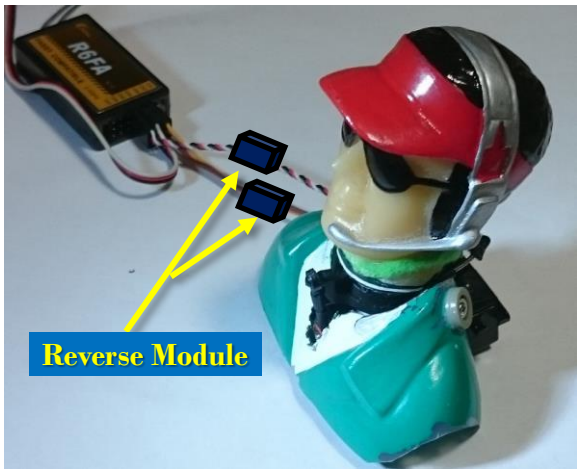
Moving Head --Servo Reverse Module by PIC12F1822 --

2019.5.10 (Fri.)
mtakapii

機 能

既に組込み済のMoving Head に対し、Head の傾斜あるいは回転方向を逆転させたい場合がある。そんな時、このモジュールを受信機とサーボの間に挿入すれば, 簡単に思いの方向・動作量で作動させることができる。



中型機用パイロット人形　H83 x W89 x D46mm

動画： https://www.youtube.com/watch?v=Zi5VtE28Yr8&feature=youtu.be

背 景

或る日、サーボの動作方向を良く考えずに Moving Head を作ってしまった。そもそもこれが、開発するきっかけ（羽目）となった（必要は発明の母ってか！）。動作は、ラダーとエレベータに連動させたい。 しかも、動作方向はスティックと同一としたい。一度、作り込んだサーボなどを作り直すのは大変な作業となる。やる気も失せる。ハードウエアをいじくらずに、なんとかならないものかと考えた末に今回の方式に辿り着いた。 が、開発期間を考えれば作り直した方が早かった。

開発に当たっては、ネットの方々に大変お世話になりました（感謝します）。 また、製作の参考にしたのは、自作の“Serial Servo”の経験が大きい（これもネットからヒント）。 後で判明したが“Serial Servo”は既に製品化されていた（近藤科学さん）。“Serial Servo”は“S-BUS”に類するものと考えている。 筆者の“S-BUS”は、UART通信ポートを用い、Servo Name を1キャラクタ（文字）で表現するので、60個以上のサーボを並列接続でき、通常 rate （50Hz程度）では同時的動作が可能となる。

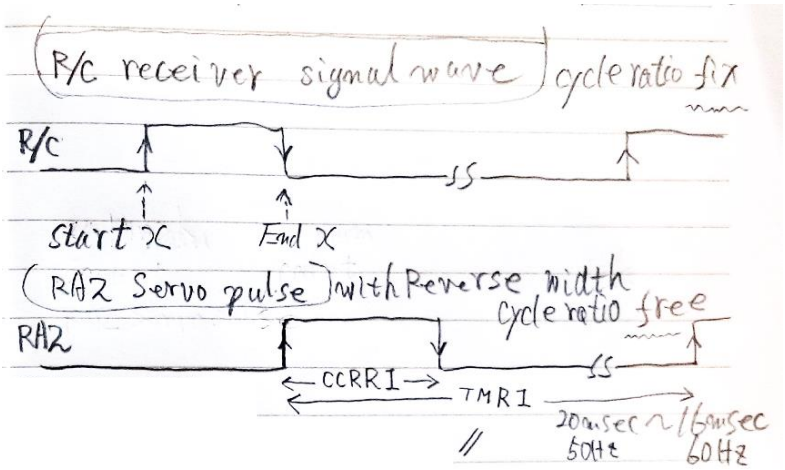
特 長

- ハードウエアをいじくらずに、受信機とサーボの間に挿入するのみで実装可能
- サーボ信号は、PIC12F1822で作り出しているので、繰り返し rate を任意に設定可能
  通常は50Hz だが、これを60Hz にしてしまうとか。これにより、素早いサーボ応答が可能
- 受信機からの信号パルス幅計測はディジタルピン（RA5）設定することで、計測可能。
  アナログ設定も試したが（当然だが）、今一、解像度が足りない（8MHzクロック時）
  ディジタル ➡ 260 ～ 590
  アナログ ➡ 23 ～ 53
  C 言語でコーディングして、この値なら実質的に充分と考える。 チリチリ音も消える。
  ノイズによる計測ミスも想定したが、シュミット入力のためか全く感じられない。
- ピン割当てはICSP を可能としている（書込の際いちいち配線代えをするのは面倒臭い）。
- Serial Pin(RA4:UART:TX)でモニタもできるので都合が良い（パルス幅計測時など）。

動 作

- R/C受信機からのパルス幅をディジタル（RA5）でカウントする => 260～590 (int)
- それをマッピング関数 map() でサーボ信号幅に反転変換する
- 反転変換値はタイマ機能によって繰り返しサーボ信号を出力する（連続出力）
- タイマ割り込み処理が不定期に行なわれる。（ main() 関数が実行されている時 ）

タイムチャート



ピン割当て

PIC12F1822 pin assignment

	1	VDD (+)	8	VSS (Connect to -)
R/C	2	RA5 in	7	RA0
TX	3	RA4 out	6	RA1 out --> FlashLED(10);
	4	RA3, MCLR	5	RA2 out --> Servo Signal (CCP1)& LED

回　路

回路と言う程でもないが、今回は“おもちゃ用”なので、ディジタル部品に必須の電源パスコンを省略している。 プロの方は、0.1uF程度の積層セラミックキャパシタなどを追加して頂きたい。

搭載用の機体

今回は、ショップ“雷”さん発売の“LASER”：

3D 機として、ジャイロとの相性が非常に良い。
（筆者の操縦はまだまだ素人ですが……）

エレベーター：ダウン

ラダー：左

動画： https://www.youtube.com/watch?v=bX_C-hxbeXo

Moving Head：*Servo Reverse Module with* PIC12F1822 Program list

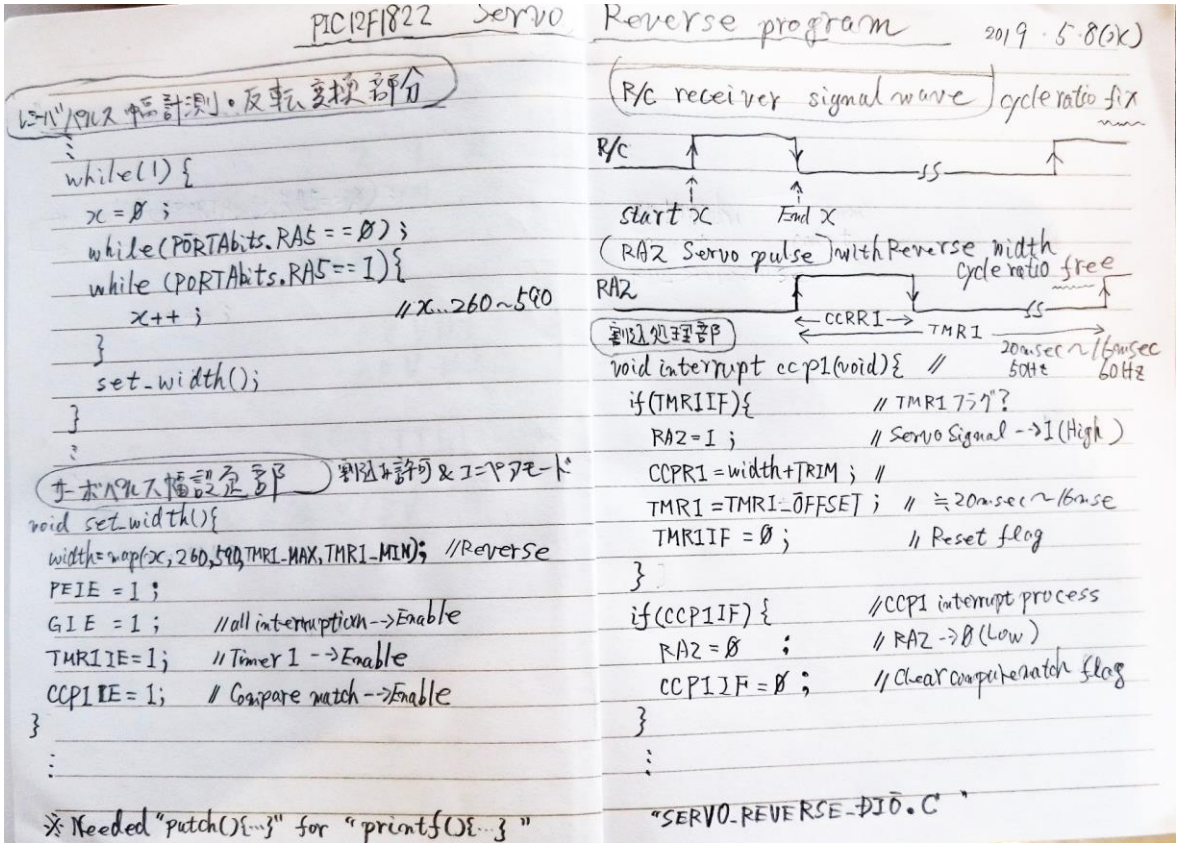
エレベーターとラダーサーボ制御はほぼ同じだが、
***warm_up*() が異なる（面白味？）。**

```
/* ********************************************
 * SERVO_REVERSE_DIO.c  for Elevater Servo
 * Renewal 2019.5.12 mtakapii
 *
 * ●動作：サーボ反転
 * ・R/C受信機からのパルス幅をカウントする=>260～590 (int)
 * ・それをマッピング関数でサーボ信号幅に反転変換する
 * ・反転変換値はタイマ機能によって繰り返しサーボ信号を出力する
 * ●特長
 * ・組込み済のサーボを反転（逆転）可能
 * ・PIC 12F1822(8pin)だけで実現可能
 * ・サーボ出力信号繰り返しrateを変更可能（50Hz,60Hz...）
 *
 *  PIC12F1822
 *  MLAB X IDE v4.01,  XC8 v1.43
 *  8MHz  Baud Rate:38400 bps, 16-bit Baud Rate
 *
 *  PIC12F1822 pin assignment
 *      1 VDD(+)    8 VSS (Connect to - )
 *   R/C 2 RA5 in    7 RA0
 *   TX  3 RA4 out   6 RA1 out-->FlashLED(5);
 *      4 RA3,MCLR  5 RA2 out Servo Signal (CCP1)& LED
 * FlashLED(5);
 *
 * 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+4600    // Retracted Position
#define TMR1_MIN   TMR1_OFFSET+2600    // Pendulum Position
#define TMR1_CEN   (TMR1_MIN+TMR1_MAX)/2  // Center

#define WIDTH_MIN   260               // Receive Pulse width MIN
#define WIDTH_MAX   590               // Receive Pulse width MAX
#define WIDTH_CEN   (WIDTH_MIN+WIDTH_MAX)/2  // Receive Pulse width CENTER

unsigned short width = TMR1_CEN;
int x = 0;

#define LED RA1
#define TRIM -300
```

Program list

```
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);
  printf("¥r¥nStarted Servo Reverse !!");
  printf("¥r¥nfor Elevator Servo  ");
  printf("¥r¥nmtakapii 2019.5.12  ");
  width = TMR1_CEN;
  warm_up();
  warm_up();
  warm_up();

  __delay_ms(300);
  printf("¥r¥n");

  while(1){
    //printf("%d  ", PORTAbits.RA5);
    x = 0;
    while( PORTAbits.RA5 == 0 );   // find rise up RA5
    while( PORTAbits.RA5 == 1 ){    // count until Low Level
      x++;                    // Result ==> x
    }
    //printf("  :%d", x);          // for Measuring
    set_width();
    //__delay_ms(200);
  }
}

/* Needed below putch() */
void putch(unsigned char byte){
  while(!TXIF)
    continue;
  TXREG = byte;
}

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

void warm_up(){
  int cent = WIDTH_CEN;

  x = cent;
  set_width();
  __delay_ms(500);
  for( x=cent ; x < WIDTH_MAX;  x += 5) {
    set_width();
      __delay_ms(15);
  }
  __delay_ms(500);
  for(x=WIDTH_MAX; x > WIDTH_MIN;  x -= 5) {
    set_width();
      __delay_ms(15);
  }
  __delay_ms(500);
  for(x=WIDTH_MIN; x<cent;  x += 5) {
    set_width();
      __delay_ms(15);
  }
  __delay_ms(500);
  set_width();
}

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

開発メモ


PIC12F1822 Servo Reverse program 2019・5・8(水)

レシーバパルス幅計測・反転変換部分

```
while(1){
  x=0;
  while(PORTAbits.RA5==0);
  while(PORTAbits.RA5==1){
    x++;                    //x…260~590
  }
  set_width();
}
```

サーボパルス幅設定部　割込み許可 & コンペアモード

```
void set_width(){
  width=map(x,260,590,TMR1_MAX,TMR1_MIN); //Reverse
  PEIE=1;
  GIE=1;      //all interruptions->Enable
  TMR1IE=1;   //Timer1->Enable
  CCP1IE=1;   //Compare match->Enable
}
```

※Needed "putch(){…}" for "printf(){…}"

R/C receiver signal wave cycle ratio fix

R/C: Start x, End x

RA2 Servo pulse with Reverse width, cycle ratio free

RA2: CCPR1, TMR1 (20msec~16msec, 50Hz~60Hz)

割込処理部

```
void interrupt ccp1(void){ //
  if(TMR1IF){             // TMR1フラグ?
    RA2=1;                // Servo Signal->1(High)
    CCPR1=width+TRIM;     //
    TMR1=TMR1-OFFSET;     // ≒20msec~16msec
    TMR1IF=0;             // Reset flag
  }
  if(CCP1IF){             //CCP1 interrupt process
    RA2=0;                // RA2->0(Low)
    CCP1IF=0;             //Clear compare match flag
  }
```

"SERVO_REVERSE_DIO.C"

開発環境