

Lab 3

Interfacing 7-Segment Display Using Digital I/O Ports

Objectives

- To understand and use digital I/O ports of AVR microcontroller.
- Learn to interface components like seven-segment display using digital I/O ports.

Software

- Microchip Studio / AVR Studio
- Proteus ISIS
- AVRDUDE

Hardware

- 1 × Arduino Nano
- 1 × 7-Segment Display
- 8 × Resistors (470 Ω)
- 1 × LED
- Wires

Theory

I/O in ATmega328p

Just like any microcontroller, a variety of components can be interfaced to the ATmega328p through its numerous pins.

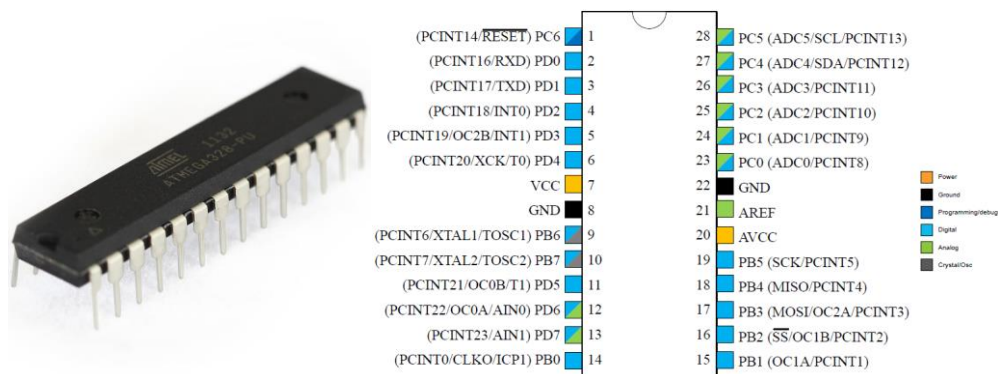


Figure 3.1 – Atmega328p pinout. All pins coloured completely or partially blue are digital I/O pins

The ATmega328p can generate output on these pins (to be consumed by connected devices) or accept input from producing components. The input may be digital (in the form of 1s and 0s) or analogue (in the form of variation in voltages). This lab session will concentrate on **digital I/O only**.

The ATmega328p has 23 digital I/O pins, i.e. 23 pins that can produce or accept data in the forms of 1s and 0s. These 23 pins have been organized in 3 groups:

1. Port **B** (8 pins)
2. Port **C** (7 pins) &
3. Port **D** (8 pins)

Each port is associated with **3, 8-bit registers each**. The registers are:

1. DDRx
2. PORTx
3. PINx

Where 'x' can be either B, C or D. **Please note** it is often a source of confusion with students. The phrase "Port" is used in two contexts, either in terms of the hardware pins or the registers. Case-in-point, PORTD may imply **the hardware pins** (numbers 2-6 & 11-13) or it may imply **one of the three registers** (8-bit memory storage element) associated with the pins, as shown in Figure 3.2. *Each bit in each register corresponds to one pin.*

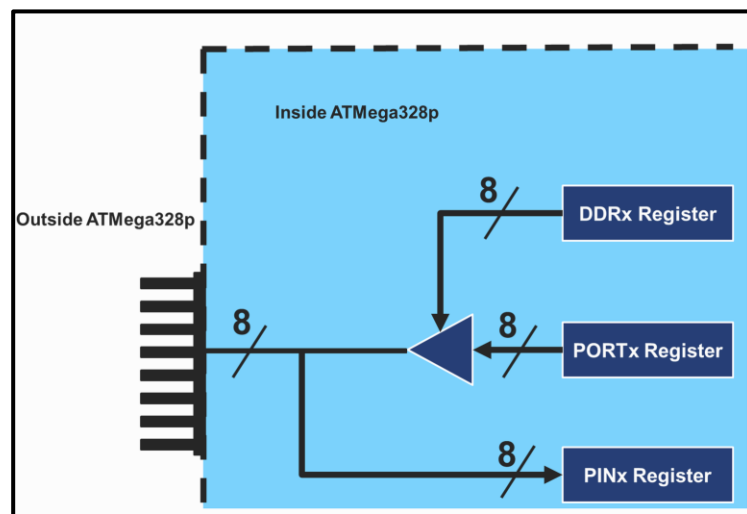


Figure 3.2 – Pins and corresponding registers

The programmers control the input and the output by controlling the 3 registers. A programmer can control:

1. the direction of a pin (input or output) by using the **DDRx register**. I.e. if DDRx is equal to 1110 0000, then the first 5 pins will act as input and last 3 bits will act as the output.
2. in case of output, control the data that is going out. I.e. in case of DDRx equal being equal to 1111 1111, any data that is put in **PORTx register** will **reflect on the outgoing pins**.
3. in case of input, receive the data that is coming in. I.e. in case of DDRx being equal to 0000 0000, any data that is coming into controller will be **stored into PINx register**.

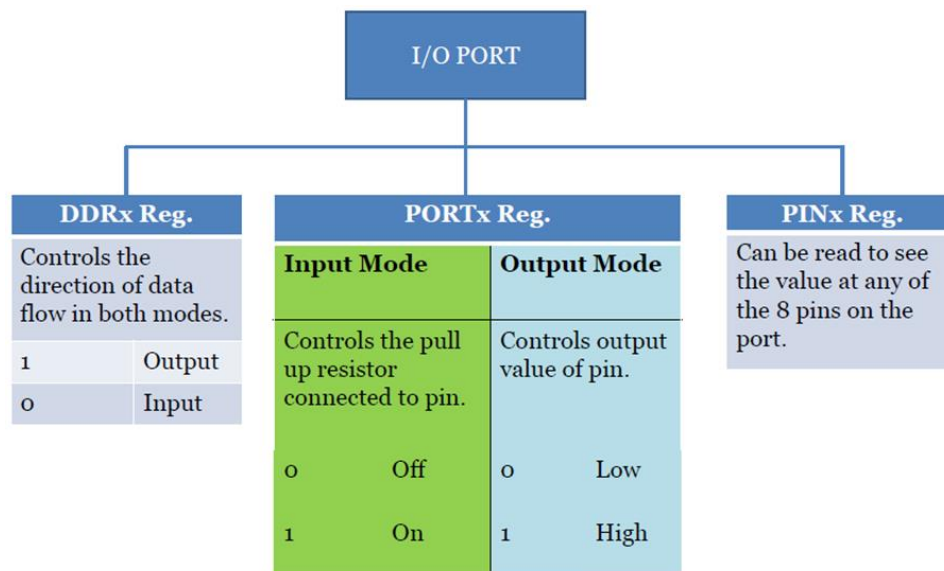


Figure 3.3 – Role of IO Registers in the ATmega328p

Figure 3.3. summarises the role of the I/O registers. Hopefully, by this point you would have gained some understanding of the digital I/O in the ATmega328p. In the next section the working of a 7-segment display is discussed so that you can learn to drive it using digital output.

Seven Segment Display

Seven-segment displays are commonly used in a variety of devices today. Each display unit can display a single alphanumeric character. A seven-segment display is a set of seven bar-shaped LED (light-emitting diode) or LCD (liquid crystal display) elements, arranged to form a squared-off Figure 8. Every LED is assigned a name from 'a' to 'h' and is identified by its name. Seven LEDs 'a' to 'g' are used to display the numerals while eighth LED 'h' is used to display the dot/decimal.

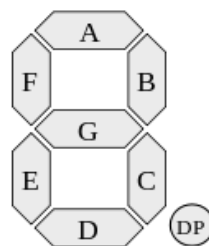


Figure 3.4 – Seven Segment Display

Types of 7-Segment Displays

There are two types of LED 7-Segment displays:

- Common Cathode
- Common Anode

In Common Cathode configuration, the negative terminals of all LEDs are connected to the common pin. The common is connected to ground and a particular LED glows when its corresponding pin is given high.

In Common anode arrangement, the positive terminals of all LEDs are connected to common pin. The common pin is given a high logic and the LED pins are given low logic to display a number.

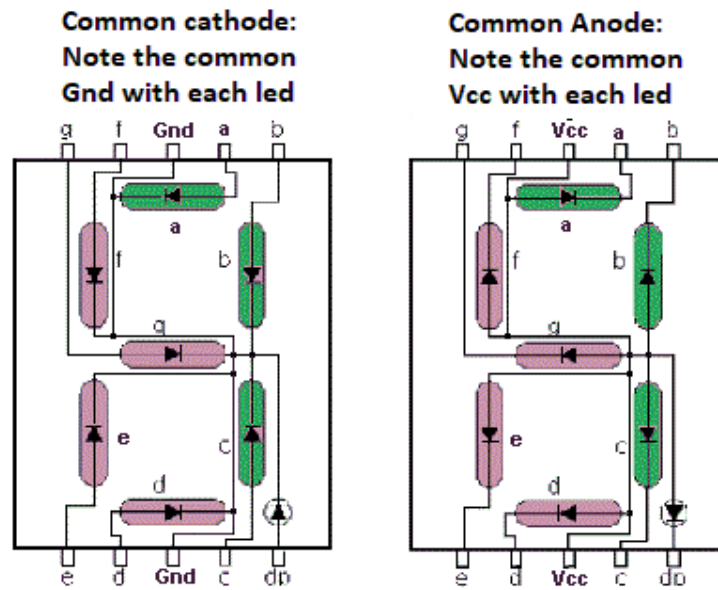


Figure 3.5 – Common cathode and common anode LED displays

Pin Configuration

The figure below shows Pin diagram of a common cathode 7-segment display. For common anode, pin 3 and 8 are connected to V_{cc} instead of ground.

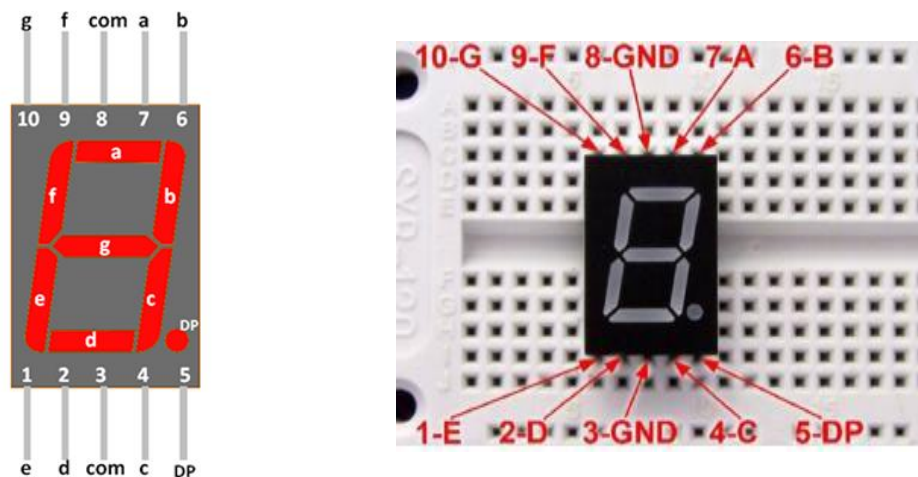


Figure 3.6 – Pin Configuration of 7-Segment Display

Pre-Lab Tasks

Task-1

Students are required to read the theory section in detail. They are encouraged to address any queries with regards to the theory during your theory session.


Task-2

Please fill out the following table, explaining what each value for DDRx means.

#	DDR _x	Meaning
1	DDRB = 0b1010 1010	<i>Pins 0, 2, 4 and 6 of port B will be inputs while pins 1, 3, 5 and 7 will act as outputs</i>
2	DDRD = 255	
3	DDRB = 0x0B	
4	DDRD = 0x45	
5	DDRC = 45	
6	DDRB = 0b1100 0110	

Task-3

Imagine a 7-segment display is connected to 8 pins of Port-B of an ATmega328p. The students are required to complete the contents of the following look-up table so that the numbers mentioned in the table are displayed. The task may be completed on the manual itself.

Digit to be displayed	H	G	F	E	D	C	B	A	
	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0	Hex
0	0	0	1	1	1	1	1	1	0x3F
1									
2									
3									
4									
5									
6									
7									
8									
9									

Assembly Code

Write a piece of code in assembly that toggles between displaying 1 and 0 on a connected 7 segment display with a suitable delay in between them.

C Code

Replicate the following code and fill out “**lookUpTable**” from the table that you filled out before:

```
#include <avr/io.h> // Most basic include files
#define F_CPU 16000000UL
#include<util/delay.h>
int main(void){
    DDRB=0xFF;
    DDRD=0xFF;

    unsigned char lookUpTable[10]={
        /* fill out the 10 hex codes for the 10 digits*/
    };
    unsigned char counter= 0;
    while(1) {    // Infinite loop; define here the
        PORTB=counter;
        PORTD=~ lookUpTable [counter];
        _delay_ms(250);
        _delay_ms(250);
        counter++;
        if(counter==10)
            counter=0;
    }
    return 0;
}
```

Task-4

Once the code in Task-3 is up and running, the students are required to simulate it in Proteus after connecting 2 7-Segment displays to your ATmega328p as shown in Figure 3.7.

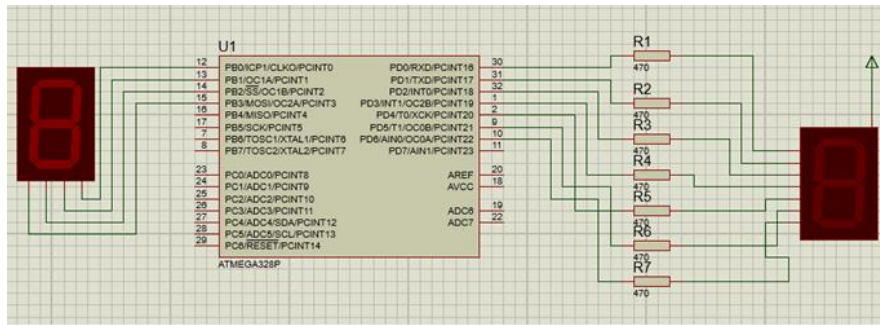


Figure 3.7 – Pre-lab Simulation Task

In-Lab Tasks

Task-1

The students are required to wire their hardware to run the code completed as Pre-Lab Task-2, on the breadboard, (schematics shown in Figure 3.8).

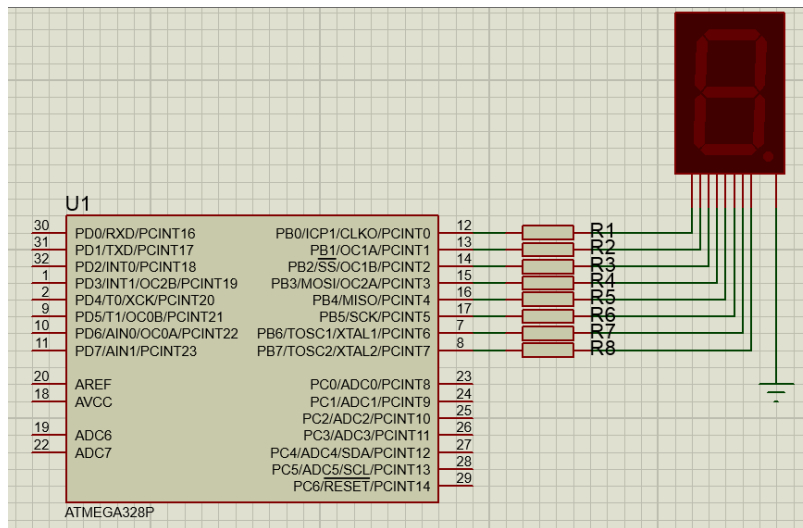


Figure 3.8 – In-Lab Task-1

Post Lab

Task-1

Students are required to connect 2 seven segments with ATmega328P, as shown in Figure 3.9. They are further required to code to count from 0-99 on these 7-segments.

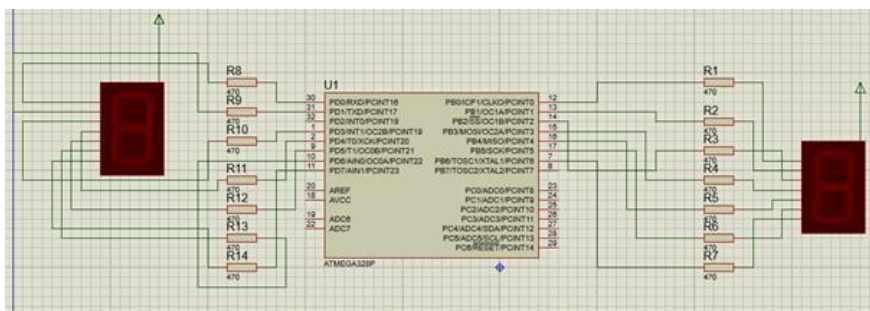


Figure 3.9 – Post-Lab Task-1

Critical Analysis / Conclusion (To be filled in by the student)**Lab Assessment** (To be filled by the lab-instructor)

Pre-Lab	/5	/25
In-Lab	/5	
Results	/5	
Viva	/5	
Critical Analysis	/5	

Comments:

Instructor Name_____
Instructor Signature