CG2271 Real Time Operating Systems

Lab 2 – General Purpose I/O

1. Introduction

In this lab we will explore how to use the general purpose I/O pins on the Arduino, by programming the Atmel Atmega328 in "bare-metal", i.e. by directly accessing the registers rather than through the Arduino library.

There are several good reasons for doing this:

i) It is faster. The Arduino library includes significant overheads in mapping between Arduino pin specifications and Atmega register accesses.
ii) It is more compact, because you write just enough code to manipulate the port and pins you are interested in, and don't need to support any other ports in your code.
iii) It is more reliable because of the reduced amounts of code and overheads, and it is easier to understand and debug your own code.
iv) It is more educational because the basic concepts you need to program the GPIO ports are transferrable to other microcontrollers, although the actual programming details will differ.

If you use Arduino library functions you will not receive any credits for this lab.

2. Submission and Demo Instructions

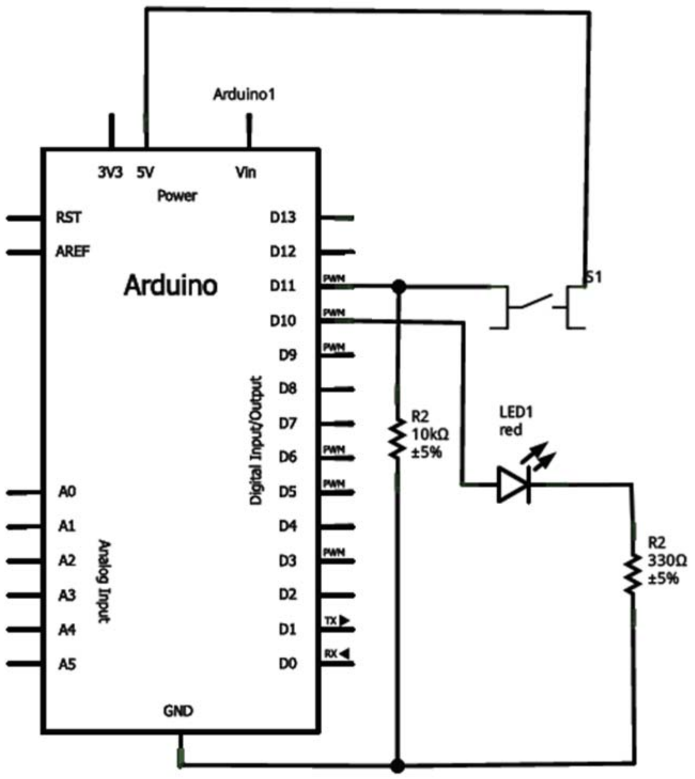
You will demonstrate your lab to your lab tutor at the start of your following lab session on the week of **12 September 2011**. Demonstrations will take place in the first 20 minutes of the lab session, so please do not be late.

After the demo you will submit your PRINTED copy of the answer book to your lab tutor.

3. Questions

Part 1.

Set up the circuit shown below on the breadboard provided for you in the Sparkfun Inventor's Kit. You are NOT REQUIRED to demonstrate this part of the lab.



You will need the following components:

Part	**Components**	**Quantity**
R1	Resistor 330 ohms (orange-orange-brown)	1
R2	Resistor 10K (brown-black-orange)	1
S1	Push-button switch	1
LED1	Red LED	1
	Breadboard	1
	Arduino Uno	1
	Wires	As needed

Question 1 (3 marks)

Consult a diagram that shows the mapping between Arduino pins and the pins on the Atmega328, and complete the table in the answer book.

Question 2 (3 marks)

Using your answers to Q1 and the circuit diagram, write the code to properly set the directions for arduino digital pins 10 and 11.

<u>Question 3</u> (10 marks)

Write a complete program that blinks the LED twice every second when the push button is not depressed, and four times every second when the push button is depressed. Cut and paste your code onto your answer book.

Note:

To use _delay_ms properly, you must declare F_CPU BEFORE you #include <util/delay.h>. So you must have the following at the top of your code before you can use _delay_ms:

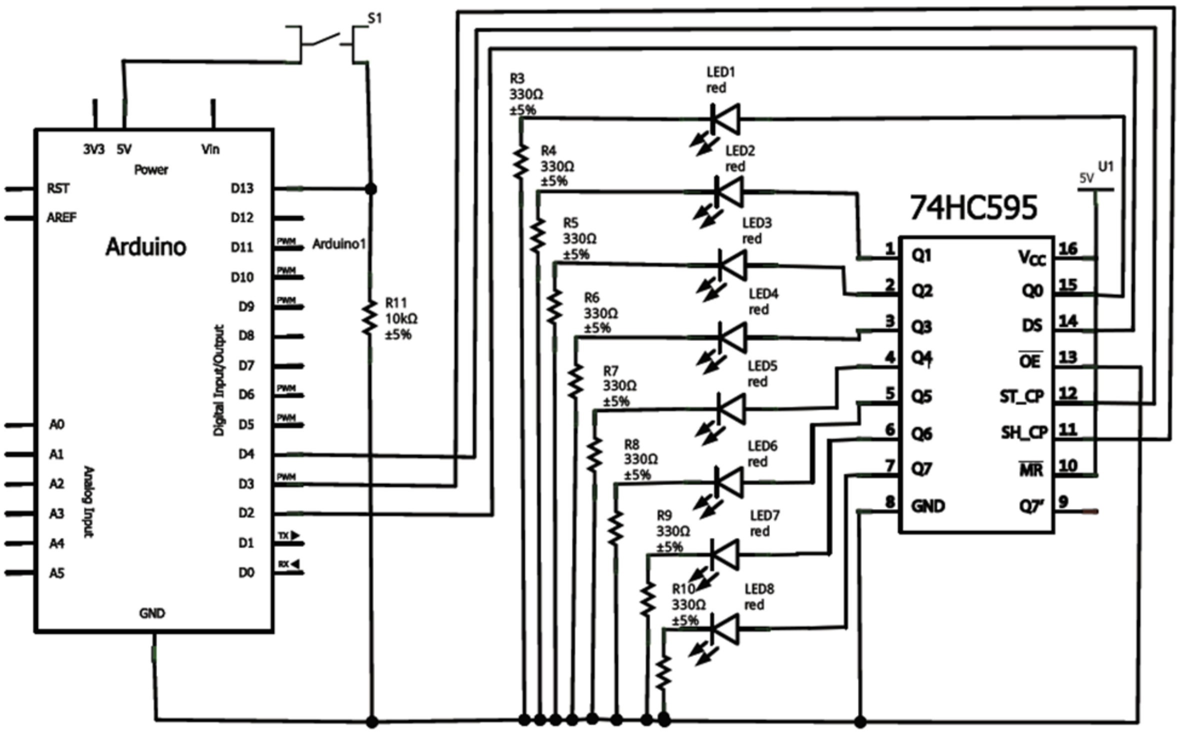
```
#define F_CPU   16000000
#include <util/delay.h>

.. rest of your code…
```

F_CPU is a symbol that specifies your system clock speed (16 MHz in this case).

Part 2.

Assemble the circuit shown below. You will demonstrate this part of the lab to your TA at the start of the next lab session.



It looks complicated but basically:

i. Pins 15, 1 to 7 of the 74HC595 are connected to the positive (longer) legs of 8 LEDs. The negative (shorter) legs are connected via 330 ohm (orange-orange-brown) resistors to the GND on the Arduino.
ii. Pin 16 (VCC) and Pin 10 (MR#) of the 74HC595 is connected to 5v on the Ardunio.
iii. Pin 13 (OE#) and Pin 8 (GND) of the 74HC595 is connected to GND on the Arduino.
iv. Pins 11, 12 and 14 of the 74HC595 is connected to digital pins 3, 4 and 2 respectively on the Arduino.
v. Connect the push-button switch between 5V and digital pin 13 on the Arduino.

The diagram below shows the pin-out for the 74HC595:

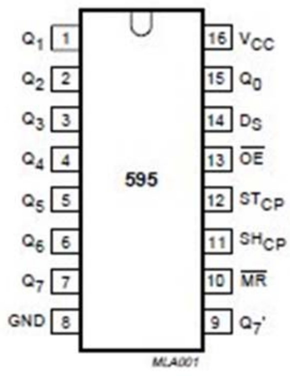


Fig.1 Pin configuration.

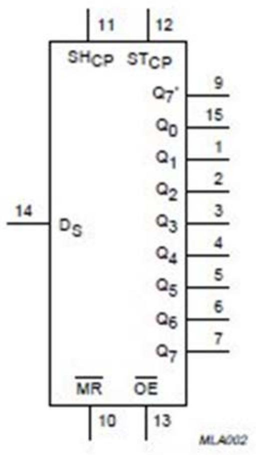


Fig.2 Logic symbol.

The components that you require are:

Components	Quantity
74HC595 Shift Register	1
Resistor 330 ohms (orange-orange-brown)	8
Resistor 10K (brown-black-orange)	1
Push-button switch	1
Red LED	8
Breadboard	1
Arduino Uno	1
Wires	As needed

The 74HC595 is known as a “serial to parallel converter”, as it takes bits one at a time on a “data” line (i.e. serially), then puts all 8 bits simultaneously onto the 8 output lines (i.e. in parallel). This is how it works:

(i) Pull the ST_CP (LATCH) pin low.

(ii) Pull the DS (DATA) pin either low or high to write in a 0 or a 1.

(iii) Pulse the SH_CP (CLOCK) (i.e. write a high, wait 1ms, then a low). The bit on the data pin is copied.

(iv) Repeat steps (i) and (ii) for the remaining 7 bits.

(v) Pull the ST_CP (LATCH) pin HIGH to copy all 8 bits onto the 8 outputs Q7 to Q0.

Question 4 (4 marks)

Consult a diagram showing the mapping between Atmega328 and Arduino pins, and complete the table in the answer book.

Question 5 (12 marks)

We will write a function with the following prototype:

```
void writeLED(uint8_t num);
```

This function will take in an 8-bit number “num”, and output the binary equivalent onto the LEDs through the 74HC595, from LSB to MSB. Cut and paste the code into your answer book.

Question 6 (5 marks)

Now write a complete program that repeatedly flashes the binary equivalents of 0 to 255 on the LEDs, with a 250 ms pause in between. Cut and paste only this portion of code into your answer book.

Question 7 (8 marks)

Modify your program so that it repeatedly flashes the binary equivalents of 0 to 255 on the LEDs with a 250 ms pause in between when the button is not depressed, and flashes all 8 LEDs when the button is depressed at a rate of one update every 500 ms. Detail the modifications that you have made.

Demo: 5 marks

4. Summary

In this lab you’ve explored input and output using GPIO ports. In particular you’ve learnt how to control another integrated circuit (a 74HC595 shift register) using the GPIO ports.