USART シリアルサーボ化

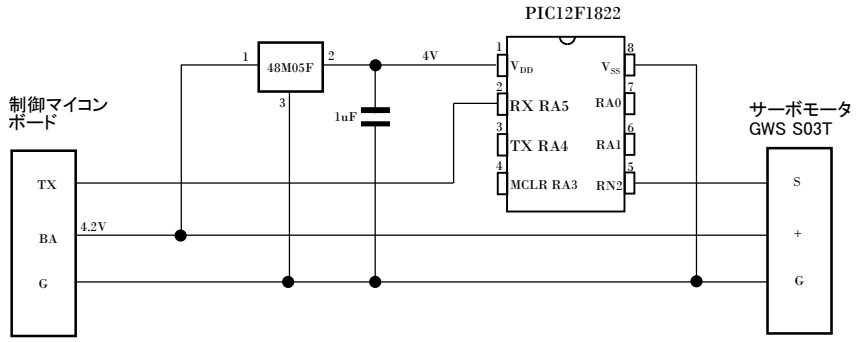
2017.12.21

USARTから角度を入力してサーボモーター信号に変換します。シリアル（テキスト）で角度を指定します。

A45 '¥n'

角度は1度から指定できます。 サンプル動画（ https://www.youtube.com/watch?v=rfplXDQ_yE4&feature=youtu.be ）では1度づつ変化させて、かなりスムーズな動きを実現しています。 タイマ割込を使っているので，反応が早く、シリアル転送時間の影響も殆どありません（38400bps）。但し指定後次の指定までは**最低20[ms]以上**の時間をおいてから実行します。 繰り返しレートはおよそ**60Hz**です。USARTバスの様な形式です。

サーボには英数１文字で名前(A)を付けます。 数字、英字（大文字、小文字）などを使えば、**62（ASCIIコード16進で、0x30～0x39, 0x41～0x5a, 0x61～0x7a）**個のサーボを識別できます※（身近な図形文字（graphic character）のみ）。 この場合は、制御マイコン側に電圧ホロアが必要でしょう。 AVRマイコンの場合の最大接続サーボ数はAVRのファンアウト数に依ります。 ここでは1個のサーボにつき、PIC12F1822を1個用います。サーボ信号は連続出力します。レギュレータとともに各サーボに固定して使う形が良いです。電源はバッテリー直結を奨めます。ただしサーボにも依りますが、通常6Vが上限です。レギュレータは仮に48M05Fとしています。制御マイコンの電源部に48M05Fを用いたからです。 このためレベルコンバーターや電圧減衰機構は不要です。



回路図（48M05Fは省略可）

※ マイコン同士で繋ぐ場合は、 どんなコードでも使えますが、OSの元で使うと意外なトラップがあります。例えば、0x1bはESCコードなのでUSART処理が中断されるでしょう。また、0x19=0x59-0x40='X'-0x40=ctrl+X ってであり、終了の意味があります。どうしても使う時はOSに、それなりの手続きを必要とします。これを嫌って図形文字に限った紹介としました。

参考にさせて頂いたページ

https://www.youtube.com/watch?v=rfplXDQ_yE4&feature=youtu.be

角度（パルス幅）の指定が電圧値なのに対し、USARTからの数値（ディジタル）に対応させています。

```
/* ******************************************
 * Serial Servo  v5
 * Renewal 2019.4.29 mtakapii
 * Use for Automatic Inverted Pendulum ...
 * ref...
 * UART_2_Servo_TMR_Converter.c    Completed!
 *                  2017.10.01 mtakapii
 *  PIC12F1822
 *  MLAB X IDE v4.01,   XC8 v1.43
 *
 *  8MHz   Baud Rate:38400 bps, 16-bit Baud Rate
 *
 *  Command ... 'a90¥n','a0¥n', 'a180¥n'
 *  'a'  ... Servo Name
 *  '90' ... Servo Operation Angle
 *  '¥n' ... End of Command Sequence
 *
 *  PIC12F1822 pin assignment
 *     1 VDD(+)     8 VSS (Connect to - )
 *   RX 2 RA5 in     7 RA0 (Connect to R/C receiver)
 *   TX 3 RA4 out    6 RA1 out-->LED
 *     4 RA3,MCLR   5 RA2 out Servo Signal (CCP1)& LED
 *
 * http://atalpha.web.fc2.com/pic/pic_204_RS232C_2.html
 * http://diy.ease-labs.com/?page_id=1584
 * http://physics.cocolog-nifty.com/weblog/2014/10/pwm-ea25.html
 * http://mitt.la.coocan.jp/pic/pic5_23.html
 * ********************************************************** */

#include <xc.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>

#define _XTAL_FREQ 8000000        // for delay_ms(x)...

//*********************************************
#pragma config FOSC = INTOSC, WDTE = OFF, PWRTE = OFF, MCLRE = OFF, CP =OFF
//#pragma config FOSC = INTOSC, WDTE = OFF, PWRTE = OFF, MCLRE =ON, CP =OFF
#pragma config CPD = OFF, BOREN = ON, CLKOUTEN = OFF, IESO = OFF, FCMEN= OFF
#pragma config WRT = OFF, PLLEN = OFF, STVREN = ON, BORV = LO, LVP = OFF
//#pragma config CCPMX = RA2      // CCP1 ==> RA2

long map(long x, long in_min, long in_max, long out_min, long out_max)
{
 return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min ;
}

//#define TMR1_OFFSET 25536
  #define TMR1_OFFSET 15000
  //#define TMR1_MAX    TMR1_OFFSET+4600   // Retracted Position
  //#define TMR1_MIN    TMR1_OFFSET+1300   // Pendulum Position
  #define TMR1_MAX    TMR1_OFFSET+4500   // Retracted Position
  #define TMR1_MIN    TMR1_OFFSET+1600   // Pendulum Position
  #define TMR1_CEN    (TMR1_MIN+TMR1_MAX)/2

  #define Servo_Name 'a'

unsigned short width = TMR1_CEN;
int x = 0;

#define LED RA1
#define TRIM -300

void interrupt ccp1( void ){  // Timer Interrupt Process
  if( TMR1IF ){
    //putch("t");
    RA2 = 1;              // Servo Signal ==> HIGH
    CCPR1 = width + TRIM;         // Set width
    TMR1 = TMR1_OFFSET;    // 20msec
    TMR1IF = 0;           // Reset flag
  }
  // CCP1 Interrupt Process
  if( CCP1IF ){
    //putch("c");
    RA2 = 0;       // to LOW Level
    CCP1IF = 0;    // Compare match Flag Clear
  }
}

void FlashLED(int i);
void warm_up();
void set_width();

// ********* main *********
void main() {
  char ch;

  //OSCCON  = 0b01101010;       // Int clock -->4MHz
  OSCCON  = 0b01110000;         // Int clock --> 8MHz
  TRISA   = 0b11101001;         // RA1, RA2, RA4 -->out
  PORTA   = 0x00;
  //CCP1CON = 0b00001100;       // ==>PWM Mode
  ANSELA  = 0x00;
  CCP1CON = 0b00001010 ;        // --> compare mode
  //T1CON = 0b00100100 ;        // Frequency DIvid --> 1/4
  T1CON   = 0b00000100 ;        // Frequency DIvid -->
```

```
  // -----------------    UART  -----------------------------
  APFCONbits.RXDTSEL = 1;      // RX --> RA5 pin
  APFCONbits.TXCKSEL = 1;      // TX --> RA4 pin
  WPUA   = 0b00011111;         // RA5
  nWPUEN = 0;
  RCSTA   = 0b10010000;        // 0x90 8 bit, Continus
  TXSTA   = 0b00100100;        // 0x24 US, 8bit, Non parity
  BAUDCON = 0b00001000;        // Baud Rate set is 16 bit mode
  SPBRGH  = 0;
  //SPBRGL  = 103;             //  2400 bps at  4 MHz
  //SPBRGL  = 207;             // 38400 bps at 32 MHz
  //SPBRGL  = 25;              // 38400 bps at  4 MHz
  SPBRGL  = 51;                // 38400 bps at  8 MHz
  // ------------------------------------------------------

  TMR1   = TMR1_OFFSET;
  TMR1ON = 1;                  // Start Timer 1
  TMR1IF = 0;                  // Timer flag clear
  CCP1IF = 0;

  __delay_ms(300);
  FlashLED(10);
  // putch('S');putch('¥n');putch('¥r');
  printf("¥r¥nStarted Serial Servo !!");
  printf("¥r¥nmtakapii 2019.5.10  ");
  width = TMR1_CEN;
  warm_up();

  while(1){
    x = 0x0000;
    ch = getch();   putch(ch);
    if(ch == Servo_Name){
      while(1) {
        ch = getch();   putch(ch);
        if( (ch == 0x0d || ch == 0x0a) ){
          set_width();
          putch('¥n');putch('¥r');
          putch('o');putch('k');
          break;
        }
        else{
          x = x * 10 + (ch - 0x30);
        }
      }
    }
  }
}
```

```
void putch(unsigned char byte){
  while(!TXIF)
    continue;
  TXREG = byte;
}

char getch(){
  RCREG = 0;
  while(!RCIF)
    continue;
  if(RCIF){
    if(FERR || OERR){
      CREN = 0;
      CREN = 1;
      RCREG = '?';
    }
    RCIF = 0;
  }
  return RCREG;
}

//  LED
void FlashLED(int i){
  while(i--){
    RA1 = 1; __delay_ms(100);
    RA1 = 0; __delay_ms(100);
  }
}

void warm_up(){
  for(x=0; x<180;  x=x+10) {
    set_width();
      __delay_ms(25);
  }
  __delay_ms(500);
  x=90;
  set_width();
}

void set_width(){
  width = map(x, 0,180, TMR1_MIN,TMR1_MAX) ;  // mapping
  PEIE = 1;
  GIE  = 1;      // all interruption --> Enable
  TMR1IE = 1;    // Timer 1 --> Enable
  CCP1IE = 1;    // Compare match --> Enable
}
```