試作検討 3

R/C サーボ, 状態遷移プログラム

創造設計第二 TA：草野 正巳, 杉浦 元將

平成 20 年 10 月 20 日, 30 日

1. はじめに

今回の試作検討では，マイコンを用いた R/C サーボの駆動方法と，状態遷移を利用したプログラムの組み方を学びます．また，今後, 実際にプログラムを書くときに有効なので, きちんと理解しましょう.

2. R/C サーボ

2.1 R/C サーボ 概要

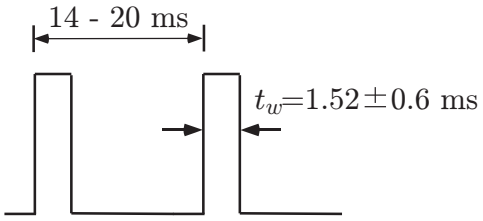

Fig. 1: R/C サーボ・モータの入力信号

創造設計第二では，マニピュレーション用にラジコン用サーボ・モータ (R/C サーボ・モータ) が A 類部品として用意されています.

R/C サーボは，小型 DC モータとポテンショメータ (可変抵抗器)，ギヤ，制御回路を小さなユニットにまとめたもので，入力信号のパルス幅に対応した角度を保つように内部でフィードバック制御を行なっています.

Fig. 1 のように制御パルスの周期は 14-20 ms で，パルス幅 (**Fig.** 1 中の t_w) が 1.52 ms のときにニュートラル (中立) 位置になり，1.52 ± 0.60 ms のときに，±60° となります．なお R/C サーボのハードウェア的な中立位置とパルス幅を 1.52 ms 与えたときの位置は異なりますので注意してください.

サーボの内部では駆動軸にポテンショメータが取り付けられており，ポテンショメータの角度に対応した幅のパルスが発生するようになっています．入力されたパルス幅とサーボ内部のパルス幅が異なっている場合, R/C サーボ内部のモータが動作し，二つのパルス幅が等しくなるまで駆動軸が回転します．この様な方式にすることで，サーボの駆動軸角度が外力によって変化した場合でもサーボの駆動力の範囲内で元の角度に戻ろうとする力を発生させることができるのです.

今回配布する R/C サーボ・モータの仕様は次の通りです.

- 双葉電子工業 (株) 製 S3003
- 寸法: 40.4 × 19.8 × 36 mm
- 重量: 37.2 g
- 動作電源: +4.8〜6 V
- 動作スピード: 0.23sec/60° (4.8 V 時)　0.19sec/60° (6.0 V 時)
- 出力トルク: 3.2kg·cm (4.8 V 時)　4.1kg·cm (6.0 V 時)

- 動作角度: ± 60° 程度[*1]

線の割り当ては，1 ピン (白): 制御信号，2 ピン (赤): 電源，3 ピン (黒): GND となっています．

2.2 マイコンを用いた R/C サーボの駆動方法

R/C サーボはマイコンの PWM 出力モードを用いて駆動させます．R/C サーボは PWM 信号の周期およびデューティ比を適切に調節する必要がありますので，タイマを 8bit モードで使用しなければなりません．

3. 課題 1 - R/C サーボと状態遷移

課題 1 では，マイコンの PWM 出力を使った R/C サーボの角度指定と，状態遷移の記述を行います．

3.1 課題 1.1

まず **Fig.** 2，**Fig.** 3 を参考にして Control Board と R/C サーボを接続して下さい．R/C サーボは Motor Driver Board の RCS0 に，白いケーブルが外側に来るように差し込んで下さい．また右下の HB0 のジャンパは，必ず上側に接続してください．

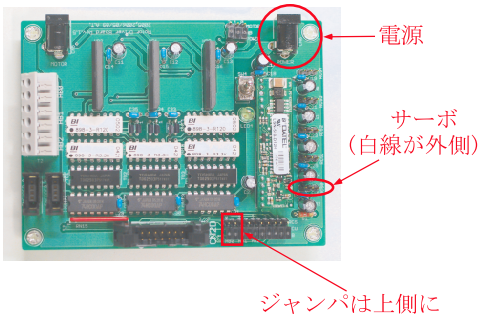

Fig. 2: R/C サーボ接続図

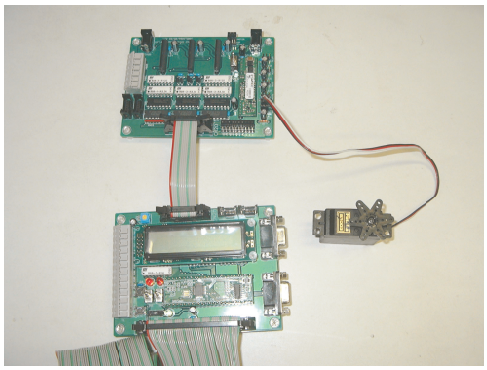

Fig. 3: マイコン-R/C サーボ接続写真

ホームページからプログラムをダウンロードして `shisaku04\kadai1` のプログラム全体を一読して，ハードウェアマニュアル 84 ページを参考にしながら次のことをヒントにして `RCServo.c` を完成させ，実際に R/C サーボが動作するかどうかを確認してください．また，それぞれのボタンを押すことで記述どおりに角度が変わることを確認してください．配布したプログラムの状態遷移は課題 1.2 中に記されています．

> **注意** 課題 1.1 のプログラムの穴埋めが終わった後はすぐにサーボを動作させず，必ず TA に答えの確認をしてもらった後にサーボを動作させてください．誤った値がレジスタに入力されると，サーボが破壊されるおそれがあります．

- High 幅 (“H”幅) の設定 (TA0 レジスタの上位番地の設定値，`ta0h=n` を入力)
  - マイコンのメインクロックが 20MHz で，カウントソースは f_{32} を使用しているので，$f_{32} = 20 \times 10^6/32$[Hz]
  - 1.52 ms でニュートラル，±0.6 ms で ±60°，では任意の角度 `deg`[°] を実現したいときは？変数 `deg` は関数 `RCServo()` の引数として与えられています．
  - 変数 `RCS_ROUND_COUNT` で TA0 レジスタの下位番地の設定値 `ta0l=m` が与えられています．
  - ハードウェアマニュアルの式は単位 [s] で与えられているので，これを n について解けばよい．

3.2 課題 1.2

配布したプログラムはボタンを押すことによって 4 つの状態を行き来しています．プログラムを参考にして，状態を 1 つ増やして 5 つにしてください．

[*1] 仕様では ±60° となっていますが，実際には約 ±90° 程度まで動作します．ただし，R/C サーボの回転角度が構造上の可動範囲を超えた場合，サーボ内部のギアが破損する可能性があるので注意してください．

配布したプログラムの状態遷移は以下のようになっています．

各モードでの R/C サーボの角度

[**初期状態** (Mode 0)]　0° ニュートラル (中立) 位置

[Mode 1]　30°

[Mode 2]　60°

[Mode 3]　-30°

モードの遷移条件

赤スイッチ押下 (SW_E)	モード i からモード $i+1$ へ モード 3 の時はモード 0 へ
白スイッチ押下 (SW_F)	モード i からモード $i-1$ へ モード 0 の時はモード 3 へ
黒スイッチ押下 (SW_G)	どのモードでもモード 0 へ

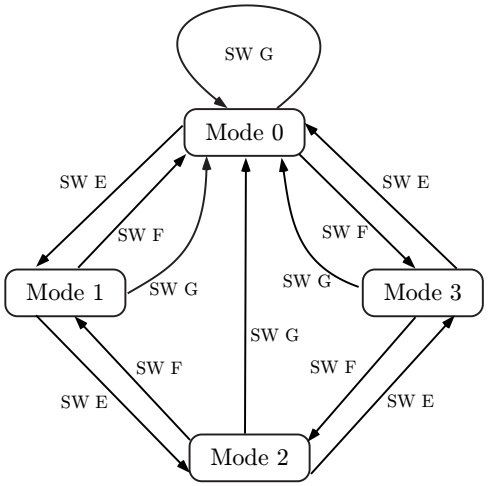

Fig. 4: プログラムに実装されている状態遷移図

これを以下のように状態を一つ増やしてください．変更点は赤で記しています．

各モードでの R/C サーボの角度

[**初期状態** (Mode 0)]　0° ニュートラル (中立) 位置

[Mode 1]　30°

[Mode 2]　60°

[Mode 3]　-30°

[Mode 4]　-60°

モードの遷移条件

赤スイッチ押下 (SW_E)	モード i からモード $i+1$ へ モード 4 の時はモード 0 へ
白スイッチ押下 (SW_F)	モード i からモード $i-1$ へ モード 0 の時はモード 4 へ
黒スイッチ押下 (SW_G)	どのモードでもモード 0 へ

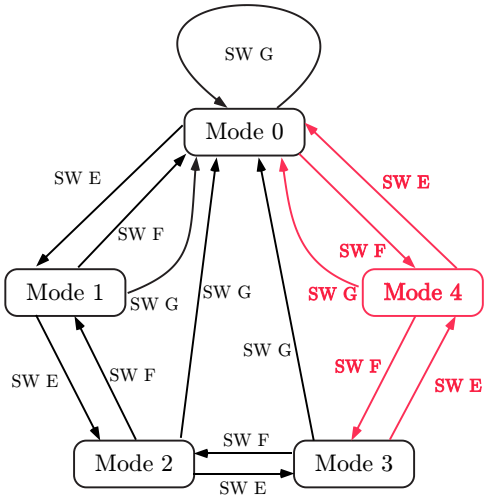

Fig. 5: 状態追加後の状態遷移図

3.3 プログラムの説明

課題 1 で使用するプログラムの主な関数の役割は以下の通りです.

- `main()`
  メイン関数．モードの切り替えを行っています.
- `RCServo(port, deg)`
  R/C サーボを駆動させる関数．中身は `RCServo.c` に記述されています．引数はそれぞれ `port` で出力ポート，`deg` で回転角度を与えます．配布したプログラムにはポート 0 以外で動作させるように記述されていませんので，他のポートを使いたいときは記述してみてください.

この課題で考えてほしいことは自分たちがマシンを作るときに状態遷移図を書き，それをリアルタイムプログラミングに基づいて実装してもらうことです.

> **注意** 最終報告書を書く段階になって，改めて状態遷移図を考えているようでは正常に動作するソフトウェアは期待できません.

3.4 余裕があったら

余裕があったら，この機会に R/C サーボや状態遷移についていろいろ試してみるとよいと思います（任意）. 例えば,

- R/C サーボの角度変化を遅くしてみる．一定周期で揺らしてみる
- 動作開始時に動作モードを選択できるようにしてみる
- スイッチを 3 回押すと状態が変わるようにしてみる
- A/D 変換の値やタイマ割り込みで状態遷移するとしたらどう記述するか？

など.

4. リアルタイムプログラミング

今回の試作検討レベルの単純な状態遷移ならともかく，本番でロボットを動かすくらいの複雑さになったときには，switch 文を 2 つに分け 1 つ目でそのモードでの動作を記述，2 つ目で状態遷移条件や遷移する瞬間に一度だけ行いたいことを記述すると見通しのつけやすいプログラムになります．

```
while(1){
  //1 つ目の switch 文でそのモードで行いたい動作を記述
  switch(mode){
    case 0:  //モード 0 の時に行うことを記述
      hoge();
      break;  //break は絶対に忘れずに

    case 1:  //モード 1 の時に行うことを記述
      hoge();
      break;  //break は絶対に忘れずに

      ⋮

    default:  //どのケースにも当てはまらない場合
      hoge();
      break;  //break は絶対に忘れずに
  }

  //2 つ目の switch 文で次のモードへの状態遷移条件や遷移する瞬間に行いたいことを記述
  switch(mode){
    case 0:
      if() mode = hoge;
      break;  //break は絶対に忘れずに

    case 1:
      if() mode = hoge;
      break;  //break は絶対に忘れずに

      ⋮

    default:  //どのケースにも当てはまらない場合
      hoge();
      break;  //break は絶対に忘れずに
  }
  //状態に関わらず行うことを記述
  hoge();
  //sleep 以外に for や while(1) など長い時間を待つような場所を作ってはだめ
  //リアルタイムプログラミングの基本
  sleep();
}
```