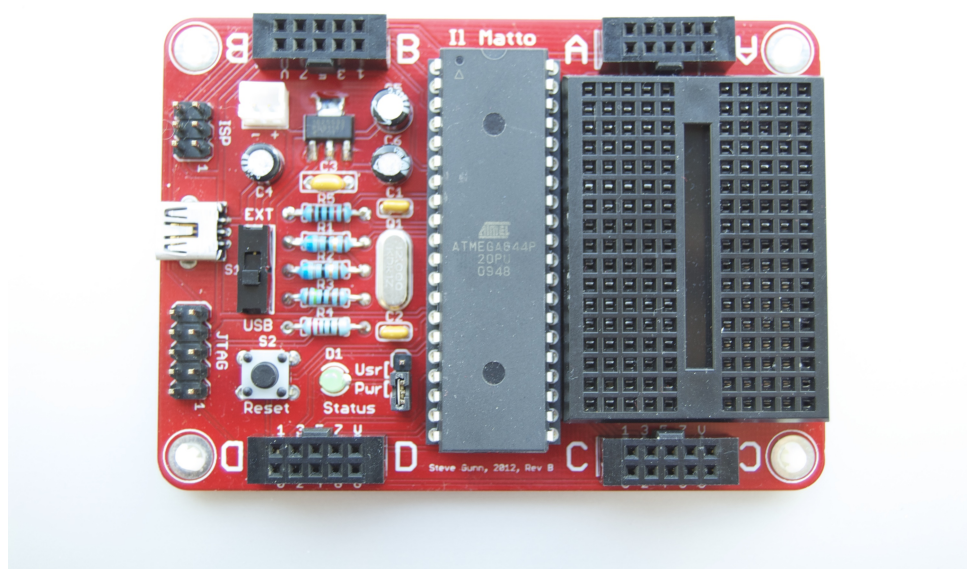


C6

Digital Input & Output

This exercise is an introduction to embedded programming in C using the Il Matto development board. The focus of the exercise is to learn to control digital outputs and read digital inputs. You will discover how to interface a seven-segment display as a simple digital output device and a rotary encoder as an input device. An important aspect of reliably reading digital inputs from mechanical switches is to use the concept of de-bouncing. You will investigate both hardware and software approaches to achieve this.



Schedule

Preparation Time : 3 hours

Lab Time : 3 hours

Items required

Tools :

Components : Seven Segment LED Display, Rotary Encoder, Piezo Speaker,
7×current limiting resistors, 2×10kΩ resistors, 2×10nF capacitors
[Components to be retained by the student after the exercise.]

Equipment :

Software : avr-gcc, text editor (e.g. notepad++)

Version: October 28, 2020

©2012


Steve R. Gunn

Electronics and Computer Science

University of Southampton

Before the laboratory you should read through this document and complete the preparatory tasks detailed in section [2](#).

Academic Integrity – *If you wish you may undertake the preparation jointly with other students. If you do so it is important that you acknowledge this fact in your logbook. Similarly, you will probably want to use sources from the internet to help answer some of the questions. Again, record any sources in your logbook.*

You will be working on your own in this laboratory. During the exercise you should use your logbook to record your observations, which you can refer to in the future – perhaps to write a formal report on the exercise, or to remind you about the procedures. As such it should be legible, and observations should be clearly referenced to the appropriate part of the exercise. As a guide the  symbol has been used to indicate a mandatory entry in your logbook. However, you should always record additional observations whenever something unexpected occurs, or when you discover something of interest.

For each Task you should create a new directory so that you have a working version of every program at the end of the lab. Remember to place comments in your code.

You will be marked using the **standard laboratory marking scheme**.

Notation

This document uses the following conventions:



An entry should be made in your logbook



Care should be exercised

Remarkable text

A point of note

`command input`

Command to be entered at the command line

`output response`

Response produced at the command line

1 Introduction

This lab introduces you to embedded programming in C with simple digital interfacing to hardware.











1.1 Outcomes

At the end of the exercise you should be able to:

- ▶ Configure microcontroller pins as outputs and turn them on/off.
- ▶ Configure microcontroller pins as inputs and read their state.
- ▶ De-bounce mechanical switches to provide reliable input.

2 Preparation

Answer the following questions with reference to the I1 Matto board running at 3.3V:

1. What is the command in C to configure a port as an output, and which header file do you need to include? 
2. Calculate the value of the current limiting resistors for the seven segment display [3] to provide a current of 10mA per segment. 
3. Why is it not possible to replace the seven segment display current limiting resistors with a single resistor in the common line? 
4. Justify that your board can supply enough current to power the display when all segments are illuminated, when powered from a USB port. 
5. Write a C array initialiser, `const uint8_t segments[10] = { ... };`, to store the bit patterns required to display the digits 0 1 2 3 4 5 6 7 8 9 on the seven segment display when the segments are connected according to table 1. The relevant bit should be set if the segment is to be illuminated so that the 8-bit value can be written to the output port to display the digit. Write the values using hexadecimal notation. 
6. What is the command in C to configure a port as an input? 
7. When configuring a pin as an input, why is it often a good idea to enable the internal pull-up resistor? What is the value of this internal pull-up resistor? What is the command in C to enable the pull-up? 
8. What voltage levels on an input pin will be read as logic High and what levels as logic Low? 
9. What is switch bounce? 
10. Compare and contrast one hardware and one software approach to switch de-bouncing. 

11. With reference to the rotary encoder datasheet [1] and possibly other resources, describe how the encoder signals the direction of movement and outline how you could detect this in a program.
12. Verify that the microcontroller has enough pin drive capability to directly drive the speaker.



3 Laboratory Work

Create a new directory ELEC1201/Labs/C6 to store your files for the laboratory. For each task you should create a new sub-directory so that you have a working version of every program at the end of the lab.

This time you are going to invoke the `avr-gcc` compiler to produce executable code for your microcontroller, in contrast to the `gcc` compiler you have used up until now to generate code to run on the PC.

You can execute all of the build and programmer commands from the command line as you did in X2. Navigate to the directory containing the source file and compile with

```
avr-gcc -mmcu=atmega644p -DF_CPU=12000000 -Wall -Os prog.c -o prog.elf
```

generate the target file with

```
avr-objcopy -O ihex prog.elf prog.hex
```

and download to your microcontroller with

```
avrdude -c usbasp -p m644p -U flash:w:prog.hex
```

The download command also assumes you are using the USB cable and that you have the boot loader successfully installed from X2.

3.1 Digital Output

Wire up the seven segment LED display on the I1 Mat to breadboard and connect to Port A according to the mapping in table 1. This is easily achieved if the display is placed at the top of the breadboard near port A and the current limiting resistors should be long enough to make the connection from the port connector to the appropriate points on the breadboard. Care will need to be taken to ensure that the resistor leads do not make contact with each other. Remember to also connect the common cathode to ground.

Write a program which initialises the seven segment display to 0 and then increments the number every second. When it reaches 9 it should wrap around to 0 and continue counting.

TABLE 1: Seven segment display wiring

Pin	7	6	5	4	3	2	1
Segment	a	b	c	d	e	f	g

3.2 Digital Input

The rotary encoder contains two parts: a push switch and the rotary control. Your rotary encoder should have been prepared to make it breadboard friendly by removal of its mounting lugs and twisting of the pins by 45 degrees. Place the encoder in the breadboard below the seven segment display. Wire up the push switch of the rotary encoder between ground and PC7 on the microcontroller. Remember to enable the pull-up on the pin.

Take your previous program and modify it to increment the count on the display by one every time the switch is pressed. Do you notice any odd behaviour?



3.3 De-bouncing

Mechanical switches are not ideal and often the contacts will bounce which can cause programs to see multiple changes on the input when only one was intended. To address this issue the input should be de-bounced. This can be achieved in hardware or in software. Implement one hardware and one software approach to de-bounce the switch and compare the results. You have been supplied with enough components to implement a simple hardware solution using a low-pass RC filter.



3.4 Rotary Encoder

Wire up the rotary encoder direction signals A and B to PC1 and PC0 respectively and connect the common pin to ground. Remember to enable the pull-ups on these pins.

Write a program which uses the rotary encoder to change the number displayed on the seven segment display. When the rotary encoder is rotated counter clockwise one click, the number should be decremented by one, and when clockwise the number incremented by one. Numbers should wrap so decrementing goes from 0 to 9 and incrementing from 9 to 0. Do you need to de-bounce the encoder? If so, describe your chosen approach, remarking on any limitations.



3.5 Simple Sound

You have been supplied with a piezo speaker [4] which has an impedance of 100Ω . Connect up the speaker between PA0 and ground. The pins are too thick to be pushed into a breadboard, so strip 3cm from the end of some hook-up wire and wrap the wire tightly around the pin.

Modify your program from section 3.4 so that it emits an audible click every time the number on the display changes.

4 Optional Additional Work

4.1 Roulette Wheel

Write a program that simulates a roulette wheel. When the rotary encoder is turned the program should cycle through the numbers on the display according to the direction and speed of rotation of the encoder. You may use the time taken for the first three clicks to calculate the initial frequency. After this the program should ignore any input from the rotary encoder until the wheel has stopped. You can model the slowing down of the wheel with the following equation for the rotor frequency [2]

$$v(t) = v_0 - kt$$

where k is a constant related to the friction and v_0 is the initial frequency of the rotor. When the rotor stops you should flash the final number three times.

References

- [1] Bourns. PEC11 Series - 12mm Incremental Encoder Datasheet, 2011. URL <http://www.bourns.com/PDFs/pec11.pdf>.
- [2] J.-I. Eichberger. Roulette Physics. 2004. URL [http://www.roulette.gmxhome.de/roulette\[1\].pdf](http://www.roulette.gmxhome.de/roulette[1].pdf).
- [3] Kingbright. Green 0.8" Single Digit Numeric Display Datasheet, 2006. URL <http://www.kingbrightusa.com/images/catalog/SPEC/sc08-11gwa.pdf>.
- [4] Projects Unlimited. PCB Mounting Miniature Speaker Datasheet, 2011. URL <http://www.puiaudio.com/pdf/ast-030c0mr-r.pdf>.