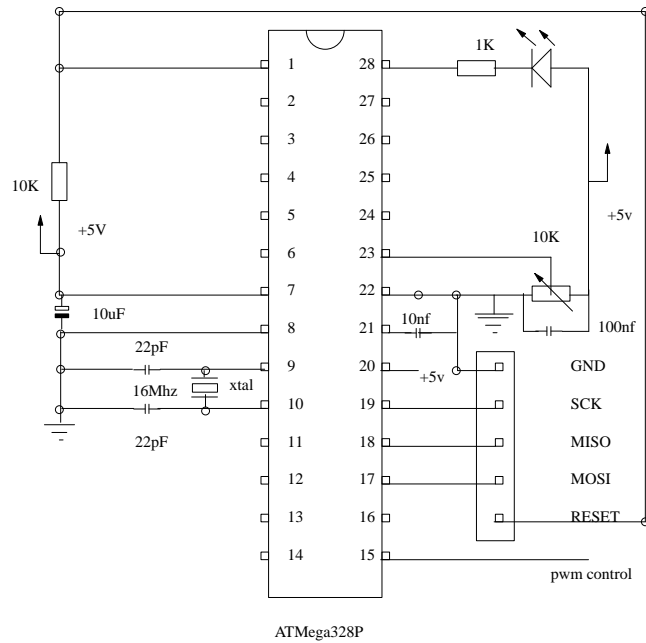


Using an ATmega328p to test a servo

- here we put together
 - an LED as an output device
 - a servo as an output device
 - and a potentiometer as an input device
 - all programmed in C

Circuit diagram



Circuit diagram

Circuit diagram

```
#define F_CPU 16000000UL

#include <inttypes.h>
#include <avr/io.h>
#include <avr/pgmspace.h>
#include <avr/eeprom.h>
#include <avr/interrupt.h>
#include <stdio.h>
#include <util/delay.h>

#if !defined(TRUE)
# define TRUE (1==1)
#endif

#if !defined(FALSE)
# define FALSE (1==0)
#endif
```

```
#define ePRESCALER8
#if defined(ePRESCALER8)
# define eICR1value 39936
# define ePRESCALEBITS (_BV(CS11))
# define eLOW 2048
# define eHIGH 3968
# define eDELAYms 3
#endif

#if defined(ePRESCALER256)
# define eICR1value 1248
# define ePRESCALEBITS (_BV(CS12))
# define eLOW 64
# define eHIGH 124
# define eDELAYms 100
#endif

#if defined(ePRESCALER1024)
# define eICR1value 1248*4
# define ePRESCALEBITS (_BV(CS10) | _BV(CS12))
# define eLOW 16
/* not a good choice as eHIGH is not exact 124/4 is not a
# define eHIGH 31
# define eDELAYms 400
#endif
```

Circuit diagram

```

/*
 * initServo - initialises timer 1 for fast PWM operation
 *             servo is enabled on B1.
 */

void initServo (void)
{
  /* initialize TMR1 (PWM) */
  TCCR1A = _BV(COM1A1) | _BV(COM1B1) | _BV(WGM11); /* cle
  TCCR1B = _BV(WGM12) | _BV(WGM13) | ePRESCALEBITS; /* se
  ICR1 = eICR1value;
  OCR1A = -1; /* off */
  OCR1B = -1; /* off */
  DDRB |= _BV(1) | _BV(2); /* output on PB1 and PB2 */
}

```

Circuit diagram

```

/*
 * set_servo1 - sets the pwm length for servo1. Value s
 */

void inline set_servo1 (int pwm)
{
  OCR1A = pwm;
}

/*
 * set_servo2 - sets the pwm length for servo1. Value s
 */

void inline set_servo2 (int pwm)
{
  OCR1B = pwm;
}

```

Circuit diagram

```

static void initLed (void)
{
  asm volatile ("sbi 7, 5" );
}

static void turnLED (int value)
{
  if (value)
    asm volatile ("cbi 8, 5");
  else
    asm volatile ("sbi 8, 5");
}

```

Circuit diagram

```

void initADC (void)
{
  /* 16Mhz/128 = 125Khz the ADC reference clock */
  ADCSRA |= ((1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0));
  /* use voltage reference from Avcc */
  ADMUX |= (1<<REFS0);
  /* Turn on the ADC */
  ADCSRA |= (1<<ADEN);
  /* Now start an A to D conversion as the initial one is
  ADCSRA |= (1<<ADSC);
}

```

Circuit diagram

```

uint16_t readADC (uint8_t channel)
{
  /* mask out the channel value */
  ADMUX &= 0xf0;
  /* set the channel bits */
  ADMUX |= channel;
  /* tell the hardware to start a new A to D conversion */
  ADCSRA |= (1<<ADSC);
  /* Wait for the A to D conversion to complete */
  while (ADCSRA & (1<<ADSC))
  ;
  /* and return the A to D value (0..1023) as a 16 bit un.
  return ADCW;
}

```

Circuit diagram

```

main()
{
  uint16_t adc;
  int pos;

  initLed ();
  initADC ();
  initServo ();

  while (TRUE) {
    adc = readADC (0);
    _delay_ms (eDELAYms);
    pos = adc * 2 + eLOW;
    if (pos > eHIGH)
      pos = eHIGH;
    set_servo (pos);

    if (adc > 512)
      turnLED (0);
    else
      turnLED (1);
  }
}

```

Compile our code

```

$ avr-gcc -mmcu=atmega328 atod.c -o atod.elf
$ avr-objcopy -R .eeprom -O ihex atod.elf atod.hex

```

Download the executable

```

sudo avrdude -p m328p -U lfuse:w:0xff:m -U hfuse:w:0xd9:m
-U efuse:w:0x07:m -c stk500v2 -e -v -U flash:w:atod.hex

```

Servo light meter

- obviously we could substitute the potentiometer for an LDR and introduce a limiting resistor
 - which turns the servo according to the light level
 - code remains the same

Circuit diagram

