Joystick Operation Using A2D Conversion

(1 Day)

How the Joystick works?

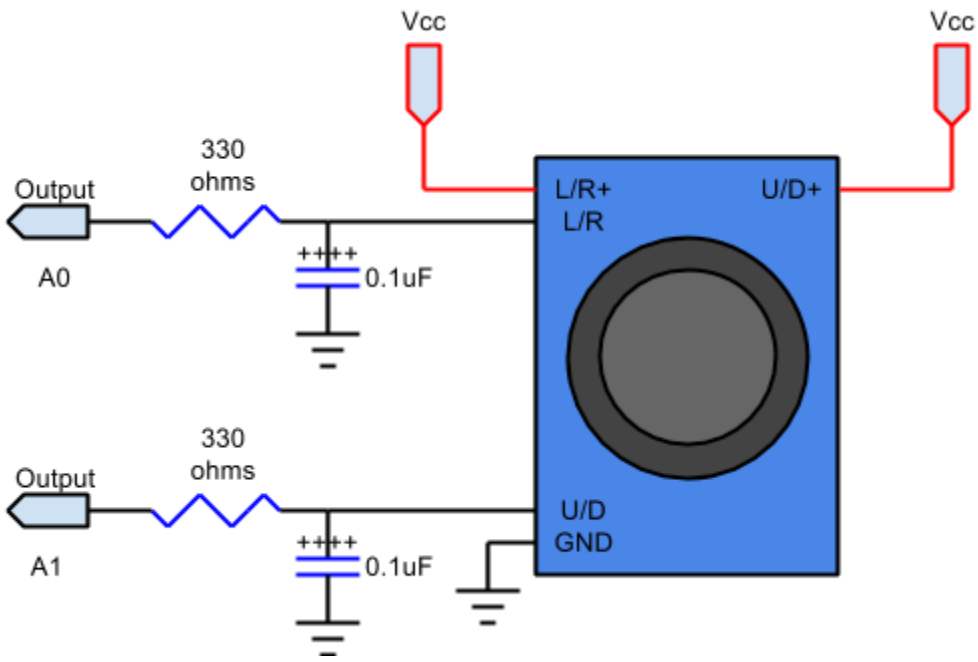
The joystick has two voltage inputs, one for Up/Down tilting, and the other for Left/Right tilting. A variable resistor is connected to each of the two voltage inputs. The amount of resistance provided by the variable resistors is dependent upon how far the joystick is tilted from center.

For example, when the joystick is tilted to the right, the resistance on the variable resistor, connected to the L/R voltage supply, is **decreased**, resulting in a greater voltage value being output on the L/R pin. When the joystick is tilted to the left, the resistance on the variable resistor is **increased**, resulting in a lower voltage value being output on the L/R pin. Using the ATmega1284's built in Analog to Digital conversion, the value of the voltages on the output pins of the joystick can be used to determine which direction the joystick is tilted, and how far the joystick is tilted.

Follow the diagram below to correctly connect the joystick to the micro-controller.

Note: Your capacitors may have a polarity that must match up.

Important: *Make sure to include the resistors and capacitors when connecting the joystick to the microcontroller. Without them, there is a risk of burning out the microcontroller.*



How the Analog to Digital Conversion Works?

The ATmega1284 has built-in Analog to Digital conversion. The ATmega1284 uses an input voltage and a reference voltage to calculate a 10-bit binary number using the following equation:

$$\mathbf{ADC = (V_{IN} * 1024) / V_{REF}}$$

Where V_{IN} corresponds to the ADC pins on PORTA (A7-A0) V_{REF} corresponds to a special pin on the microcontroller. The resulting 10-bit binary number is stored in a register named ADC. This register can be read from at any time. The values of ADC range between 0 and 1024. The higher the number, the closer the Input Voltage is to the Reference Voltage.

```
Example:  unsigned short input = ADC; // short is required to store all
                                      // 10 bits.
```

Analog to Digital conversion will only work with the ATmega1284 if specific flags are set within the microcontroller. The function below sets those flags. Copy and paste the function to the top of the program. Make sure to call the function within main, but before the while loop.

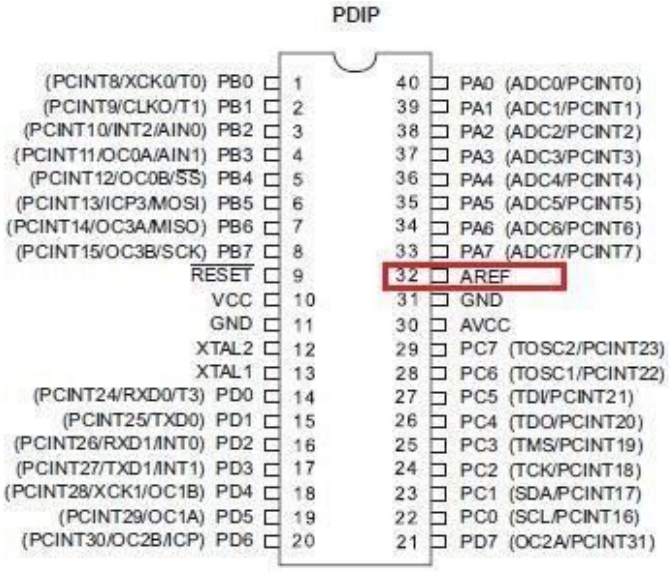
```
void A2D_init() {
      ADCSRA |= (1 << ADEN) | (1 << ADSC) | (1 << ADATE);
```

```
        // ADEN: Enables analog-to-digital conversion
        // ADSC: Starts analog-to-digital conversion
        // ADATE: Enables auto-triggering, allowing for constant
        //         analog to digital conversions.
}
```

Note: You do not need to set the DDRA to enable the Analog to Digital circuitry

After the above function is called, Analog to Digital conversion will work as follows.

- V_{in} is connected to pin PA0.
- The AREF (V_{ref}) pin is connected directly to voltage. AREF is the pin located between pin PA7 and the ground pin. See the diagram below.



- The value of ADC will be updated constantly, ensuring that the the value read from the ADC register is an accurate representation of what is happening NOW.
- For more information about how Analog to Digital conversion is working in the ATmega1284, consult page 240 of the datasheet: ATmega1284 datasheet

Prelab:

Come prepared with exercise 1 of the lab completed.

Exercise 1 -- Display result of A2D

Design a system where the 10-bit binary result from an A2D conversion is displayed on a bank of 10 LEDs. Use the Joystick to adjust the supplied voltage to PA0.

Criteria:

- Connect the L/R output pin of the joystick to PA0
- Display the result of the A2D conversion (stored in ADC) on a bank of 10 LEDs

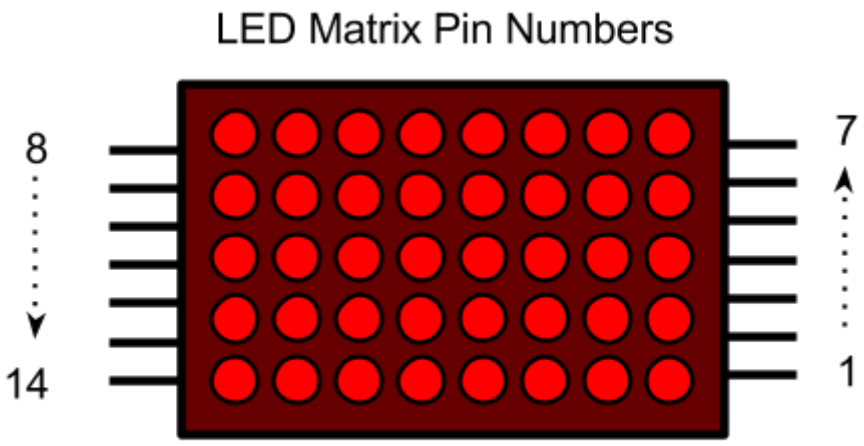
Hints:

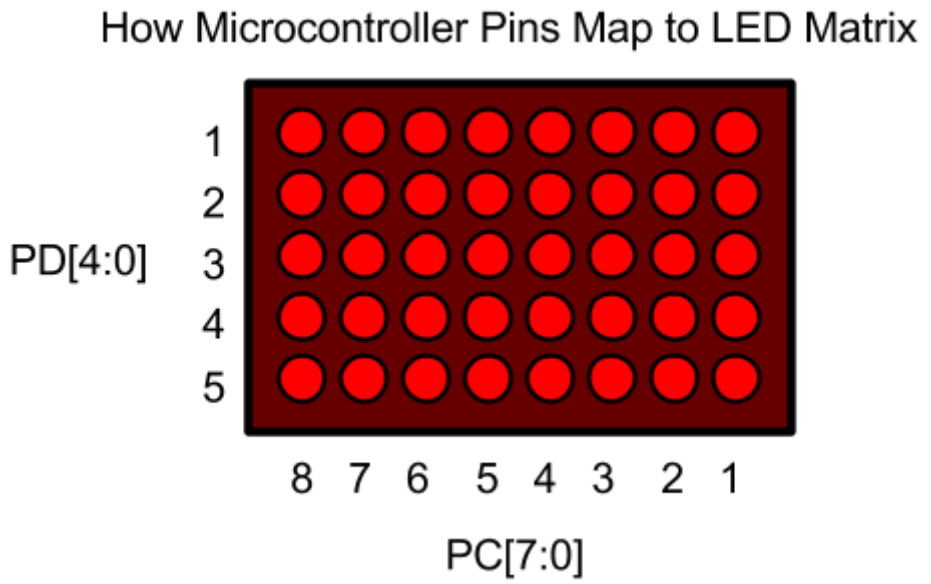
- Since ADC is a 10 bit register, 8 bits will be output on one port, and the remaining two bits will be output on another port.

Video Demonstration: **http://youtu.be/PJaN-ncFbxw**

Exercise 2 -- Shifting an LED with a Joystick

Using the A2D result stored in ADC, design a system where an illuminated LED is shifted left or right along the top row an LED matrix. For this part of the lab, only use the L/R output from the joystick. Connect the LED matrix according to the diagram and pinouts given below.





Microcontroller to LED matrix connections

PC7: pin 9 COL1 PC6: pin 14 COL2 PC5: pin 8 COL3 PC4: pin 12 COL4 PC3: pin 5 COL5 PC2: pin 1 COL6 PC1: pin 7 COL7 PC0: pin 2 COL8	PD4: pin 13 ROW5 PD3: pin 3 ROW4 PD2: pin 4 ROW3 PD1: pin 10 ROW2 PD0: pin 6 ROW1 No connection: pin 11

Criteria:

- PORTC outputs the pattern to be displayed on the LED matrix
- PORTD grounds the row to display the pattern.
- When the joystick is tilted LEFT, the illuminated LED should shift to the left. If the joystick is still tilted left when the illuminated LED reaches 0x80, the illuminated LED should reset to the far right (0x01), and continue shifting left as long as the joystick is tilted left.
- When the joystick is tilted RIGHT, the illuminated LED should shift to the right. If the joystick is still tilted left when the illuminated LED reaches 0x01, the illuminated LED should reset to the far left (0x80), and continue shifting right as long as the joystick is tilted right.
- When the joystick is in the middle position, the LED should remain stationary

Hints:

- One solution is to use two synchSMs. One synchSM drives the LED matrix, the other synchSM shifts the LED.
- When the joystick is at rest in the middle position, the value stored in ADC should be around 512. Use this value as a reference when determining which direction the LED should shift.

Video Demonstration: http://youtu.be/GCoqTI5VZ6w

Exercise 3 -- Variable Speed LED

Expand upon exercise 2 of the lab by adjusting the speed the illuminated LED shifts depending on how far left or right the joystick is tilted. For example, if the joystick is tilted only slightly to the right, the LED will move slower. However, if the joystick is tilted all the way to the right, the LED will move much faster.

Criteria:

- Same as exercise 2
- The illuminated LED should shift once every 1000ms, 500ms, 250ms, or 100ms, depending on how far the joystick is tilted left or right.
- If the joystick is in the center position, the illuminated LED should remain stationary.

Hints:

- One solution is to use three synchSMs. One synchSM drives the LED matrix, a second synchSM shifts the LED, and a third synchSM adjusts how often the LED

shifts.

- Remember, the the farther away from 512 the A2D conversion result it, the farther the joystick is being tilted left or right.

Video Demonstration: http://youtu.be/doYS6cGk49I

Exercise 4 -- Shifting an LED in 8 Directions

Design a system that moves an illuminated LED (controlled by the joystick) around a 5x8 LED matrix. The LED can be moved up, down, left, right, up/left, up/right, down/left, or down/right. An additional function is needed for this exercise to allow for switching to other ADC channels to handle the additional axis.

```
// Pins on PORTA are used as input for A2D conversion
//    The default channel is 0 (PA0)
// The value of pinNum determines the pin on PORTA
//    used for A2D conversion
// Valid values range between 0 and 7, where the value
//    represents the desired pin for A2D conversion
void Set_A2D_Pin(unsigned char pinNum) {
       ADMUX = (pinNum <= 0x07) ? pinNum : ADMUX;
       // Allow channel to stabilize
       static unsigned char i = 0;
       for ( i=0; i<15; i++ ) { asm("nop"); }
}
```

Examples:

```
// Sets PA0 to be input pin for A2D conversion
Set_A2D_Pin(0x00);

// Sets PA7 to be input pin for A2D conversion
Set_A2D_Pin(0x07);

// Invalid option. Input pin for A2D remains unchanged
Set_A2D_Pin(0x0F);
```

IMPORTANT: The microcontroller takes approximately 14 clock cycles to convert a new ADC value. When switching the A2D pin, ensure enough time has passed (1 ms is plenty of time) before reading the contents of ADC. Otherwise, the ADC value read could be the conversion result from the previous A2D pin.

Criteria:

- The L/R pin from the joystick remains connected to PA0.
- Connect the U/D pin from the joystick to a pin on PORTA of your choosing.
- If the illuminated LED reaches an edge of a matrix, it remains there unless moved away from the edge. For example, if the illuminated LED reaches the far left edge of the matrix, continuing to tilt the joystick left will have no effect, but tilting the joystick right will move the illuminated LED away from the edge.

Hints:

- One approach is to use three synchSMs to complete the exercise. One synchSM stores ADC results for L/R and U/D. A second synchSM updates

Video Demonstration: **http://youtu.be/NdMrFF2uKQU**

Exercise 5 (challenge) -- Variable Speed LED in 8 Directions

Expand upon the exercise 4 of the lab by increasing the speed at which the illuminated LED shifts depending upon how far the joystick is tilted from center.

Criteria:

- Same as part 4 only with the added criteria of variable shifting speed.
- Possible shifting speeds are: 1000 ms, 500 ms, 250 ms, and 100 ms.
- The farther the joystick is tilted away from center, the faster the LED is shifted.

Hints:

- Refer to the hints provided in exercise 3 for help with adjusting shifting speed.

Video Demonstration: **http://youtu.be/RbGQpOfd-SA**