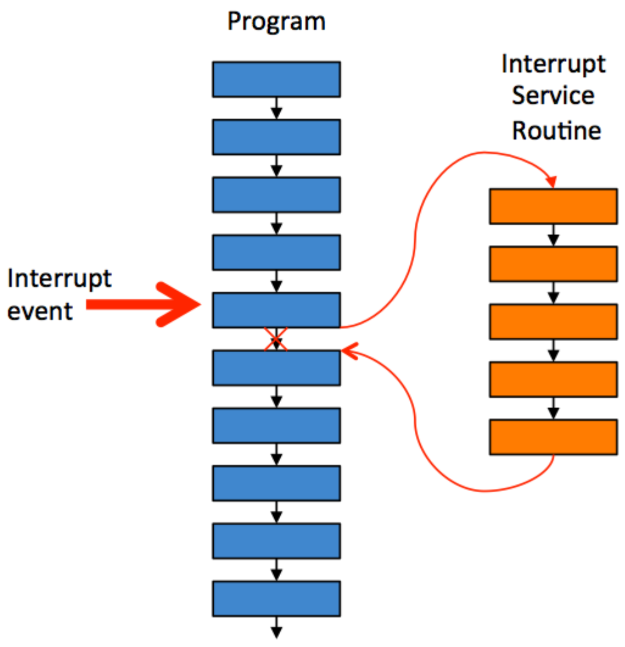
C10

Interrupts

This exercise is an introduction to working with interrupts using the `Il Matto` development board. The focus of the exercise is to learn to write programs that are event driven by interrupts. You will convert some of the previous programs that you have developed to work with interrupts.



Schedule

Preparation Time	:	3 hours
Lab Time	:	3 hours

Items provided

Tools	:	
Components	:	
Equipment	:	Saleae Logic Analyser [2], Oscilloscope [3]
Software	:	`avr-gcc`, `Eclipse`

Items to bring

Il Matto

Identity card

Laboratory logbook

Toolkit

Version: December 3, 2012

Steve R. Gunn
Electronics and Computer Science
University of Southampton

Before entering 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 undertake the exercise working with your laboratory partner. 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; at the beginning of the exercise one of the laboratory demonstrators will mark your preparatory work and at the end of the exercise you will be marked on your progress, understanding and logbook.

Notation

This document uses the following conventions:

✎	An entry should be made in your logbook

1 Introduction

This lab introduces you to working with interrupts on an embedded platform.

1.1 Outcomes

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

- Install an interrupt handler routine.
- Understand the importance of the `volatile` keyword when using interrupts.
- Use timer interrupts to improve the implementation of repetitive tasks.
- Use external interrupts to respond to external events.

2 Preparation

Study and familiarise yourself with the lecture notes on interrupts. Answer the following questions with reference to the `Il Matto` board running at 3.3V:

1. Describe the different interrupt sources available on the AVR.
2. What does the `volatile` keyword do?
3. How are interrupts enabled and disabled?
4. How do you install an interrupt handler in C to act on the *ADC conversion complete* event?
5. Give an example of a program where you would choose to use interrupts. Explain your decision.

3 Laboratory Work

Open the Eclipse application on the computer you are using and when asked for the workspace you wish to use, make sure the checkbox for *use as default* is not ticked and type `H:\ELEC1201\Labs\C10`.

Remember to create a new C project (File -> New -> New project.. -> C/C++ -> C Project), for each task of the exercise and name it with the appropriate part of the exercise for easy future reference, e.g. `C10_3.1`. This time you should select "AVR Cross Target Application" -> "Empty Project" as "Project type" and add in a new C source file.

3.1 Timer Interrupts

Take the counter program that you wrote in section 3.1 of Lab C6 and convert it to use a timer interrupt rather than the `_delay_ms` function. The code in the `main` function should now do the initialisation and then spin in a `while(1)` loop waiting for an interrupt.

It is not necessary to connect the seven segment display. Confirm the timing and program operation using the Saleae Logic analyser connected to port A.

3.2 ADC Interrupts

Take the proximity detector program that you wrote in section 3.2 of Lab C9 and convert it to use the ADC in free-running mode at maximum sampling rate. Use the interrupt service routine to handle all of the actions. The code in the `main` function should now do the initialisation and then spin in a `while(1)` loop waiting for an interrupt.

Modify your program so that you flip a bit on pin PB7 every time you enter the ADC interrupt handler. Use this to measure the maximum ADC sampling rate with the oscilloscope.

3.3 External Interrupts

Write a program which modifies an 8-bit value according to the following conditions: if a falling edge is detected on `INT1` the value should be incremented; if a falling edge is detected on `INT0` the value should be decremented. No wrapping should occur and hence it should not be incremented past 255 or decremented below 0. Whenever the value changes, your program should write it to port C. Your program should use interrupts to achieve the tasks.

To test your program you may use hook-up wire to act as a simple switch between ground and `INT0` or `INT1`. You may use the Saleae Logic analyser connected to port C to confirm correct operation.

4 Optional Additional Work

4.1 Multiple Interrupts

Combine the programs from section 3.2 and 3.3 so that the ADC sampling rate can be adjusted by pressing the two "buttons" on `INT0` and `INT1`. Modify the ADC auto trigger source to use *Timer/-Counter0 Compare Match* to trigger the ADC and write the 8-bit value from the external interrupt code to the register `OCR0A`. Now setup timer 0 in CTC mode and you should be able to manipulate the sampling rate of the ADC with the external inputs to the AVR.

What sampling rates can you get with your setup?

References

[1] Atmel Corporation. ATMEGA164PA/324PA/644PA/1284P Datasheet. Datasheet 8152G-AVR-11/09, 2011. URL http://www.atmel.com/Images/doc8152.pdf.

[2] Saleae. Logic Analyser (Logic). User's Guide 1.1.15, 2012. URL https://secure.ecs.soton.ac.uk/notes/ellabs/reference/equipment/std-bench/Logic%20Analyser-%20Saleae%20User%20Guide.pdf.

[3] Tektronix. Digital Storage Oscilloscope (TDS1000C). User Manual Rev. A, 2006. URL https://secure.ecs.soton.ac.uk/notes/ellabs/reference/equipment/std-bench/Tektronix_TDS1000C_2000C_User_Manual.pdf.