ワンチップマイコンを用いた温度コントローラの開発と冷却駆動実験

2009 年 2 月 20 日 市川隆 (東北大理)

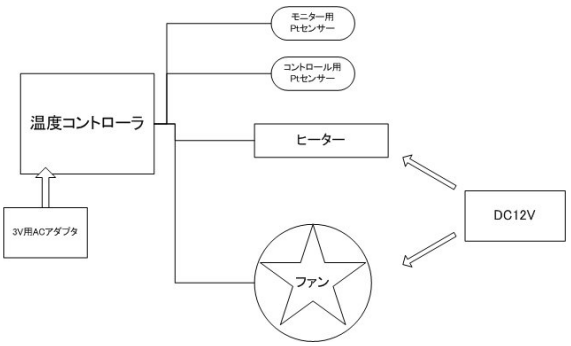
1. はじめに

南極の極寒環境において通常の電子機器・電気機器は動作しない。電子部品を厳選することでマイナス 40℃でも運用可能な装置の開発も一部可能であるが、一般的にマイナス 10℃程度が限界であろう。従って、真空断熱材を用いた保温性の高い温蔵庫の中に入れ(「真空断熱材 BOX 実験」村田)、自己発熱を利用した保温が有効と考えられる。しかしその場合にも温度が下がり過ぎないように、また上がりすぎないように温度コントロールが必要である。市販の温度コントローラは高価であり、しかも南極の環境下での動作は保証されない。また本目的には市販温度コントローラにあるような多機能は必要としない。そこでワンチップマイコンを用いたマイナス 40℃でも動作可能な温度コントローラを開発した。

2. マイコンによる制御システム

Texas Instruments 社のミックスド・シグナル・マイコロプロセッサ MSP430F4270 は、32KB+256B フラッシュメモリ、256BRAM、16 ビットタイマ、4 チャンネル並列の 16 ビット A/D コンバータ、12 ビット D/A コンバータ 、32 個の I/O 端子、LCD ディスプレイ・ドライバを搭載した 16 ビット汎用マイコンである(トランジスタ技術 2007 年 1 月号に特集あり)。しかも 1 個 1200 円と安価である。4K バントまでのプログラムならば、無償で開発プラットホーム(IAR Embedded Workbench)が使え、C 言語で開発ができる。(このマイコンを用いた周期誤差補正回路については「周期誤差補正回路」(市川)参照)。図 1 に今回開発した温度コントローラのシステムの構成を示す。A/D コンバータを 2 つ用いて Pt 温度センサーからの入力を 2 チャンネル用意した。片方をモニター用、もう片方をコントロール用として使用する。温度差に比例する電流を出力するトランジスタを D/A コンバータに接続する。温度が上がりすぎた場合に備えて、冷却のために外気を取り込むファンを駆動するリレーを I/O ポートに取り付ける。ヒーターとファンの駆動用には別電源(12V、スイッチング式)を用意する。

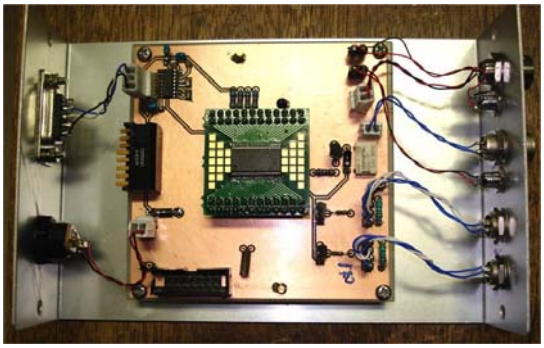
図 1
システム構成

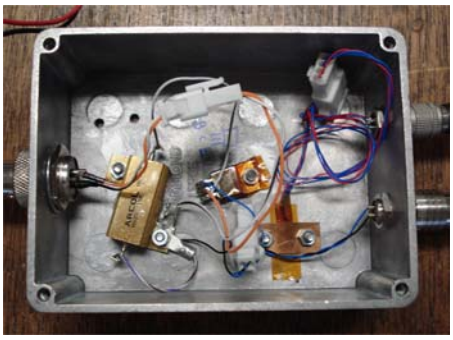


3. 温度制御回路

制御回路を図 2 に示す。左はコントロール回路、右はヒーターと温度センサー(Pt)である。回路図は添付資料(1)。部品は南極での運用のためマイナス 40℃仕様のものを選んだ。パソコンからデータを取り込むために、RS232C 用ポートを用意した。

図2 制御回路　　ヒーターと温度 Pt センサー





4. 温度設定

今回の目的では、2 つのモードを用意する。ひとつは設定した温度に維持するモードであり、もうひとつは 2 つモニターの温度差が常に一定になるようにコントロールするモードである。前者は冷却化では動作しない装置を一定の温度に保つものであり、後者は霜が着かないように、望遠鏡などの装置と外気の温度差が常に一定になるようにコントロールする(例えば、望遠鏡やミラーを外気よりも 0.1℃高くして、着霜を防ぐ)。設定はディップスイッチから 2 進数で与える。前者の場合は摂氏温度(1℃単位)、後者は温度差(0.01℃単位)をディップスイッチで与える。マイコンにロードするソフトを書き換えることで、その他の使用方法も可能である。

5. パソコンによるモニター

本システムには RS232C 機能を取り入れてある。マイコン用 RS232C ソフトはトラ技 2007 年 1 月号を用いた。またトラ技 2007 年 7 月号付属の Windows 用ソフトによって、エクセルの VBA 機能を用いてエクセルに直接読み込むことができる。バイナリ読みだしならば、フリーの Acknowrich などの通信ソフトでの読み出しも可能である。Linux 用には読み込みソフトを新たに開発した(添付資料)。ただし、本システムからのデータを垂れ流すだけなので、パソコンからマイコンを制御することはできない。本システムの開発に使用した無料のプラットホームはソフトの長さに 4K バイトの制限があり、複雑なソフトの開発ができないためである。有償のものを購入することで、マイコンのメモリ(32KB)をすべて利用できるようになるので、もっと幅の広い使い方も可能になる。

6. 室内実験

制御には PID 制御をしている。ここでは P と I の制御での一例を紹介する。図は室温約 25 度、設定温度 31 度での結果である。十分に時間がたち、安定した状態での結果を図 3 に示す。平均値は 30.75℃、標準偏差は 0.21℃である。31℃にたっしていないのは、ヒーターの能力が足りないものと思われる。誤差が大きい原因はまだ不明である。

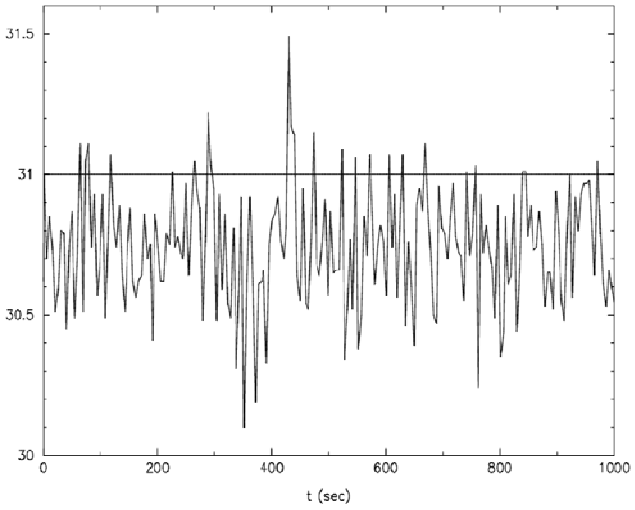
7. 冷却実験

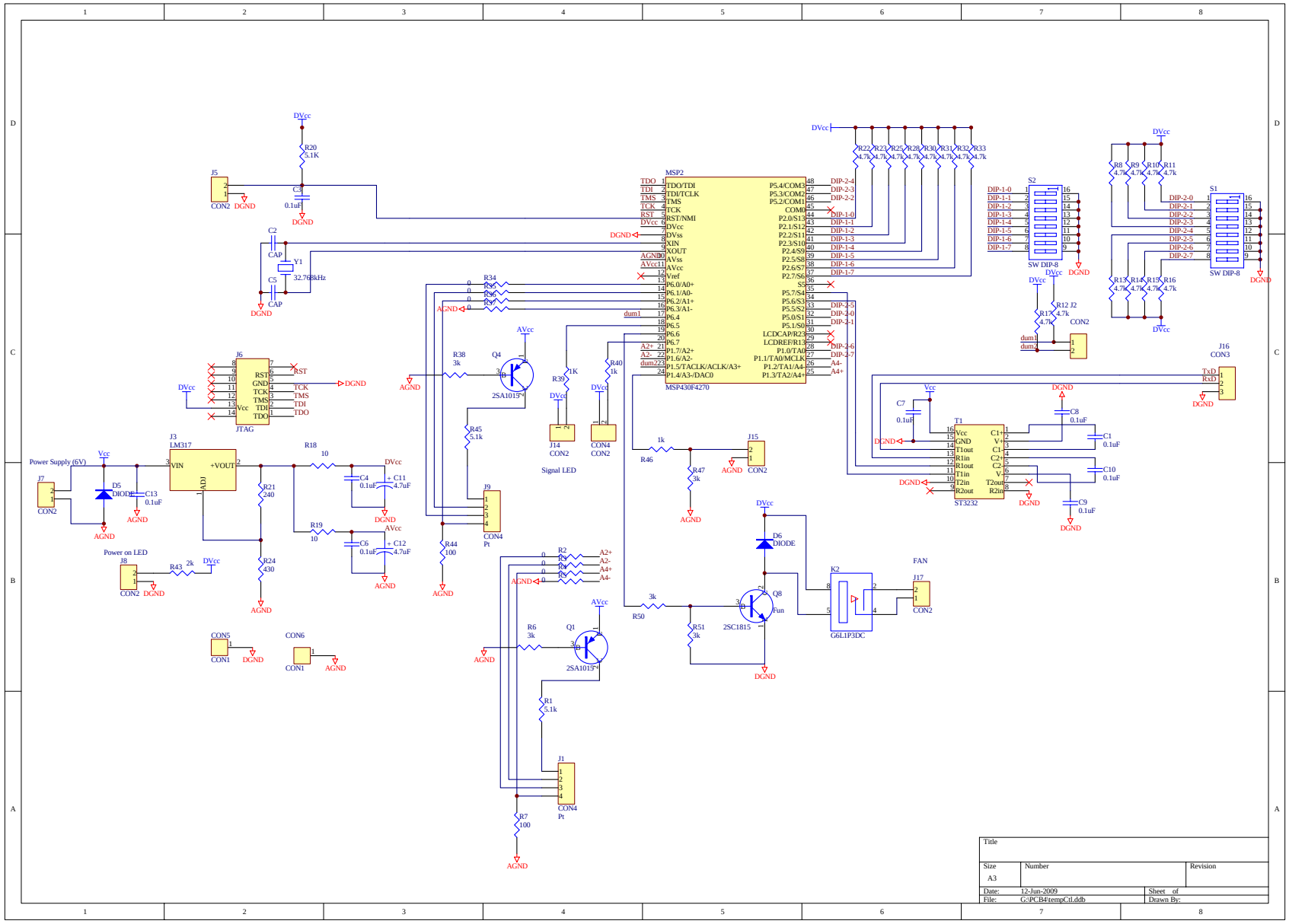
電源を除く本システムを冷凍庫の中で駆動実験を行った。まず室温から冷凍庫の限界(今回は－75℃までしか下がらなかった)まで動かした。その結果、最低温度まで正常に動作した。次に、

電源を切り、−75℃で長時間放置した後、コールドスタートした。この場合も問題なく動作したので、本装置は南極の環境でも問題なく動作することが期待される。

今後はPID制御パラメータの設定、高精度Pt温度センサーへの交換、ヒーターのバワーアップなどで、精度を高めていく予定である。

図3 温度コントロール例




MSP2
MSP430F4270
JTAG
LM317
Power Supply (6V)
Power on LED
Signal LED
2SA1015
2SC1815
ST3232
G6L1P3DC
SW DIP-8
FAN
Title
Size
A3
Number
Revision
Date: 12-Jun-2009
Sheet of
File: G:\PCB4\tempCtl.ddb
Drawn By:

(2) 温度コントロール用プログラム (Linux 用)

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <termios.h>
#include <unistd.h>
#include <time.h>

int main(int argc, char *argv[] ){
  FILE *measureFile;
  struct tm *date;
  time_t now;
  int year,month,day;
  int hour,minute,second;
  time_t time();
  int fd;
  struct termios tio;
  int err;
  int i,j;
  int heater,fun;
  int time0,dtime;

  unsigned char buff1,buff2;
  float T,Tset;

  if ((fd = open("/dev/ttyS0", O_RDWR | O_NOCTTY | O_NONBLOCK))<0) {
    exit(-1);
  }
  bzero(&tio, sizeof(tio));
  tio.c_cflag = B9600 | CS8 | CLOCAL | CREAD;
  tio.c_iflag = IGNPAR;
  tio.c_oflag = 0;
  tio.c_lflag = 0;
  tio.c_cc[VTIME] = 0;
  tio.c_cc[VMIN] = 1;

  tcflush(fd, TCIFLUSH);
  tcsetattr(fd, TCSANOW, &tio);
  fcntl(fd, F_SETFL, FNDELAY);
```

```
time(&now);
date=localtime(&now);
year=date->tm_year + 1900;
month=date->tm_mon+1;
day=date->tm_mday;
hour=date->tm_hour;
minute=date->tm_min;
second=date->tm_sec;

measureFile=fopen(argv[1],"a");
printf("%5d/%2d/%2d %2d %2d %2d¥n",year,month,day,hour,minute,second);
fprintf(measureFile,"%5d/%2d/%2d %2d %2d %2d¥n",year,month,day,hour,minute,second);
fclose(measureFile);
time0=time(NULL);
while(1){
  measureFile=fopen(argv[1],"a");
  dtime=time(NULL)-time0;
  err=-1;
  while(err<0){
    err = read(fd,&buff1,1);
  }
  err=-1;
  while(err<0){
    err = read(fd,&buff2,1);
  }
  Tset=(buff1*256+buff2)/100.0;
  err=-1;
  while(err<0){
    err = read(fd,&buff1,1);
  }
  err=-1;
  while(err<0){
    err = read(fd,&buff2,1);
  }
  T=(buff1*256+buff2)/100.0;
  err=-1;
  while(err<0){
    err = read(fd,&buff1,1);
  }
```

```
    heater = buff1;
    err=-1;
    while(err<0){
      err = read(fd,&buff2,1);
    }
    fun=buff2;

    printf("%8d %7.2f %7.2f %1d %1d ¥n",dtime,Tset,T,heater,fun);
    fprintf(measureFile,"%8d %7.2f %7.2f %1d %1d ¥n",dtime,Tset,T,heater,fun);
    fclose(measureFile);
  }
  close( fd );
}
```

温度コントロール用プログラム (マイコン用)

```
// （トラ技2007年1月号、9月号を参照して作成)
#include "DMM0.h"
#include "msp430x42x0.h"
#include <math.h>
#include <stdlib.h>
#define REPEAT 16
float V0, V1, V7;                       // output voltages from AD converter
float R;                                // register to be obtained
float T;                                // Temperature
int heater, fun;
float Tset;                             // setup Temperature from DIGIswitch
float Po;                               // output power
char ch2[2];
const float Tupper = 40;                // upper temperature above which fan is operated
const float AR = 3.9083e-3;             // conversion coefficient of Pt-RTD
const float BR = -5.775e-7;             // converstion coefficient of Pt-RTD
const float R0 = 100.;                  // reference registor for Pt-RTD
// R=R0(1+AT+BT^2)
const float KI = .5;                    // I for PID control
const float KP = 1000.;                 // P for PID control
const float bias = 940;
unsigned short TAR_period;              // sampling rate
unsigned short Cnt_OVF;                 // overflow for TAR
unsigned short Num_OVF;                 // number of TAR overflow
long cnt0;
```

```
void MEA_Temp(void);
void main(void) {
   WDTCTL = WDTPW + WDTHOLD;                    // stop Wotch Dog Timer
   SCFQCTL = 94 - 1;                           // MCLK = 94 * ACLK * 2 = 6.16MHz, D = 2
   FLL_CTL0 = DCOPLUS + XCAP18PF;               // DCOPLUS = 1, capacitance 18pF
   SCFI0 = FLLD_2 + FN_2;                      // D = 2, fDCOCLK =     2.2-17Mhz
    BTCTL = BT_ADLY_2000;                       // Basic Timer
   t1ms(5);                                    // wait for 5ms until FLL+ stabilized
    P1SEL = 0x10;
 // P1DIR = 0xEF;                               //P1.4 should be DIR=0 for DAC output
   P1DIR = 0x00;
   P2DIR = 0x00;
   P5DIR = 0x10;
   P6DIR |= 0xF0;
    P6OUT |= 0xA0;
   P6OUT &= 0xAF;
    DAC12_0CTL = DAC12OPS + DAC12SREF_0 + DAC12LSEL_0 + DAC12AMP_7 +
DAC12IR;
                                               // (see manual)
   DAC12_0DAT = 0;
    _BIS_SR(GIE);                               // enable interupt
   do    {
     MEA_Temp();                                // measure temperature
   }while(1);
}
void MEA_Temp(void)
{
    long AD_Sum;                                // sampling times
   int i,j;
   float IR;
// setup AD converter
   SD16CTL = SD16VMIDON + SD16REFON + SD16SSEL_1 + SD16DIV_3 +
SD16XDIV_2;
                                               // 1.2V ref, SMCLK, /8, /16
   t1ms(1);
   SD16CTL &= ~SD16VMIDON;                      // VMID off: used to settle ref cap
   SD16CTL = SD16REFON + SD16SSEL_1 + SD16DIV_3 + SD16XDIV_2;
                                               // 1.2V ref, SMCLK, /8, /3, fM = 44kHz
   SD16CCTL0 =    SD16UNI + SD16OSR_1024 + SD16BUF_0;
                                               // unipole, OSR = 1024, bufferoff
```

```
  IR = 0;
  do  {
    j = (j + 1) & 0x3;                          // if j = 0, transfer data to COM
    Tset   = P2IN;                                  //read DIGIswitch
                                                    // 1st bit = sign 0 = +; 1 = -1
    if(Tset >= 128) Tset = -(Tset-128);     // 8th bit MLB
// measurement of offset
    SD16INCTL0 = SD16INTDLY_0 + SD16GAIN_8 + SD16INCH_7;
    SD16CCTL0 |= SD16SC;                            // start measuremment
    AD_Sum = 0;
    for (i = 0; i < REPEAT; i++)  {
      while ((SD16CCTL0 & SD16IFG) == 0); // wait until finish
      AD_Sum = AD_Sum + SD16MEM0;
    }
    SD16CCTL0 &= ~SD16SC;                           // stop AD convert
    V7 = AD_Sum / (float)REPEAT / 65535. * Vref2/8.;
// measurement of Pt
    SD16INCTL0 = SD16INTDLY_0 + SD16GAIN_8 + SD16INCH_0;
                                                    // Pt (CH = 0)
    SD16CCTL0 |= SD16SC;
    AD_Sum = 0;
    for (i = 0; i < REPEAT; i++) {
      while ((SD16CCTL0 & SD16IFG) == 0);
      AD_Sum = AD_Sum + SD16MEM0;
    }
    SD16CCTL0 &= ~SD16SC;
    V0 = AD_Sum / (float)REPEAT / 65535. * Vref2/8.;
 // reference registor R0=100
    SD16INCTL0 = SD16INTDLY_0 + SD16GAIN_8 + SD16INCH_1;
                                                    // CH = 1
    SD16CCTL0 |= SD16SC;
    AD_Sum = 0;
    for (i = 0; i < REPEAT; i++) {
      while ((SD16CCTL0 & SD16IFG) == 0);
      AD_Sum = AD_Sum + SD16MEM0;
    }
    SD16CCTL0 &= ~SD16SC;
    V1 = AD_Sum / (float)REPEAT / 65535. * Vref2/8.;
    R = (V0 - V7) / (V1 - V7) * 100.;
    T = (-AR + sqrt(pow(AR,2)-4*BR*(1-R/R0)))/2.0/BR;
```

```
    if (abs(Tset - T)/Tset > .2)                // deviation of temperature
    {
      IR = 0;                                   // clear integration
    }
    else   {
      IR = IR+(Tset - T);                       // integrate difference
    }
    Po = KP * (Tset - T) + KI * IR + bias;          // bias voltage to be sent
    if (Po > 4095) Po = 4095;
    if (Po < bias) Po = 0;
    DAC12_0DAT = Po;
                                                  // p6.4 NC
                                                  // p6.5 heater LED (1=off,0=on)
                                                  // p6.6 fan (1=on, 0=off)
                                                  // p6.7 fan LED (1=off,0=on)
    if(Po <=   bias) {
      heater=0;
      P6OUT |= 0x30;
    }
    else {
      heater=1;
      P6OUT &= 0x1F;
    }
//    if(T > Tupper) {
    if(T > Tset) {
      P6OUT |= 0x40;
      P6OUT &= 0x7F;
      fun=1;
    }
    else {
      P6OUT &= 0xBF;
      P6OUT |= 0x80;
      fun=0;
    }
    if ( j == 0) {
                                          // 温度データをシリアル通信で出力
      TX_DAT((int)(Tset*100)>>8);
      TX_DAT((int)(Tset*100));
      TX_DAT((int)(T*100)>>8);
      TX_DAT((int)(T*100));
```

```
        TX_DAT(heater);
        TX_DAT(fun);
      }
   } while(1);
}
/*************************************
   1ms wait routine
*************************************/
void t1ms(unsigned short dat) {
   unsigned short j;
   unsigned long i;
   i = 32.768 * dat - 22;
    do {
     j = BTCNT1;
     while(BTCNT1 == j);
     i--;
   }while (i != 0);
}
/*****************************************************
   EIA574 Data transmit service routine
*****************************************************/
void TX_DAT(unsigned int j)
{
   unsigned int i;                          // 変
   P5OUT &= ~0x10;                          // TX = 0   スタート・ビット
   BTCNT1 = 0;                                  // BTCNT1 をクリア
   while (BTCNT1 < tb[0]);                  // 1 周期待つ
   for (i = 0; i < 8; i++)                  // データの送信, D0 から
   {
     if ((j & 0x1) == 0)
     {
       P5OUT &= ~0x10;                          // 送信データが 0 なら TX = 0
     }
     else
     {
       P5OUT |= 0x10;                           // 送信データが 1 なら TX = 1
     }
     j = j / 2;                                 // 送信データを 1 ビット右シフト
     BTCNT1 = 0;                                  // BTCNT1 をクリア
     while (BTCNT1 < tb[i + 1]);                // 1 周期待つ
```

```
    }
    P5OUT |= 0x10;                          // TX = 1、ストップ・ビット
    BTCNT1 = 0;                             // BTCNT1 をクリア
    while (BTCNT1 < tb[9]);                 // 1 周期待つ
}
```