させます。
サーボモータはロボットの関節やアミューズメント機器に使われており、デューティ比によって回転する角度が決まります。

## 13.1　サーボモータとは

本キットには、ラジコンなどで多く利用されているサーボモータユニットが搭載されています。このユニットには、DC モータやギヤ（減速機構）、位置センサなどが一体になって組み込まれており、電源と目標値（角度情報）を供給するだけで簡単に制御することができます。

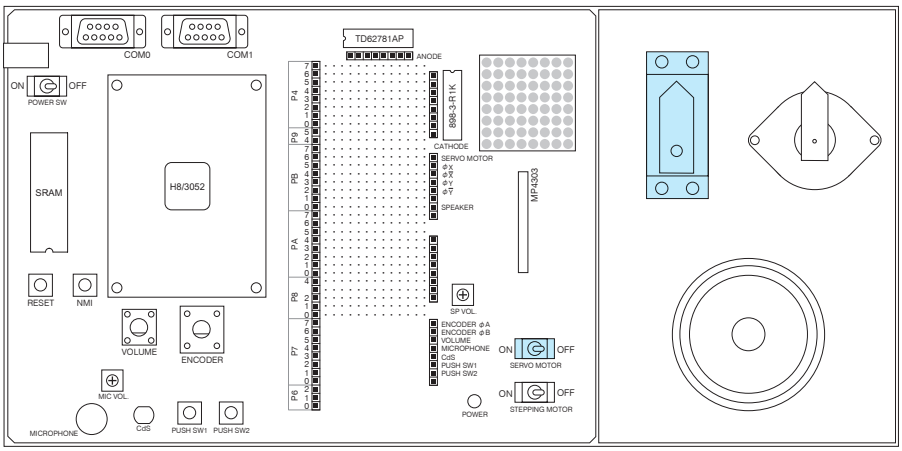



図 13-1　サーボモータ

本キットでは主電源とは別に、サーボモータ用の電源スイッチがあります。サーボモータを駆動するときはスイッチを ON してください。

# STEP 13 サーボモータを使おう

サーボモータは、普通のモータのように軸が回転し続けるのではなく、回転角度を保持する動作を続けます。回転角度は仕様で決まっており、本キットの場合は、左 60 度～右 60 度です。

サーボモータは、PWM で制御します。回転角度はパルス幅で決まります。回転角度とパルス幅の関係は、メーカや機種によって違いがありますが、キット付属のサーボモータは下図のような関係になります。周期は 10 ～ 20[ms] の範囲なら問題ないので、L 時間は固定でパルス幅だけ変更してもいいでしょう。

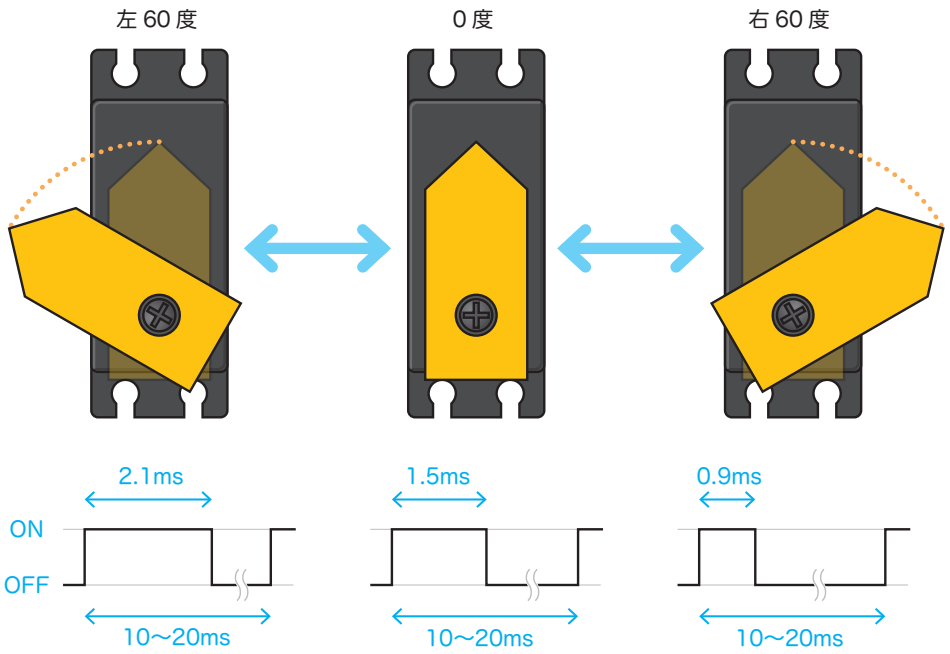



図 13-2　サーボモータの制御パルス

⚠ パルス幅が短か過ぎたり長過ぎると、モータが最大回転角を超えて動作し、サーボのギヤや制御回路が壊れてしまう事があります。パルス幅の設定値は十分注意してください。

## 13.2　サーボモータを回転させよう

それでは実際に、サーボモータの位置がデューティ比によって決まることを確かめてみましょう。図 13-3 のように $PA_2$ を SERVOMOTOR に追加配線してください。

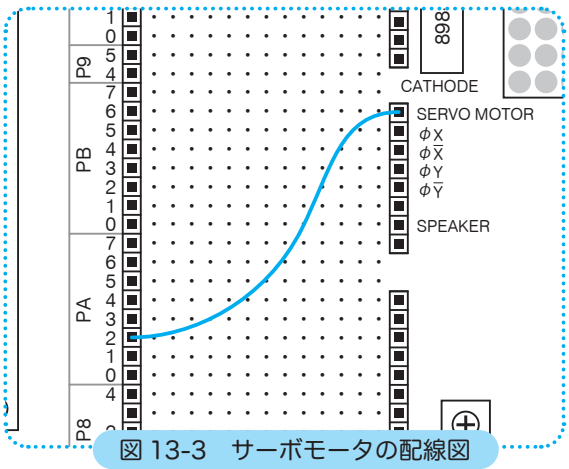



図 13-3　サーボモータの配線図

サーボモータと接続したポート A は、ポート 4 やポート B と同じ、8 ビット 8 ピンのポートです。PADDR、PADR について見てみましょう。

ポート A の入出力設定は、PADDR に値を書き込んで行います。
サーボモータは出力装置です。サーボモータと接続した $PA_2$ は出力端子に設定してください。また、未使用ピンは出力端子に設定することが推奨されるので、ポート A 端子はすべて出力に設定すればいいでしょう。

## 13.3 PADDR と PADR

| ビット： | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $PA_7DDR$ | $PA_6DDR$ | $PA_5DDR$ | $PA_4DDR$ | $PA_3DDR$ | $PA_2DDR$ | $PA_1DDR$ | $PA_0DDR$ |
| 初期値： | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W： | — | W | W | W | W | W | W | W |

図 13-4　PADDR と初期値

| ビット： | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $PA_7$ | $PA_6$ | $PA_5$ | $PA_4$ | $PA_3$ | $PA_2$ | $PA_1$ | $PA_0$ |
| 初期値： | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W： | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

図 13-5　PADR と初期値

※ 本キットの H8/3052F は、モード 6 で動作します。上記初期値はモード 6 の場合です。
詳細は「H8/3052F ハードウェアマニュアル」をご参照ください。

モード 6 では、$PA_7DDR$ は 1 に固定され、$PA_7$ はアドレス出力として機能します。

課題　13-1

サーボモータを図のように中央 0 度の位置まで回転させる。

中央 0 度はパルス幅 1.5ms ですから、待ち時間関数を利用してパルスを出力するフローチャートを考えてみましょう。

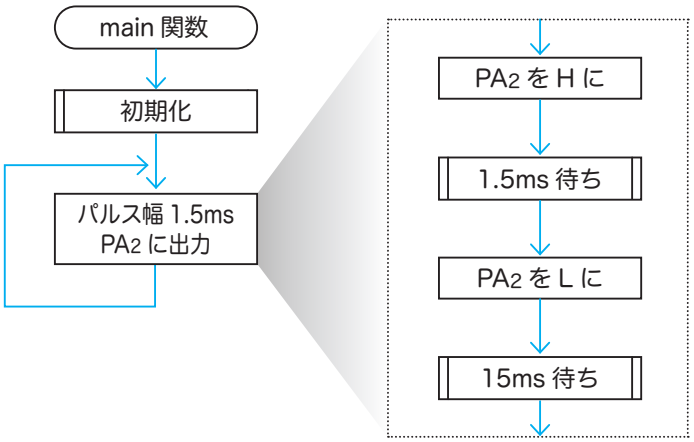



図 13-6　フローチャート

# STEP 13 サーボモータを使おう

では、図 13-6 を基にプログラム 13-1 を組んでみましょう。
また、サーボモータの取り扱いついて注意事項を記載しておくので、サーボモータを故障させないために、絶対に守ってください。

・プログラムを実行する前にトグルスイッチ (SERVO MOTOR) を ON にしておく。
・プログラムを実行中にトグルスイッチを ON/OFF しない。

プログラム 13-1 の待ち時間関数は単位が 0.1[m 秒 ] になっています。
なぜ、単位が 0.1[m 秒 ] になっているのかと言うと、待ち時間関数の引数である int ms に小数を入れることはできないからです。また、少数を入れることができたとしても 0.1 回ループさせることはできません。そこで 156 回ループさせて 0.1m 秒を作る関数に改造します。

### プログラム例　13-1

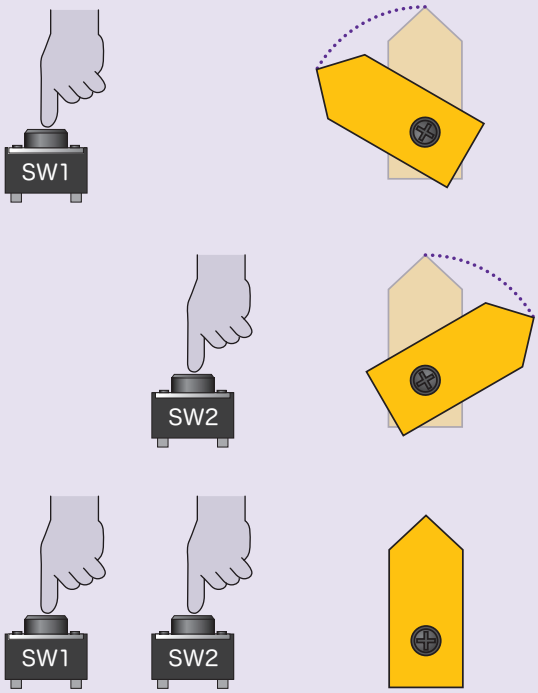
```
15  /*
16   *  待ち時間関数 0.1ms
17   */
18  void wait01Ms(int ms)
19  {
20      int i, j, k;
21      for (j = 0; j < ms; j++)
22      {
23          for (i = 0; i < 156; i++)
24              k++;
25      }
26  }
27
28  /*
29   *  main 関数
30   */
31  int main(void)
32  {
33      initIO(); // 初期化関数の呼び出し
34
35      // サーボモータにパルスを送る
36      while (1)
37      {
38          // 1.5[ms] 間、H にする
39          PA.DR.BIT.B2 = 1;
40          wait01Ms(15); // 待ち時間関数の呼び出し
41
42          // 15[ms] 間、L にする
43          PA.DR.BIT.B2 = 0;
44          wait01Ms(150); // 待ち時間関数の呼び出し
45      }
46
47      return 0;
48  }
```

⚠ 0 度になるプログラムを実行しても矢印板が中央位置にならない場合は、矢印板の取付けねじを緩めて、中央位置に固定しなおしてください。

プログラム 13.1 で、サーボモータは中央 0 度の位置まで動いたでしょうか？
中央 0 度に動かせたら、左 60 度、右 60 度も試してみましょう。L 時間は 15ms 固定で、パルス幅を変更するだけで構いません。

## 課題　13-2

SW1 のみ押されている間、目標角度が左へ。
SW2 のみ押されている間、目標角度が右へ。
SW1 かつ SW2 を押すと中心 (0 度 ) へ戻る。
※ 目標角度は、左右とも 60 度を超えないように補正すること。



# STEP 13 サーボモータを使おう

フローチャートを考えてましょう。
目標角度が左右 60 度を超えないようにパルス幅を 0.9ms 以上 2.1ms 以下に制限します。

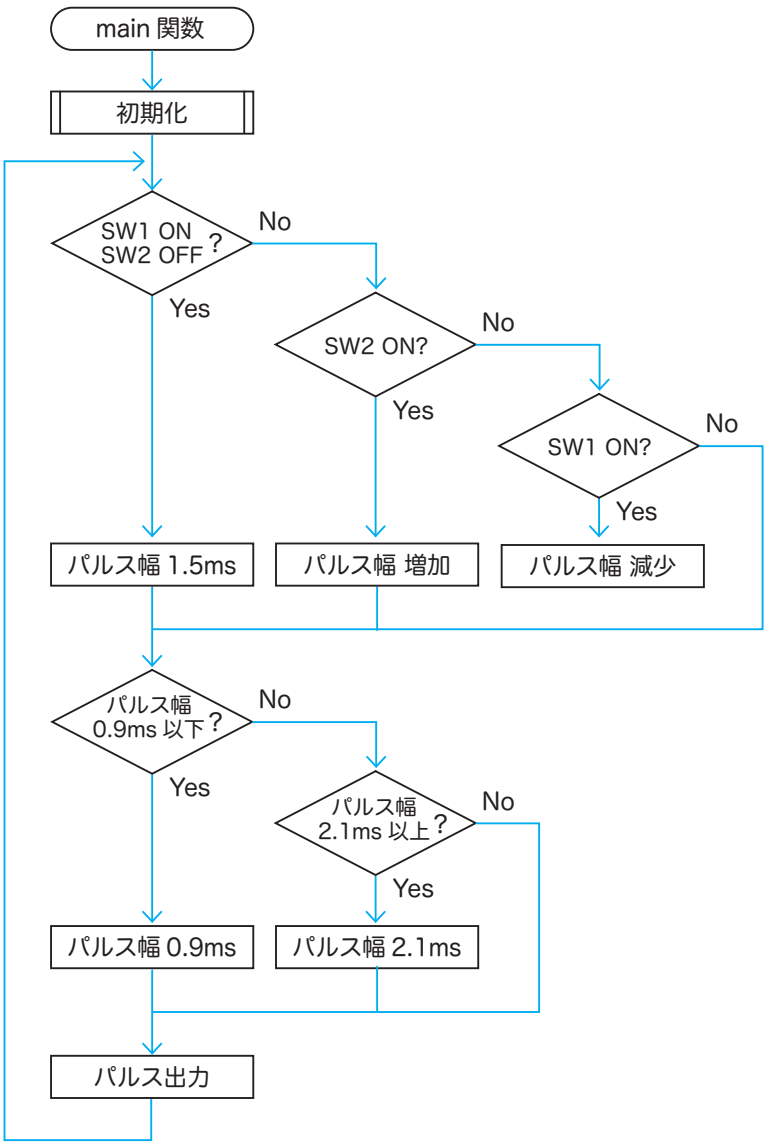



図 13-7　フローチャート

プログラム例　13-2

```
/*
 *  main 関数
 */
int main(void)
{
    initIO();     // 初期化関数の呼び出し
    int h_ms = 9; // H の時間

    while (1)
    {
        // パルス幅を変化
        if (SW1_ON && SW2_ON)    // SW1,SW2 ともに ON
            h_ms = 15;         // H の時間を 1.5ms にする
        else if (SW1_ON)         // SW1 のみ ON
            h_ms++;          // H の時間を 0.1ms 長くする
        else if (SW2_ON)         // SW2 のみ ON
            h_ms--;          // H の時間を 0.1ms 短くする

        // H の時間が 0.9 ～ 2.1ms 以内になるよう補正
        if (h_ms < 9)
            h_ms = 9; // H の時間を 0.9ms にする
        else if (h_ms > 21)
            h_ms = 21; // H の時間を 2.1ms にする

        // サーボモータにパルスを送る
        PA.DR.BIT.B2 = 1;
        wait01Ms(h_ms); // 待ち時間関数の呼び出し（h_ms × 0.1[ms]）
        PA.DR.BIT.B2 = 0;
        wait01Ms(150); // 待ち時間関数の呼び出し（150[ms] 固定）
    }

    return 0;
}
```

本 STEP では、ソフトウェアタイマを使用して PWM 信号を発生していました。しかし、このようなプログラムでは、CPU が無駄な処理をしている時間が多く、タスクを効率よく処理することができません。

H8/3052F は、高機能なタイマ（ITU：16bit Integrated Timer Unit）を内蔵しています。実は、このタイマには CPU の動作とは独立に PWM 波形を生成する機能があるのです。詳しくは STEP22 で解説します。