CG1112 Engineering Principles and Practices II

Week 5: Studio 1: GRADED LAB

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.

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.

E-Lecture: https://youtu.be/jYdjvMgTkfg

There are several good reasons for this:

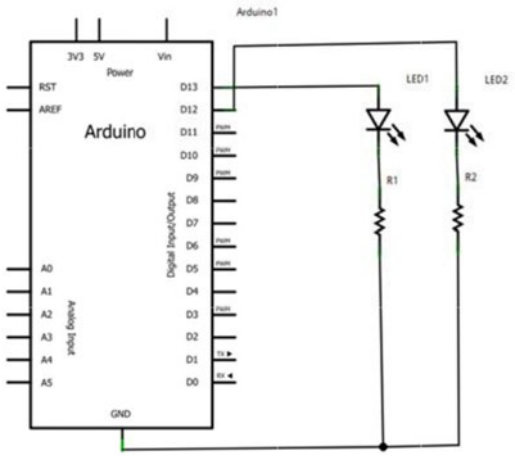
I. There is a significant performance improvement. The Arduino library introduces significant overheads in mapping between Arduino pin specification and the Atmega's register accesses. This adds computational overheads which will now be avoided.

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.

2. Exercise A

You will assemble your circuits on the breadboard that is provided. Connections are made between the breadboard and the Arduino using the wire jumpers provided. 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.

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 = B00100000;
}

void loop()
{
  PORTB = B00100000; // Turn on LED
  delay(1000);
  PORTB = B00000000; // Turn off LED
  delay(1000);
}
```

3. Exercise B

Now, connect a pushbutton to Arduino Pin 10. The button's logic must be such that by default it is LOW, when it is activated, the Arduino must a detect a logic HIGH.

Draw the switch circuit in the space below.

Lab Report

Q1. Draw the schematic to show the connection.

Code B.1

Modify the code such that by default one LED is blinking. While the switch is being pressed, the current LED will stop blinking and the other LED will start blinking. Once the switch is released, it will revert back to the default LED.

For e.g.
RED LED
RED LED
RED LED
(Switch Pressed)
YELLOW LED
YELLOW LED
(Switch Released)
RED LED
RED LED

Lab Report

Q2. Provide the complete code for Exercise B.1.
Follow the guidelines on Good Coding Practice.

** You can retype the code or take screenshots from Arduino IDE. **

Code B.2

Modify the code such that at anytime, only ONE LED will be blinking. When the pushbutton is pressed and released, it will stop blinking the current LED and switch over to the other LED. The whole process will keep repeating indefinitely.

For e.g.
RED LED
RED LED
RED LED
(Switch Pressed & Released)
YELLOW LED
YELLOW LED
YELLOW LED
YELLOW LED
(Switch Pressed & Released)
RED LED
RED LED

Lab Report

Q3. Provide the complete code for Exercise B.2.
Follow the guidelines on Good Coding Practice.

** You can retype the code or take screenshots from Arduino IDE. **

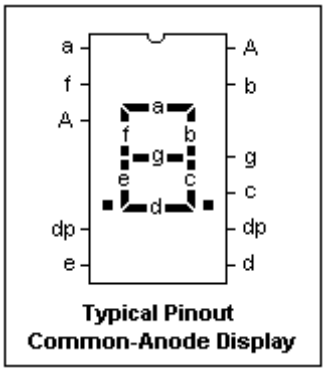
Q4. Do you notice any issues with the behavior of your code.
Explain your observation and give a possible reason for this issue.

4. Exercise C

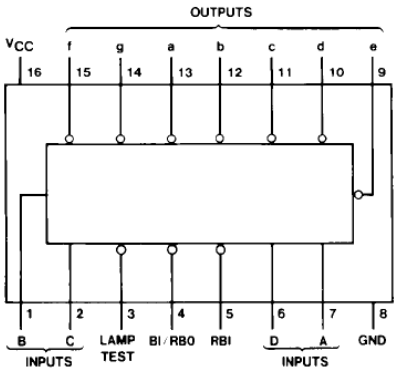
In this exercise, we will be developing a 7-Segment Display Counter. The objective of the counter is to count from 0 – 9 and roll-over to start from 0 again. The expected behavior can be seen here:

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

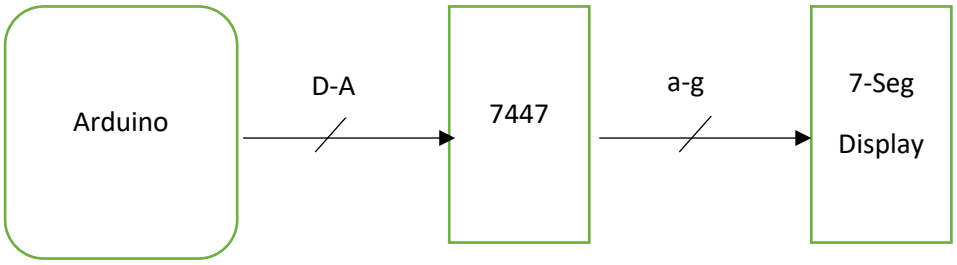


7447 Pin Out

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 Report

Q5. Demonstrate your working prototype to your Lab TA.
Provide the complete code for Exercise C.2.
Follow the guidelines on Good Coding Practice.

** You can retype the code or take screenshots from Arduino IDE.

Q6. Explain why you selected the GPIO pins and how it helps with a more efficient SW implementation.

5. 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! ☺