3. PWM Signal Generators for servos and brushless X-UFO with modified ESC
UFO Doctor, June 24th, 2010

1. Introduction

This report describes PWM signal generators for:

- Standard Servo, period 20ms, pulse 1-2ms
- Modified ESC, period 0.68ms, pulse 0.1-0.67ms (fast PWM 1500Hz)

Thanks to Oliver Mezger (http://mezdata.de/avr/370_servotester/), the programming of the myAVR Board MK2 USB for a signal generator is an easy task. The C-program is in the appendix and the Servo_Tester_Hex_Dat and Fast_PWM_Gen_Hex_Dat are on the homepage www.ufo-doctor.ch in "BL-modification tutorials"

2. Standard servo tester

2.1. Hardware

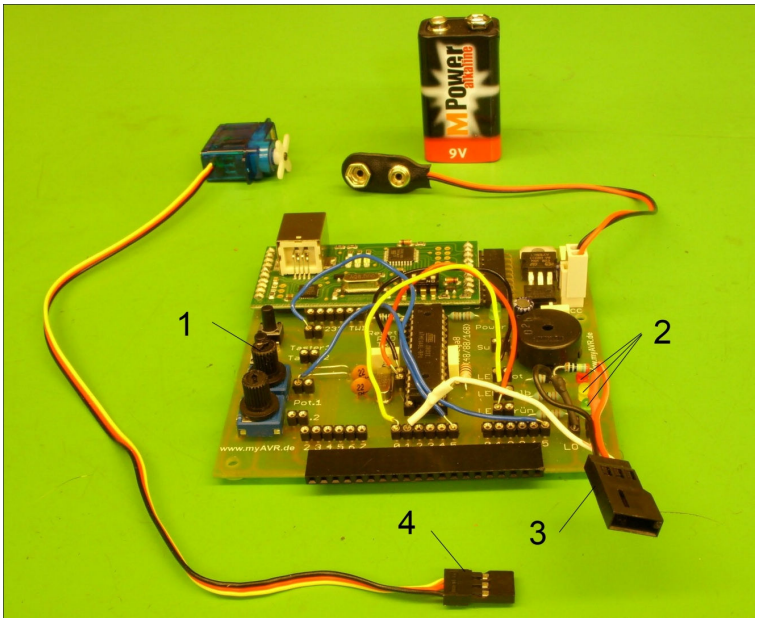


Fig. 1. Standard servo tester

1: Potentiometer (Control)
2: LEDs (left, middle, right)
3: Plug for servo
4: Standard Servo

An optional 3-Digit display can be attached to this myAVR board, giving the precise pulse width

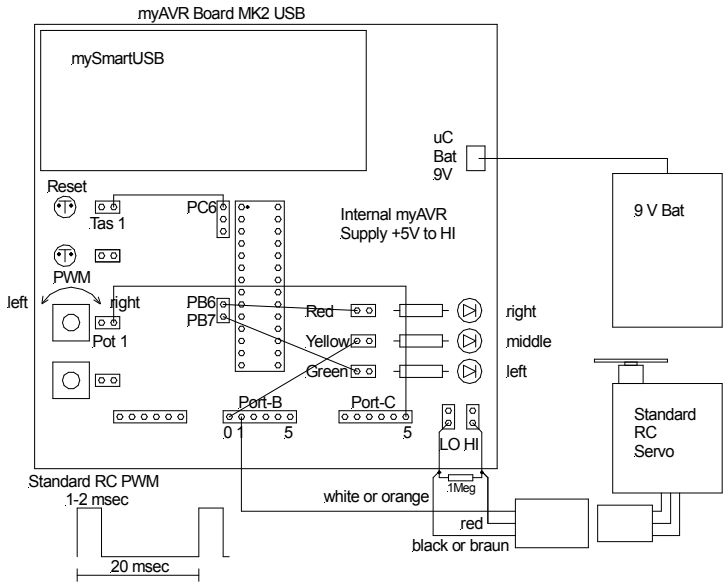


Fig. 2. Connection on myAVR Board MK2 USB for a standard servo.

The plug for the servo is connected to LO (Gnd) and HI (+5V) by help of a 1MOhm resistor.

The servo follows the control by the potentiometer; the three LED's displays the setting.

2.2. Preparing the software

- Copy the original C-program into the folder “PWM_Test_Gen_Docs”
- Start AVR Studio, create a new project “Servo_tester” in the location “PWM_Test_Gen_Docs” and build the hex-file following the former tutorial
- This hex-file is now created in the folder “default” in “PWM_Test_Gen_Docs” with the name “Servo_tester” (extension ”.hex” is not clearly visible)
- Copy this hex file into “PWM_Test_Gen_Docs” and rename it as “Servo_tester_Hex_Dat”
- At present, there is the old hex file on your uC. You might store that old file, which makes sense if you will later reprogram your ESC, thus:
- Make a copy of “Servo_tester_Hex_Dat” and store it the folder “PWM_Test_Gen_Docs” and rename it as “Old_SW_LED_Hex_Dat” (but only if this name is correct with the actual uC-content, else give another name)

2.3. Reading the existing old and flashing the new software for the servo tester

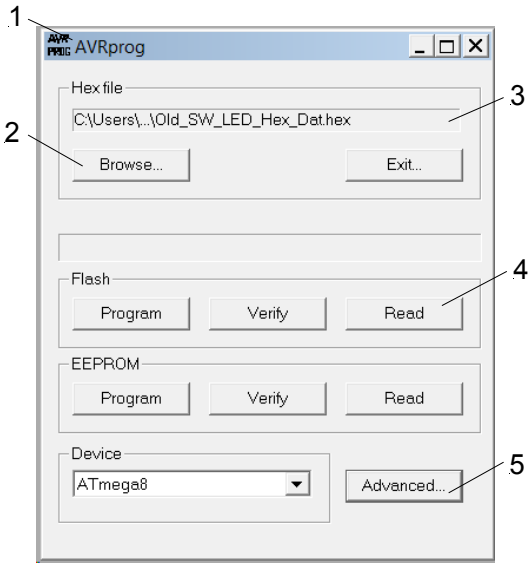


Fig. 3. Reading old hex file
1. Connect the board to the PC, open “Tools” and AVRprog (first line)
(Hint: if you see a message: “connection failed”, disconnect the USB cable and plug it in again)
2. Browse to… \PWM_Test_Gen_Docs\ Old_SW_LED_Hex_Dat
3. Check path: Old_SW_LED_Hex_Dat
4: In “Flash” click “Read”.
The program asks you: This file already exists. Replace existing file? Click “Yes”
Now your Old_SW_LED_Hex_Dat is stored for later use!
5. Push now the button “Advanced”

2.4. Read and write the fuse correct settings

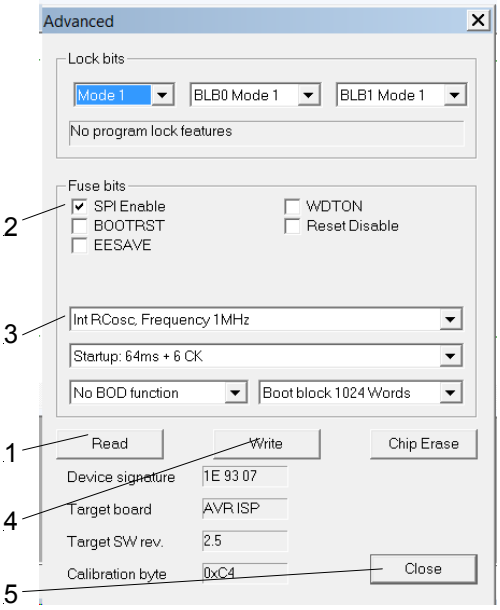


Fig. 4. Fuse Bits

1: Click “Read” to look at the momentary fuse settings on your Atmega8 with the already existing old program

Do change only the need parameters!

2: Check or set SPI Enable
3: Check or set the oscillator conditions:
- Standard is Int RCosc Freq. 1MHz
“Start-up time: 64ms +6k CK”

4: Click “Write” (if you have changed something)
5: Click “Close”

2.5. Flash the Servo Tester Hex program

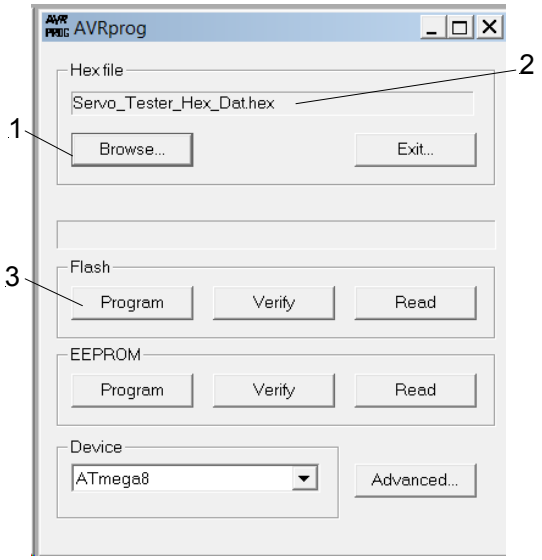


Fig. 5. Load and flash the program "Servo_tester_Hex_Dat"

1: Browse to ….\PWM_Test_Gen_Docs\ Servo_tester_Hex_Dat

2: Check the path for this Hex file again

3: In "Flash" click "Program"

You will notice a blue running bar on the AVRprog screen and the little green LED's on mySmart USB are blinking

Programming done, congratulations!

Connect now a standard servo and a 9V battery as shown in Fig.2
The servo should follow the control of the potentiometer and the 3 LED display the setting.

If the servo tester works as intended you may connect a standard (NOT modified) ESC with BL.

3. Fast PWM signal generator for modified ESC
3.1. Software

The original C-program by Oliver needs only minor changes for fast PWM.
You may make the changes listed below by your own or flash the hex file:
"Fast_PWM_Gen_Hex_Dat" in the folder "BL-modification tutorials" within:
http://ufo-doctor.ch

```
//Fast PWM signal generator
//based on Servortester V 0.5.1 by Oliver Mezger 23.02.2010
//C program, internal oscillator fuse bit set on 1 MHz

//following 3 lines have to be found and changed:

//Original standard PWM by Oliver:
const: unsigned char SMIN = 100, mitte = 150, SMAX = 200;
const:  n = wert / messungen / 4 + 30;
const:  ICR1 = 19999; // Signal alle 20 ms;

//Changes for fast PWM by UFO-Doctor:
const: unsigned char SMIN = 1, mitte = 34, SMAX = 67;
const:  n = wert / messungen / 15;
const:  ICR1 = 680; // Signal alle 0.68 ms;
```

Follow the former steps 2.2. to 2.5, but now for this "Fast_PWM_Gen_Hex_Dat"
This is a good exercise for the next tutorial!

3.2. Hardware

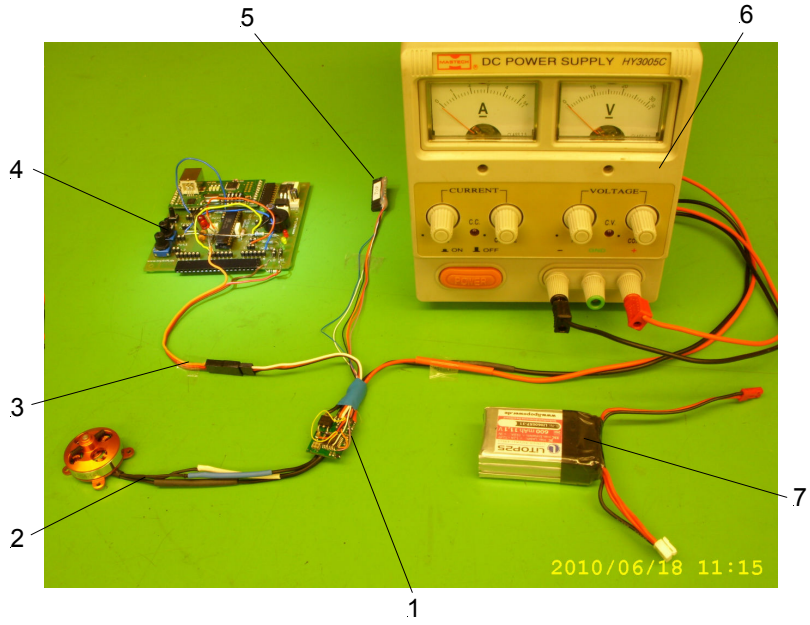


Fig. 6. First test set-up with a modified ESC

1: Modified ESC
2: BL-motor
3: Standard RC-plug/adapter
5: ICP-adapter, but disconnected
6: Power supply with current limiter

If the system works fine, connect the:

7: Lipo Battery

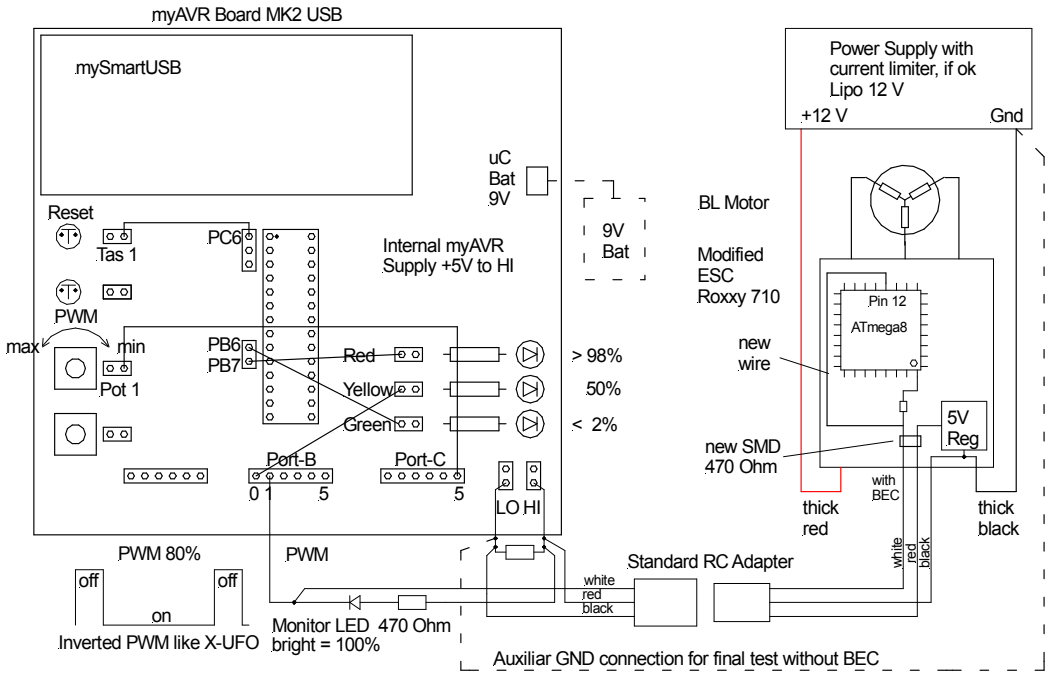


Fig. 7. Circuit of the fast PWM signal generator with modified ESC and BL motor
The monitor LED shows the power setting by dimming, please note that the minimum power setting is with the potentiometer turned fully clockwise!

First test: programmed ESC (see tutorial 4) with a new wire from input to Pin 12, the new resistor is not required yet, but has no influence.

Second test: Standard RC adapter removed, leaving the white input cable only. Now an auxiliary ground connection and a 9V battery is required (dashed lines)

Note: the new 470 Ohm SMD pull-up resistor is required only for the modified X-UFO. It can be attached already, but has no function in this test set-up.

4. Appendix: original C-program for a standard servo tester

Note: the 3 lines with bold letters must be changed for fast PWM!

```
// Servotester V 0.5.1 (c) Oliver Mezger 23.2.2010
#include <avr/io.h>             // Definitionen laden
#include <util/delay.h>         // Delay-Bibliothek laden
#include <avr/interrupt.h>

// Zuordnungen zu Ports
#define SegmenteIn          PIND
#define SegmenteOut         PORTD
#define SegmenteDdr         DDRD
#define SteuerungIn         PINB
#define SteuerungOut        PORTB
#define SteuerungDdr        DDRB
#define StelleIn            PINC
#define StelleOut           PORTC
#define StelleDdr           DDRC

const unsigned char bcd_7[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
const unsigned char SMIN = 100, mitte = 150, SMAX = 200;
const unsigned char mtoleranz   = 3;
const unsigned char LEDmask   = 0b00111110;
const unsigned char LEDrot      = 0b01000000;
const unsigned char LEDgn       = 0b10000000;
const unsigned char LEDblau     = 1;
volatile unsigned char anzeige[3] ={0,0,0};

ISR(TIMER0_OVF_vect){
  static unsigned char stelle=6,dzeiger=0;
  SegmenteOut = bcd_7[anzeige[dzeiger++]];        // Zeile laden
  if (dzeiger == 2) SegmenteOut &= 0b01111111;  // Dezimalpunkt setzen
  StelleOut = (StelleOut & 0b11111000) | stelle;    // Stelle anschalten
  stelle = (stelle * 2 + 1) & 0b111;                // naechste Stelle neg. Logik
  if (dzeiger >2){
    stelle = 6;                                     // 110 wegen neg. Logik
    dzeiger = 0;
  }
}
ISR(ADC_vect){
  static unsigned int wert = 0;
  static unsigned char i = 0,blink = 0,dunkel = 0;
  const unsigned int messungen = 60;                // mehrere Messungen mitteln
  unsigned int n;
  wert += ADC;
  i++;
  if (i >= messungen){
    i = 0;
    n = wert / messungen / 4 + 30;
    wert = 0;
    OCR1A = n * 10;
    blink++;                                        // Blinken erzeugen
    if (blink > 2){
      dunkel = ~dunkel;
      blink = 0;
    }
    if (n <= mitte-mtoleranz) // Binaerer Baum fuer 5 Faelle
      if (n < SMIN)
        SteuerungOut = (SteuerungOut & LEDmask) | LEDgn;            // unter Minimum
      else
        SteuerungOut = (SteuerungOut & LEDmask) | (LEDgn & dunkel); // zwischen Minimum und unterer Toleranz
    else if (n < mitte+mtoleranz)
      SteuerungOut = (SteuerungOut & LEDmask) | LEDblau;            // in der Neutral-Toleranz
    else if (n <= SMAX)
      SteuerungOut = (SteuerungOut & LEDmask) | (LEDrot & dunkel);  // zwischen Toleranz und Maximum
    else
      SteuerungOut = (SteuerungOut & LEDmask) | LEDrot;             // ueber Maximum
    anzeige[0] = n % 10;
    n /= 10;
    anzeige[1] = n % 10;
    anzeige[2] = n / 10;
  }
}
```

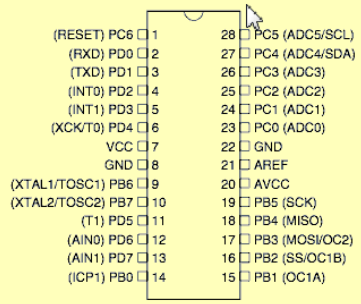
```
int main(){
  TCCR0 = 2;                        // Systemtakt / 8
  TIMSK |= (1<<TOIE0);              // Timerinterrupt frei geben
  SegmenteDdr=0xff;                 // Ausgang
  StelleDdr=0x07;                   // Ausgang
  ADMUX = 5;                        // ADWandler 5
  ADCSRA = 0b11111100;               // ADWandler Free Running, Interrupt frei
  TCCR1A = 0b10000010;              // Timer1 Ausgang OC1A, Fast PWM
  TCCR1B = 0b00011001;              // Keine Vorteilung Timer mit CPU-CLK
  ICR1 = 19999;                      // Signal alle 20 ms
  SteuerungDdr = 0b11000011;        // Ausgänge LED und Servo
  sei();                            // globale Interruptfreigabe
  while(1){                         // Endlosschleife

  }
  return 0;
}
```

5. Connections to ATmega8 PDIP for the standard servo tester by Oliver

7. ATmega 8

PDIP



Belegung	Bedeutung	Pin		Pin	Bedeutung	Belegung
Reset	(Reset) PC6	1		28	PC5 (ADC5)	Poti
a	(RXD) PD0	2		27	PC4 (ADC4)	
b	(TXD) PD1	3		26	PC3 (ADC3)	
c	(INT0) PD2	4		25	PC2 (ADC2)	Dig1
d	(INT1) PD3	5		24	PC1 (ADC1)	Dig2
e	(T0) PD4	6		23	PC0 (ADC0)	Dig3
	VCC	7		22	GND	
	GND	8		21	AREF	
LED_{Rot}	(XTAL1) PB6	9		20	AVCC	
$LED_{Grün}$	(XTAL2) PB7	10		19	PB5 (SCK)	Taster
f	(T1) PD5	11		18	PB4 (MISO)	
g	(AIN0) PD6	12		17	PB3 (MOSI/OC2)	
DP	(AIN1) PD7	13		16	PB2 (SS/OC1B)	
LED_{Blau}	(ICP1) PB0	14		15	PB1 (OC1A)	Servo

ATMEGA8

Please notice that the pin numbers and functions of the dual-in-line package ATMEGA8 **PDIP** here are different from the SMD package ATMEGA8 **TQFP** as used for the ESC Roxxy 710!