# Interrupts and Timers on Arduino Uno Example

ATmega328P Data Sheet - https://content.arduino.cc/assets/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

**Using Timer1**

```
static uint8_t saveTCCR1A, saveTCCR1B;

static inline void capture_init(void)
{
  // save existing timer settings
  saveTCCR1A = TCCR1A;
  saveTCCR1B = TCCR1B;

  // initialize timer1
  TCCR1B = 0;
  TCCR1A = 0;

  //Initialize counter value to 0
  TCNT1 = 0;

  //Set Timer/Counter1 Interrupt Flag Register
  //ICF1: Timer/Counter1, Input Capture Flag
  //TOV1: Timer/Counter1, Overflow Flag
  TIFR1 = (1<<ICF1) | (1<<TOV1);

  //Set Interrupt Mask Register
  // ICIE1: Timer/Counter1, Input Capture Interrupt Enable
  // TOIE1: Timer/Counter1 Overflow Interrupt Enable
  TIMSK1 = (1<<ICIE1) | (1<<TOIE1);}
}

static inline void capture_start(void)
{
  //Set Timer/Counter1 Control Register B
  // ICNC1: Input Capture Noise Canceler
  // ICES1: Input Capture Edge Select
  // CS10: Clock Select - clkIo/1 (no prescaling)
  TCCR1B = (1<<ICNC1) | (1<<ICES1) | (1<<CS10);
}
static inline uint16_t capture_read(void)
{
  //input capture register
  return ICR1;
}
static inline uint8_t capture_overflow(void)
{
  //Set Timer/Counter1 Interrupt Flag Register
  // TOV1: Timer/Counter1, Overflow Flag
  return TIFR1 & (1<<TOV1);
}
static inline void capture_overflow_reset(void)
{
   //Set Timer/Counter1 Interrupt Flag Register
   // TOV1: Timer/Counter1, Overflow Flag
   TIFR1 = (1<<TOV1);
}
static inline void capture_shutdown(void)
{
  TCCR1B = 0;
  TIMSK1 = 0;
  //Restore old timers
  TCCR1A = saveTCCR1A;
  TCCR1B = saveTCCR1B;
}
```

```
#define TIMER_OVERFLOW_VECTOR TIMER1_OVF_vect
#define TIMER_CAPTURE_VECTOR TIMER1_CAPT_vect
#define FREQMEASURE_BUFFER_LEN 12
static volatile uint32_t buffer_value[FREQMEASURE_BUFFER_LEN];
static volatile uint8_t buffer_head;
static volatile uint8_t buffer_tail;
static uint16_t capture_msw;
static uint32_t capture_previous;
ISR(TIMER_OVERFLOW_VECTOR)
{
  capture_msw++;
}

ISR(TIMER_CAPTURE_VECTOR)
{
  uint16_t capture_lsw;
  uint32_t capture, period;
  uint8_t i;

  // get the timer capture
  capture_lsw = capture_read();
  // Handle the case where but capture and overflow interrupts were pending
  // (eg, interrupts were disabled for a while), or where the overflow occurred
  // while this ISR was starting up. However, if we read a 16 bit number that
  // is very close to overflow, then ignore any overflow since it probably
  // just happened.
  if (capture_overflow() && capture_lsw < 0xFF00) {
    capture_overflow_reset();
    capture_msw++;
  }

  // compute the waveform period
  capture = ((uint32_t)capture_msw << 16) | capture_lsw;
  period = capture - capture_previous;
  capture_previous = capture;
  // store it into the buffer
  i = buffer_head + 1;
  if (i >= FREQMEASURE_BUFFER_LEN) i = 0;
  if (i != buffer_tail) {
    buffer_value[i] = period;
    buffer_head = i;
  }
}
```