Introduction to Robotics

Lecture 4: Mastering MCU Peripherals

8. 10. 2018

ParaDiSe

Today Goal

- learn about MCU Peripherals
- look "under the hood" of the Arduino libraries
- using the knowledge, fix the imperfections from last time

The Bible

ATmega328p Datasheet

Also handy: Arduino Uno pinout

Playing with the Bits

Given `n` and a number `x`:

- set bit n: `x |= (1 << n)`
- reset bit n: `x &= ~(1 << n)`
- read bit n: `x & (1 << n)`

ParaDiSe Note

Given `n` and a number `x`:

- ATmega328p is an 8-bit MCU with no caches
- there is no mutex, use bool
- do not forget volatile keyword!

Register vs. Register

- general purpose register in the core
- peripheral memory mapped to the address space

- for convenience, symbolic names for the registers
- constants for bit positions

GPIO Pins

- no pins 0–13
  - ports B, C, D
  - 8 pins for each port
  - e.g. PC1, PD0, PB7
- 3 control registers: DDRn, PORTn, PINn
  - DDRn controls direction (1 – output, 0 – input)
  - PORTn in output mode: sets value
  - PORTn in input mode enable (1)/disable(0) pullup
  - PINn – read pin value
  - PINn – write 1 = toggle PORTn
- PORTn can be override by a peripheral
- peripheral can read from PINn

GPIO Example

```
#include <Arduino.h>
// const int LED_PIN = 13; -- in Arduino, our pin is PB5
void setup() {
    Serial.begin( 9600 );
    DDRB |= (1 << 5); // pinMode( LED_PIN, OUTPUT );
}

void loop() {
    PORTB |= (1 << 5); // digitalWrite( LED_PIN, LOW );
    delay( 300 );
    PORTB &= ~(1 << 5); // digitalWrite( LED_PIN, HIGH );
    delay( 300 );
}
```

Datasheet Excursion

Timers

Timer consists of:

- a counter register
- a clock source (internal with a prescaler, external)
- a output compare registers

Timers

Timer consists of:

- a counter register
- a clock source (internal with a prescaler, external)
- a output compare registers

There are 3 timers in ATmega328p:

- timer0 – 8-bit (timing, 2 output captures, PWM)
- timer1 – 16-bit (timing, 2 output captures, PWM, input capture)
- timer2 – 8-bit (timing, 2 output captures, PWM)

Timer as... Well Timer

- on each clock tick counter is incremented
- clock source divided by a prescaler (1, 8, 64, 256, 512)
- timer is reset when reaches the top value
  - either natural range of the timer
  - or output compare register
- interrupt on counter top

Note: using timers breaks Arduino's: delay, servo, millis, micros, analogWrite

Timer Example

```
void setup() {
    TCCR0A = (1 << WGM01); // set the timer mode to CTC
    OCR0A = 249; // set the value that you want to count to
    TIMSK0 = (1 << OCIE0A);  // set the ISR COMPA vect
    sei();          //enable interrupts
    TCCR0B = (1 << CS02); // set prescaler to 256 and start the timer
}
void loop() {}

ISR(TIMER0_COMPA_vect) {  // timer0 overflow interrupt
    // Action every 4 milliseconds
}
```

Datasheet Excursion

Timer as PWM generator

Many modes, we describe only the simplest one (Fast PWM).

- compare counter with an output compare register and send it to a pin
- using natural range $\rightarrow$ two PWM signals (synchronised)
- using one compare register for top $\rightarrow$ one PWM signal

Note: The output pin is hard-wired, we cannot change it

PWM Example

```
void setup() {
    pinMode( 5, OUTPUT );  pinMode( 6, OUTPUT ); // configure pins
    // set PWM value; power = n / 255 %
    OCR0A = 0; OCR0B = 20;
    TCCR0A = (1 << WGM00) | (1 << WGM01) | (1 << COM0B1) | (1 << COM0A1);
        // enable Fast PWM mode and enable both outputs
    TCCR0B = (1 << CS02) | (1 << CS00); // start timer at 1 / 1024
}
uint8_t power = 0;
void loop() {
    OCR0A = power++;
    _delay_ms( 50 ); // delay is broken now, include <util/delay.h>
}
```

Datasheet Excursion

Timer as Input Capture

- counter is rising
- when an ege on input is detected
  - store counter value in special register
  - trigger interrupt

Note: Only timer 1 supports this mode.

PWM Example

```
void setup() {
    pinMode( 5, OUTPUT );  pinMode( 6, OUTPUT ); // configure pins
    // set PWM value; power = n / 255 %
    OCR0A = 0; OCR0B = 20;
    TCCR0A = (1 << WGM00) | (1 << WGM01) | (1 << COM0B1) | (1 << COM0A1);
        // enable Fast PWM mode and enable both outputs
    TCCR0B = (1 << CS02) | (1 << CS00); // start timer at 1 / 1024
}
uint8_t power = 0;
void loop() {
    OCR0A = power++;
    delay( 50 );
}
```

Input Capture Example

```
void setup() {
    TCCR1A = 0; // Normal operation
    TCCR1B = (1 << CS12) | (1 << CS10) | (1 << ICES1);
        // set clock source, detect rising edge
    TIMSK1 = (1 << ICIE1); // enable interrupt
    sei();
}
void loop() {}
ISR(TIMER1_CAPT_vect) {
    // ICR1 contains captured value
}
```

Datasheet Excursion

Task 1: Generate PWM for the Motors

Use timer 0 to generate 2 PWM signals for motors.

- use its natural range
- use 1024 prescaler to obtain frequency around 60 Hz
- implement function(s) to set motor power and direction

Task 2: Generate Signal for Servo

Use timer 1 to generate signal for servo.

- make its period 20 ms (choose prescaler and top value)
- implement functions to set servo position
- reminder: 0.5 ms = left, 1.5 ms = center, 2.5 ms = right

Task 3: Read Ultrasonic Sensor With Timer

Use timer 1 to read pulse width of the ultrasonic sensor.

- use input capture mode
- do not break servo signal generator (or at least make it switchable)
- hint: you can change orientation of the edge being detected in the interrupt