Two-Digit BCD Display

Purpose

Construct a 2-digit binary coded decimal LED display, interface it to an Arduino microcontroller board, and use it to test the accuracy of the Arduino's internal clock.

Equipment Required

Qty.	Description
2	MAN71A 7-segment LED displays
2	7447 BCD/7-Segment Decoder/Drivers
14	resistors, 300-400 ohm (as available)
1	Protoboard
1	Arduino Uno microcontroller board
1	USB cable (to program and power the Arduino)
1	DMM

Pre-Lab (done before lab)

1. Copy the Arduino program from the next page and paste it into a text document. Modify the program to count *down* from 59 to 0 and start again at 59. Hint: the opposite of the `++` operator is `--`. Make sure your revised program will be accessible to you from lab on Thursday. For example, put it on your z-drive, Google Docs, or a thumb drive.

2. The variable t0 is declared as an unsigned long integer. On the Arduino, that's a 4-byte positive number. This matches the output of the millis() function, which is used to output how many seconds have elapsed since power on (or since the reset button was pressed). How many milliseconds can be counted before the variable rolls over to zero?

Arduino Program

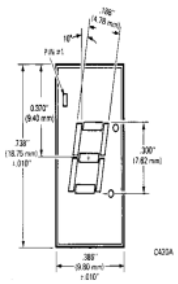
```
/* Count seconds from 0-99 BCD on using pins 2-5 and 6-9.
   This facilitates output on a 7-segment display driven by a 7447. */

unsigned long t0; //This will be a global variable.

// The setup routine runs once when you press reset or power up:
void setup() {
  // initialize pins 2-5 for digital output. This is the ones digit.
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  // Initialize pins 6-9 for digital output. This is the tens digit.
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  // Start the timer here:
  t0 = millis(); // Record current time in milliseconds.
}
// The loop routine runs over and over again forever:
void loop() {
  for (byte tens=0; tens <= 9; tens++) {
    digitalWrite(6, (tens >> 0) % 2); // LSB
    digitalWrite(7, (tens >> 1) % 2);
    digitalWrite(8, (tens >> 2) % 2);
    digitalWrite(9, (tens >> 3) % 2); // MSB
    for (byte ones=0; ones <= 9; ones++) {
      digitalWrite(2, (ones >> 0) % 2); // LSB
      digitalWrite(3, (ones >> 1) % 2);
      digitalWrite(4, (ones >> 2) % 2);
      digitalWrite(5, (ones >> 3) % 2); // MSB
      while( millis() < t0+1000 ){} // Delay until 1s after t0.
      t0 = millis(); // Record current time in milliseconds.
    }
  }
}
```

Procedure

1. Place a 7-segment BCD decoder/driver chip (7447) and a 7-segment LED display on the breadboard. Connect the 7447 outputs to the cathodes of the 7 individual LED segments through resistors as shown. *The LEDs can be destroyed if connected directly across the 5V source.* With the remaining components, make a second copy of this for the second digit of your display.



Pin No.	A MAN3410A, 3610A, 71A, 3810A
1	Cathode A
2	Cathode F
3	Common Anode
4	No Pin
5	No Pin
6	No Connection
7	Cathode E
8	Cathode D
9	Cathode D.P.
10	Cathode C
11	Cathode G
12	No Pin
13	Cathode B
14	Common Anode

Figure 1: Pinout of the 7-segment LED display.

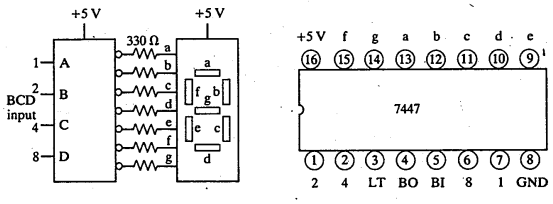


Figure 2: Pinout of the 7447, with wiring to the 7-segment display.

2. Connect the Arduino to the computer using the USB cable. Now start the Arduino development software. Under File/Examples/01.Basics, open the Blink program. Try compiling and loading this to the Arduino by pressing the right-arrow button. If it works, then you will see some blinking of the Tx and Rx LEDs on the Arduino. Then the LED near pin 13 will begin to blink (on 1s, off 1s). Note "Arduino blink test passed" in your log book.

3. The Arduino will supply power to the circuits on the breadboard. With the Arduino unpowered, connect the Arduino +5V and GND leads to the breadboard. Now make power and ground connections to the 7447. Connect the common anode of the 7-segment display to +5V. Power the Arduino Uno by connecting it to the computer via the USB cable (note that the "blink" program begins to run again; the Arduino didn't forget!). Use the LAMP TEST function of the 7447 to test each digit of your circuit. Did all 7 segments light up? If not, fix it before you proceed.

4. Remove power from the Arduino again. Connect pins 2-5 to the ones, twos, fours, and eights inputs of the 7447 that you will use for the "ones" digit of the decimal display. Connect pins 6-9 of the Arduino to the inputs of the other 7447.

5. Reconnect the Arduino to the computer. Open a new sketch, and paste in the source code from the second page of this lab. Load the program to the Arduino and see that it counts all the way from 0 to 99, and restarts at 0.

6. Now try out the code you modified, and have your TA or instructor initial your lab book.

7. Using whatever timekeeping device you have at hand, compare the time kept by the Arduino. Record the time elapsed, explain how you measured it, and note any apparent discrepancy with the time kept by the Arduino. Estimate the percentage error (i.e., uncertainty in the discrepancy divided by total time measured). If you have more than 1% error, try to improve your measurement. If you have a stopwatch or are willing to wait a long time, you can achieve much less than 0.1% error.

The TA is authorized to award 10% extra credit for the most carefully executed measurement in each lab section.