14 ４軸アザラシ型ロボットを動かす

14.1　４軸アザラシ型ロボット

では、いよいよ多関節歩行ロボットを作ってみましょう。

いきなり人型ロボットを作ってみたいところですが、すでに書いたように、人型ロボットはサーボモータを 20 個近く必要としますし (図 14.1 参照)、モータ自体の性能も重要になってきます。サーボモータの数が多いと、それらを動かすだけでも大変です。モーション (動き) を作る際も、倒れないように、かなり正確な制御が必要になってきます。

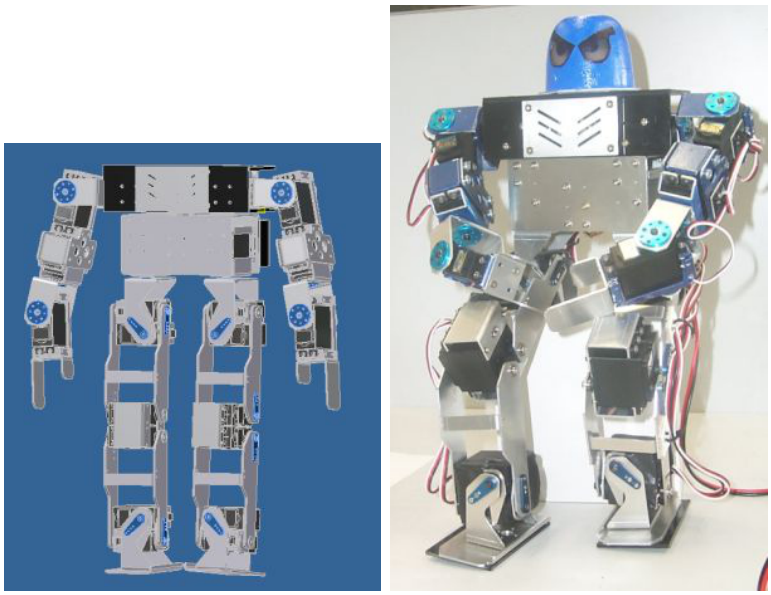

図 14.1　人型ロボット

多関節歩行ロボットの原理を理解したいということでしたら、サーボモータの数はできるだけ少ないほうが、基板を作るにもプログラムを作るにも簡単です。

サーボモータの数が最も少ない多関節歩行ロボットは、1 本の脚にモータ 2 個を使い、2 本の前脚で移

動する「アザラシ型」のロボットでしょう (図 14.2 参照)。

このロボットならば、サーボモータの数が少ないだけでなく、最初から胴体が接地しているため、倒れる心配がありません。

ここでは、このアザラシ型ロボットを作って、動かしてみましょう。

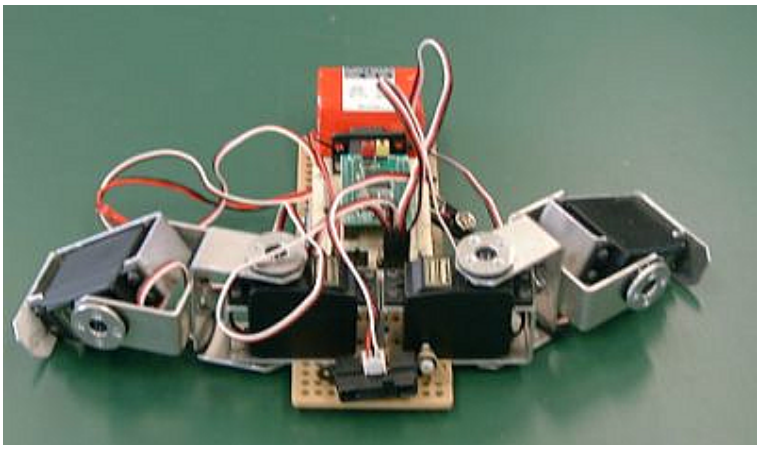

図 14.2　アザラシ型ロボット (ごまちゃん)

なお、この後で紹介する 8 軸亀型ロボットは、胴体の後に同様の脚を 2 本追加することで作成できます。

図 14.2 のロボットは、安価なサーボモータ Futaba S3003 を使い、それらをつなぐブラケットは自作しました。

これらは後ほど 8 軸亀型ロボットで用いる、近藤科学株式会社の KRS788HV(図 14.3) とサーボアーム 700A(図 14.4) で構成することもできます。PWM で制御するサーボモータでしたら、ほかの製品でも問題ありません。

サーボアームもそろっているサーボモータは、ヴイストン株式会社、有限会社浅草ギ研、ミニスタジオ有限会社など、いくつかの会社から販売されています。

好みに応じて選択してください。

図 14.3　KRS788HV

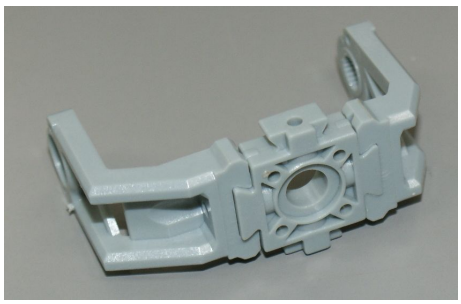

図 14.4　サーボアーム 700A

胴体は株式会社タミヤのユニバーサルプレートを使うと便利です。

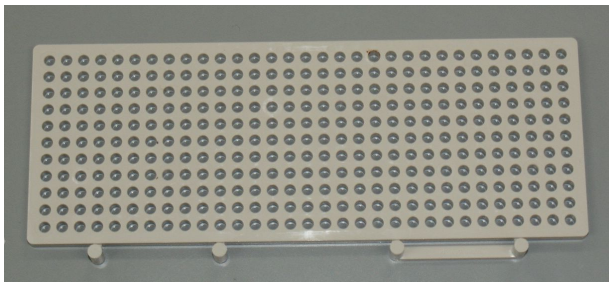

図 14.5　ユニバーサルプレート

では、モータを図 14.6 のようにつないでください。P.370 の図 14.2 も参照してください。

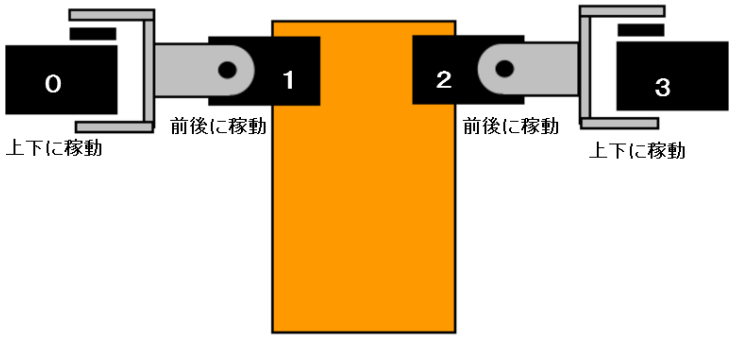

図 14.6　サーボモータの配置

14.2　ロボット用マイコン基板を作る

では、ロボット用にマイコン基板を作りましょう。ここまでサンハヤト株式会社のマイコンボード MB-H8A と書込み・拡張 I/O ボード MB-RS10 を利用してきました。

しかし、書込み・拡張 I/O ボードを自作しようとすると、マイコンボード MB-H8A には三端子レギュレータも RS232C 用のレベル変換 IC も乗っていません。これらの回路も実装しなくてはならないわけです。

一方、株式会社秋月通称の AKI-H8/3694F(QFP) タイニーマイコンモジュールには、これらの回路が実装されていますし、32.768kHZ のサブクリスタルも実装されています。しかも、こちらのほうが安価です。

今回はこのマイコンボードを利用することにしましょう。サンハヤト株式会社のマイコンボード MB-H8A を使う場合には、電源用回路と RS232C 用のレベル変換 IC 回路を適宜付け加えてください。

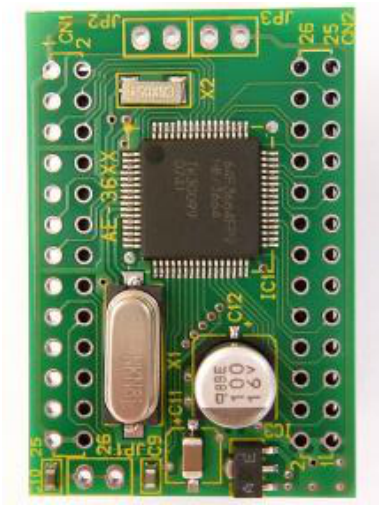

図 14.7　AKI-H8/3694F

14.2.1 回路図

今回はソフト的に PWM を割り振ることにします。回路図は図 14.8 のとおりです。

8 軸ロボットを作る際に、TC4051B を用いて回路的に割り振りをしますので、そちらのボードを作成しても結構です (P.387 図 15.5 参照)。

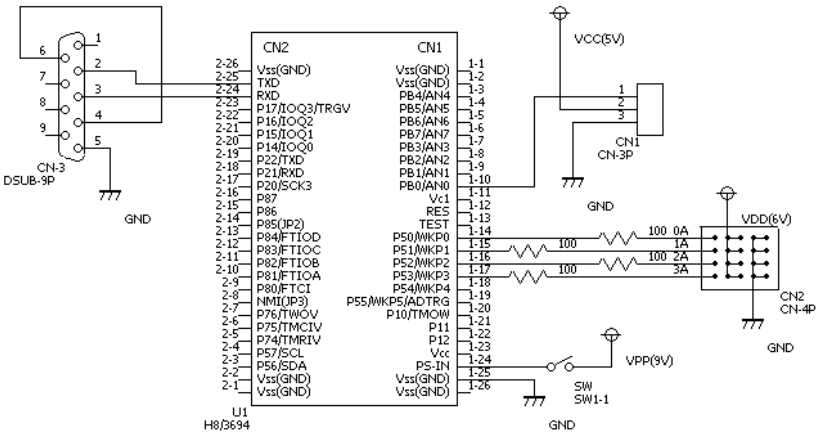

図 14.8　4 軸歩行ロボット用書込み・拡張 I/O ボード回路図

CN1 には後ほど PSD センサを接続して、障害物を回避するようにプログラムしてみたいと思います。

マイコンに利用する電池は 6P 電池 (9V) です。CN1 の 2 番ピンはマイコンボードの VCC(CN1-23) に

つないでください。

モータの電源 (6V) は、マイコンとは別電源にしてください。なお、ここではサーボモータは Futaba S3003 を用いているので、6V にしましたが、利用するサーボモータにあわせて電源を用意してください。

KRS788HV の場合には、9V〜12V です (6V でも動くようですが、動作は遅いようです)。

14.2.2 書込み・拡張 I/O ボードを作る

では、ユニバーサル基板に必要な回路を作ってみましょう。

以下に、実態配線図を示しますので、参考にしてください。

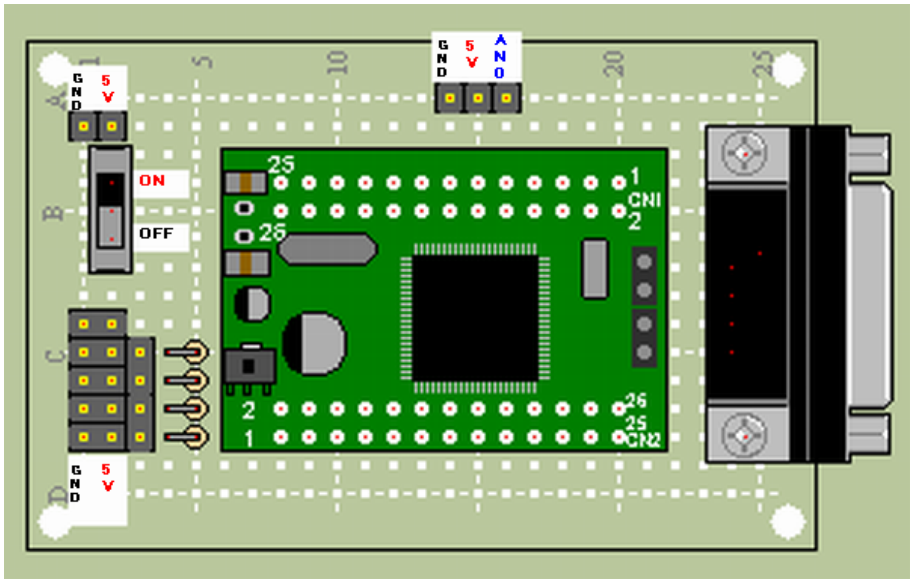

図 14.9　4 軸歩行ロボット用書込み・拡張 I/O ボード実体配線図 (表)

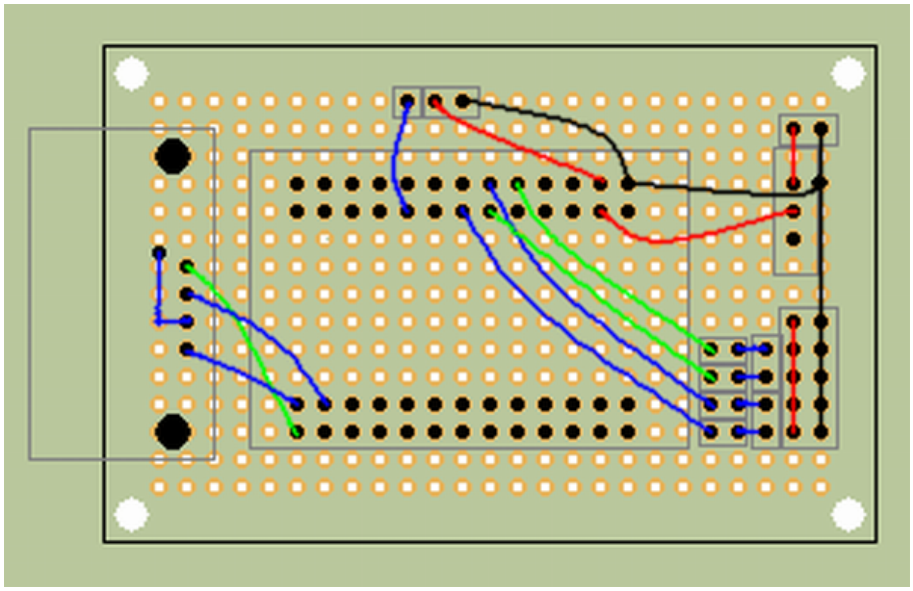

図 14.10　4 軸歩行ロボット用書込み・拡張 I/O ボード実体配線図 (裏)

ここでは、基板取りつけ用 Dsub コネクタがそのままでは基板に取り付けられないので、1 番ピンと 7 番ピン以降を切り取りました。

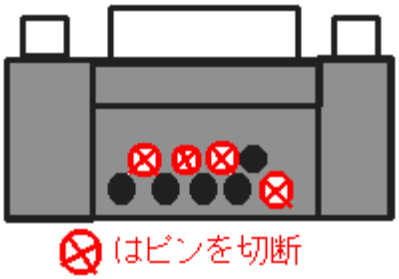

図 14.11　基板取りつけ用 Dsub コネクタ (メス) のピンを切る

なお、株式会社秋月通称では、DSUB コネクタ DIP 化キット (メス) も売っていますので、これを利用すれば、ピンを切る必要はありません。

14.3　モーション (動き) を考える

4 軸ロボットの動きを作りましょう。前脚が 2 本だけですので、あまり多くの動きは作れません。

ここでは、前進 2 パターン、右旋回、左旋回の 4 パターンを考えてみました。後退は胴体が床に引っかかってうまくいかない場合があるので、ここでは作らないことにしました。

14.3.1 前進 (クロール形)

まずは、左右の前脚を交互に前に出して前進させることを考えましょう。ここでは、クロール形の前進と呼んでおきます。

クロール形の前進は、以下の図 14.12 のように 2 本の前足を動かすことで実現できます。

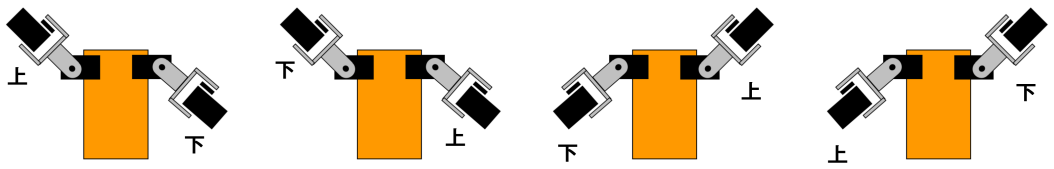

図 14.12　前進 (クロール形) 上から見た図 (画面上が前)

ここで、サーボモータを上にするとは、以下の図 14.13 のように、前脚の先を水平に上げることです。

図 14.13　サーボモータを上にしたところ 前から見た図

また、サーボモータを下にするとは、以下の図 14.14 のように、前脚の先を下げることです。

図 14.14　サーボモータを下にしたところ 前から見た図

したがって、図 14.12 は以下のような動作になります。

- 左前脚を上げ、前に出す。右前脚は下にし、後にさげる。
- 左前脚を下にし、右前脚は上げる。
- 左前脚を後ろにさげ、右前脚は前に出す。
- 左前脚を上げ、右前脚はさげる。

この動作を繰り返すことで、前進することができるわけです。

14.3.2 前進 (バタフライ形)

次に、左右の前脚を同時に前に出して前進させることを考えましょう。ここでは、バタフライ形の前進と呼んでおきます。

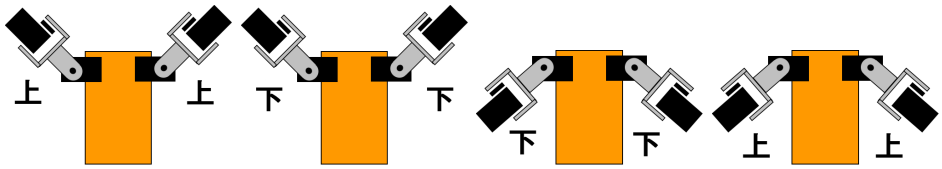

図 14.15　前進 (バタフライ形) 上から見た図 (画面上が前)

図 14.15 は以下のような動作になります。

- 両脚を上げ、前に出す。
- 両脚を下にする。
- 両脚を後ろにさげる。
- 両脚を上げる。

14.3.3 右旋回

方向転換として、右旋回を作ります。

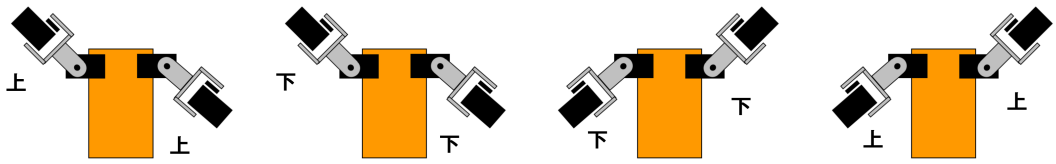

図 14.16　右旋回 上から見た図 (画面上が前)

- 左前脚を上げ、前に出す。右前脚は上げ、後にさげる。
- 両脚を下にする。
- 左前脚を後ろにさげ、右前脚は前に出す。
- 両脚を上にする。

14.3.4 左旋回

左旋回も作りましょう。左旋回は、右旋回を (右図から左図へと) 逆に実行していけば実現できます。

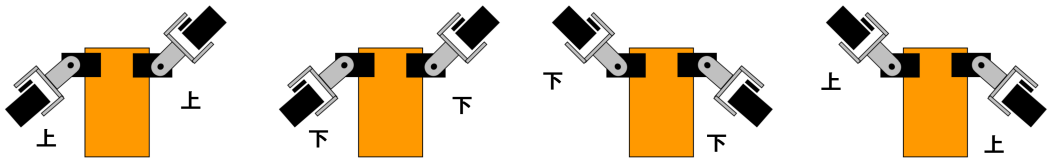

図 14.17　左旋回 上から見た図 (画面上が前)

- 左前脚を後ろにさげ、上にする。右前脚は前に出して上にあげる。
- 両脚を下にする。
- 左前脚を前に出し、右前脚は後ろにさげる。
- 両脚を上にする。

14.4　モーション (動き) を作る

それでは、実際に動きをプログラムしてみましょう。P.368 の課題 13.1.4 で作ったプルグラムを使って、各モーションのデータを作ってください。

モーションデータを作る

P.368 の課題 13.1.4 で作ったプルグラムを使って、各モーションのデータを作りましょう。

サーボモータの種類や、接続の仕方によってデータは多少異なります。以下のサンプルプログラムは、著者の 4 軸ロボット「ごまちゃん」の場合のデータです。各自自分のロボットのデータは、実際に動かして決めてください。

プロジェクトを 10_01_GOMA という名前で作り、intprg.c の INT_TimerW をコメントアウトしてください。

intprg.c の INT_TimerW をコメントアウト

```
// vector 21 Timer W
//__interrupt(vect=21) void INT_TimerW(void) {/* sleep(); */}
// vector 22 Timer V
__interrupt(vect=22) void INT_TimerV(void) {/* sleep(); */}
```

以下は、あらかじめ作っておいたモーションデータを再生するプログラムです。

モーションデータは、自分のロボット用のものに置き換えてください。

10_01_GOMA.c

```
/************************************************************************/
/*                                                                      */
/*  FILE        :10_01_GOMA.c                                           */
/*  DESCRIPTION :ソフト的に割り振り (4 軸ロボット「ごまちゃん」)        */
/*  CPU TYPE    :H8/3694F                                               */
/*  モータ 0 P50                                                        */
/*  モータ 1 P51                                                        */
/*  モータ 2 P52                                                        */
/*  モータ 3 P53                                                        */
/************************************************************************/

#include "iodefine.h"
#include<machine.h>

#define PWM_MAX_DUTY 12000-1 //2.4mS
#define PWM_MIN_DUTY  2000-1 //0.4mS
#define PWM_PERIOD   12500-1 //2.5mS

#define SRV_CH_NUM 4 /* モータ数 */
#define MOTION_NUM 4 /* モーション数 */

int srv_ch;     /* PWM の出力先 */
int srv_flg;    /* 20ms をカウント */
unsigned short pwm_duty[SRV_CH_NUM]; /* PWM のデューティのデータ */

/*******************************
   PWM 出力端子を初期化
*******************************/
void PwmInit(void)
{
  srv_ch=0;
  srv_flg=0;
  set_imask_ccr(1);
  TW.TMRW.BYTE=0x48;  /* FTIO 端子は使用しない */
  TW.TCRW.BYTE=0xA0;  /* 内部クロックの 1/4, 出力なし */
  TW.TIERW.BYTE=0x73; /* A,B の割り込みを可能にする */
  TW.TSRW.BYTE=0x70;  /* 割り込みフラグをクリア */
  TW.TCNT=0x0000;     /* TCN の初期化 */
  TW.GRA=PWM_PERIOD;  /* 周期 (2.5mS) */
  TW.GRB=7500;        /* 1.5mS */
  TW.TMRW.BIT.CTS=1;  /* TCNT カウンタスタート */
  set_imask_ccr(0);

  IO.PCR5=0xFF;
  IO.PDR5.BYTE=0x00;
}

/*********************************
  サーボ動作開始
 *********************************/
void PwmStart(void)
{
  TW.TMRW.BIT.CTS=1;
}

/*********************************
  サーボ動作停止
 *********************************/
void PwmStop(void)
{
  TW.TMRW.BIT.CTS=0;
}

/*********************************
  デューティの設定
 *********************************/
void PwmSetDuty(int ch, unsigned short duty)
{
  /* サーボチャンネルが範囲外なら無視 */
  if(ch >= SRV_CH_NUM){
    return ;
  }

  if(duty < PWM_MIN_DUTY){          /* デューティを最小値以上に限定 */
    duty = PWM_MIN_DUTY;
  }else if(duty > PWM_MAX_DUTY){    /* デューティを最大値以下に限定 */
    duty = PWM_MAX_DUTY;
  }
  /* デューティを設定 */
  pwm_duty[ch] = duty;
}
```

```
 82
 83  void main(void)
 84  {
 85    int i, j=0;
 86  /* バタフライ形前進のデータ */
 87  unsigned short pwm_data[MOTION_NUM][SRV_CH_NUM]=
 88      {{ 6000, 4500, 7500, 6000}, /* 上、前、前、上 */
 89       { 9500, 4500, 4500, 2500}, /* 下、前、前、下 */
 90       { 9500, 7500, 4500, 2500}, /* 下、後、後、下 */
 91       { 6000, 7500, 4500, 6000}};/* 上、後、後、上 */
 92
 93    PwmInit(); /* PWM の初期化 */
 94
 95    while(1){
 96      /* 次のモーションデータを設定 */
 97      for(i=0; i<SRV_CH_NUM; i++){
 98        PwmSetDuty(i, pwm_data[j][i]);
 99      }
100      while(srv_flg<100){
101        ;
102      }
103      srv_flg=0;
104      j++;
105      if(j>=MOTION_NUM){
106        j=0;
107      }
108    }
109  }
110
111  /*********************************
112    割り込み関数
113   *********************************/
114  __interrupt(vect=21) void INT_TimerW(void)
115  {
116    /* デューティのコンペアマッチ */
117    if(TW.TSRW.BIT.IMFB==1){
118      switch(srv_ch){
119        case 0:
120          IO.PDR5.BIT.B0=0;
121          break;
122        case 1:
123          IO.PDR5.BIT.B1=0;
124          break;
125        case 2:
126          IO.PDR5.BIT.B2=0;
127          break;
128        case 3:
129          IO.PDR5.BIT.B3=0;
130          break;
131      }
132      TW.TSRW.BIT.IMFB=0;
133    }
134    /* 2.5mS ごとのコンペアマッチ */
135    if(TW.TSRW.BIT.IMFA==1){
136      switch(srv_ch){
137        case 0:
138          IO.PDR5.BIT.B1=1;
139          break;
140        case 1:
141          IO.PDR5.BIT.B2=1;
142          break;
143        case 2:
144          IO.PDR5.BIT.B3=1;
145          break;
146        case 3:
147          IO.PDR5.BIT.B4=1;
148          break;
149      }
150        TW.TSRW.BIT.IMFA=0;
151        srv_ch++;
152        if(srv_ch>= SRV_CH_NUM){
153          srv_ch=0; /* モータ出力先を 0 に戻す */
154          srv_flg++;
155        }
156        TW.GRB=pwm_duty[srv_ch];
157    }
158  }
```

End Of List

実行結果

バタフライ形の前進をする。

課題 14.4.1 (提出) 他のモーションも実行してみる

上のサンプルでは、バタフライ形の前進を実行してみましたが、それ以外の、クロール形の前進、右旋回、左旋回も同様にして、データを作り実行してみましょう。

プロジェクト名：e10_01_GOMA

14.5 障害物を回避する

次は、PSD センサをつないで障害物回避をしてみましょう。

今回は PSD センサの出力を、増幅せずにそのまま使いました。前方に障害物があるかの測定だけなので、これでも充分だと思います。

障害物を見つけたら、右に旋回するようにしましょう。

では、プロジェクトを 10_02_GOMA という名前で作り、intprg.c の INT_TimerW をコメントアウトしてください。

intprg.c の INT_TimerW をコメントアウト

```
// vector 21 Timer W
//__interrupt(vect=21) void INT_TimerW(void) {/* sleep(); */}
// vector 22 Timer V
__interrupt(vect=22) void INT_TimerV(void) {/* sleep(); */}
```

10_02_GOMA.c

```
/************************************************************************/
/*                                                                      */
/*  FILE        :10_01_GOMA.c                                           */
/*  DESCRIPTION :ソフト的に割り振り (4 軸ロボット「ごまちゃん」)        */
/*  CPU TYPE    :H8/3694F                                               */
/*  モータ 0 P50                                                        */
/*  モータ 1 P51                                                        */
/*  モータ 2 P52                                                        */
/*  モータ 3 P53                                                        */
/************************************************************************/

#include "iodefine.h"
#include<machine.h>

#define PWM_MAX_DUTY 12000-1 //2.4mS
#define PWM_MIN_DUTY  2000-1 //0.4mS
#define PWM_PERIOD   12500-1 //2.5mS

#define SRV_CH_NUM 4 /* モータ数 */
#define MOTION_NUM 4 /* モーション数 */
#define HIGH 68 /* PSD の閾値 */

int srv_ch;     /* PWM の出力先 */
int srv_flg;    /* 20ms をカウント */
unsigned short pwm_duty[SRV_CH_NUM]; /* PWM のデューティのデータ */

/*******************************
   PWM 出力端子を初期化
*******************************/
void PwmInit(void)
{
  srv_ch=0;
```

```
33    srv_flg=0;
34    set_imask_ccr(1);
35    TW.TMRW.BYTE=0x48;  /* FTIO 端子は使用しない */
36    TW.TCRW.BYTE=0xA0;  /* 内部クロックの 1/4, 出力なし */
37    TW.TIERW.BYTE=0x73; /* A,B の割り込みを可能にする */
38    TW.TSRW.BYTE=0x70;  /* 割り込みフラグをクリア */
39    TW.TCNT=0x0000;     /* TCN の初期化 */
40    TW.GRA=PWM_PERIOD;  /* 周期 (2.5mS) */
41    TW.GRB=7500;        /* 1.5mS */
42    TW.TMRW.BIT.CTS=1;  /* TCNT カウンタスタート */
43    set_imask_ccr(0);
44
45    IO.PCR5=0xFF;
46    IO.PDR5.BYTE=0x00;
47  }
48
49  /*********************************
50    サーボ動作開始
51   *********************************/
52  void PwmStart(void)
53  {
54    TW.TMRW.BIT.CTS=1;
55  }
56
57  /*********************************
58    サーボ動作停止
59   *********************************/
60  void PwmStop(void)
61  {
62    TW.TMRW.BIT.CTS=0;
63  }
64
65  /*********************************
66    デューティの設定
67   *********************************/
68  void PwmSetDuty(int ch, unsigned short duty)
69  {
70    /* サーボチャンネルが範囲外なら無視 */
71    if(ch >= SRV_CH_NUM){
72      return ;
73    }
74
75    if(duty < PWM_MIN_DUTY){          /* デューティを最小値以上に限定 */
76      duty = PWM_MIN_DUTY;
77    }else if(duty > PWM_MAX_DUTY){    /* デューティを最大値以下に限定 */
78      duty = PWM_MAX_DUTY;
79    }
80    /* デューティを設定 */
81    pwm_duty[ch] = duty;
82  }
83
84  void main(void)
85  {
86    int i, j=0;
87    short ad_data;
88    unsigned short *pwm_duty; /* PWM のデューティのデータ */
89    /* バタフライ形前進のデータ */
90    unsigned short butter_data[MOTION_NUM][SRV_CH_NUM]=
91      {{ 6000, 4500, 7500, 6000}, /* 上、前、前、上 */
92       { 9500, 4500, 4500, 2500}, /* 下、前、前、下 */
93       { 9500, 7500, 4500, 2500}, /* 下、後、後、下 */
94       { 6000, 7500, 4500, 6000}};/* 上、後、後、上 */
95    /* 右旋回のデータ */
96    unsigned short right_data[MOTION_NUM][SRV_CH_NUM]=
97      {{ 6000, 4500, 4500, 6000}, /* 上、前、後、上 */
98       { 9500, 4500, 4500, 2500}, /* 下、前、後、下 */
99       { 9500, 7500, 7500, 2500}, /* 下、後、前、下 */
100      { 6000, 7500, 7500, 6000}};/* 上、前、後、上 */
101
102   PwmInit(); /* PWM の初期化 */
103   AdInit();  /* AD 変換の初期化 */
104
105   AdStart();
106   ad_data=AdRead();
107   if(ad_data>HIGH){
108     pwm_data=right_data[0];
109   }else{
110     /* バタフライ歩行の再生 */
111     pwm_data=butter_data[0];
112   }
113
114   while(1){
115     /* 次のモーションデータを設定 */
116     for(i=0; i<SRV_CH_NUM; i++){
117       PwmSetDuty(i, *(pwm_data+j*SRV_CH_NUM+i));
118     }
119     while(srv_flg<100){
120       ;
121     }
```

```
122       srv_flg=0;
123       j++;
124       if(j>=MOTION_NUM){
125         j=0;
126         AdStart();
127         ad_data=AdRead();
128         if(ad_data>HIGH){
129           pwm_data=right_data[0];
130         }else{
131           /* バタフライ歩行の再生 */
132           pwm_data=butter_data[0];
133         }
134       }
135     }
136   }
137
138   /*********************************
139     割り込み関数
140    *********************************/
141   __interrupt(vect=21) void INT_TimerW(void)
142   {
143     /* デューティのコンペアマッチ */
144     if(TW.TSRW.BIT.IMFB==1){
145       switch(srv_ch){
146         case 0:
147           IO.PDR5.BIT.B0=0;
148           break;
149         case 1:
150           IO.PDR5.BIT.B1=0;
151           break;
152         case 2:
153           IO.PDR5.BIT.B2=0;
154           break;
155         case 3:
156           IO.PDR5.BIT.B3=0;
157           break;
158       }
159       TW.TSRW.BIT.IMFB=0;
160     }
161     /* 2.5mS ごとのコンペアマッチ */
162     if(TW.TSRW.BIT.IMFA==1){
163       switch(srv_ch){
164         case 0:
165           IO.PDR5.BIT.B1=1;
166           break;
167         case 1:
168           IO.PDR5.BIT.B2=1;
169           break;
170         case 2:
171           IO.PDR5.BIT.B3=1;
172           break;
173         case 3:
174           IO.PDR5.BIT.B4=1;
175           break;
176       }
177         TW.TSRW.BIT.IMFA=0;
178         srv_ch++;
179         if(srv_ch>= SRV_CH_NUM){
180           srv_ch=0; /* モータ出力先を 0 に戻す */
181           srv_flg++;
182         }
183         TW.GRB=pwm_duty[srv_ch];
184     }
185   }
```

End Of List

実行結果

障害物を検知すると、右旋回して回避する。

課題 14.5.1 (提出) 左に避ける

前進はクロール形で、障害物回避は左旋回で行うようにプログラムを変更してください。

プロジェクト名：e10_02_GOMA