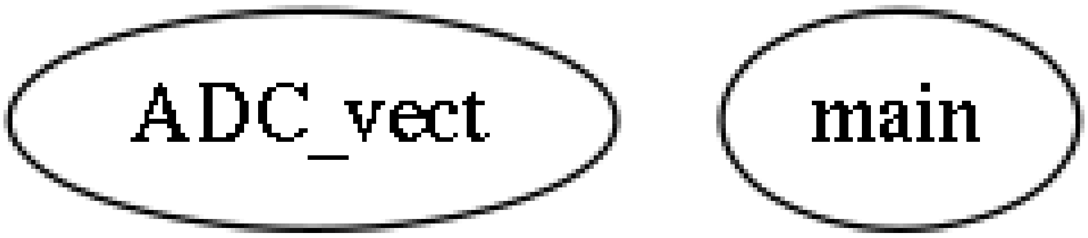
October 3, 2015 at 08:08

1. Introduction.

This converts the voltage from a potentiometer to a PWM output for heavier loads. Heavy loads just need a switching devide like a transistor.



Extensive use was made of the datasheet, Atmel “Atmel ATtiny25, ATtiny45, ATtiny85 Datasheet” Rev. 2586QAVR08/2013 (Tue 06 Aug 2013 03:19:12 PM EDT) and “AVR130: Setup and Use the AVR Timers” Rev. 2505AAVR02/02.

⟨ Include 3 ⟩

2. "F_CPU" is used to convey the Trinket clock rate.

```
#define F_CPU 8000000UL
```

3. ⟨ Include 3 ⟩ ≡

```
#include <avr/io.h>    /* need some port access */
#include <avr/interrupt.h>    /* have need of an interrupt */
#include <avr/sleep.h>    /* have need of sleep */
#include <stdlib.h>
#include <stdint.h>
  int main(void)
      {
      ⟨ Initialize ADC 11 ⟩
      ⟨ Initialize Timer 13 ⟩
      uint8_t pot = 0;
```

This code is used in section 1.

4.

```
  sei();
```

5. Rather than burning loops, waiting for something to happen, the “sleep” mode is used. The specific type of sleep is ‘idle’. In idle, execution stops but timers continue and ADC conversion begins.

```
  for ( ; ; )
  {
```

6. The selected pot toggles at the beginning of each loop.

```
  pot = (pot ≡ 0) ? 1 : 0;
```

7. Next a potentiometer is selected by setting the MUX.

```
if (pot ≡ 0) ADMUX = (ADMUX & #F0U) | #03U;     /* ADC3 at PB3 */
else ADMUX = (ADMUX & #F0U) | #02U;     /* ADC2 at PB4 */
```

8. Now we wait in “idle” for an ADC conversion (7.1.1).

```
sleep_mode( );
```

9. If execution arrives here, some interrupt has been detected. Usualy we would check for which interrupt but we will assume that it’s the ADC.

Next the pot position, determined by the ADC, is written to the timer of the respective PWM output.

```
if (pot ≡ 0) OCR0A = ADCH;     /* PB0 PWM */
else OCR0B = ADCH;     /* PB1 PWM */
}     /* end for */
return 0; }     /* end main() */
```

10. These are the configuration blocks.

11. ⟨ Initialize ADC 11 ⟩ ≡

```
{
  ADMUX |= (1 ≪ ADLAR);     /∗ Left adjust for 8 bits ∗/
  ADCSRA |= (1 ≪ ADPS2);    /∗ 500 kHz ADC clock ∗/
  ADCSRA |= (1 ≪ ADIE);     /∗ Interrupt on completion ∗/
  ADCSRA |= (1 ≪ ADEN);     /∗ Enable ADC (prescaler starts up) ∗/     /∗ DIDR0 17.13.5 ∗/
  DIDR0 |= (1 ≪ ADC2D);     /∗ Disable digital on ADC2 ∗/
  DIDR0 |= (1 ≪ ADC3D);     /∗ Disable digital on ADC3 ∗/
}
```

This code is used in section 3.

12. PWM setup isn't too scary. Timer Count 0 is configured for "Phase Correct" PWM which, according to the datasheet, is preferred for motor control. OC0A and OC0B are set to clear on a match which creates a non-inverting PWM.

13. ⟨ Initialize Timer 13 ⟩ ≡

```
{    /∗ 15.9.1 TCCR0A  Timer/Counter Control Register A ∗/
  TCCR0A |= (1 ≪ WGM00);     /∗ Phase correct, mode 1 of PWM (table 15-9) ∗/
  TCCR0A |= (1 ≪ COM0A1);    /∗ Set/Clear on Comparator A match (table 15-4) ∗/
  TCCR0A &= ∼(1 ≪ COM0A0);    /∗ Set on Comparator A match (table 15-4) ∗/
  TCCR0A |= (1 ≪ COM0B1);    /∗ Set/Clear on Comparator B match (table 15-7) ∗/
  TCCR0A &= ∼(1 ≪ COM0B0);    /∗ Set on Comparator B match (table 15-7) ∗/
    /∗ 15.9.2 TCCR0B  Timer/Counter Control Register B ∗/
  TCCR0B |= (1 ≪ CS02);     /∗ Prescaler set to clk/8 (table 15-9) ∗/
    /∗ 14.4.9 DDRD  The Port D Data Direction Register ∗/
  DDRB |= (1 ≪ DDB0);     /∗ Data direction to output (sec 14.3.3) ∗/
  DDRB |= (1 ≪ DDB1);     /∗ Data direction to output (sec 14.3.3) ∗/
}
```

This code is used in section 3.

14. Interrupt Handling.

```
ISR(ADC_vect)
{
}
```