CG1112 Engineering Principles and Practices II

Week 3: Studio 2:

Arduino Bare-Metal GPIO Programming

Objectives:

1. Develop Low-Level (Bare-Metal) code for the Atmel Microcontroller.
2. Understand Bit-Masking techniques.
3. Develop simple Embedded C programs.
4. Understand how to extract meaningful information from the datasheet.

Equipment Needed:

1. Laptop with Arduino Uno installed
2. Arduino Uno Board
3. Breadboard with LED's, Resistors and Wires.

** The **Lab Journal** Questions are to help with your understanding. Seek clarification from your TA if you are not sure about what is required. **

GUIDELINES on Good Coding Practice:

1. Declaration and use of Constants when necessary.
2. Meaningful Comments. Don't comment (// for-loop) when you are starting a loop. Give some useful info on the functionality of the loop.
3. Proper Indentation, Spacing and Alignment.

1. Introduction

In this lab, we will explore how to use the General Purpose Input / Output (GPIO) pins on the Arduino by programming the Atmel Atmega328 in "bare-metal" / "low-level". This is done by directly accessing the HW registers in the microcontroller rather than using the Arduino library functions.

There are several good reasons for this:

I. There is a significant performance improvement. The Arduino library introduces significant overheads in mapping between Arduino pin specification and the Atmega's register accesses. This adds computational overheads which will now be avoided.

II. You code exactly what you want. Since you are now writing code by directly manipulating the registers, you don't need to deal with other peripherals or registers that are not needed for your application.

III. Better understanding of Microcontroller Concepts. This allows you to easily transfer your code from one microcontroller to another. Though the low-level registers may be different, the idea and concept is still the same.

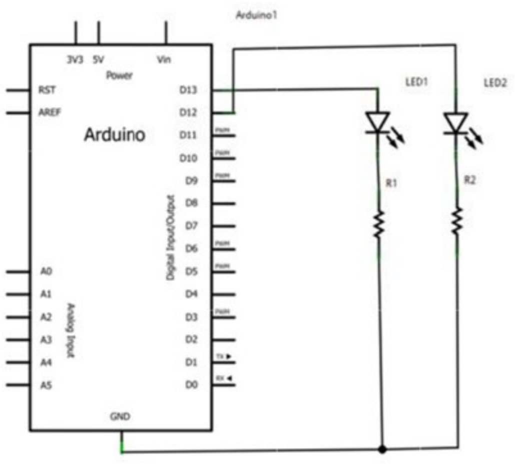
Before you proceed, please view the following E-Lecture's here:

Demo and Explanation on transitioning from Scratch to Bare-Metal: https://youtu.be/k_8NDkzExik

E-Lecture of Slides: https://youtu.be/gkg9MZd5WbQ

2. Exercise A

You will assemble your circuit on the breadboard that is provided. Connections are made between the breadboard and the Arduino using the wire jumpers provided. Use 220 ohms for the resistors. An example circuit is shown below:



Note that you should **ALWAYS** color code your wires. E.g. black for GND, red for 5V/3.3V, green for inputs, orange for outputs, etc. This makes your circuits much easier to debug.

IMPORTANT NOTE: Arduino Pin Numbers refer to the pins 0 – 13 that is labelled on the board. 328p Pin Numbers refer to the PORT and the Bit/Pin numbers based on the microcontrollers registers.

Mapping between Arduino Pin and 328p pin.

Arduino Pin	328p Pin
12 (Red LED)	Port B Pin 4
13 (Green LED)	Port B Pin 5

Launch your Arduino Uno and enter the following code.

** Remember to add the line <#include "Arduino.h"> at the start. All the register definitions are included in that file. **

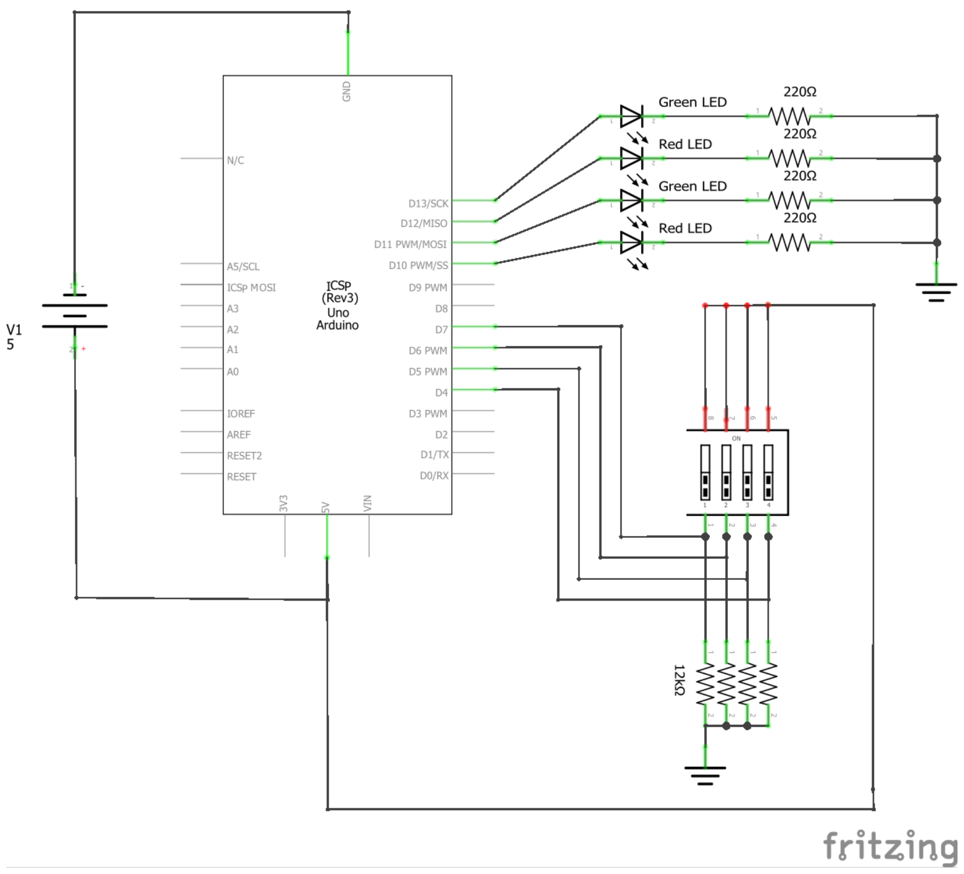
```
#include "Arduino.h"

void setup()
{
  // setting DDRB as output
  DDRB = B00110000; // Set PB5 & PB4 as output

}

void loop()
{
  // toggle both LED's one at a time
  PORTB = B00100000; //Green LED: ON, Red LED: OFF
  delay(1000);
  PORTB = B00010000; //Green LED: OFF, Red LED: ON
  delay(1000);
}
```

Putting a binary string in your code as shown above, can be quite cumbersome and not so intuitive. Defining a BIT MASK and using that in your code is good alternative. Enter the following code into a new sketch and observe the similar behavior as before.

```
#include "Arduino.h"

#define PIN5  (1 << 5)
#define PIN4  (1 << 4)

void setup() {
  // setting DDRB as output
  DDRB = ((PIN5) | (PIN4));

}

void loop() {
  // toggle both LED's one at a time
  PORTB = PIN5; //Green LED: ON, Red LED: OFF
  delay(1000);
  PORTB = PIN4; //Green LED: OFF, Red LED: ON
  delay(1000);

}
```

3. Exercise B

Connect the 4-way DIP switch and 4 LED's (Two Green and Two Red) to the Arduino as shown below. Use 220 ohm resistors for the LED's and 1.2k/12k resistors for the Switches.



Lab Journal

Q1. When the DIP switch is toggled to the ON position, what is the voltage level seen by the Arduino Pin?

Q2. Do we need any Pull-Up resistors at the top of the DIP switch?

Q3. The DIP switch pins are connected to D7 to D4 on the Arduino Uno. Which PORT and PINS does it map to within the AT328 microcontrollers ports?

The DIP switches are now going to control the LED's through the AT328P. Develop the code such that each Switch maps to a single LED based on this mapping of the Uno Pin Numbers.

Switch Pin	LED Pin
7	13
6	12
5	11
4	10

For e.g., when SW Pin 7 is toggled ON, then the LED at Pin 13 should be ON. At any time, multiple switches can be in the ON position and the corresponding LED's must be switched ON.

Lab Journal

Q4. Provide the full code for the above objective.

4. Exercise C

Modify the code from Exercise B to fulfill these 2 objectives.

- When ALL the switches are in the OFF position, the LED's continuously light up in a running sequence from Uno Pin 13 down to Pin 10.
- When ALL the switched are in the ON position, the LED's continuously light up in a running sequence from Uno Pin 10 up to Pin 13.
- Other than the above TWO cases, the appropriate LED will still continue to be mapped to its own Switch like in Exercise B.

The expected behavior can be viewed here: https://youtu.be/IdEg4F647Wg

Lab Journal

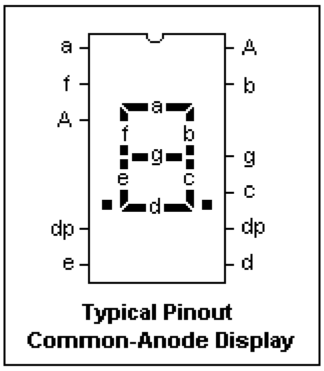
Q5. Provide the full code for the above objective.

5. Exercise D

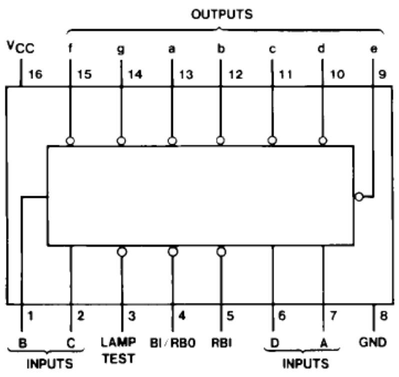
In this exercise, we will be developing a 7-Segment Display Counter. The objective of the counter is to count from 0 – 9 and roll-over to start from 0 again. The expected behavior can be seen here:

https://youtu.be/8VGkVpW2zcM

You will use a Common-Anode display as in your EPP1 studio. However, instead of directly connecting the GPIO pins to the display, you will be using a 7447 BCD-to-7 Segment Decoder IC. The full datasheet is available in IVLE. The main sections are shown here.



Common-Anode 7 Segment Display

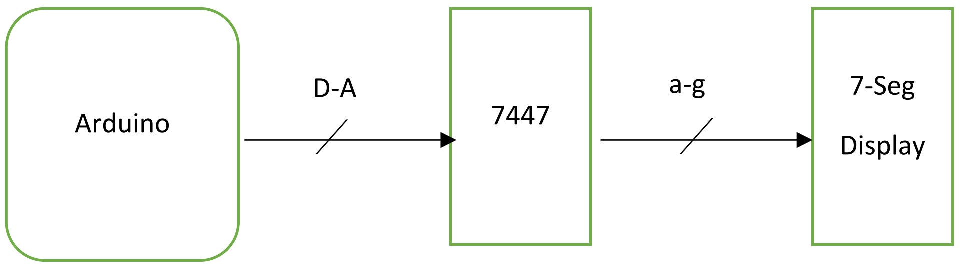


7447 Pin Out

Decimal or	Inputs						BI/RBO	Outputs							Note
Function	LT	RBI	D	C	B	A	(Note 1)	a	b	c	d	e	f	g	
0	H	H	L	L	L	L	H	L	L	L	L	L	L	H	
1	H	X	L	L	L	H	H	H	L	L	H	H	H	H	
2	H	X	L	L	H	L	H	L	L	H	L	L	H	L	
3	H	X	L	L	H	H	H	L	L	L	L	H	H	L	
4	H	X	L	H	L	L	H	H	L	L	H	H	L	L	
5	H	X	L	H	L	H	H	L	H	L	L	H	L	L	
6	H	X	L	H	H	L	H	H	H	L	L	L	L	L	
7	H	X	L	H	H	H	H	L	L	L	H	H	H	H	(Note 2)
8	H	X	H	L	L	L	H	L	L	L	L	L	L	L	
9	H	X	H	L	L	H	H	L	L	L	H	H	L	L	
10	H	X	H	L	H	L	H	H	H	H	L	L	H	L	
11	H	X	H	L	H	H	H	H	H	L	L	H	H	L	
12	H	X	H	H	L	L	H	H	L	H	H	H	L	L	
13	H	X	H	H	L	H	H	L	H	H	L	H	L	L	
14	H	X	H	H	H	L	H	H	H	H	L	L	L	L	
15	H	X	H	H	H	H	H	H	H	H	H	H	H	H	
BI	X	X	X	X	X	X	L	H	H	H	H	H	H	H	(Note 3)
RBI	H	L	L	L	L	L	L	H	H	H	H	H	H	H	(Note 4)
LT	L	X	X	X	X	X	H	L	L	L	L	L	L	L	(Note 5)

7447 Functional-Table

You are to connect the Arduino Pins to the 7447 Decoder. The Decoder will then interface to the 7-Seg Display. The block level view is shown below.



You are free to decide which Arduino Pins you want to use. Choosing wisely will make your SW implementation very easy.

Lab Journal

Q6. Demonstrate your working prototype to your Lab TA.
Provide the complete code for this exercise.

6. Summary

In this Studio, you have learnt how to develop a bare-metal code for the Arduino using its IDE. Controlling GPIO pins are the most fundamental operations you must understand before going onto more advanced peripherals in the Microcontroller. In subsequent Studio's you will learn how to develop the code for many other sub-systems like Timers, PWM, Interrupts, USART, etc.

Good Luck! ☺