Experiment #9 Counter/Timers – Interrupts

References:

https://www.robotshop.com/letsmakerobots/arduino-101-timers-and-interrupts

Timer-counters are special circuits and registers which can be programmed to count the number of clock cycles during an interval or respond at periodic intervals independent of the threads of a running program. They work by interrupting the main program when a counting interval occurs and then executing what is called an interrupt service routine (ISR). A properly written ISR is short enough so that its execution time does not adversely affect the main program.

Arduino mega microcontrollers have 5 16-bit counter/timers – see references. 16-bit counter/timers have a maximum count of 65,536 counts equal about 4ms on a 16 Mhz clock with no prescaler. This is ideal for driving stepper motors at 1000 steps per second.

Functions

To program a counter/timer interrupts requires three components; a general purpose setup function which is called every time you execute a move operation, a counter/timer setup function for each counter timer you wish to use, and a separate ISR for each counter timer.

The varsInt function is the general purpose setup function. It first calls the counter/timer setup functions. Then it sets the count interval by setting the output compare register for the counter/timer using OCR#A. In this example ctc4 is used for the master wheels and ctc3 is used for the slave wheels

* Variable speed with interrupts

```
void varsInt(byte dir, float dist, long del, float ratio, byte master, byte slave) {
  ctc3_setup();
  ctc4_setup();
  PORTL = dir;
  float stepf = dist*steps_per_inch;
  masterWheels = master;
  slaveWheels = slave;

  noInterrupts();
  //OCR4A = 16 * del - 1;//setup speed for master
  OCR4A = 16 * del;
  TCNT4 = 0;//reset
  float temp = del * ratio;
  long slaveDelay = temp;
  //OCR3A = slaveDelay * 16 - 1;//setup speed for slave
  OCR3A = slaveDelay * 16;
  TCNT3 = 0;//reset
  steps = stepf;
  steps_counter = 0;  //reset step counter
```

```
  interrupts();

}
```

The counter/timer setup functions write individual bits into the counter timer registers to setup mode, clock prescaling, and enable interrupts.

To initialize counter-timers 3 and 4 for count to compare mode at full clock speed

```
/*********************SETUP**********************/

void ctc4_setup() {
  noInterrupts();
  TCCR4A = 0;  // clear counter control register
  TCCR4B = 0;
  TCNT4 = 0;

  OCR4A = 16000; // compare match register – 1000 usecond delay
        // countCompareValue = delayInMicroseconds * 16
        // countCompareValue = 16000000 / prescaler / desired frequency
  TCCR4B |= (1 << WGM42); // count to compare mode
  TCCR4B |= (1 << CS40); // 1 prescaler
  TIMSK4 |= (1 << OCIE4A); // enable timer compare interrupt
  interrupts();
}
void ctc3_setup() {
  noInterrupts();
  TCCR3A = 0;  // clear counter control register
  TCCR3B = 0;
  TCNT3 = 0;
 // OCR3A = 16000; // compare match register – 1000 usecond delay
        // countCompareValue = delayinmicroseconds * 16
        // countCompareValue = 16000000 / prescaler / desired frequency
  TCCR3B |= (1 << WGM32); // count to compare mode
  TCCR3B |= (1 << CS30); // 1 prescaler
  TIMSK3 |= (1 << OCIE3A); // enable timer compare interrupt
  interrupts();
}
```

ISRs have specific purposes and names. They behave similar to functions with severe limitations. You cannot pass variables to an ISR. They do not return any values. All variables used in an ISR must be declared volatile with global scope. Since the variables are global scope, any function can change their values. Care must be exercised in manipulating these variables.

The ISRs only function is to step the motors

```
//********************Set upt ISR
//3: Slave
ISR(TIMER3_COMPA_vect) { // timer compare ISR
  if (steps > 0) {
    PORTL ^= slaveWheels;
    PORTL ^= slaveWheels;
    steps_counter++;
  }
}

//4: Master
ISR(TIMER4_COMPA_vect) { // timer compare ISR
  if (steps > 0) {
    PORTL ^= masterWheels;
    PORTL ^= masterWheels;
    steps--;
  }
}

//********************End Set upt ISR
```