C7 タイマ割り込み同期型リアルタイム制御プログラム

C7.1 はじめに

今回は，自律ロボットや機械システムなど**自律エージェント**のふるまいを表現する原理と，これをマイクロコントローラを使って実現するためのプログラムの基本構造について実習する．さらに後半は，この基本構造を応用して，各自のチャレンジテーマ・マシンのプログラム設計に入る．

今日の目標

1. 自律エージェントのモデルとして有限状態マシン：FSM の概念を理解する．
2. モード遷移図を使って自律ロボットのプログラムを設計できるようになる．
3. タイマ割り込みに同期して入出力処理，モード決定，行動計算を実現する，単純化したリアルタイム制御プログラムの構造を理解し，自律ロボットのプログラムを書けるようになる．
4. チャレンジテーマ・マシンのプログラムの基本設計を行なう．

C7.2 自律エージェントと FSM

知識 C7.1 自律エージェント

外界の状況を認識し，自分で判断して行動する実体を**自律エージェント**と呼ぶ．たとえば，動物や人は天然の，ロボットや自動機械は人工の自律エージェントといえる．自律エージェントには，外界からの情報を得る**入力**，自分自身の行動や外部への作用を生じる**出力**が必要である．前者を実現する手段がセンサ，後者を実現する手段がアクチュエータなどである．自律エージェントが状況や経過に応じてふるまうための仕組みとして，エージェント自身が**状態**あるいは**モード**という属性を持つと考えると理解しやすい．さらに，状況によってモードを変える**モード決定**，モードに応じて入力から出力を決定する**行動計算**という機能を実装すれば，そのまま自律エージェントのプログラムの枠組みができる．

知識 C7.2 有限状態機械：FSM

このような自律エージェントを，有限個の状態を条件によって遷移するメカニズムとしてモデル化したものを**有限状態機械：Finite State Machine** と呼

ぶ．FSM は，たとえば図 C7.1 のような**状態遷移図**と呼ばれるグラフで表現できる．円が 1 つの状態（モード），矢印が遷移を表わす．矢印には，その遷移が起こる条件を付記する．自律ロボットなどの自律エージェントのプログラム設

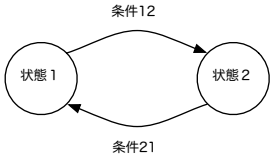

図 C7.1: 簡単な FSM の状態遷移図の例

計の第一歩は，基本的な行動を FSM で表現することである．

例 C7.1 チャレンジテーマ・バッターマシンの FSM 表現

モード遷移の具体例として，チャレンジテーマのバッターマシンの例を考えてみよう．球の検出，球速とコースの計測，計測結果に応じた打撃（タイミング，運動姿勢）などの主要機能実行にモードを割り当てて，スイッチや球の移動に応じて適宜実行するように遷移を表わすと，図 C7.2 のようになる．

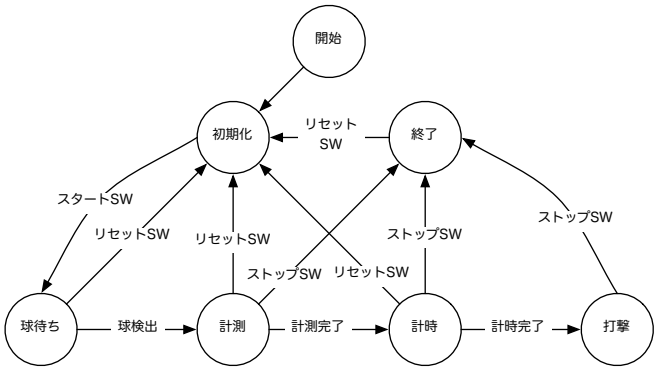

図 C7.2: バッターマシンの状態遷移図の例

C7.3 タイマ割り込み同期型モード遷移プログラム

知識 C7.3 タイマ割り込み同期

自律エージェントのように外界と相互作用するシステムでは，プログラムの実行時に時間や時刻を把握し，正確なタイミングで動作させる必要がある．たとえば，図 C7.2 のバッターマシンの例では，球速の計測や，球の検出からバッティングまでのタイミング設定，バッターの運動制御を有効に行なうには，時間や時刻を正確に検出・計測できなければならない．

このような時間計測・時刻同期を MCU で簡単に実現する手段としては，次のようにタイマ割り込みを利用することが有効である．

- タイマの 1 つを自由計数モードに設定して，一定数カウントするごとに割り込みを発生させる．
- タイマ割り込みによって実行されるタイマ割り込み関数の中で，一定時刻ごとに速やかに行なうべき処理を実行する．
- 外部事象の時間計測も，タイマ割り込み周期を単位としてカウントすると処理しやすい．

この方法は，ちょうど人が時計のアラームを設定して，それに合わせて行動するのに似ている．このアラームの間隔を十分短く設定しておけば，マシンは外部から見てタイミングよく機敏に動作することができる．また，タイマ割り込みは，プログラムにとって時間という物理量のセンサの役割を果たす．

図 C7.1，図 C7.2 のようなモード遷移図で表わされる自律エージェントを時間同期で制御するには，入出力処理，モード決定，行動計算を，上記のタイマ割り込みに同期して行なえばよい．このような構造のプログラムを，**タイマ割り込み同期型モード遷移プログラム**と呼ぶことにする．この仕組みは，自律エージェントのプログラムの基本構造となる．

知識 C7.4 簡易版タイマ割り込み同期型モード遷移プログラムの構造

今回扱うタイマ割り込み同期型モード遷移プログラムのフローチャートを図 C7.3 に示す．プログラムが実行されると，メイン関数の中でさまざまな初期化処理のあとにタイマ割り込みを設定・許可する．以後，MCU 内でタイマ割り込みが発生するごとに，最小限のタイムラグでタイマ割り込み関数が自動的に実行される．図 C7.3 では，タイマ割り込み関数に入ると，まず 1 つ前のタイマ割り込みの際に計算しておいたデータを出力し，現在の情報を入力したあと，モード決定を行ない，決定したモードに応じたマシンの行動計算を行なって次回の出力データを用意し，メインに戻る．すなわち，出力，入力，モード決定，行動計算のすべての制御演算を，タイマ割り込み関数の中で実行している（下記**注意**参照）．

このうち，入出力処理はエージェントと外部との直接の相互作用なので，タイマ割り込みとの時間ずれができるだけ少ないことが望ましい．モード決定は，FSM の構造を忠実に実装した処理である．行動計算は，各モードごとのエージェントのふるまいを実現するもので，具体的には，最新の入力から次の出力を決定する演算処理をモード決定直後に実装する．これらは，モードを表わすための**モード変数**を用いて，これに対する case または switch 文などによる場合分けによって構成する．そして，具体的な判定や計算は，最新の入力データと過去の入出力データなどを参照して行なう．

メイン関数では，ディスプレイへの表示や操作インターフェイス，シリアル通信など，タイマ割り込みに同期する必要のない処理や，タイマ割り込みの周期より時間のかかる処理などを行なう．これらの実行中にタイマ割り込みが発生しても，制御演算が確実に 1 回だけ実行されるようにしている点の本構造の特徴がある．ただし前提として，

- CPU が実行すべき仕事の中で，制御演算がもっとも高い頻度で，かつ定常的に行なわれる
- 操作インターフェイスなど，制御演算の周期よりも時間がかかる，あるいは非同期の仕事が存在する

という状況を想定している．

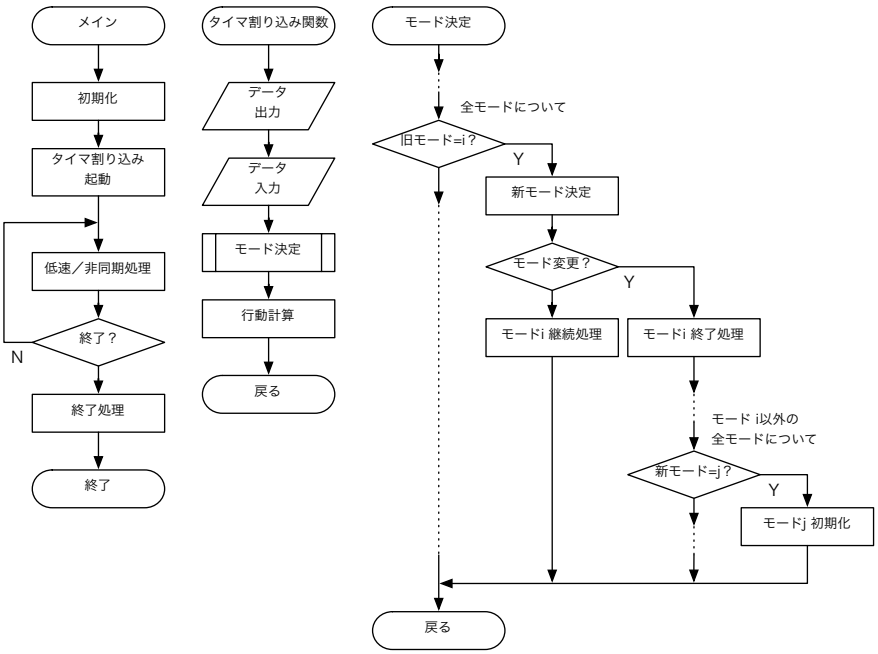

図 C7.3: 簡易版タイマ割り込み同期型モード遷移プログラムの構造

注意 図 C7.3 は，一般的には決してよい作法のプログラム構造とは言えないので注意してほしい．本来，このような割り込み処理関数内で行なう処理は，割り込み直後に行なわなければならない必要最小限のものだけに限定すべきである．今回の場合，入出力処理は割り込み処理関数内で行なうのが妥当だが，モード決定，行動計算などのモード処理はメインの中で処理する方が望ましい．それには，遅い，あるいはタイマに同期しない処理を実行している途中に，タイマ割り込みが発生するごとに確実にモード処理を実行する必要がある．しかし，このような性質の違うタスクを管理するためには，通常 OS（オペレーションシステム）が必要となる．ここでは OS を使用しない簡単な実装を想定している

ため，メイン関数の中でタスクの管理を見通しよく実現することは困難である．そこで，入出力処理とモード処理を合わせた制御演算全体をタイマ割り込みごとに確実に実行できること，遅い処理はこれを気にせずに実装できることを重視して，ここでは割り込み関数にすべての制御演算を含めている．

C7.4 タイマ割り込み同期型モード遷移プログラムの実習

上のプログラム構造に基づいて，簡単化したチャレンジテーマ・マシンのプログラムで下記の手順で実現し，実験を行なう．さらに，実現したプログラムを利用して自分のマシンのプログラムを設計しよう．ただし，具体的な作業については，実習資料および実習指導者に従うこと．

実習 C7.1 基本的な構造の実装と実験

図 C7.2 のうち，球待ちモード，球検出，他のモードにおけるスイッチによる遷移論理をタイマ割り込み関数内のモード決定部分に実装する．このほかのモードでの行動計算はなし，または適切なダミーとする．LED 等で，モードを表示できるようにしておくまた，モード遷移とは直接無関係だが，マシンの操作や実験に必要なインターフェイスをメイン関数側に実装する．実装が終わったら，実行して球の検出やスイッチによりモード遷移が行なわれることを確認する．

実習 C7.2 計測モードの実装と実験

PSD ユニットを使って球速・コースを計測する例を実装し，実験によって自分のマシンのプログラムのためのデータを得る．出力波形を MCU 取り込むとどのようなデータが得られるのか，PSD ユニットの配置を工夫しながら検討する．

各自の方法については自由時間，および C8 や S コースでさらに検討することとし，今回は共通のサンプルを利用する．ただし，自分のプログラムへのヒントとして以下をあげておく：

- 時間計測はメインループ数のカウントでおこなう
  - 割り込み周期が計測の時間単位となる（今回は RC サーボの PWM 周期と同じ 10 − 20ms とするが変更可能）．
  - カウント処理は，フローチャートにおける「モード i 継続処理」に含めればよい．
- 球速計測は，例えば次のような方法が考えられる．

  - PSD 1つで，出力波形の幅（面積を高さの平均で割ってもいい）から判断.
  - PSD 2つをボールの経路に沿って並べ，そのピークの時間差から判断.

- コース判定は，例えば次のような方法が考えられる.
  - PSD 1つで，出力波形の高さ（平均高さを取るか，面積を幅で割ってもいい）から判断
  - PSD 2つを，ストライク判定ゾーンの両脇に向かいあわせて置き，両出力の差から判断
  - PSD 2つをボールの経路に沿って並べ，そのピーク値（または積分値）の差から判断

なお，効率的な実験のため，終了モード時，あるいはメインループ側で計測結果を表示できるようにするとよい.

実習 C7.3 計時モードの実装と実験

球速・コースに応じた打撃開始タイミングを得る仕組みを実現する．計測モード以降のタイマ割り込みの回数を，対応するモード遷移計算の中でカウントダウンし，その値を打撃モードへの遷移条件とする．球速・コースに考慮したタイミングで打撃モードに遷移できるように，カウントダウンの初期値を計測モードへの初期化処理で設定する.

実習 C7.4 打撃モードの実装と実験

球速・コースの計測結果に応じた打撃運動計算を実装できるようにする．各自のマシンに違いがあるので，今回はどのように運動パターンを与えるかだけ扱う．多自由度のマシンについては，どうすればいいか自分で考える．パラメータを実験で簡単に調整できるようにするのが重要.

実習 C7.5 終了モードの実装と全体実験

終了モードに入れるべき処理を考え，実装する．たとえば，マシンの姿勢を戻すなど，次の打撃へ備える処理，マシンの撤収を容易にする処理．実験やデバッグを容易にする球速・コースの計測結果などの記録・表示機能.

実習 C7.6 自分のマシンのためのプログラミング準備

残った時間は，実現したプログラムを利用して，自分のマシンのプログラムを考えるための実験をしよう．センサーマシンの配置，球のどんな情報をどう計測するかなどを検討しよう.

C7.5 まとめ

今回の内容を整理すると，自律ロボットなど自律エージェントのプログラムの設計の流れは以下のとおり．

1. 制御周期に同期して実現すべきエージェントの全ふるまいを FSM として表現する．
2. FSM の状態（エージェントのモード）のうち，制御周期に同期して行なうべき部分をタイマ割り込み関数内のモード決定部分に実装する．
3. 各モードに応じたふるまいをタイマ割り込み関数内の行動計算として実装する．
4. 制御周期に同期して行なうべき入出力処理を，タイマ割り込み関数の最初に出力，入力の順に実装する．
5. タイマ割り込み外に行なうべき低速あるいは非同期の処理をメイン関数に実装する．なお，こちら側の処理を別の非同期あるいは低速の FSM とする場合もある．

なお，今回の実装はタイマ割り込み関数内にすべての制御演算を実装するという点で変則的なものである．これを改善するにはリアルタイム OS を搭載して，入出力処理以外の制御演算は，タイマ割り込みごとに実行されるが割り込み関数とは独立したタスクとして実装すべきである．