AVR: Session 3

Timers, Interrupts and Counters

Timers and Counters

- As the name suggest, keeps track of time elapsed since start of microcontroller.
- All Microcontrollers have a clock inside them. (Frequency: 1MHz!!!)
- Counters tick with the clock. (8 bit or 16 bit)
- Prescaling: Simply a way to make the counter skip a certain number of clock ticks

Related Registers

Timer

- TCCR0 / TCCR1(Timer/Counter Control Register)
    - FOC (Force Output Compare)
    - WGM (Waveform Generation Mode)
    - COM (Compare Match Output)
    - CS (Clock Select)
- TCNT1 (holds Timer Count)

LED Blink: Timer and Counter Style

- We want LED to blink every 1/5th of second.
- When counter reaches 200,000, instruct the microcontroller to toggle the output.
- But our counter is 16 bit => we cannot count beyond 65536.

Prescaling to the rescue

Deciding the Prescalar:

- 1,000,000 Ticks per second, Maximum Count till 65535.
- Available Prescalars: 8,64,256 and1024
    - 8 Prescalar: 125000 Counts/Second
    - 64 Prescalar: 15625 Counts/Second
    - 256 Prescalar: 3906 Counts/Second
    - 1024 Prescalar: 976 Counts/Second

Let us use 64 Prescalar => Toggle with every 3205 counts

Pseudo Code

```
// Include header files
int main(void)
{
  /* Designate pin as output and pull it down to LOW */
  /* Select the Prescalar */
  //Start of MAIN LOOP
     if (Timer Count > 3204)
    {
      /* Set Timer Count to Zero */
      /* Toggle LED */
  // End of MAIN LOOP
}
```

Actual Code

```
#include <avr/io.h>
int main(void)
{
 PORTB |= 1<<PINB0;
 DDRB |= 1<<PINB0;
 TCCR1B |= (1<<CS10 | 1<<CS11); //Page No. 110
```

```
while(true)
{
if(TCNT1 > 3204)
  {
  TCNT1=0;
  PORTB ^= 1<<PINB0;
  }
}
```

Interrupts

- Exactly like it sounds like: When X event occurs, stops the main code and executes a particular block of code.
- Event can be Counter reaching a number, pin changing state, Analog to Digital Conversion, Serial Communication, PWM
- We will use Interrupts to make LED blink example more efficient.

Timer Interrupts

- We will ask TCNT1 to a number to match.
- This number will be stored in OCR1A/ OCR1B
- When the number is matched we want to put the counter back to zero - CTC
- Timer/Counter will need to know we are using the Interrupt feature - TIMSK register

Pseudo Code

```
// Include header files
int main(void)
{
  /* Enable Global Interrupt */
  /* Designate pin as output and pull it down to LOW */

  /* Select the Prescalar  and Enable CTC Mode*/
  /* Enable use of OCR1A Register */
  //Start of MAIN LOOP
  // End of MAIN LOOP
}
//Interrupt  Code
{
/*Toggle LED*/
}
```

Actual Code - Part 1

```
#include <avr/io.h>
#include <avr/interrupt.h>

int main(void)

{
  sei();
  DDRB |= 1<<PINB0;
  TCCR1B |= 1<<CS10 | 1<<CS11 | 1<<WGM12; // Page No. 110
  TIMSK |= 1<<OCIE1A; // Page No. 112
  OCR1A = 15624;
  while(1)
  {
  }
}
```

Actual Code - Part 2

```
ISR(TIMER1_COMPA_vect) // Page No. 44
{
  PORTB ^= 1<<PINB0;
}
```

PWM - Application of Timers and Interrupts

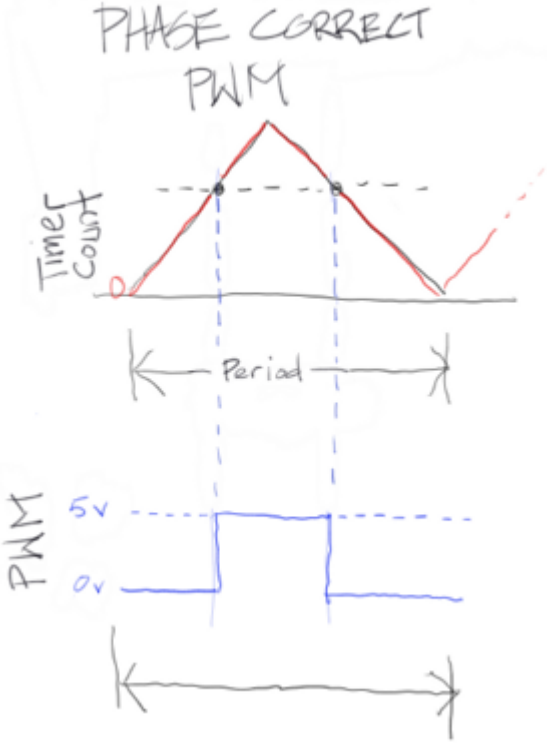
- Finds an important application

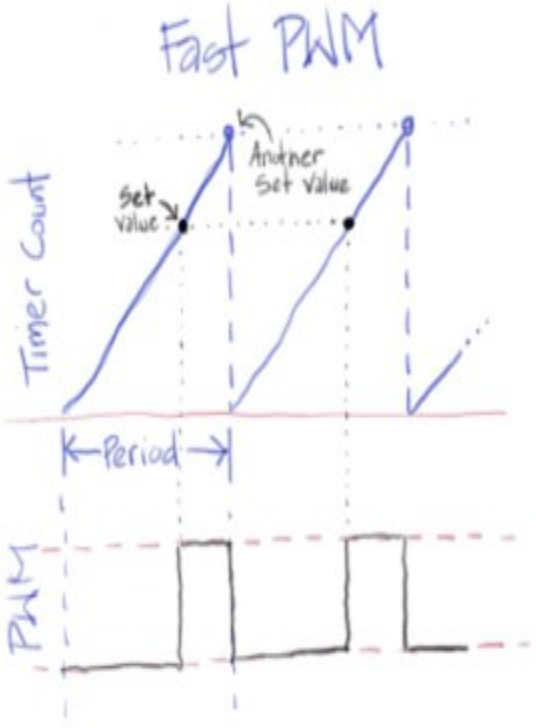
in motor control.

- Types of PWM:
  - Non-inverted
  - Inverted
  - Phase Correct PWM

Waveform Generation: Types

- Non Inverted
- Inverted
- Phase Correct





PWM to control a servo

- Servo needs a PWM with 20ms period and high time from 0.9-2.1ms.
- We will use the inverting mode here.
- No Prescaling => 20,000 counts as period (Set ICR1 as 20,000)
- Set OCR1A = ICR1 - (pulse width)

Pseudo Code

```
// Include header files
int main(void)
{
  /* Enable Global Interrupt */
  /* Designate pin as output */

  /* Select the Prescalar and Enable CTC Mode, Set ICR1 as the top of
  Waveform */
  /* Enable use of OCR1A Register */
  //Start of MAIN LOOP
  // End of MAIN LOOP
}
//Interrupt  Code
{
/*Toggle LED*/
```

Actual Code

```
#include <avr/io.h>
#include <util/delay.h>
int main(void)
{
  DDRD |= 0xFF;
  TCCR1A |= 1<<WGM11 | 1<<COM1A1 | 1<<COM1A0; // Page No.
  108
  TCCR1B |= 1<<WGM13 | 1<<WGM12 | 1<<CS10; // Page No. 110
  ICR1 = 19999;
  OCR1A = ICR1 - 2000; //18000
```

```
  while (1)
  {
    OCR1A = ICR1 - 800;
    _delay_ms(100);
    OCR1A = ICR1 - 2200;
    _delay_ms(100);
  }
}
```