16 無線通信 (XBee) の利用

16.1 XBee

P.393 の「シリアルからコントロールする」では、シリアル通信を用いて有線でロボットを操縦しました。

ここでは、これを無線化してみましょう。

無線化にあたっては、シリアル通信をそのまま無線化できる XBee を利用することにします。

16.2 パソコンから無線で操縦する

XBee にはいくつかの種類がありますが、ここではシリーズ 2 の S2B を利用します (図 16.1)。

図 16.1 XBeeS2B

パソコンにつなぐには、XBee エクスプローラ USB を利用します (図 16.2)。

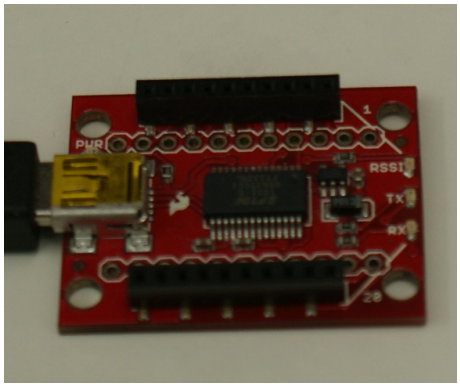

図 16.2　XBee エクスプローラ USB

マイコン側には、XBee エクスプローラ 5V マイコン用を用います (図 16.3)。

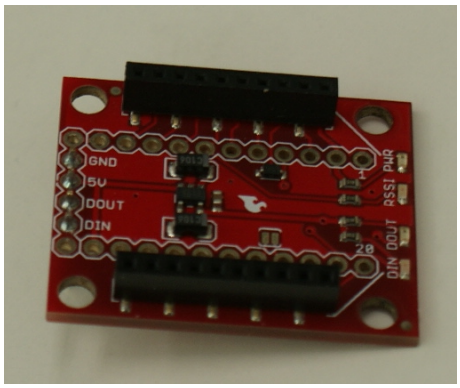

図 16.3　XBee エクスプローラ 5V マイコン用

16.2.1 XBee を設定する

ここでは、シリアル通信を XBee を用いて無線化することが目的ですので、トランスペアレント・モードというものを使います。そのためには、送信側・受信側の XBee の設定をする必要があります。

設定はソフトウェア X-CTU を用いて行います。

X-CTU は Digi International のホームページから無料でダウンロードできます。

では、これを用いて設定をしてみましょう。

以下のものを設定します。

表 16.1 トランスペアレント・モードの設定

XBee	1 台目 (0013A20040704B1B)	2 台目 (0013A200406C14AA)
Function Set	ZIGBEE ROUTER AT	ZIGBEE COORDINATOR AT
ID-PAN ID (1 以上で同じ値にする)	3	3
DH-Destination Address High	0013A200	0013A200
DL-Destination Address Low	406C14AA	40704B1B

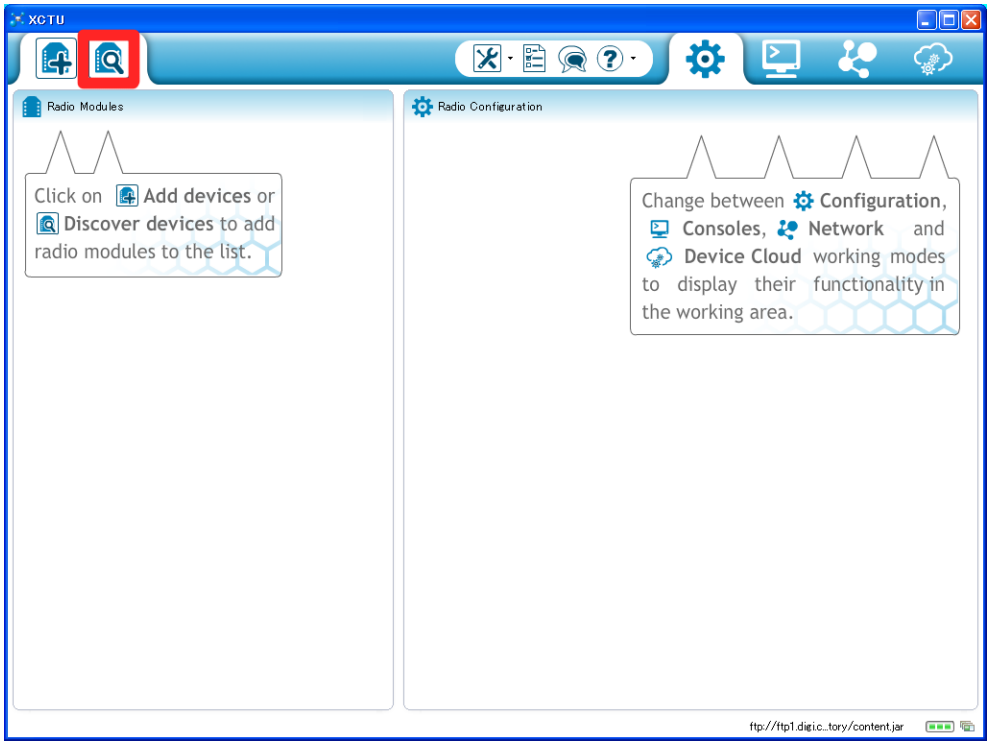

図 16.4 X-CTU の起動

X-CTU を移動すると、図 16.4 のような画面が表示されます。

「Discover devices」ボタンをクリックして、パソコンに接続されている XBee を探ししましょう。

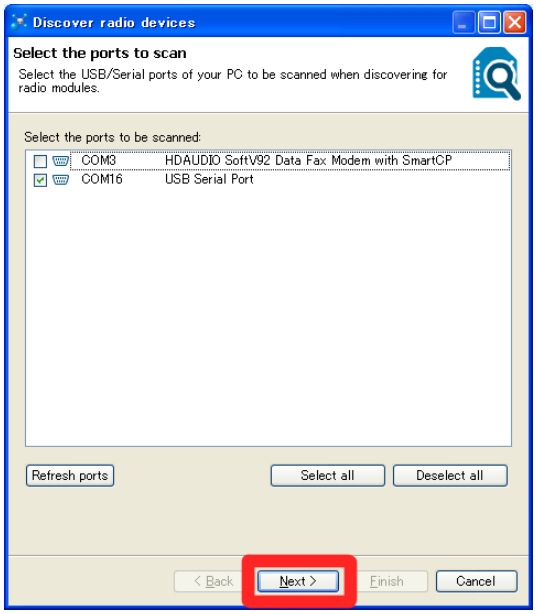

図 16.5　検索するポートの選択

まずは、検索するポートを指定します。XBee がつながっているポートにチェックをつけます (図 16.5)。

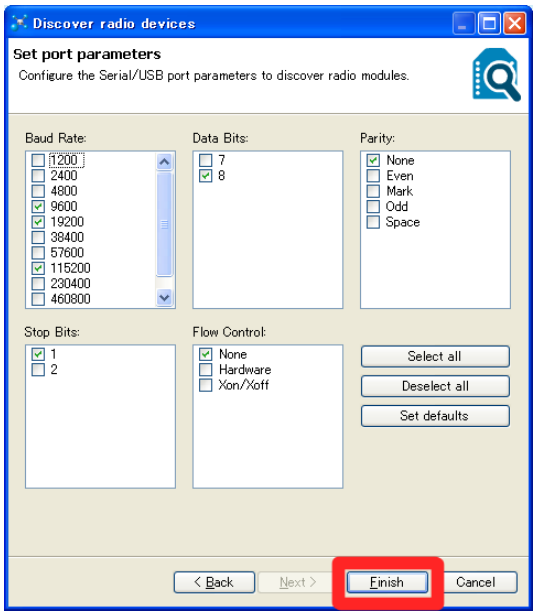

図 16.6　検索条件の選択

次に、検索するデバイスの条件を選択します (図 16.6)。「Select all」とすると、全ての条件で検索してくれますが、時間がかかります。

条件を選んだら、「Next」をクリックします。

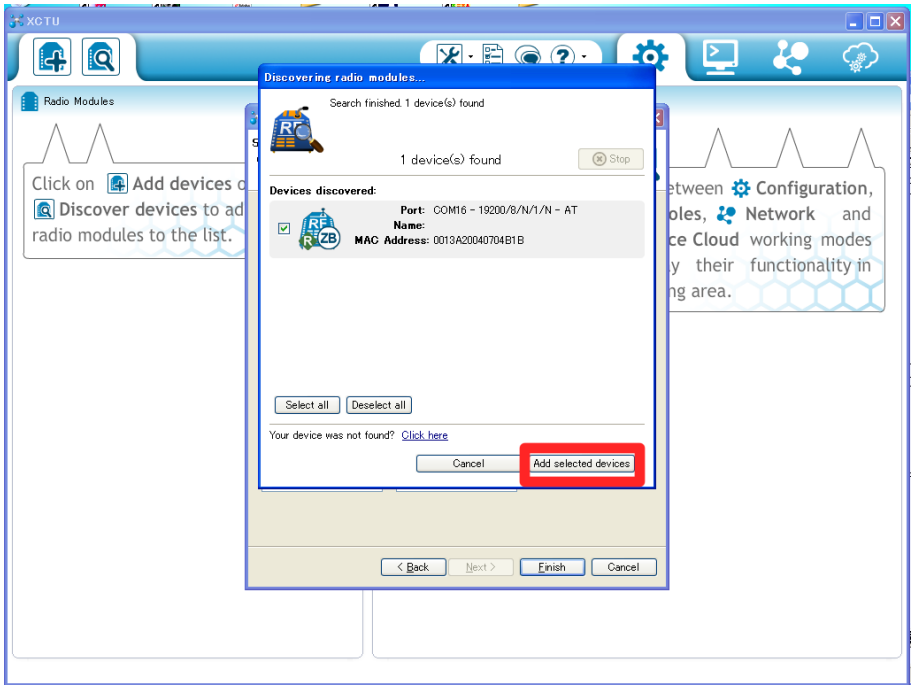

図 16.7　見つかった表示される

しばらく検索に時間がかかります。

検索が終わると、見つかったデバイスが表示されます (図 16.7)。

「Add select devices」をクリックします。

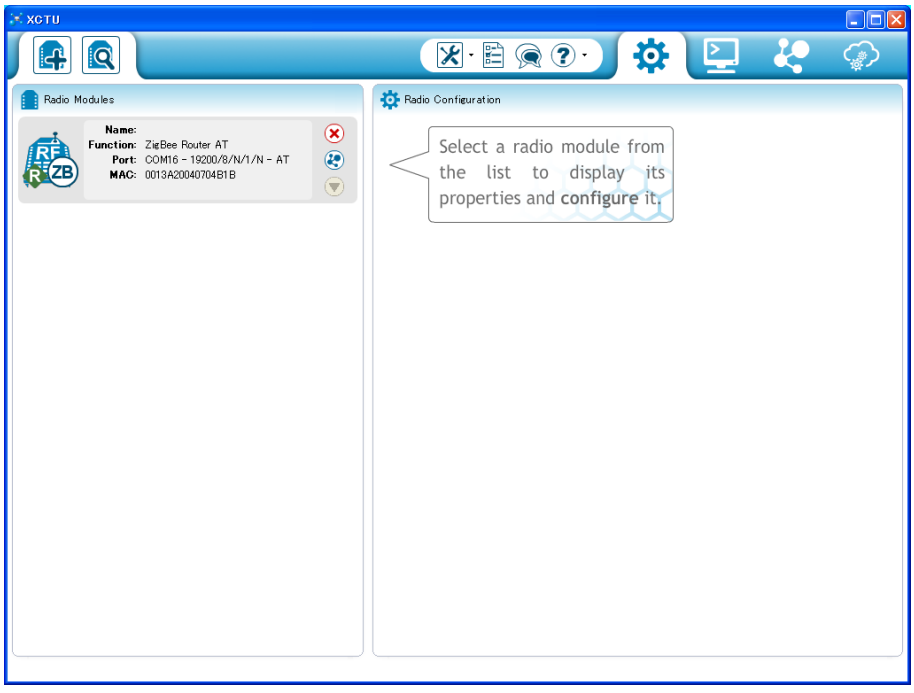

図 16.8　デバイスが追加される

デバイスが追加されるので、クリックします (図 16.8)。

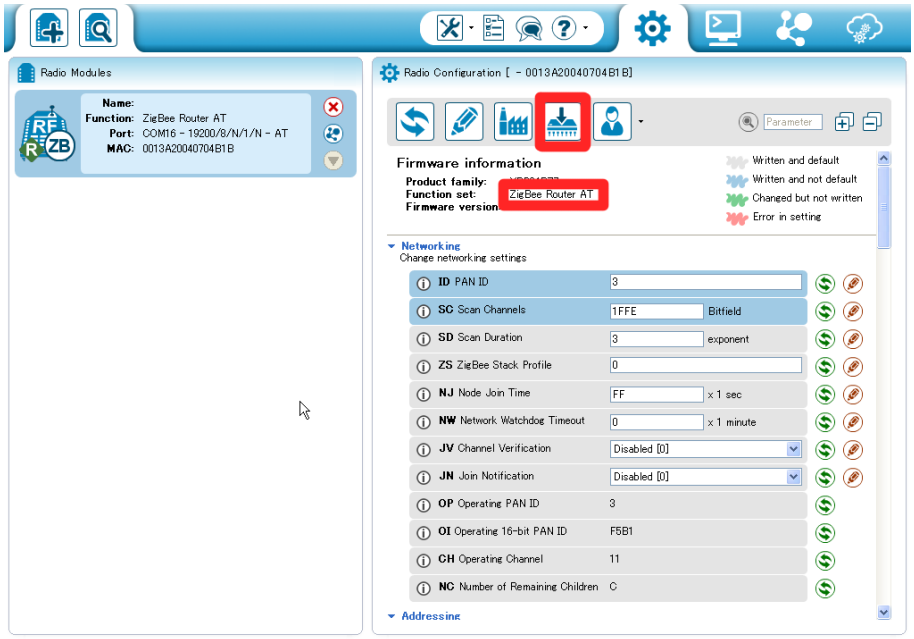

図 16.9　Function set の確認

デバイスの情報が表示されるので、「ID PAN ID」を必要に応じて変更します。

「Function set」を確認します。変更したい場合には、「Update firmware」ボタン  をクリックします (図 16.9)。

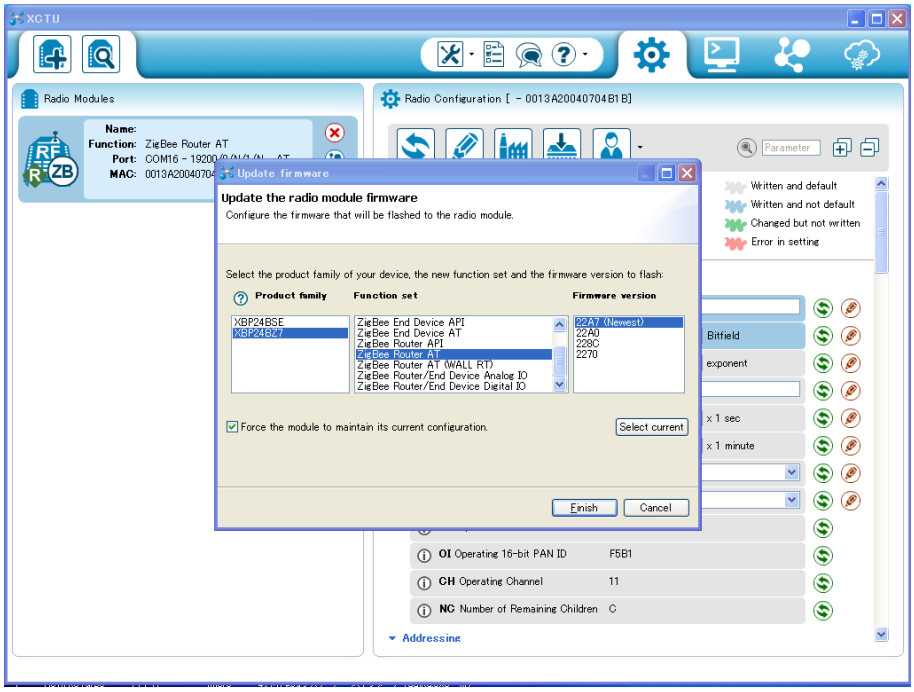

図 16.10　Function set の変更

「Update firmware」ウィンドウが表示されるので、必要な Function set を選んで「Finish」ボタンを押します (図 16.10)。

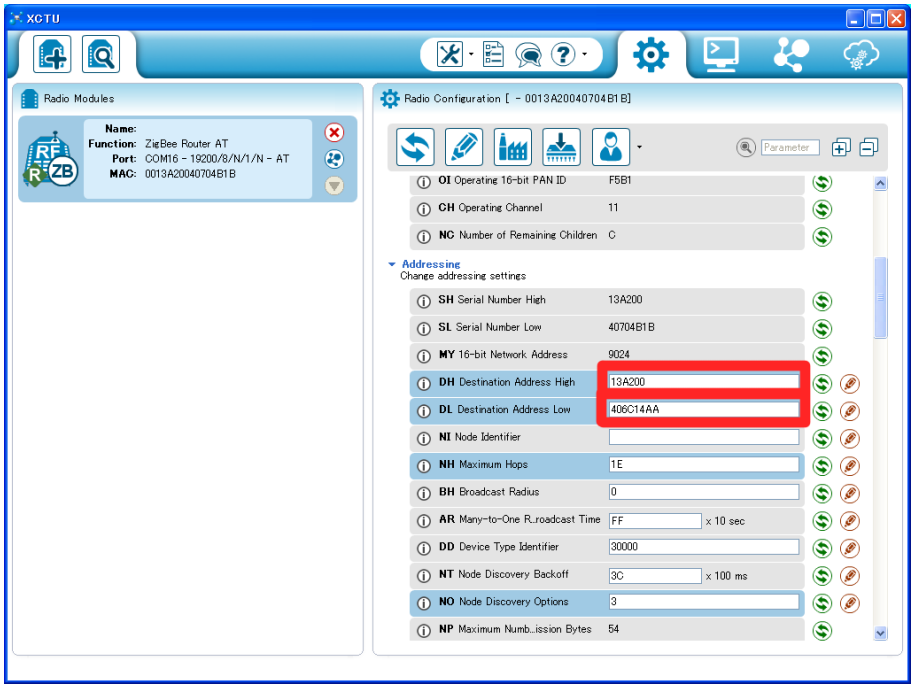

図 16.11　相手のアドレスを登録

相手のアドレスを登録します (図 16.11)。

全ての設定が終わったら、「Write radio settings」ボタン  を、クリックします。

以上で設定は終了です。

ターミナルソフトを起動して、無線でロボットをコントロールしてみましょう。

プログラムは、P.393 の 07_TMRW08.c を使います。

モーション再生ソフト仕様

- f : 前進。
- b : 後退。
- r : 右旋回。
- l : 左旋回。

ボーレート 19200bps。

実行結果

シリアルケーブルを通して、ターミナルソフトから指令を送る。
前進、後退、右旋回、左旋回の動作を指令可能。

16.3　コントローラを作って無線で操縦する

では、P.328 の「3 軸加速度センサを利用する」で使った、KXP84 モジュールを利用してロボットのコントローラを作ってみましょう。

回路図

コントローラのマイコンも、AKI-H8/3694F(QFP) タイニーマイコンモジュールを利用します。

前後と左右の傾きが分かればよいので、X 軸と Y 軸のアナログ出力のみを計測することにします。

加速度センサの傾きと、出力の関係は P.330 の図 12.10 を確認してください。

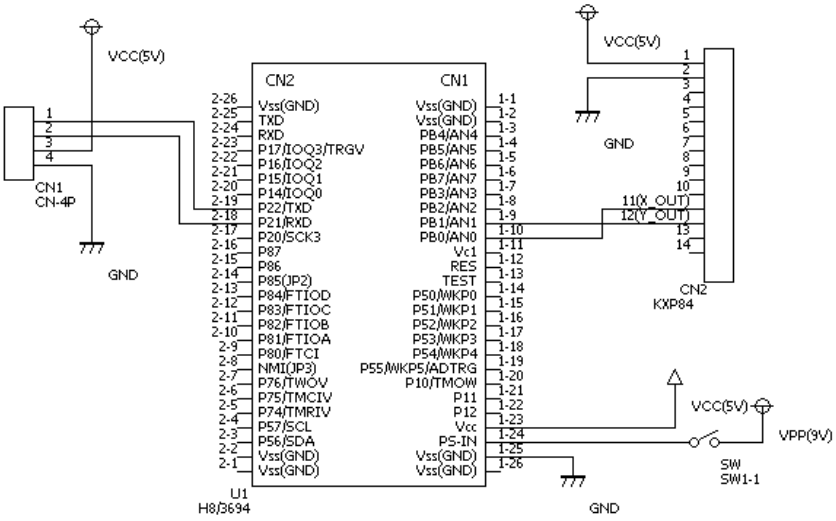

図 16.12　無線コントローラの回路図

以下にコントロール基板の画像を掲載しておきます (図 16.13)。

XBee で無線通信するための構成です。パソコンと有線でつなぐには、XBee の代わりにレベル変換 IC の回路につなぎます。

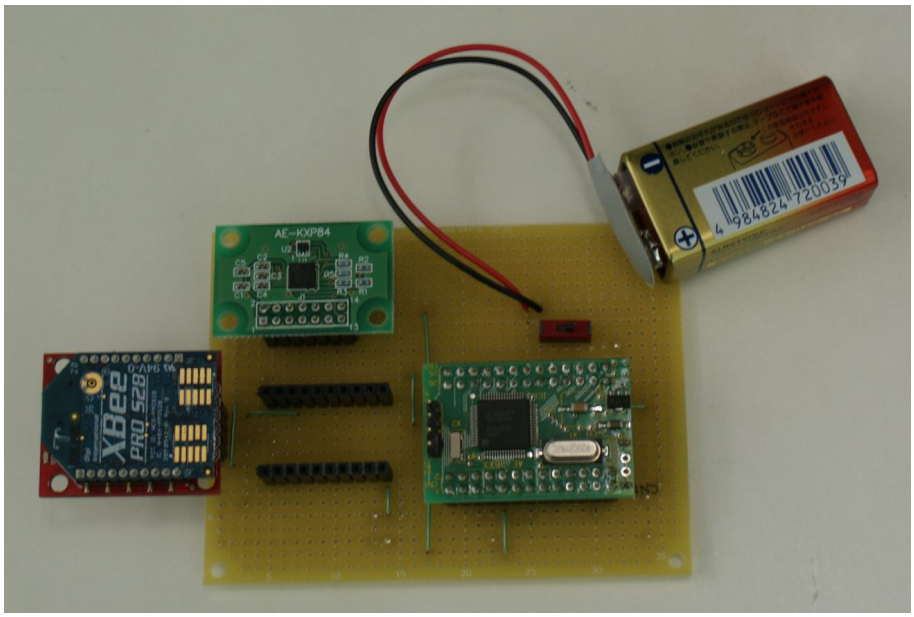

図 16.13　無線コントローラ基板

以下に示すのは、3 軸加速度センサの傾きによってロボットを歩かせるプログラムです。ロボット側のプログラムは、P.396 のサンプルプログラム 07_TMRW08.c を使います。

コントローラソフト仕様

- コントローラを前に傾ける (図 16.13 では右側に傾ける) 「f」を送信：前進。
- コントローラを後に傾ける (図 16.13 では左側に傾ける) 「b」を送信：後退。
- コントローラを右に傾ける (図 16.13 では上側に傾ける) 「r」を送信：右旋回。
- コントローラを左に傾ける (図 16.13 では下側に傾ける) 「l」を送信：左旋回。

ボーレート 19200bps。

なお、コントローラの傾きの判定は、前後を優先し、前後のどちらかに傾いている場合には、左右については傾きを見ないようにしました。これは、斜め方向の歩行を作っていないためです (もちろん、作ることは可能です)。

09_ADC03.c

```
/************************************************************************/
/*                                                                      */
/*  FILE        :09_ADC03.c                                             */
/*  DESCRIPTION :AN0-1 の値を 4 回平均してシリアルで表示                */
/*  CPU TYPE    :H8/3694F                                               */
/*                                                                      */
/*  AD0  CN1-10                                                         */
/*  AD1  CN1- 9                                                         */
/*                                                                      */
/************************************************************************/

#include "iodefine.h"
#include "sci.h"

#define WAIT_COUNTER 0x000FFFFFL /* ループ回数 */
#define WAIT_LOOP 0x06FFFFFL /* ループ回数 */

/*******************************
  wait 関数
*******************************/
void WaitLoop(void)
{
  unsigned long i;

  for(i=0; i<WAIT_LOOP; i++){
    ;
  }
}

/*******************************
   AD のつながった端子を初期化
*******************************/
void AdInit(void)
{
  AD.ADCSR.BIT.ADST = 0;      /* A/D 変換停止 */
  AD.ADCSR.BIT.SCAN = 1;      /* スキャンモード */
  AD.ADCSR.BIT.CKS  = 1;      /* 高速変換 */
  AD.ADCSR.BIT.CH   = 1;      /* AN0-1 */
}

/*******************************
   AD 変換開始
*******************************/
void AdStart(void)
{
  AD.ADCSR.BIT.ADST = 1;      /* A/D 変換スタート */
}

/*******************************
   AD 変換中止
*******************************/
void AdStop(void)
{
  AD.ADCSR.BIT.ADST = 0;
}

/*******************************
   AD 変換読み取り
```

```
 59  ******************************/
 60  unsigned short AdRead(int a)
 61  {
 62    unsigned short ad_r;
 63
 64    while(!AD.ADCSR.BIT.ADF){
 65      ;
 66    }
 67    if(a==0){
 68      ad_r=(AD.ADDRA>>6) ;
 69    }else if(a==1){
 70      ad_r=(AD.ADDRB>>6) ;
 71    }
 72    AD.ADCSR.BIT.ADF=0;
 73    return(ad_r);
 74  }
 75
 76  /******************************
 77     main 関数
 78  ******************************/
 79  void main(void)
 80  {
 81    char a[7];
 82    unsigned short i1, i2, j;
 83    unsigned int avr,avr2;
 84    unsigned long k;
 85    int count, avr_c;
 86
 87    WaitLoop(); /* XBee 用 */
 88
 89    Sci1Init(br19200);
 90    Rx1BuffClear();
 91    AdInit();
 92
 93    Sci1Puts("***** H8/3694F *****\n");
 94
 95    while(1){
 96      avr=0;
 97      /* 4 回 A/D 変換をして平均を取る */
 98      for(avr_c=0; avr_c<4; avr_c++){
 99        AdStart();
100        i1=AdRead(0);
101        avr=avr+i1;
102      }
103      i1=avr>>2;  /*  4 で割る */
104      avr2=0;
105      /* 4 回 A/D 変換をして平均を取る */
106      for(avr_c=0; avr_c<4; avr_c++){
107        AdStart();
108        i2=AdRead(1);
109        avr2=avr2+i2;
110      }
111      i2=avr2>>2;  /*  4 で割る */
112      if(i1>600){
113        Sci1Putchar('f'); /* 前進 */
114      }else if(i1<350){
115        Sci1Putchar('b'); /* 後退 */
116      }else if(i2<330){
117        Sci1Putchar('r'); /* 右旋回 */
118      }else if(i2>640){
119        Sci1Putchar('l'); /* 左旋回 */
120      }else{
121        Sci1Putchar('h');
122      }
123      /*  ウェイトループ  */
124      for(k=0; k<WAIT_COUNTER; k++){
125        ;
126      }
127    }
128  }
```

End Of List

実行結果

コントロール基板の傾きによって、ロボットを歩かせることができる。

課題 16.3.1　(提出) 動きの追加

ほかの動きを追加し、コントローラから動かしてみましょう。

プロジェクト名：e09_ADC03

17 モーション作成ソフト

17.1 モーション作成ソフト

モータの数が少なく、転倒する危険性が少ないロボットの動きを作ることは、それほど難しいことではありません。

ここまで紹介してきたサンプルプログラムを少し拡張すれば、ターミナルソフトからロボットのポーズを作り、それらをつなぐことによって一連の動きを作ることも可能です。

この作業は少々面倒ですが、四足歩行ロボットなどでは、各足の周期を変更するだけのことが多いので、現実的な方法であるといえます。

しかし、二足歩行ロボットのように転倒しやすいものの動きを作るときや、四足歩行ロボットなどでも少し複雑な動きを作るときなどには、手軽にデータを作り、再生して確認することができるソフトがあると便利です。

ここでは、パソコン上で動作し、マイコンとシリアル通信を用いて、手軽にロボットの動きを作成、確認できるプログラム、モーション作成ソフトを使ってみることにしましょう。

なお、ここで紹介するモーション作成ソフト「意信電信」は、著者のホームページにて実行ファイルを公開しています。必要な方はご利用ください (ただし、モータ数は 24 個になっているので、マイコン側のプログラムもそれにあわせる必要があります)。

通信データの仕様も公開していますので、同様のソフトを自分で作ることも可能です。

このテキストでは、モーション作成ソフトの作成に関しては扱いません。

17.1.1 モーション作成ソフト「意信電信」操作マニュアル

モーション作成ソフト「意信電信」の基本的な操作について説明しておきます。

ここには書きませんでしたが、シリアル通信の設定を変更する機能なども備わっています。

起動と画面の説明

本ソフトは、図 17.1 のアイコンをダブルクリックすることによって実行できます。必要なファイルやフォルダを生成しますので、あらかじめフォルダを作り、実行ファイルを入れておくことをお勧めします。

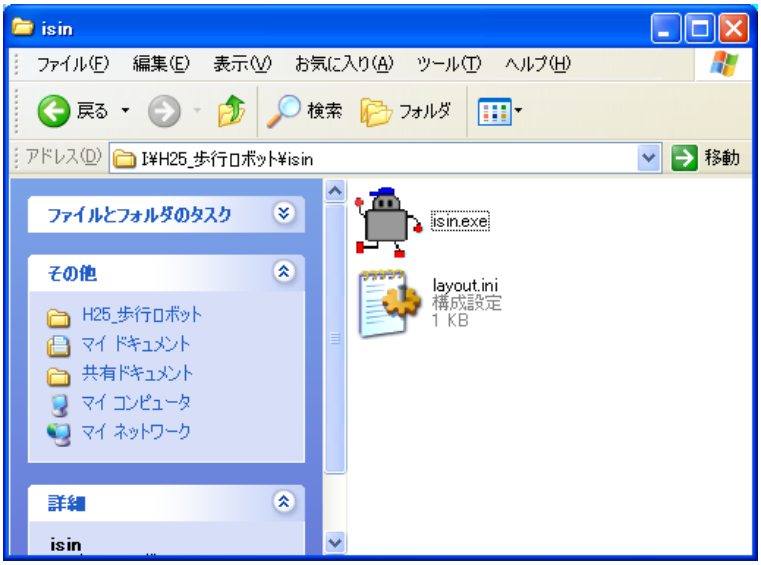

図 17.1　「意信電信」アイコン

図 17.2 がメイン画面です。初めて起動したときには、カテゴリ「歩行ロボット」、モーション「通常歩行」、ポーズデータ「ホームポジション」が作られ表示されるようになっています。

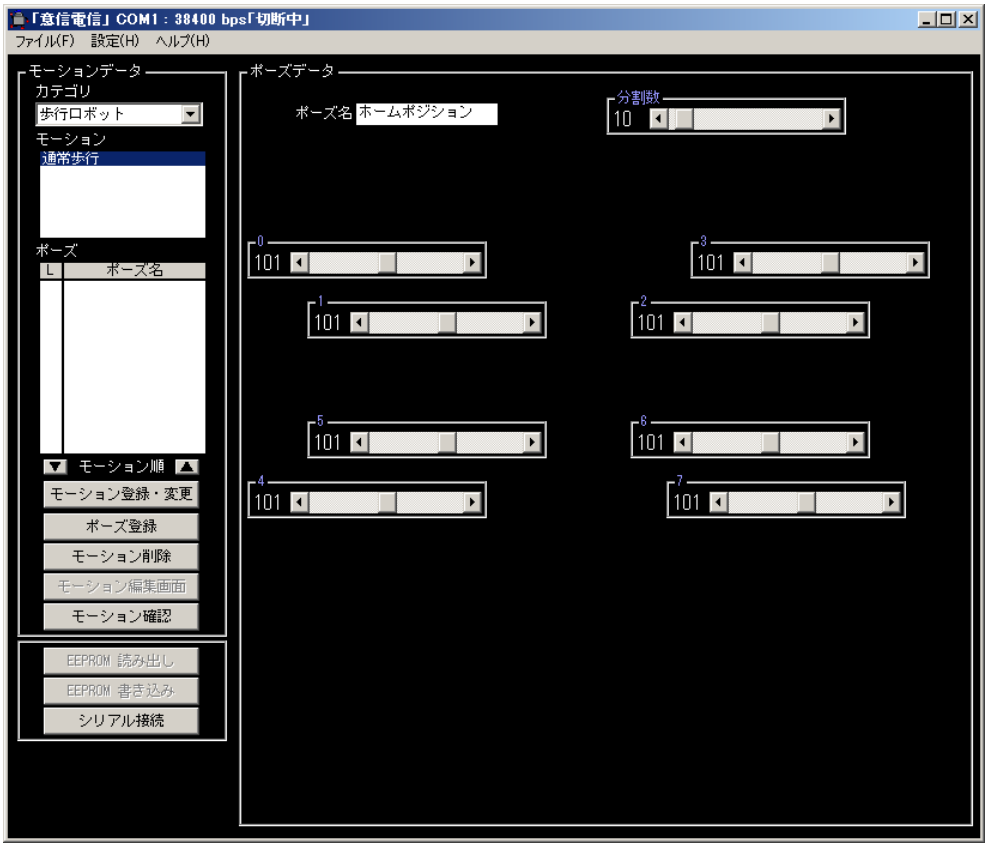

図 17.2 「意信電信」操作画面

「ポーズ」はロボットの姿勢を意味します。

「モーションデータ」の「ポーズ」を選択すると、そのデータ内容が「ポーズデータ」に表示されます。

「ポーズデータ」はスライダを動かすことで変更できます。スライダを動かすと変更されたデータが送信されます。

変更した「ポーズ」は「ポーズ登録・変更」ボタンで変更されるまで保存データに反映されません。「分割数」は前の「ポーズ」からの移行時間を表しています。単位は、約 20ms です。

「ポーズ」をいくつか連続してつないだものが、「モーション」になります。「カテゴリ」は「モーション」の集まりで、同種の「モーション」を分類するために使います。

一つの「モーション」に登録できる「ポーズ」の数は、最大で 12 に制限しています。「カテゴリ」や「モーション」の数に制限はありません。

「モーション」の確認は、「モーション確認」ボタンでおこないます。

現時点では、「モーション編集画面」「EEPROM 読出し」「EEPROM 書き込み」の各機能はこのテキストの範囲のプログラムでは利用できません (非表示にしています)。

ボタンの機能に関する注意

ボタンの機能は、「モーション」「ポーズ」のどちらを選択しているかによって変わってきます。

図 17.3　「モーション」選択状態

図 17.4　「ポーズ」選択状態

同名の「ポーズ」関する注意

「ポーズ名」は、同一「モーション」内でも同名のものを複数使うことができます。ただし、同じデータを利用しますので、ひとつのデータを変更した場合には、同名の全ての「ポーズ」データが変更されることになります。なお、異なる「モーション」内の同名の「ポーズ」は影響を受けません。

ループの設定

各モーション内では、ひとつだけループを設定することができます。「ポーズ」の「L」リストボックスで、複数のセルを選択すると、ループ設定ダイアログが出てきます。セルの選択は、ドラッグで選択しても良いし、SHIFT や Ctrl を使っても結構です (Shift+下向き矢印などを使った場合には、2 個のセルが選択された時点で ループ数の要求が出ます)。ループ数は、半角で入力してください。

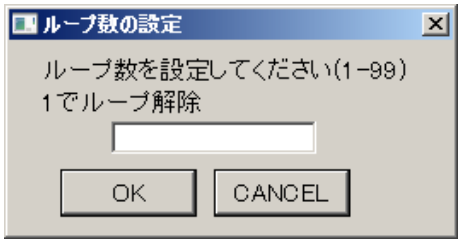

図 17.5 ループ数の設定

ループ数を入力すると、ループの最初に対応する項目にループ数が、ループの最後には「E」のマークが出ます。

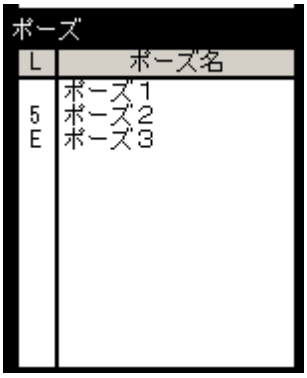

図 17.6 ループ数の表示

初期化ファイル

お勧めはしませんが、スライダの最大値最小値は「minmax.ini」に、レイアウトは「layout.ini」に保存されています。値を変更することによって、スライダの最大最小値やレイアウトを変更することもできます。データ送信時などの最大値最小値のチェックはおこなっていません。デフォルトの値に戻したい場合には、削除してください。次回プログラム終了時にデフォルト値で作成されます。なお、「homepos.ini」はホームポジションの値ですが、「設定」メニューの「ホームポジションに設定」を使って設定することをお勧めします。

17.1.2 通信の仕様

通信データは、制御コードとデータから構成します。モータの角度を指定するデータは、0 から 200 までを使うことにします。100 が稼動範囲の中央に来るように、マイコン側のプログラムで変換することにします。

201 から 255 までは制御コードとします。

今回は以下のように割り振ってみました。

ただし、後のサンプルで実装しているのは、一部機能のみです。

表 17.1　通信コード一覧

記号	数値	意味
	0-200	モータの角度データ
	201-238	未使用
SDV	239	以下、通信終了 (EOT) まで、1 個のモータを駆動するためのデータ (分割数、モータ番号、角度)
DV	240	次の 1 バイトは電圧のデータ (マイコンから)
SMD	241	以下、通信終了 (EOT) までモーションデータ
DIV	242	次の 1 バイトは分割数 (指定のポーズ到達までの時間)
SLO	243	ループ開始 (次の 1 バイトはループ回数)
ELO	244	ループ終了
CSM	245	次の 1 バイトは、チェックサム (保存データでは削除)
MEX	246	マイコン上にあるモーションを実行
RTM	247	外部記憶装置のモーションをマイコンへ読み出し
MTR	348	マイコン上にあるモーションを外部記憶装置へ書き込み
SSD	249	以下通信終了まで、1 箇所のモータを駆動するためのデータ (モータ番号, 角度のみ)
SPWM	250	PWM 開始
EPWM	251	PWM 停止
ACK	252	肯定応答 (マイコン側から)
NAK	253	再送信要求 (マイコン側から)
SOH	254	通信開始
EOT	255	通信終了

1 モータをスライダで動かす場合 (分割数無し)

上記の制御コードを用いた通信データの例をあげておきます。

1 モータをスライダで動かす場合は、SOH(254)、SSD(249)、モータ番号、角度データ、EOT(255) の 5 バイトのデータを送ります (図 17.7 参照)。

通信速度を必要をするために、チェックサムなどは送りません。

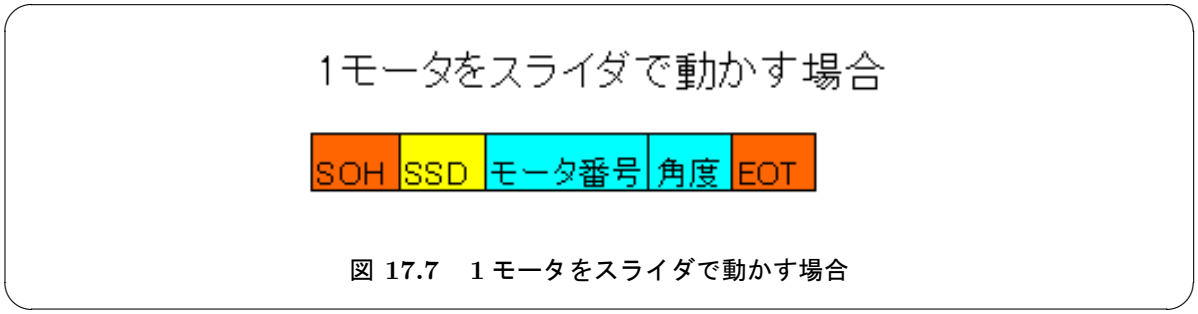

図 17.7　1 モータをスライダで動かす場合

1 モーションを転送する場合

ロボットの歩行動作などの動きは、いくつかのポーズを連続して再生することで実現します。

データは図 17.8 のように決めました。

「全モータのデータ」は、モータ番号無しで角度データをモータ番号順に送ります。モータ番号を入れないのは、データ数を少しでも減らすためです。

チェックサムには、データ部をすべて加算して 256 で割った余りを入れます。

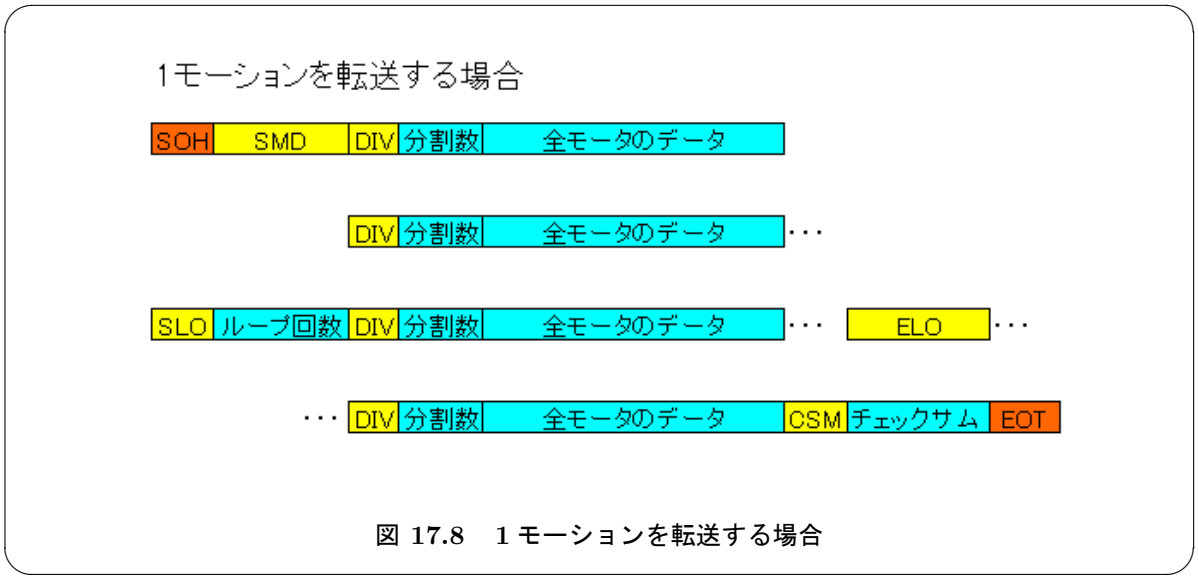

図 17.8　1 モーションを転送する場合

17.2　モーション作成ソフトから制御する

では、上で説明した通信の仕様に沿って、マイコンのプログラムを変更してみましょう。

以下にサンプルプログラムを掲載します。

07_TMRW09.c

```
 1  /************************************************************************/
 2  /*                                                                      */
 3  /*  FILE        :07_TMRW09.c                                            */
 4  /*  DESCRIPTION :サーボモータ 8 個「意信電信」スライダー対応版          */
 5  /*  CPU TYPE    :H8/3694F IC(TC4051B)                                   */
 6  /*                                                                      */
 7  /*  COM P82/FTIOB                                                       */
 8  /*  A P10                                                               */
 9  /*  B P11                                                               */
10  /*  C P12                                                               */
11  /*                                                                      */
12  /************************************************************************/
13
14  #include "pwm1.h"
15  #include "sci.h"
16
17  unsigned char HOME[SRV_CH_NUM]={100,100,100,100,100,100,100,100};
18  int DAT_curIdx;
19  int i;
20  unsigned char data[100];
21  unsigned short id_c, dn;
22  /* id_c:データカウント用  */
23  unsigned short loop, startf;
```

```
24  /*   loop:ループ数格納用
25     startf:ループの始まりを格納  */
26
27  #define WAIT_LOOP 0x06FFFFFL /* ループ回数 */
28
29  /*******************************
30    wait 関数
31  *******************************/
32  void WaitLoop(void)
33  {
34    unsigned long i;
35
36    for(i=0; i<WAIT_LOOP; i++){
37      ;
38    }
39  }
40
41  /*******************************
42    モーション再生
43  *******************************/
44  void Play_motion(void)
45  {
46    int loop_flag=0,loop_flag2=0;
47    /*  loop_flag:データの終了判定用
48     loop_flag2:ループの終了判定用  */
49    id_c=2;
50    do{
51      dn++;
52      switch(data[id_c]){
53        case DIV:
54          id_c++;
55          PwmSetPose(&data[id_c]);
56          id_c=id_c+SRV_CH_NUM;
57          break;
58        case SLO:
59          id_c++;
60          loop=data[id_c];
61          id_c++;
62          startf=id_c;
63          while(loop){
64            loop_flag2=0;
65            id_c=startf;
66            do{
67              switch(data[id_c]){
68                case DIV:
69                  id_c++;
70                  PwmSetPose(&data[id_c]);
71                  id_c=id_c+SRV_CH_NUM;
72                  break;
73                case ELO:
74                  loop_flag2=1;
75                  break;
76              }
77              id_c++;
78            }while(!loop_flag2);
79            loop--;
80          }
81          id_c--;
82          break;
83        case EOT:
84          loop_flag=1;
85          break;
86      }
87      id_c++;
88    }while(!loop_flag);
89  }
90
91  /*******************************
92    main 関数
93  *******************************/
94  void main(void)
95  {
96    WaitLoop();
97    Sci1Init(br19200); /* ボーレートの設定 */
98    PwmInit();
99    for(i=0; i<SRV_CH_NUM; i++){
100     PwmSetDuty(i, D_CONST+D_PROPO*HOME[i]); /* モータの初期位置設定 */
101   }
102
103   while(1){
104     DAT_curIdx=0;
105     while((data[DAT_curIdx]=Sci1Read())!= SOH){ /* SOH が来るまで読み捨てる */
106       ;
107     }
108     while((data[++DAT_curIdx]=Sci1Read())!=EOT){ /* EOT まで読み込む */
109       ;
110     }
111     switch(data[1]){
112       case SSD:
113         PwmSetDuty(data[2], D_CONST+D_PROPO*data[3]);
114         /* スライダからのデータを処理 */
115         break;
116       case SMD:
```

```
117         /* モーション再生 */
118           Play_motion();
119           break;
120       }
121     }
122   }
123
```

End Of List

pwm.h

```
 1  #ifndef _PWM_H_
 2  #define _PWM_H_
 3
 4  #define PWM_MAX_DUTY (12000-1) //2.4mS
 5  #define PWM_MIN_DUTY  (3000-1) //0.6mS
 6  #define PWM_PERIOD   (12500-1) //2.5mS
 7
 8  #define SRV_CH_NUM 8
 9
10  #define D_CONST PWM_MIN_DUTY //char_duty と duty の変換式の定数部分
11  #define D_PROPO 45 //char_duty と duty の変換式の比例係数部分 (12000-3000)/200
12
13  #define SDV 239
14  #define SMD 241
15  #define DIV 242
16  #define SLO 243
17  #define ELO 244
18  #define SSD 249
19  #define SOH 254
20  #define EOT 255
21
22  void PwmInit(void);
23  void PwmStart(void);
24  void PwmStop(void);
25  void PwmSetDuty(int ch, unsigned short duty);
26  unsigned short PwmGetDuty(int ch);
27  void PwmSetPose(unsigned char char_duty[]);
28
29  #endif
```

End Of List

pwm.c

```
 1  #include "pwm1.h"
 2  #include "iodefine.h"
 3  #include <machine.h>
 4
 5  #define P1_MASK 0x07
 6
 7  char buf[6];
 8  unsigned long n;
 9  int c;
10  unsigned short srv_curr_index=0;
11  unsigned short srv_ch;     /* PWM の出力先 */
12  unsigned short calc_duty_flg;    /* 20ms をカウント */
13  unsigned long pwm_duty[SRV_CH_NUM]; /* PWM のデューティのデータ */
14
15  /*******************************
16     PWM 出力端子を初期化
17  *******************************/
18  void PwmInit(void)
19  {
20    srv_ch=0;
21    calc_duty_flg=0;
22    IO.PCR1|=0x07;
23    IO.PDR1.BYTE|=(srv_ch & P1_MASK);
24    IO.PDR1.BYTE&=(srv_ch | (~P1_MASK));
25    set_imask_ccr(1);
26    TW.TMRW.BYTE=0x49;  /* FTIOB 端子を PWM 出力に設定 */
27    TW.TCRW.BYTE=0xA0;  /* 内部クロックの 1/4, コンペアマッチ B で 1 出力 */
28    TW.TIERW.BYTE=0x71; /* A の割り込みを可能にする */
29    TW.TSRW.BYTE=0x70;  /* 割り込みフラグをクリア */
30    TW.TCNT=0x0000;     /* TCN の初期化 */
31    TW.GRA=PWM_PERIOD;  /* 周期 (2.5mS) */
32    TW.GRB=PWM_PERIOD-7500;        /* 1.5mS */
33    TW.TMRW.BIT.CTS=1;  /* TCNT カウンタスタート */
34    set_imask_ccr(0);
35
36  }
37
38  /*********************************
39    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;
}


/********************************
  デューティの取得（パラメータから）
 ********************************/
unsigned short PwmGetDuty(int ch)
{
  /* サーボチャンネルが範囲外なら無視 */
  if(ch >= SRV_CH_NUM){
    return (0);
  }

  /* デューティを返す */
  return (pwm_duty[ch]);
}

/*-----------------------------------------------------------------------*/
/* ポーズ補間関数 0-200 で指定*/
/*-----------------------------------------------------------------------*/
void PwmSetPose(unsigned char char_duty[])
{
  long cur_duty[SRV_CH_NUM], new_duty[SRV_CH_NUM];
  long diff_duty[SRV_CH_NUM];
  int i;
  unsigned short div, count_i;

  div=char_duty[0]; /* 分割数：20mS 単位 */
  for(i=0; i<SRV_CH_NUM; i++){
    new_duty[i]=D_CONST+D_PROPO*char_duty[i+1];/* 目標値 */
    cur_duty[i]=PwmGetDuty(i); /* 現在のデューティ */
    /* 現在のデューティと目標デューティの差 */
    if(new_duty[i]-cur_duty[i]>=0){
      diff_duty[i]=new_duty[i]-cur_duty[i];
    }else{
      diff_duty[i]=cur_duty[i]-new_duty[i];
    }
  }
  /* 補間の計算 */
  for(count_i=1; count_i<=div; count_i++){
    /* 20mS 待ち */
    while(!calc_duty_flg){
      ;
    }
    calc_duty_flg=0;
    /* 20mS ごとの値を計算 */
    for(i=0; i<SRV_CH_NUM; i++){
      if(new_duty[i]-cur_duty[i]>=0){
        PwmSetDuty(i, cur_duty[i]+(diff_duty[i]*count_i)/div);
      }else{
        PwmSetDuty(i, cur_duty[i]-(diff_duty[i]*count_i)/div);
      }
    }
  }
}

/********************************
  割り込み関数
 ********************************/
```

```
130  __interrupt(vect=21) void INT_TimerW(void)
131  {
132    /* 2.5mS ごとのコンペアマッチ */
133    TW.TSRW.BIT.IMFA=0;
134    srv_ch++;
135    if(srv_ch>= SRV_CH_NUM){
136      srv_ch=0; /* モータ出力先を 0 に戻す */
137      calc_duty_flg=1;
138    }
139    IO.PDR1.BYTE|=(srv_ch & P1_MASK);
140    IO.PDR1.BYTE&=(srv_ch | (~P1_MASK));
141    TW.GRB=PWM_PERIOD-pwm_duty[srv_ch];
142  }
```

End Of List

実行結果

モーション作成ソフトから、ポーズを作ったり、モーションを作って確認することができる。

課題 17.2.1　(提出) モーション作成ソフトで作ったモーションで自律動作

モーション作成ソフトで作ったモーションを、(シリアルケーブル無しで) 自動的に再生するプログラムを作りましょう。

上記サンプルプログラム 07_TMRW09.c にデータを書き込むようにしてください。

プロジェクト名：e7_TMRW09

18 DCモータの駆動

18.1 DCモータ駆動回路

DC モータの回転を制御するには、FET で H ブリッジを組むなどの方法があります。

ここでは、市販の DC モータドライブ IC を用いて、DC モータを制御してみましょう。

DC モータドライブ IC としては、入手性が良く、使用法が簡単な TA8428K を利用することにします。TA8428K は DC モータを、正転、逆転、ストップ、ブレーキの 4 モードが動作可能です (表 18.1)。

表 18.1　TA8428K の動作

入力		出力		出力モード
IN1	IN2	OUTA	$\overline{\text{OUTA}}$	
H	H	L	L	ブレーキ
L	H	L	H	逆転
H	L	H	L	正転
L	L	OFF(ハイインピーダンス)		ストップ

ここで、ブレーキとはモータの軸を回すと抵抗を感じる状態を言います。一方ストップは軸をまわすと抵抗無く回る状態のことです。

18.1.1 回路図

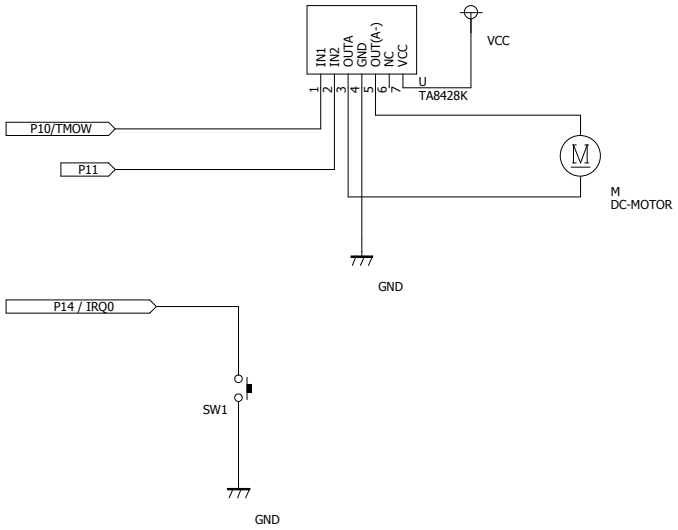

図 18.1　DC モータ駆動回路回路図

13_DCMOT01.c

```
/*****************************************************************************/
/*                                                                           */
/*  FILE        :13_DCMOT01.c                                                */
/*  DATE        :Fri, Mar 07, 2014                                           */
/*  DESCRIPTION :スイッチを押したら DC モータが駆動                          */
/*  CPU TYPE    :H8/3694F                                                    */
/*                                                                           */
/*  タクトスイッチ                                                           */
/*  SW0 P14                                                                  */
/*  モータ IC                                                                */
/*  IN1 P10                                                                  */
/*  IN2 P11                                                                  */
/*                                                                           */
/*****************************************************************************/

#include "iodefine.h"
#include "sw.h"

void MotInit(void)
{
  IO.PCR1 |= 0x03;
  IO.PDR1.BIT.B0=0;
  IO.PDR1.BIT.B1=0;
}

void MotDrive(int a)
{
  switch(a){
    case 0: //ブレーキ
      IO.PDR1.BIT.B0=0;
      IO.PDR1.BIT.B1=0;
      break;
    case 1: //逆転
      IO.PDR1.BIT.B0=0;
      IO.PDR1.BIT.B1=1;
      break;
    case 2: //正転
      IO.PDR1.BIT.B0=1;
      IO.PDR1.BIT.B1=0;
      break;
    case 3: //ストップ
      IO.PDR1.BIT.B0=1;
      IO.PDR1.BIT.B1=1;
      break;
  }
}
```

```
47
48  void main(void)
49  {
50    SwInit();       /* タクトスイッチ接続ポートの初期化 */
51
52    while(1){
53      SwWaitPush();  /* タクトスイッチが押されるまで待つ */
54      SwWaitDetach();  /* タクトスイッチが離されるまで待つ */
55      MotDrive(1);
56      SwWaitPush();  /* タクトスイッチが押されるまで待つ */
57      SwWaitDetach();  /* タクトスイッチが離されるまで待つ */
58      MotDrive(2);
59      SwWaitPush();  /* タクトスイッチが押されるまで待つ */
60      SwWaitDetach();  /* タクトスイッチが離されるまで待つ */
61      MotDrive(3);
62    }
63  }
```

End Of List

18.2 DC モータ制御

ここでは、ロータリスイッチの値に応じで回転数を変える。タイマ V を用いる。

18.2.1 回路図

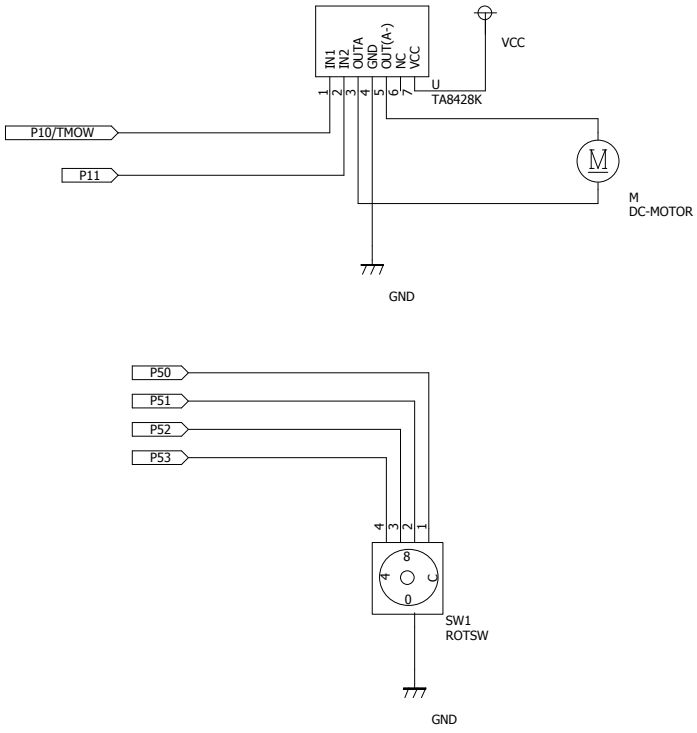

図 18.2 DC モータ駆動回路回路図 (ロータリスイッチ)

intprg.c

```
// vector 21 Timer W
__interrupt(vect=21) void INT_TimerW(void) {/* sleep(); */}
// vector 22 Timer V
//__interrupt(vect=22) void INT_TimerV(void) {/* sleep(); */}
// vector 23 SCI3
__interrupt(vect=23) void INT_SCI3(void) {/* sleep(); */}
// vector 24 IIC2
__interrupt(vect=24) void INT_IIC2(void) {/* sleep(); */}
// vector 25 ADI
__interrupt(vect=25) void INT_ADI(void) {/* sleep(); */}
```

13_DCMOT02.c

```
 1  /************************************************************************/
 2  /*                                                                      */
 3  /*  FILE        :13_DCMOT02.c                                           */
 4  /*  DATE        :Mon, Mar 10, 2014                                      */
 5  /*  DESCRIPTION :ロータリスイッチで DC モータ制御                       */
 6  /*  CPU TYPE    :H8/3694F                                               */
 7  /*                                                                      */
 8  /*  ロータリスイッチ                                                    */
 9  /*  ROTSW1 P50                                                          */
10  /*  ROTSW2 P51                                                          */
11  /*  ROTSW3 P52                                                          */
12  /*  ROTSW4 P53                                                          */
13  /*                                                                      */
14  /*  モータ IC                                                           */
15  /*  IN1 P10                                                             */
16  /*  IN2 P11                                                             */
17  /*                                                                      */
18  /************************************************************************/
19
20  #include "iodefine.h"
21  #include "sw.h"
22
23  void MotInit(void)
24  {
25    IO.PCR1 |= 0x03;
26    IO.PDR1.BIT.B0=0;
27    IO.PDR1.BIT.B1=0;
28  }
29
30  void TmvInit(void)
31  {
32    TV.TCRV0.BYTE = 0xD3; /* A,B 割込み,B でクリア,128 分周 */
33    TV.TCRV1.BIT.ICKS=1; /* 128 分周 */
34    TV.TCORA=0x10;
35    TV.TCORB=255;
36  }
37
38  void MotDrive(int a)
39  {
40    TV.TCORA=((a&0x0F)<<4);
41  }
42
43  void main(void)
44  {
45    RotSwInit();   /* ロータリスイッチ接続ポートの初期化 */
46    MotInit();      /* DC モータ接続ポートの初期化 */
47    TmvInit();
48
49    while(1){
50      MotDrive(RotSwGet());
51    }
52  }
53
54  __interrupt(vect=22) void INT_TimerV(void) {
55    if(TV.TCSRV.BIT.CMFB==1){
56      IO.PDR1.BIT.B0=1;
57      TV.TCSRV.BIT.CMFB=0;
58    }
59    if(TV.TCSRV.BIT.CMFA==1){
60      IO.PDR1.BIT.B0=0;
61      TV.TCSRV.BIT.CMFA=0;
62    }
63  }
```

End Of List