(2 pts) Name:

AME 3623: Embedded Real-Time Systems: Final Exam

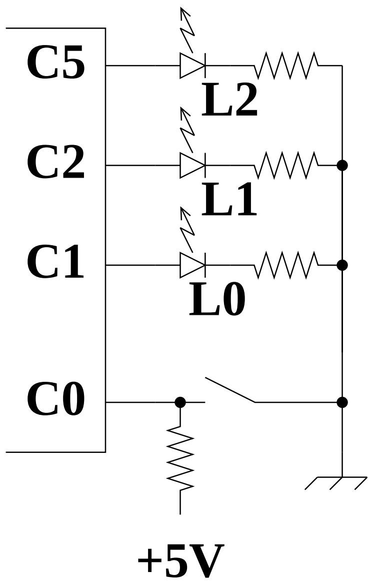
Solution Set

May 6, 2013

Problem	Topic	Max	Grade
0	Name	2	
1	Interrupt Service Routines and Digital I/O	25	
2	Number Representation and Arithmetic	20	
3	Finite State Machines	50	
4	Analog Processing	40	
5	Analog Circuits	20	
6	Microprocessors	20	
7	Serial Communication	25	
Total		202	

1. **Interrupt Service Routines and Digital I/O** (25 pts)

Carefully consider the following circuit:



And consider the following program:

```
ISR(TIMER2_OVF_vect) {
  static uint8_t counter = 0;

  PORTC = PORTC & ~0x26 | counter << 4;
  counter += 1;
}

int main(void) {
  DDRC = 0x26;
  PORTC = 0;
  timer2_config(TIMER2_PRE_32);
  timer2_enable();
  sei();

  while(1) {
  }
}
```

(a) (5 pts) Assuming a system clock of $16MHz$, at what frequency is the timer 2 counter incrementing? (give the ratio with units)

$\frac{16{,}000{,}000}{32}\ tocks/sec$

(b) (5 pts) At what frequency is the timer 2 overflow interrupt being generated? (give the ratio with units)

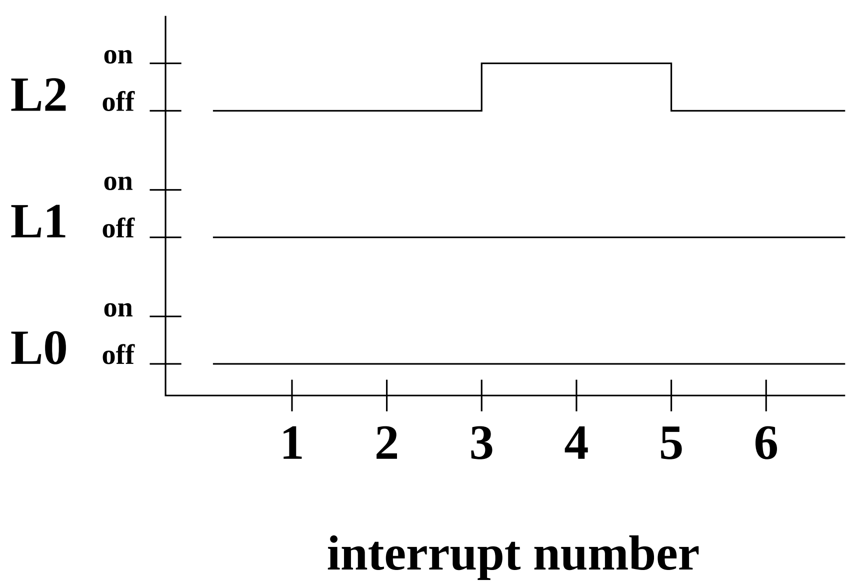
$\frac{16{,}000{,}000}{32 \times 256}\ interrupts/sec$

(c) (5 pts) What is the sequence of values that *counter* takes on for the first 6 interrupts?

0 (before first interrupt), 1, 2, 3, 4, 5, 6.

(d) (10 pts) Show the state of LEDs 0, 1, 2 as a function of interrupt number for interrupts 1 through 6.

Note: for the first interrupt, the counter is zero while deciding the state of the LEDs.
Note 2: After the shift, LEDs L0 and L1 are lined up with zeros (always) and L2 is lined up with bit 1 of the counter.



2. **Number Representation and Arithmetic** (20 pts)

(a) (5 pts) What is the decimal equivalent of hexadecimal 87? Show your work.

*8 * 16 + 7 = 135*

(b) (5 pts) Take the two's complement (the negative) of hexadecimal 87 (assume an 16-bit, signed representation). Show your work.

```
0x87 = 0000 0000 1000 0111

flip: 1111 1111 0111 1000

add one: 1111 1111 0111 1001  (ANSWER)
```

(c) (5 pts) Compute $24 - 0x1b$ using binary arithmetic. Show your work and give your answer in 8-bit binary two's complement.

```
24   = 0001 1000
0x1b = 0001 1011
-0x1b = 1110 0100 + 1 = 1110 0101


    00011000
+   11100101
    --------
    11111101  (ANSWER)
```

(d) (5 pts) What is the decimal equivalent of the above result?

-128+64+32+16+8+4+1 = -3

3. **Finite State Machines and Control** (50 pts)

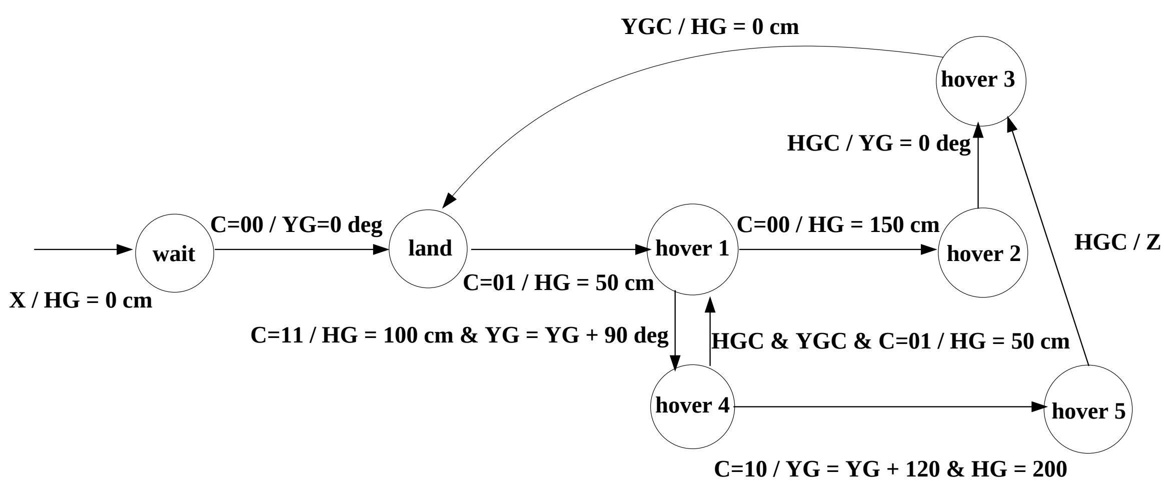
Consider a helicopter whose state can be described with two variables: height and yaw (x, y, roll and pitch are **not** part of the state). One proportional-derivative controller is used for each to produce the appropriate control signals to close the gap between the desired and actual state. On top of these PD controllers is a Finite State Machine controller (pictured below). This controller has access to the following actions:

- HG = xxx cm. Set the height goal to xxx. You may assume that the underlying PD controller will eventually bring the height to the currently set goal.
- YG = yyy deg. Set the yaw goal to yyy. The associated PD controller will eventually bring the true yaw to the yaw goal.
- Z. Do nothing

The FSM also has the following events:

- HGC. Height goal complete. The helicopter has reached its height goal.
- YGC. Yaw goal complete.
- C = bb. A control input that can be one of four different values: 00, 01, 10 or 11 (so, 4 different events). This control input is set by some remote pilot.

The FSM is as follows:



(a) (5 pts) Starting from the *wait* state, assume that the following commands are issued in this order: 00, 01, 11, 10. What is the state that the FSM stops in?

The **land** *state.*

(b) (5 pts) What is the orientation of the heli at this state?

210 *degrees.*

(c) (5 pts) What is the maximum height obtained by the heli during the sequence?

200 *cm.*

(d) (5 pts) Starting from the *wait* state, assume that the following sequence of commands is issued: 00, 01, 11, 01. What is the state that the FSM stops in?

The **hover1** *state.*

(e) (5 pts) What is the orientation of the heli at this state?

90 *degrees.*

(f) (5 pts) What is the maximum height obtained by the heli during the sequence?

100 *cm.*

Consider the problem of a robot dribbling a ball down a soccer field until it reaches the far end of the field (we will call this *the goal*). The robot is equipped with a camera that will automatically move to follow the ball if it sees it. We will design a FSM that performs the high level control for this robot.

The actions are as follows:

- Search ball (SEARCH). This is a generic action that takes a variety of strategies to find the ball (and will ultimately succeed)
- Move toward location behind ball (BEHIND): the robot drives toward a position so that the ball is between the robot and the goal (but not necessarily near the ball)
- Approach ball (APPROACH): the robot drives directly to the ball
- KICK. (immediately after a kick, a GOAL event will occur if a goal is achieved)
- CELEBRATE

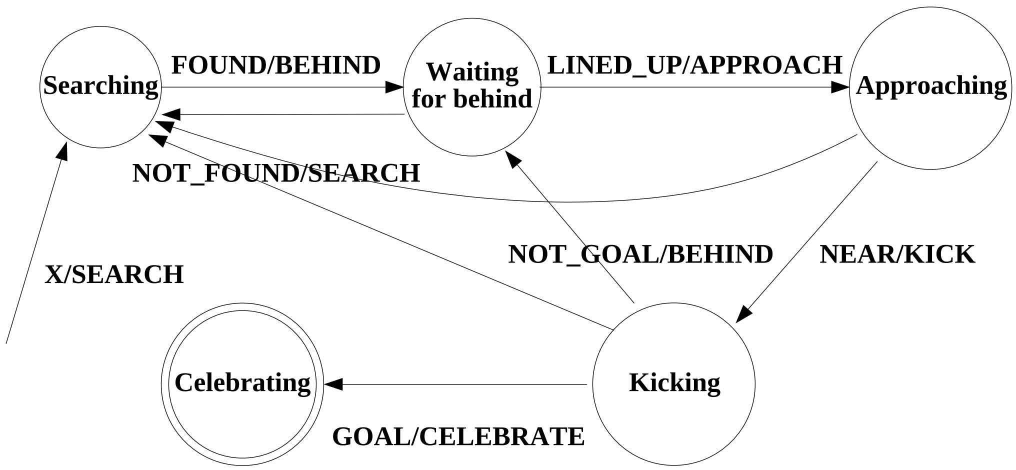
The events are:

- FOUND: the camera sees the ball
- NOT_FOUND: the camera does not see the ball
- NEAR: the ball is very close to the robot (and kickable)
- GOAL: the ball is at the goal
- NOT_GOAL: the ball is not at the goal
- LINED_UP: the ball is between the robot and the goal

Note: Although the camera will move to try to keep the ball in view, it can lose visual track of the ball. The FSM must take this into account by initiating a new search.

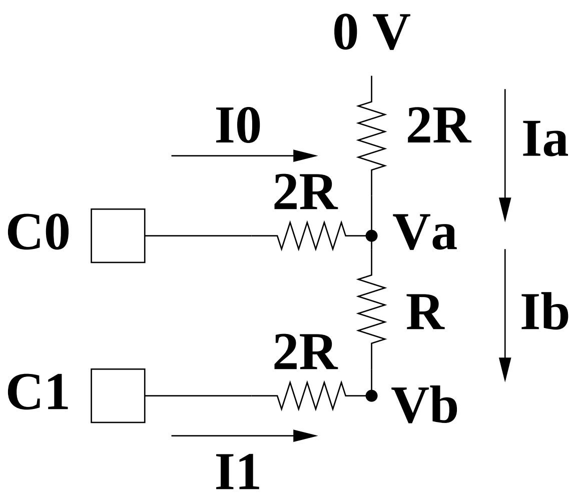
(g) (20 pts) Draw the FSM that will result in the ball arriving at the goal (and the robot celebrating this fact)

Given the specification, there are several ways to do this. This solution includes the essentials.



4. **Analog Processing** (40 pts)

Consider the following circuit:



C_1 and C_0 are logical values determined by your Atmel Mega processor (i.e., 0 and 1). The voltage at pin i is $5C_i$. These are considered known variables for the following analysis.

(a) (5 pts) What two equations are always true according to Kirchoff's Current Law? Label them **A** and **B**

$$\begin{aligned} I_0 + I_a &= I_b \ \ (A) \\ I_b + I_1 &= 0 \ \ (B) \end{aligned}$$

(b) (5 pts) What remaining equations are always true?

All from Ohm's Law as applied to the resistors:

$$\begin{aligned} 0 - V_a &= 2\,R\,I_a \\ V_a - V_b &= R\,I_b \\ 5\,C_0 - V_a &= 2\,R\,I_0 \\ 5\,C_1 - V_b &= 2\,R\,I_1 \end{aligned}$$

(c) (10 pts) Given equation **A**, derive a simplified equation that contains only Va, Vb and known variables. Show your work.

$$\begin{aligned} \frac{-V_a}{2\ R} + \frac{5\ C_0 - V_a}{2\ R} &= \frac{V_a - V_b}{R} \\ -V_a + 5\ C_0 - V_a &= 2\ V_a - 2\ V_b \\ 5\ C_0 &= 4\ V_a - 2\ V_b \end{aligned}$$

(d) (10 pts) Given equation **B**, derive a simplified equation that contains only Va, Vb and known variables. Show your work.

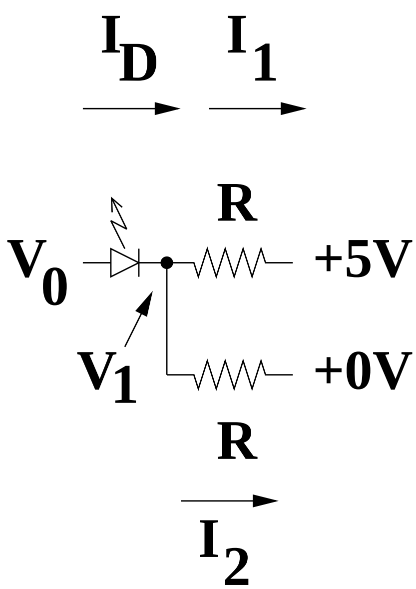
$$\begin{aligned} \frac{V_a - V_b}{R} + \frac{5\ C_1 - V_b}{2\ R} &= 0 \\ 2\ V_a - 2\ V_b + 5\ C_1 - V_b &= 0 \\ 3\ V_b - 5\ C_1 &= 2\ V_a \end{aligned}$$

(e) (10 pts) Solve for Vb given only known variables.

$$\begin{aligned} 5\ C_0 &= 6\ V_b - 10\ C_1 - 2\ V_b \\ V_b &= \frac{5}{4}(2\ C_1 + C_0) \end{aligned}$$

5. **Analog Circuits** (20 pts)

Given the following circuit:



Assume that $R = 1000\Omega$ and $V_f = 2V$.

(a) (5 pts) What are the equations that are always true?

$$\begin{aligned} V_1 - 5 &= I_1 R \\ V_1 - 0 &= I_2 R \\ I_D &= I_1 + I_2 \end{aligned}$$

(b) (10 pts) Assume $V_0 = 4V$. What is V_1?

Guess: $I_D = 0$ *and* $V_0 - V_1 < V_f$
Therefore:

$$\begin{aligned} 0 &= I_D \\ &= I_1 + I_2 \end{aligned}$$

$$
\begin{aligned}
&= \frac{V_1 - 5}{R} + \frac{V_0}{R} \\
&= \frac{2V_1 - 5}{R}
\end{aligned}
$$

$$V_1 = 2.5\ V$$

Check: $V_0 - V_1 < V_f$
$4 - 2.5 < 2$ *Correct!*

(c) (5 pts) True or False and briefly explain. A Pulse Width Modulated signal **IS** an analog signal.

False. A PWM signal is a digital signal, though it approximates an analog signal assuming that there is some form of smoothing of it (e.g., by a capacitor or a motor).

6. **Microprocessors** (20 pts)

(a) (5 pts) Briefly explain why our programs are stored in the EEPROM of the microcontroller.

By storing the program in the EEPROM, it is persistent through power loss. This means that our embedded system can then be in a position to function as soon as power is restored.

(b) (5 pts) True or False and briefly explain. The instruction decoder "tells" the ALU which mathematical operation to perform.

True. The instruction decoder specifies what all components should be doing on each clock cycle.

(c) (5 pts) True or False and briefly explain. The ALU stores the results of its computation in the RAM.

False. Results are stored in a general purpose register.

(d) (5 pts) True or False and briefly explain. The program counter increments (adds one) at each clock cycle.

False. This is true much of the time, but the PC can also implement a jump (a branch) or conditional jump by adding/subtracting an arbitrary value.

7. **Serial Communication** (25 pts)

(a) (15 pts) The code below is taken from our implementation of get_dec(), which takes in a sequence of decimal digits from the serial port and returns the corresponding integer value. Make the necessary changes to this function so that it instead interprets the serial input as a sequence of hexadecimal digits. For example, the sequence of characters '1', 'B', '\n' will result in the return of a decimal value of 27; the sequence of characters 'a', '0', '\n' will result in a return value of 160.

```
uint32_t get_hex(){
  char c;   // Last character read

  uint32_t val = 0;   // Value of the number read so far

  // Loop until a non-digit character is received
  while(1) {
    c = fgetc(fp);      // Assume that fp is global and is already initialized


    if(c >= '0' && c <= '9') {
      val = val * 16 + c - '0';
    }else if(c >= 'A' && c <= 'Z') {
      val = val * 16 + c - 'A' + 10;
    }else if(c >= 'a' && c <= 'z') {
      val = val * 16 + c - 'a' + 10;
    }else{
      return(val);
    };

  };

};
```

(b) (10 pts) Briefly describe the role of the *output buffer* in serial communication.

The output buffer temporarily stores the bytes (characters) until the serial hardware can send them out the serial line [which can take some time]. The buffer allows operations such as fprintf() to return as soon as the characters are placed in the buffer, rather than waiting for the characters to be sent.