Electronic Design Project 2
Project: Sound level meter

John H. Davies, Fernando Rodríguez Salazar and J. M. R. Weaver

2011 March 7

Advance information

The project will start immediately after the Christmas vacation so we thought that you might like to have the briefing documents in advance. This gives you the option of doing a little background reading before the first laboratory. You will carry out the project in teams of about four students, which will be assigned at the first laboratory session.

Here is a reminder of the timetable for the first set of laboratories. It is part of the second year timetable issued in October by Dr Rodríguez.

• Groups 1 and 2: Monday 2011 January 10 and Monday 2011 January 17
• Groups 3 and 4: Friday 2011 January 14 and Friday 2011 January 21

The first laboratory for groups 1 and 2 is on the first day of teaching. It is essential that you attend on the correct day because the project is carried out in teams and all members must be present together. Please arrive on time because we start with a briefing. Laboratories take place in Rankine 709. You must have a laboratory record book with you at all times to keep notes of your work throughout this project.

Contents

1 Introduction 2
2 Preliminary investigation 5
3 Hardware design assignment 7
4 Final design and construction 11
5 Complete hardware 12
6 Construction and testing 14
7 PCB design flow 18
8 Software 20
9 Reflections 24
1 Introduction

The aim of this project is to design and build a sound level meter to show the intensity of sound picked up by a microphone. The system should be cheap, compact and ‘solid state’ – no moving-coil meters. This might seem familiar: the design of the analogue part of a sound level meter was the final part of the analogue laboratories for Electronic Engineering 1Y. We will use a similar specification and components for simplicity but you can’t simply copy your design from last year because some features make it unsuitable for integrating into a complete system.

That’s it: The details are up to you. A string of questions should come to mind. Here are a few to get you started in case your mind goes blank.

- What type of scale is used to measure the intensity of sound?
- Is an instantaneous display of the intensity the most helpful?
- What sort of display is appropriate?
- What is needed to drive the display?
- How can the number of components be reduced to keep the cost down?

Displays of sound level are common on audio equipment and in software for playing music, such as iTunes. These should give you some good ideas for an attractive and informative product. Credit will be given for novelty, how well the product performs and for low cost, which always means a compromise. The final product should be self-contained – it would be wonderful (but probably optimistic) if you have time to build it into a box.

The meter will almost certainly require a digital system to drive the display. A microcontroller is an obvious choice and we recommend a Freescale MC9S08QG8. This is a 16-pin device, which is similar in many respects to the larger MC9S08GB60 that you used in Electronic Engineering 1Y and will explore further in Embedded Processors 2. The MC9S08QG8 includes both an ADC and a comparator. A couple of pages from its data sheet are attached to give you an idea of its capabilities. Please consult us if you prefer to use something else. We don’t want to stifle novel ideas but we can’t support every possible device so you will need a convincing argument.

1.1 Overall plan for the project

The project is broken into three phases. These will take place at the start, middle and end of the teaching part of the second semester (before the Easter vacation) with gaps between.

1. Preliminary investigation. Any project needs some initial research. Start by drawing up a list of questions, starting with those above. The main theme of the first phase is to develop the overall design of the system and specify the analogue and digital parts.

2. Analogue design. The output of the microphone is a small analogue signal. This must be interfaced to the digital system. As you know from the lectures in this course, you can rarely connect a signal directly to an ADC but usually need an operational amplifier and so on. You will design the analogue part of the system in this phase of the project. It is a
design exercise and should not require practical work – you should have the information from the preliminary investigation. Much of this can be taken from Electronic Engineering 1Y but may need modification. For example, a bipolar power supply is inconvenient. Is it possible to get the microcontroller to do some of the operations that were carried out with analogue components?

3. **Overall design and construction.** The final part is to complete the design of the product, lay out a printed circuit board, and to build and test the board. Another major task is to programme the microcontroller.

As you can see, the project draws on many of the courses that you have studied this year and in the past. We teach you this material for good reasons! The meter is a *mixed-signal* system, which means that it includes both analogue and digital aspects. This makes the project difficult because you must bring a wide range of skills to bear on it. We have to teach different topics in separate courses for organisational reasons but the real world is not so cooperative. Almost all modern systems have both analogue and digital aspects and you will encounter the same challenges – but enhanced – in Team Design Project 3 next year.

### 1.2 Conduct of the project

You will carry out the project in teams of about 4 students. This gives you a chance to concentrate on the aspects of electronics that you enjoy most – digital, analogue or the interface between them. It cuts down the workload as well. We also hope that it will make the project more enjoyable. Teamwork introduces a few communication problems compared with the usual pairs and will help you appreciate the problems before Team Design Project 3 next year. We shall allocate the teams, following feedback from previous years.

Unfortunately teams bring a hazard: What do you do if somebody doesn’t turn up? Inevitably it will be the person who has the PCB design on his or her filespace and the rest of you have to start again. (Avoid this particular problem by keeping a copy on a USB drive.) We shall therefore monitor attendance carefully and will penalise absentees severely. It is not fair for one lazy person to wreck the work of a team and we may refuse credit to offenders.

### 1.3 Deliverables and assessment

Details are given in each of the sections that follows but here is a summary. The project counts for 40% of the overall assessment of this course.

- Each team must submit a brief report for each of the two preliminary parts of the project. These reports contribute 10% to your final mark.
- The final product and team report contribute 20% to your final mark.
- Your individual laboratory record books must be submitted at the end of the project and will be assessed. This is the main evidence of each student’s contribution and contributes 10% to your final mark.
- We may use peer-assessment software to judge the contribution of each member of the team. Alternatively we may ask for a short, confidential report. An adjustment may then
be made for each individual. In most cases this will be small or zero. However, we will refuse credit in extreme cases where we judge that a student has not made any worthwhile contribution to the project.

1.4 Laboratory book

It is vital to keep good records. There are gaps between the three parts of the project and you will have to repeat a lot of work if you don’t write sufficient notes to refresh your memory the next time. This is excellent training for future projects.

You are therefore required to use a laboratory record book throughout this project. It must be firmly bound; staples are acceptable but a loose-leaf refill pad is not. We recommend A4 size for convenience because you may wish to stick printouts in the book. It will be collected for assessment at the end and may be called in for examination at any time without advance notice – you are expected to have it with you at all times when working on the project. Here are some guidelines for using your record book, which we will consider when we assess it.

• Make brief notes of all your work for this project, including team meetings and individual study. Put the date on each entry.

• Laboratory books should be used as an immediate record. Do not scribble on loose pieces of paper and copy the result into your lab book. Beautiful, neat books with no crossings out will be penalised: real work isn’t like that. On the other hand, we must be able to read your writing.

• Just draw a line through writing that proves to be wrong, rather than obliterating it, because it often happens that the work was correct after all.

• Keep a note of all sources of information that you have used – helpful web sites, for instance – because they may be needed for the final report. Make a note of the vital points but don’t copy complete articles.

• We expect to see calculations, ideas for the design, rough sketches and so on – your thought processes.

• The most illuminating material is where you have developed your ideas for the project. Imagine that you fell ill and that the rest of the team had to take over your role: your laboratory book should enable them to reconstruct your thinking and recover from your absence.

Laboratory books will be returned after they have been assessed because the information may be valuable in future.
2 Preliminary investigation

Two afternoons in the first two weeks of the second semester have been set aside for the preliminary investigation. You will probably spend one afternoon (and more) reading and may wish to devote the other to experiments.

Your main task is to develop a specification and outline design for the overall system. This should include a diagram showing the main blocks with an estimate of the functions required for each block. Critical components may be specified but a detailed circuit is not expected at this stage. It is most important to specify as closely as possible what your system will do but not the details of how it will do it. You must also explain the reasoning behind your major decisions.

Pay particular attention to aspects that affect the overall system. An obvious example is the power supply. It is most convenient if everything runs from the same voltage, which should be matched to a convenient battery. (The system must be self-contained so you can’t use a bench power supply.) The battery must be able to supply sufficient current for the complete system, and so on.

You often have a choice of implementing functions in software, using hardware (modules) inside the microcontroller, or separate hardware. Aim to get as much value out of the microcontroller as possible.

The components from Electronic Engineering 1Y are available if you wish to do some experiments. The Electronics Stores offers a wide range of components and the catalogue is on the web at www.elec.gla.ac.uk/shopweb/ecs/ECSmain.htm (sorry, local access only). Please ask as early as possible if you need a component that is not in Stores so that we can order it in good time. Note that most of the opamps in Stores are unsuitable for low supply voltages. More appropriate devices include the MC33201, which we used last year, and the TS951, which cannot drive so high a current but has better performance in some other respects. Both are also available in dual and quad packages.

Copies of suppliers’ catalogues are available in the laboratory and here are the most useful web sites.

- RS Components – uk.rs-online.com/web
- Farnell – uk.farnell.com
- Rapid Electronics – www.rapidonline.com

The web sites work well if you have a good idea of what you want but the paper catalogues tend to be more helpful for inspiration. Look also at the reference books listed in the lecture notes, such as The Art of Electronics. Some books are available in our local library while others are in the short loan collection of the University Library.

2.1 Report

Each team should submit a formal, word-processed report on the preliminary investigation to Professor J H Davies before 1700 on Friday 28 January 2011. It should describe the overall design of the system and explain why you chose this approach. The report should not be more than two sides in length. Please show the name of your team but not the members’ names. No
individual reports are required but you must keep a record of work in your laboratory book, which will be collected and assessed at the end of the project.

Here are a few guidelines to help you write a good report. The moodle site has a longer document on *How to write a technical project report*, although it is really intended for longer reports than this.

- Obviously the report must describe your design. Even more important, it must explain why you made these choices. In other words, don’t write just ‘A non-inverting amplifier was chosen’ but follow this with ‘because its high input impedance avoids loading the filter’.

- Don’t repeat material from data sheets or anything else: Give a reference instead. Cite these in the text with numbers in square brackets, such as [1], and give a numbered list at the end of your report. These should include books, application notes and web pages but not the lecture notes for this course. References are essential.

- Be selective, particularly when quoting from data sheets. Does it matter that the op-amp has a ‘high output current’ in your circuit? If not, omit it. Concentrate on the aspects that are critical to your design.

- Be definite. Don’t say that something ‘could be suitable’, for instance. Is it or isn’t it? You need to know before you can draw the design.

- Be precise and quote numbers where possible: ‘AAA alkaline batteries should give a lifetime of 15 h at full power’ rather than ‘Alkaline batteries were chosen because they have a high capacity.’

- Use the passive voice in a formal report: ‘The voltage was 3.1 V’ rather than ‘I measured the voltage’. This document contains instructions and therefore uses a different style.

- An engineering report is not a story. Please do not write ‘First we did this, then Morag did that…’.

- Lay the document out neatly but don’t spend a lot of time on making it beautiful: It’s a formal report, not advertising. Use a clear font such as 12-point Times for the main text. Clear diagrams improve a report immensely.

- Use the equation editor for mathematics; $y = x^2$ looks far more professional than $y = x^2$.

- Read the final draft carefully. Use a spelling-checker and a grammar-checker if you wish but neither is a substitute for careful reading by a human.
3 Hardware design assignment

The second phase of the project is to complete the design of the hardware. This should prepare you to lay out the PCB in the final set of laboratories. Most work is needed for the analogue system before the microcontroller. The design of the output is simple if you are using only a few LEDs but needs more effort for a dual 7-segment display or a large number of LEDs. See the section on multiplexing displays below, which should complement Embedded Processors 2. First, here’s some feedback on the first reports.

3.1 Feedback on preliminary investigation

Some reports were inevitably better than others and a few had serious omissions, which leave you with a lot of work for the next stage. Here are the main points that I looked for.

• Clear block diagram of overall system. (This really needs a diagram.)
• Recognition that the intensity of sound is conventionally measured in decibels (dB) with a reasonable estimate of the range of values to be handled.
• Need for an amplifier after the microphone, with consideration of how this should be adapted for a single power supply.
• Practicable scheme for rectification and peak detection, which can be done in many ways.
• Appropriate display with justification for your choice.
• Idea of how the value from the ADC will be converted to a value for the display.
• Description of how often the display will be updated in time.
• Suitable voltage and source of power with reasoning to support your choice.
• List of appropriate references and citations in the text.
• Any novel features.
• A simulation to test your design (nobody mentioned one, which was disappointing).

Many reports suffered from two common problems.

• Waffly, imprecise text. ‘It might be possible to…’ is useless – can it or can’t it? Somebody has to decide and that person is you! Lack of precision was the major weakness. *Be definite.*
• Irrelevant information from data sheets. For example, ‘The MC33201 opamp was selected because of its high output current’. This is meaningless until you calculate how much output current your system needs. It may turn out to be more than the MC33201 can provide.

Many teams had clearly not spent enough time reading either the notes from Electronic Engineering 1Y or the summary of the microcontroller. You can’t do a successful project without thorough preparation.
3.2 Issues to consider for the hardware

Some of you have developed the complete circuit already and need only to calculate the values of components. Others have a lot more work to do. I have nothing to add to the suggestions in the earlier handout but it might be a good idea to remind you of the critical points.

- The supply voltage must be compatible with all components for simplicity. Try to avoid regulators. This is meant to be a cheap, simple system.
- The power supply must be able to provide sufficient current for all components.
- The amplifier must work correctly from a single power supply.
- The individual building blocks of the system must work correctly together. Draw the complete circuit and check.
- Check that you have enough free pins on the microcontroller. Two pins are needed for power and another for the debugging connection (BKGD). Look through chapter 2 of the data sheet, External Signal Description, to find the functions available on each pin.
- Aim for a realistic dynamic range. For example, a range of 100 dB means a ratio of maximum to minimum voltage of $10^5$. Can you get this from an 8-bit ADC?
- If you plan to use a numerical display, you must devise a method to calculate the value that it will show. Is it possible to implement the mathematical expression for decibels in the microcontroller, for instance?
- Specify values of all components.

I strongly recommend that you draw the circuit in Capture because that will help you to lay out the PCB later. You can also simulate much of the circuit.

Finally, because I know that this causes a lot of anxiety, I should repeat that there is no perfect design. Some designs might be better than others, and I have an idea for what I would use, but practical problems almost never have a single, perfect solution. The final report gives you a further opportunity to argue in favour of your design but this isn’t a mathematical problem with a unique answer. Real engineering isn’t like that.

3.3 Multiplexed, dual 7-segment displays

This explains how to drive a dual, 7-segment LED but the same method can be used for any display with too many LEDs to drive directly from the microcontroller.

The display can show numbers from 00 to 99 with optional decimal points to the right of each digit. This means that there are actually 8 segments to each digit, despite the name. Figure 1(a) on the following page shows the layout of a single digit and its internal circuit. The cathodes (negative terminals in forward bias) of all the LEDs are connected together to give a common cathode display; common anode displays are equally common. A data sheet for the Avago HDSP-K123 display is provided.

We need two digits, which would require 16 pins of the microcontroller if the displays were connected separately. This is excessive so we shall follow usual practice and multiplex the
Figure 1. (a) Physical layout and internal circuit of a single, common-cathode, seven-segment LED display. The segments are labelled in a standard way and ‘dp’ stands for decimal point. (b) Multiplexed connection of two digits. The eight segment lines and the gates of the n-MOSFETs, one per digit, are connected to pins of the microcontroller.

displays. The general idea is to show the two digits alternately – first the units only, then the tens only, then the units again, and so on. The eye averages the intensity and sees a steady image of both digits provided that the frequency of multiplexing is around 100 Hz or higher. The flashing becomes obvious if the frequency is too low.

The circuit is shown in figure 1(b). Note that digit 1 is on the left, the more significant digit. The corresponding segment pins of the displays are connected in parallel (both segments ‘a’ connected together, both segments ‘b’, and so on). The common cathodes are used to select the active digit by inserting a switch between the cathodes and ground. No current can flow if this switch is open and all segments remain dark, whatever the voltage applied to the anodes. A digit becomes active when its switch is closed and current can flow through the segment lines, resistors, LEDs and switch.

The switches are transistors of some sort so that they can be operated by the microcontroller. I have shown n-channel MOSFETs in figure 1(b). These work in much the same way as npn bipolar transistors but give superior performance. Briefly, this is how they work.

• When the gate is grounded (the left-hand terminal on the schematic), no current flows through the channel (between the two terminals on the right, called source and drain).

• When the voltage on the gate is raised through a critical value called the threshold voltage $V_t$, the channel starts to conduct and current can flow through the channel.

The gates of the MOSFETs can be connected directly to digital outputs of the microcontroller; no resistor is needed. We shall use a dual, n-channel MOSFET to reduce the number of components. It comes in a SOIC-8 package (or SO-8, which means the same) and the data sheet is provided. Note that a digital FET is simply a MOSFET with a threshold voltage that suits digital circuits – there is nothing digital about its operation.

Aside: Why are MOSFETs better than bipolar transistors? Here are the two main reasons. First, the gate is like a capacitor and draws no input current in a steady state. This is unlike a bipolar transistor where $I_b = I_c/\beta$ and $\beta$ is low when the transistor is used as a saturated switch.
Second, the channel of a modern MOSFET has a lower resistance than a bipolar transistor when it is turned on. A minor advantage is that no resistor is needed in series with the gate because no current flows.

A resistor should always be connected in series with each segment to limit the current through the LED to a few mA. Multi-packs of resistors are convenient for this application because we need 8. The packages are typically single-in-line (SIL) or dual-in-line (DIL) and we have some 8-pin SILs, each of which contains 4 independent 270Ω resistors. This should make it easier to lay out the PCB.

### 3.4 Report

Each team should submit a formal, word-processed report on the design of the hardware to Professor J H Davies by 0900 on Friday 25 February 2011. The main text should be one or two sides in length. Attach a separate page for the circuit diagram. Further pages may be used for the results of simulations. Please show the name of your team but not the members’ names. This time you need not list references to my lecture notes or data sheets for components in stores but should include references for any other sources that you use. No individual reports are required but you must keep a record of work in your laboratory book, which will be collected and assessed at the end of the project. Remember to note the date on entries in your laboratory book as you work on this project.
4 Final design and construction

You now have several laboratory sessions to complete the project:

- Finish the design of the hardware if necessary (it should already be complete).
- Lay out a PCB for the complete system.
- Write the software for the microcontroller, which includes several tasks.
- Populate the PCB with your components and test each aspect of the system.
- Write the report.

These are major jobs and you will not be able to accomplish them unless you work as a team. There is no point in having three people sitting and watching the fourth person at a computer: you need a strategy for getting the project done. Two people might work on the hardware and two on the software, for instance. Somebody might start on the report toward the end. Of course these jobs can’t be done in isolation so the whole team must discuss progress regularly.

Testing is critical. Don’t think that you can build the whole system, download the software and expect it to work first time. The hardware must be built in a systematic way and tested as soon as another functional block is added. This is even more true for the software. Develop the program in small modules and test each as soon as possible.

4.1 Deliverables

We hope to get some independent reviewers along for their opinion of your products. The results should be included in your final report, which must be submitted to Professor Davies by 1700 on Friday, 2011 March 18. It should include the following material.

1. Final schematic design of the hardware (printout from OrCAD) with explanations of why you chose this particular circuit. Do not repeat anything from the earlier reports.

2. A Bill of Materials (BoM, from OrCAD) with estimated cost of components. The PCB costs 20p per square inch.

3. A description of the final software, including a clearly commented listing (printout from CodeWarrior).

4. Brief operating instructions, addressed to a non-technical user.

5. Comments on the performance of the system and suggestions for improvement in both the hardware and software.

An example of a report for last year’s project is available on moodle.

Include your earlier reports so that the document covers the complete project. The new parts should not be longer than four pages plus printouts. Please show the name of your team but not the members’ names. Hand in your individual log book at the same time (with your name!!).
5 Complete hardware

You should have designed all the hardware already but I’ll run through a few aspects of the display and microcontroller that had not been considered by all teams.

5.1 Microcontroller

The system will be based on a 16-pin Freescale MC9S08QG8 microcontroller. It is similar in most respects to the larger MC9S08GB60 that you have used in Embedded Processors 2. Unfortunately its ADC is rather different but a chapter of the lecture notes covers this module.

Look through chapter 2 of the data sheet, *External Signal Description*, to find the functions available on each pin and the external components that the microcontroller needs. You may need the following connections to other parts of the system, depending on the display that you have chosen.

- 8 or more digital outputs to drive the segment lines of the display or the LEDs directly.
- 2 digital outputs to drive the MOSFETs that select the digits of a dual 7-segment display if you have chosen this.
- 1 analogue input to the ADC from the opamp or peak detector.
- 1 digital input from a pushbutton if used.

That’s at least 10 pins. The microcontroller needs a few more for itself, which are described in section 2.2 of the data sheet, *Recommended System Connections*.

- 2 pins for power supply and ground, \(V_{\text{DD}}\) and \(V_{\text{SS}}\).
- 1 pin for the BKGD connection, which is used to programme and debug the chip. This goes to pin 1 of the 6-pin background header, which also needs connections for \(V_{\text{SS}}\) and \(V_{\text{DD}}\). An interface called a BDM pod goes between this header and a USB port on a PC.
- The reset connection is optional on this device. It cannot force the microcontroller into debug mode and therefore serves little purpose.

We are now up to a minimum of 13 pins so the 16-pin package may have some left over for extra features! Most designs with microcontrollers start like this because the number of pins is often a limitation.

The microcontroller needs a clock. The internal RC clock can be trimmed to sufficient accuracy (about 1%) as described in section 8.3 on page 21.

The next job is to assign functions to the pins. Section 2.1 of the data sheet, *Device Pin Assignment*, contains the information needed. The functions of some pins, such as \(V_{\text{DD}}\) and \(V_{\text{SS}}\), are fixed but most offer a choice. There are 8 possible analogue inputs to the ADC, for example. Try to find an arrangement that will make the PCB simple without complicating the software. Here is a little guidance.

- One pin is input only and another is output only.
- Port B has the full 8 pins but port A has only 6, several of which have fixed functions.
- The display probably need 8 outputs (at least) and the software is simpler if they use a single port.
5.2 Other components

Only a few other components are needed.

- A resistor should be connected in series with each LED to limit the current to a few mA. Multi-packs of resistors are convenient because they make it easier to lay out the PCB and we have several varieties.
  - The resistors for a simple display can all be connected together at one end. This is called a resistor network and we have sets of 5 or 8 available in 6 or 9-pin single-in-line (SIL) packages.
  - A multiplexed display needs independent resistors. We have networks of 4 independent 270Ω resistors in 8-pin SILs.

- Both the opamp and microcontroller should have 100 nF decoupling capacitors connected as close to each package as possible. These should be multilayer ceramic components (available from stores). They stop digital noise spreading from the microcontroller into the opamp.

- Connect a further 10 µF electrolytic decoupling capacitor across the power supply. Its placement is not critical.

- Holders that can be mounted on the PCB for single AAA, 2 × AAA and 2 × AA cells are available.

- A sliding switch for power should also be mounted on the PCB. We have a suitable SPDT switch, which stands for ‘single pole double throw’. In other words, it switches a single input between two possible outputs. (A simple on–off or SPST switch would be sufficient but is not available.) A dual power supply needs a DPDT switch, which is effectively two SPDT switches in parallel.

Most components are in stores but a few special ones are kept for this course. These are:

- microcontroller and USB–BDM pod
- opamps: TS951, TS952, MC33201, MC33202, all in PDIP8, and MCP6281 in SOIC8
- displays: Avago HDSP-K123 dual 7-segment common-cathode display and Avago HDSP-4832 multicolour bargraph display
- dual MOSFET: FDS9926A in SOIC8
- miscellaneous: microphone, 6-pin BDM header, SPDT switch, DPDT switch, 5 × 330Ω networked resistors in SIL5 and 4 × 270Ω independent resistors in SIL8 packages

Datasheets are provided on moodle.
6 Construction and testing

*Bring your instructions for OrCAD PCB Editor.* Mr Young’s *PCB design flow* is reproduced in section 7 on page 18. Follow it carefully to avoid common errors.

6.1 Schematic drawing

You have had plenty of experience with Capture so nothing should be unfamiliar. *Make a new folder for your project first.* The only awkward aspect is where to find the components. We started with the analog library in the course at the beginning of the year because part of the exercise was to simulate the circuit. If you do not plan any simulation, the discrete library is a better choice. This is in the directory one level above the pspice directory (OrCAD16.3/tools/capture/library/). Fewer ‘fixups’ are needed with this library, which also contains some handy parts. Two local libraries are helpful:

- library for this course, edp2capturelib.olb, which can be downloaded from moodle (separate libraries are required for Capture, PSpice and PCB Editor).
- project library, Q:/TDP3_ORCAD/Projects.olb on the network

Here are a few tips to help you, but most of the procedure is in Mr Young’s summary.

- The two MOSFETs are in a single package, called a dual MOSFET. Their references are therefore Q1A and Q1B rather than Q1 and Q2. The two opamps in a dual package are treated in the same way.

- Do not run wires for the power supplies to the opamps. Connect them by name instead. The pins are called V+ and V− for all op-amps as far as I know, so add power symbols with these names to the nets for power and ground. The power supply for the microcontroller can be handled in the same way.

- Other connections can also be made by name, using a *net alias*.

- The power nets must be named so that you can increase the widths of their tracks in PCB Editor.

- In the one-transistor amplifier that you used for your first PCB design, we replaced the battery by a connector for power. You can keep the battery in this project because it will be mounted on the PCB. Use the battery from the discrete library.

- The power switch is an SPDT type, not a simple on–off (SPST) switch (page 13).

- Add test points to make your PCB easier to test. A test point is a single pin, which is extremely helpful when you use an oscilloscope probe to investigate the circuit. Provide points for ground, the analogue input to the microcontroller and other critical points.

You have a lot of choice in assigning functions to pins of the microcontroller, particularly for the display. Sketch on paper the relevant pins of the microcontroller and display and work out the simplest way of connecting them. A neat layout for this part of the board will make assembly much easier.

**Milestone:** Get your schematic drawing checked before starting to lay out the PCB.
Surface-mount components

Most modern components are designed for surface mounting, which means that they are soldered directly to tracks on the same side of the PCB as the body of the component; they do not have pins that penetrate the board to tracks on the opposite side. The dual FET for multiplexing is available only in a small outline integrated circuit (SOIC) package. Its pins are 0.05” apart, half the spacing of a traditional dual-in-line IC. This is the largest type of surface-mount package and should therefore be an ‘easy’ introduction to working with them. The MCP6281 opamp is also in a SOIC package.

6.2 Printed circuit board

Assign footprints in Capture. Use the components provided to select the correct footprint. Here are a few points to watch.

- Remember that all names of footprints begin with gu-

- Most footprints are in the departmental library on the Q: drive, which you used at the start of the year. A few extra footprints are provided in a small library for this course, edp2pcblib. Download it from moodle as a zip file and expand the archive into your allegro directory.

- The integrated circuits will be mounted in sockets so choose the dipsock footprints.

- Check the assignment of the pins carefully for unfamiliar components, such as LEDs and switches. The polarity matters! Use Edit Part if the numbers need to be changed, as we did for the electrolytic capacitor.

Lay out the components on the bench to estimate the size of the board. Err on the large side; the dimensions can be reduced later if necessary. Use the usual 25 mil design rules. Create the netlist to add the components to the board. Get help if the netlister fails. The next step is to lay the components out on the PCB, which is the most critical part of the layout procedure.

- Place decoupling capacitors close to the power pins of the integrated circuits.

- The footprint for the battery holder includes two mounting holes as well as two standard pins for the positive and negative terminals. Do not run tracks to the mounting holes.

- You can rearrange the connections from the microcontroller to the LED display to simplify the layout but don’t forget to tell the programmers! Try to work out the best arrangement, then follow the instructions for I’ve just spotted an error in the circuit to update your schematic drawing and send the changes to PCB Editor. You don’t have to restart the PCB layout from scratch.

- All components are usually mounted on the top of the board. If we do this with a surface-mount device, it means that you must use a double-sided board. Try to avoid this by mounting the SOIC package on the bottom of the board instead. Use the Mirror command in PCB Editor to swap the component from the top to the bottom surface. (Don’t do this for anything else!) It’s a bit ugly but it is much easier to work with single-sided boards. (Mr Fairbairn, who makes the PCBs, will be grateful.)
• Add text to identify your team to the bottom etch layer (mirrored) and to the top (not mirrored) if used. Make it large enough to be legible!

Route the board when the components are optimally arranged. *Manual routing is best.* If you must use the autorouter, start with a single-sided board and tracks only on the bottom. If this does not work, and only a few tracks are not routed automatically, do them by hand and put vias in convenient places. Use wires on top instead of requesting a double-sided board.

If you choose a double-sided board, follow the instructions carefully. You might as well keep the SOIC package on top. It is particularly important to set up the vias correctly. We shall install the microcontroller and op-amp in sockets, which means that *you cannot run tracks to their pads on the top of the board.* The dipsock footprints forbid this because the pads will be under the sockets and inaccessible. Try to avoid vias on the ground and power nets.

Check carefully for vias that are too close to components or underneath them. You will almost certainly have to create space by sliding tracks so that the vias can be moved away. The autorouter will happily run tracks to the top pads of many components, which will be difficult to solder. Look for these too, or you will have a miserable time soldering the board. Vias are bad news!

Add text to the top etch layer (if used) to identify the connections and mark pin 1 of the BDM header. You may wish to add text to help you connect the LEDs and other components with the correct polarity.

**Milestone:** Get your layout checked before sending it for manufacture.

When you are satisfied with your layout, send the board (.brd) and pdf files to Professor Davies, who will print it on to tracing paper for the electronics workshop. Please give the file a distinctive name, although this should be clear from the text on your board, shouldn’t it? Make the usual colour plot to help you assemble the board because we do not produce a silkscreen. You need this for the report as well.

### 6.3 Construction

Here are some tips for populating the PCB. Please wear goggles and follow the usual safety precautions for soldering. *Ask us if you are in any doubt.* Plenty of scrap components are available if you wish to practise before starting on the real thing.

• First check the printed circuit board itself: look for breaks in the track or narrow whiskers of copper that shouldn’t be there.

• If you made a single-sided board, write text on top of the PCB to identify all connectors and mark pin 1 of the BDM header. The pad for pin 1 is square rather than circular or oval.

• First solder the surface-mount package (if used). Mr C. Hardy from the Electronics Workshop will provide assistance.

• Next solder the vias. Poke a wire through each hole and solder it top and bottom. If the board is single-sided, you may need some wires on the top of the PCB instead.

• Install sockets for the microcontroller and op-amp. Orient them correctly.
• Install the remaining components, starting with the lowest and ending with the tallest.

• Check that components are correctly oriented where this matters. These include LEDs, electrolytic capacitor and battery.

• Fix a length of wire through the mounting holes on the battery holder and solder each end to fasten it more securely.

• Sticky plastic feet are available from stores.

Check your PCB visually after everything has been soldered into place. Have you accidentally bridged any gaps or missed any pins?

Test the ground and power tracks on your PCB for continuity. Use a multimeter on its ohmmeter or continuity setting (some of them beep, which is helpful to you but annoying for everybody else). Check that there is a low resistance (well below 1 Ω) between the power connector and and the sockets for the ground and power pins of the integrated circuits. If the resistance is high, suspect the vias.

Insert the cells into the holder and confirm that the correct voltages are present on the power pins of the sockets for the integrated circuits.

♫ Milestone: Show your board and test results to a member of staff. You will be issued with the ICs when the board has been approved.

6.4 Testing

Here is a suggested strategy for testing the hardware.

1. Test the analogue front end by itself before inserting the microcontroller. Confirm that its output voltage is as expected.

2. Remove the opamp, plug in the microcontroller, connect the BDM pod and try to download a simple program. If all is well, test the display by itself.

3. Replace the opamp and test the ADC. It is easy to read the result in the debugger.

4. Test the whole thing!

Now impress your friends with it. Congratulations!
7 PCB design flow

This checklist is adapted from a handout by Mr I. Young. Save your work frequently!

1. Make a new directory for each design. Create an allegro directory within it.

2. Start Capture, create a new project in the new directory and draw the circuit.
   - Select an Analog or Mixed A/D project if you wish to simulate the circuit. Alternatively, the PC Board Wizard gives you the choice of simulation or not and offers appropriate libraries.
   - Use libraries in the pspice directory if you wish to simulate your design. If not, you may prefer libraries in the library directory, one level higher. For example, you could take passive components from library/discrete rather than library/pspice/analog. These cause fewer warnings from the netlister.
   - A helpful, local library of symbols is available in Q:/TDP3_ORCAD/Projects.olb.
   - Use wires to join components, symbols and junctions – don’t just push them together.
   - Provide power connections for all integrated circuits. If power pins are hidden you must place power symbols with matching names on the supply rails.
   - Include a battery or a connector for a power supply.
   - Check the circuit carefully before starting the layout: It is much easier to correct mistakes now. If you spot an error in the circuit later, refer to the PCB handout.

3. Prepare the schematic for PCB Editor.
   - Get the real components first so that you can match them to footprints.
   - Edit components with incorrectly numbered pins, such as the electrolytic capacitor in the analog library.
   - Mark all unconnected pins with the No Connect symbol.
   - Add the No Connect property NC to packages with pins that are not shown on the symbol in Capture (for example, pin 8 of single op-amps in DIP8 packages).
   - Add footprints from our local library, prefix GU-. The symbols are in the directory Q/allegro/pcb_lib/symbols. In most cases you should use the dipsock footprints rather than dip for integrated circuits to prevent tracks running to the top pads.
   - Footprints for a few special components are provided on the moodle page for this course. Download the archive into your allegro folder and unzip it. Check these footprints against the symbols in Capture and the real components carefully. In particular, make sure that the pin numbers match.
   - Run a Design Rules Check. Check the Session Log even if no warning dialogue box is shown and correct any errors. Print the final schematic to guide the layout.

4. Create a bare board in PCB Editor using the New Board Wizard.
Set up the search paths if necessary. Add Q:/allegro/pcb_lib/symbols to psmpath and similarly for padpath.

Use 25 mil design rules so that your board will be easy to etch and solder.

Make the size of board generous; you can reduce it later.

Choose GU-VIA80 for the default via.

5. Netlist the design in Capture. Check all warnings in the the Session Log carefully even if PCB Editor appears to be launched successfully.

6. Place the components on the board in PCB Editor. Pin the Find and Options panels open to control the operation of the tools.

Take time to arrange and orient the components to simplify the ratsnest as far as possible. This step is critical to get a well-routed board.

Add mounting holes if required. Do not place them too close to the edge.

Use Setup > Constraints > Physical... to increase the width of tracks for the power supplies and other tracks that carry a heavy current.

Fix the list of vias so that only GU-VIA80 is shown and increase the spacing around vias.

Update the Design Rules Check and correct any errors.

7. Route the board.

Manual routing is best for a simple design (which includes most of ours).

If you must use the autorouter, try a single-sided board first. Use Display > Status... to check that the completion rate is 100%.

If only a few tracks are not routed automatically, complete them by hand and put vias in convenient places. Use wires on top instead of a double-sided board.

If you must use a double-sided board, review the results of the autorouter carefully. Move any vias and tracks to pads on the top that cannot be soldered easily.

Gloss the design in PCB Editor or PCB Router.

8. Add text on the etch layer(s) to identify the board and connectors. Mirror text on the bottom.

9. Write photomasks as pdf files at actual size. Follow the instructions carefully and remember to select a filmmask in the Options control panel so that guide holes appear in the pads.

10. Send your board (.brd) and pdf files to Professor Davies. He will check the layout and print the photomasks.

11. Take the photomasks to Mr S. Fairbairn in the electronics workshop (room 712A). He will etch and drill your PCB. Finished boards are left in pigeonholes in the electronics stores, just outside the workshop and data library.
8 Software

The software is a major part of this project and the final program will be larger than anything that you have written in previous electronics courses. It is therefore essential that you design the software before typing anything into the computer, just as you design a circuit before laying out a PCB. In other words, you must approach the software systematically. Break it into modules, each of which can be tested before you put them together. This should be second nature to those of you who have taken Computing Science 1P. We shall look for a professional approach in your laboratory books when we assess them.

Most of the topics in this section have been covered in Embedded Processors 2 but are repeated here for completeness. I’ll assume that you are using a dual 7-segment display, which must be multiplexed, and that the input comes from an external peak detector. This means that you have two tasks to perform at different intervals of time. Many teams have taken different approaches, which require other structures for the software.

8.1 Tasks

After configuring the system, the program requires at least two tasks to be run at regular intervals. They have rather different characteristics.

- **Multiplexing the display.** The display should be multiplexed at 100 Hz or more to avoid flicker, which means that the active digit should be swapped at 200 Hz.

- **Updating the display from the peak detector.** This should also be performed regularly, but at a lower frequency so that the user has time to read the value – perhaps every $\frac{1}{2}$ s.

You also need to trigger conversions of the ADC and may require further tasks, such as reading the pushbutton.

The professional way of performing tasks regularly is to trigger them from a timer running in hardware. Please do not use a software delay! The MC9S08QG8 offers three timers. The modulo timer (MTIM) is intended for triggering periodic tasks and you might wish to consider it. However, we recommend the familiar timer/pulse-width modulator (TPM). It usually drives periodic outputs but can equally well be used to schedule tasks in software. Both the timer core and the two capture/compare channels can set flags and request interrupts.

8.2 Paced loop

The simplest structure for a program to run tasks at regular intervals is a *paced loop*. This is an infinite loop that waits for a timer to roll over, then performs the sequence of tasks. When it has finished these tasks, it goes back to waiting for the timer. Thus the frequency of the loop is regulated by the timer. Here is an outline of a paced loop in a mixture of C and English.

```
for (;;) {  // Start of infinite, paced loop
    while (TimerOverflowFlag == 0) {  // Wait for timer to roll over
        Clear timer flag  // Do not forget this!
        ++TickCounter;  // To keep track of time
        Carry out task 1

```
The TickCounter keeps track of the total elapsed time. This name is commonly used because each overflow of the timer acts like the tick of a clock. It is used to schedule tasks that do not need to be performed during every loop. For example, the paced loop might run at 200 Hz to multiplex the display but the number displayed might be changed at only 2 Hz, once every 100 loops.

### 8.3 Configuration of microcontroller

Most of this should be familiar from Embedded Processors 2 but some registers have different names in the QG8. Disable the watchdog by writing to the system options register 1 (SOPT1). If you want to use stop mode (ambitious) or the reset pin, the corresponding bits in SOPT1 must be enabled as well. Check the data sheet and header file. You can write to SOPT1 only once, after which it locks.

Configure the input/output pins next. Set the appropriate bits in the data direction registers for pins to be outputs. Activate the internal pullup resistors for pushbutton inputs and unused inputs to stop them floating.

The clock is less familiar because we have generally used the default settings in the past. The internal clock can be calibrated (trimmed) to make its frequency more accurate. Freescale store a trim value in the non-volatile memory at the factory, which must be copied into the trim register of the internal clock source (ICS). The following lines do the trick.

```c
ICSTRM = NVICSTRM; // Retrieve stored trim value
ICSSC_FTRIM = NVFTRIM_FTRIM; // Fine trim bit
```

The bus should now run at 4.0 MHz ± 1%. Other registers in the ICS can be left with their default values. If you think that this is not fast enough, you can double the speed of the bus to 8.0 MHz with a further line. (The maximum is 10 MHz for this device.)

```c
ICSC2 = 0; // Disable bus divider (doubles frequency)
```

### 8.4 ADC

*Please read the chapter of the lecture notes on the ADC in the MC9S08QG8.* It explains how to configure and use the ADC. Important features to check include the number of bits, ADC clock, time for charging the input capacitance and resolution.

### 8.5 Display and multiplexer

This section is relevant only if you are using a numerical display. Skip it if you are using simple LEDs instead.

A dual display should be multiplexed at a frequency of about 100 Hz to avoid flicker. This means that you should swap the active digit at about 200 Hz. The procedure is straightforward.

1. Turn off the old digit selector (the MOSFET).
2. Put the pattern for the new digit on the segment lines.

3. Turn on the new digit selector.

There is a fairly long chain from the numerical value of a digit to the segments of the LED:

digit → pattern of segments → pins of port → pins of display → segments of LED.

For example, you might have digit ‘1’ → segments b and c → port pins PTB1 and PTB2 → pins 10 and 15, and 3 and 8, of the display module. This depends critically on the way in which the display module is connected to the pins of the microcontroller, which will be chosen by the PCB designer to make routing as simple as possible. Use symbolic constants in the definitions to make your program easy to read and maintain. For example, a set of \#define statements shows the mapping from segments of the display to pins of the port that you are using. Look back at figure 1 on page 9 for the standard definition of segments and the connections to the display.

\begin{verbatim}
#define SEG_A PTBD_PTBD0_MASK // AAAA
#define SEG_B PTBD_PTBD1_MASK // F B
#define SEG_C PTBD_PTBD2_MASK // F B
#define SEG_D PTBD_PTBD3_MASK // GGGG
#define SEG_E PTBD_PTBD4_MASK // E C
#define SEG_F PTBD_PTBD5_MASK // E C
#define SEG_G PTBD_PTBD6_MASK // DDDD
#define SEG_DP PTBD_PTBD7_MASK // decimal point
\end{verbatim}

This may not be the best mapping – I just chose it because it is obvious. Next define a constant array that gives the segments required to display each character.

\begin{verbatim}
const unsigned char LEDChar[] = {
    SEG_A | SEG_B | SEG_C | SEG_D | SEG_E | SEG_F, // digit 0
    SEG_B | SEG_C,       // digit 1
    // and so on for all desired patterns
};
\end{verbatim}

The element LEDChar[1] contains the appropriate values that should be driven onto the segment pins to display the digit ‘1’, for example. If your PCB designer decides to rearrange the connections between the output port and the LED display, all you have to do is change the segment definitions to match. Everything else works just the same.

### 8.6 General programming guidelines

Most of these should be familiar but a reminder is never a bad idea.

- Use symbolic constants wherever possible, not numerical values. The header (.h) files contain definitions for all bits in special function registers.

- Copy and paste constants from the header files into your program because they are cumbersome and unreliable to type. Freescale has an annoying habit of changing the names with each version of CodeWarrior, which sometimes leads to inconsistencies between the header file and data sheet.
• Lookup tables should be defined with the `const` keyword as in the `LEDChar[]` array above. This tells the compiler that the values are constant and can be stored in flash memory. Otherwise it assumes that they are variable and must go into RAM, which could easily fill up.

• Check that variables have sufficient storage to hold the range of values expected. For example, a signed byte can hold \(-128\) to \(+127\) and the range of an unsigned byte is \(0\) to \(255\). The *HC(S)08 Backend* chapter of the CodeWarrior compiler manual gives the default sizes and ranges of different types of variable.

• When you set up the project in CodeWarrior for your software, ensure that you choose the correct device. It is an MC9S08QG8, not the same as in Embedded Processors 2.

• The project must be configured for floating-point numbers if you use them (probably not). When you see *Select the floating point format supported*, I suggest that you select `float` is IEEE32, `double` is IEEE64. The other defaults should be fine.

Remember to initialise everything at the start of the program. The display should start up with something sensible, for instance. Don’t forget to activate the pullup resistor if you use a pushbutton.

### 8.7 Testing

We can’t over-emphasise the importance of writing the program in small modules and testing each thoroughly before putting them together. Almost all of this can be done using the simulator before the hardware is ready. The simulator includes all the peripherals and can even mimic the ADC. Use breakpoints and single-stepping to test critical parts. Put important variables into Data windows so that you can watch them. Check timing using the *CPU Cycles* counter in the Register window.

**Milestone:** Describe your procedures for testing to a member of staff. You will not be issued with a BDM pod until we are satisfied with your results.

When the hardware is available, you could start by testing the software for the display. Confirm that the values for the segment lines on the output port alternates between the two patterns expected for the digits. The outputs to the MOSFETs should change at almost the same times. Go on to check that numbers are displayed correctly if appropriate.

Work on the ADC next. Check that the output changes as expected when the input voltage varies. Is this mapped correctly to the expected output? Finally, put it all together.
Figure 2. Simulation of single-supply amplifier used a MCP6281 opamp. The output (green) is grossly distorted by the recovery from saturation.

9 Reflections

Here are a few comments on the final design, construction and performance of the products. Almost all of the teams produced a working product, which is excellent, and I was pleased to see the variety of approaches.

9.1 Hardware

The analogue design proved awkward with a single supply. In principle you could use the usual inverting or noninverting amplifier on a single supply and rely on the absence of a negative supply to provide the rectification. A problem is that this drives the opamp well into saturation near ground and it can take time for the circuit to recover and give the expected positive output after the input has changed sign. Many simulations gave alarming results. I simulated a circuit with an MCP6281, a modern opamp that I thought would work well in this circuit, and the results, shown in figure 2, were appalling. Several teams provided a $-1.5\text{V}$ supply, which is ugly but safe. Others may have hoped for the best and were lucky that the real circuit seems to work much better than the simulation. I found this when I tested a circuit with an MCP6281, as shown in figure 3 on the next page, but unfortunately the gain is lower in the real circuit than the simulation and this may improve the performance around saturation.

The best approach is to redesign the circuit to avoid saturation. This can be done by biassing the amplifier so that the output is at $\frac{1}{2}V_{\text{CC}}$ when there is no input (the operating point). A couple of teams did this, which was excellent design. I did the same and you can see the circuit in the schematic drawing, figure 4 on page 26. It works as follows.

- The potential divider formed by $R_2$ and $R_3$ sets $V_+ = \frac{1}{2}V_{\text{CC}}$. (Capacitor $C_1$ removes noise from this voltage.)

- Capacitor $C_2$ blocks input at DC so the only connection to $V_-$ is the feedback resistor $R_4$. The circuit is therefore a voltage follower at DC so $V_{\text{out}} = V_+ = \frac{1}{2}V_{\text{CC}}$. 


Figure 3. Measurement of input and output voltages for a noninverting amplifier with a gain of +100 using a MCP6281 with a single supply. The leading edge of the half-cycle is distorted because the opamp takes some time from being driven into saturation.

- For the AC input signal, $V_+$ appears to be grounded so the circuit looks like an inverting amplifier. Technically it is a transresistance or transimpedance amplifier rather than the usual inverting circuit because I omitted the resistor in series with the input.

The idea is that the output looks something like $V_{out}(t) = \frac{1}{2}V_{CC} + V_{signal}\cos(\omega t)$ as shown in figure 5 on the next page. The full sine wave appears on the output until its amplitude exceeds $\frac{1}{2}V_{CC}$, in which case it is clipped.

Many teams used the classic rectifier and peak detector as in Professor Weaver’s design. The problem with this is that low sound levels do not produce enough voltage for forward-bias the diode and are therefore not detected by the ADC. The alternative is to apply the signal directly to the ADC, without an analogue peak detector, and do the peak detection in software. I took this approach and so did several teams.

9.2 Printed circuit board

A few teams had problems because they used a variety of ground symbols on their schematic drawings. Always use the same symbol throughout because they will not be connected together if they have different names.

All teams but one completed a PCB, which is excellent. The layout was fairly straightforward with simple LEDs but tricky for a 7-segment display. Many teams were able to manage with a single-sided board, which pleased the workshop. I simplified the layout by connecting the LEDs active low, as shown in figure 6 on page 27, which is a bit sneaky. The two least significant LEDs are connected to port A and the remaining 8 most significant LEDs all to the complete port B, again to simplify the layout.

Several teams discovered that soldering vias is a major nuisance. They are particularly troublesome if power nets have several vias in series. Another problem was footprints that did not match the components. You must get the real components and choose footprints to match.
Figure 4. Schematic used for the prototype. The inverting amplifier is biased at the midpoint of the power supply to preserve the full AC signal. The gain proved excessive and the 330 kΩ resistor had to be reduced. The LEDs in the bargraph display are connected active low to simplify the PCB.

9.3 Software

The major problem was poor design, with a few honourable exceptions. It is always a good idea to make something simple first, and add the fancy bits later. Software is difficult to test, even with modern debugging systems.

Please use symbolic constants wherever possible. I don’t know what TPMSC = 0xA3 means without looking at the data sheet; symbols make it clear.

Interrupts are tricky. If you enable an interrupt, be sure to provide a corresponding interrupt service routine. Do not enable interrupts globally until everything has been configured; typically EnableInterrupts is the last statement before the main, infinite loop.

If the debugger reports that the clock frequency is changing, the program has lost control. The usual reasons are that you have enabled an interrupt without an ISR or the program has escaped from its infinite loop and the MCU is executing instructions from unprogrammed memory. Unexpected resets also occur if you have forgotten to disable or feed the watchdog.

Almost no reports explained how the values from the ADC were converted into dB in the software, or treated as logarithmic values. Several programs contained lookup tables to do this conversion but almost nobody justified the numbers.

Figure 5. Sketch of output of AC coupled amplifier with offset voltage when (a) no input is applied (bias point) and (b) with a sine wave input.
Figure 6. Layout of the PCB for the prototype. The board could probably have been routed entirely on the bottom but two links on top (shown green) gave a simple layout. The surface-mount opamp has a red outline because it is mirrored onto the bottom of the PCB. Power and ground tracks are run separately for the analogue and digital sections, joining only near the battery; this keeps noise generated by the microcontroller away from the opamp.

Several teams forgot to enable the analogue input pin with the APCTL1 register. I think that the analogue input works if you forget this but the signal is distorted because the digital input/output circuit is still connected. Also, you can treat the full 10-bit output from the ADC as the single 16-bit register ADCR. This means that you need not manipulate the 8-bit registers ADCRL and ADCRH yourself – let the compiler do it.

As an example, I have shown my program in listing 1 on the next page. It was designed for the hardware in figure 4, which presents an offset AC signal as in figure 5 to the ADC. The general idea is to keep track of the maximum and minimum values in the input (peak and valley) over an interval of time and deduce the amplitude from these.

• The input voltage is converted every 0.1 ms ‘tick’ and the software updates the current maximum and minimum values.

• At every 10 ms ‘tock’ the peak-to-peak amplitude is calculated from the peak and valley, which are reset for the next period. If the amplitude is larger than the displayed value, an extra LED is illuminated on the bar graph and a decay counter is reset. This controls the time constant for the attack of the display: it goes up by one LED every 10 ms.

• After 20 tocks (200 ms) of the decay counter the top LED on the bar graph is extinguished. This controls the time constant for the decay of the display: it goes down by one LED every 200 ms.

Each LED corresponds to a factor of 2 in voltage or 6 dB. I did not attempt to calibrate the absolute values. The attack rate is faster than the decay rate, which is convenient for a sound level meter.
Timing is controlled by a paced loop using the counter of the TPM module. The overflow flag TOF is set when the counter rolls over and paces the loop. Interrupts are not used, nor are the capture/compare channels.

**Listing 1. Program soundmeter2.c for sound level meter.**

```c
/**********************************************************************
Simple example program for sound level meter, EDP2
John H Davies, 2011-03-13
Built using CodeWarrior 6.3
soundmeter2, basic functionality, poor algorithm for updating peak

Pin allocation:
PTA0 - analog input, ADP0
PTA1 - not used, internal pullup to avoid floating input
PTA2 - bar graph display bit 0, see below
PTA3 - bar graph display bit 1, see below
PTA4 - BDM communication
PTA5 - pushbutton input (IRQ), not used in this program, internal pullup
Port B - bar graph display bits 9:2, see below

10-bit bar graph display, assignment for easy PCB layout:
[msb] PTB7, PTB6, ..., PTB1, PTB0, PTA3, PTA2 [lsb]
LEDs are ACTIVE LOW to ease layout of PCB

Bus clock: internal, calibrated 4MHz using trim value
although accuracy is not vital for this program

Input is AC, biassed to half supply, not rectified;
ADC finds peaks and valleys to extract amplitude

Paced loop runs at 10kHz, defines ticks at 0.1ms
ADC performs a conversion at the start of each loop
Value is used to update current top and bottom; display not changed

Amplitude is calculated from top - bottom at 100Hz (100 ticks = 1 tock)
Peak value of input is displayed in thermometer code (0..01..1)
Peak display slides up one bit immediately if new value is higher;
otherwise peak value slides down one bit after 0.2s (20 tocks)
**************************************************************************/
#include "derivative.h" // Peripheral declarations

#define TIMER_COUNTS 400 // Bus clock cycles for 0.1ms tick
#define TICK_COUNTS 100 // Ticks per tock, 10ms
#define DECAY_TIME 20 // Tocks before peak decays, 200ms
#define BUTTON PTAD_PTAD5 // Pushbutton input (unused)
```
```c
#define SOUND_IN 0b00000000 // ADCSC1 to convert ADC channel 0

void DisplayBar (unsigned int Value); // Display value on LEDs

void main(void) {
    unsigned int top, bottom; // Extreme values from ADC
    unsigned int amplitude; // Current amplitude (top - bottom)
    unsigned int PeakAmp = 0; // Current maximum, thermometer code
    unsigned char ticks = 0; // Counter for paced loop
    unsigned char DecayCounter = 0; // Counter within ticks for decay

    // Configure basic functions and trim clock to stored calibration
    SOPT1 = SOPT1_BKGDPE_MASK; // Enable BDM pin, stop watchdog
    ICSTRM = NVICSTRM; // Trim internal oscillator
    ICSSC_FTRIM = NVFTRIM_FTRIM; // Fine trim bit

    // Configure ports (described above)
    // Outputs must be driven high because LEDs active low
    PTAPE = PTAPE_PTAPE1_MASK | PTAPE_PTAPE5_MASK; // Button input and unused pin
    PTAD = PTAD_PTAD2_MASK | PTAD_PTAD3_MASK; // LEDs off
    PTADD = PTADD_PTADD3_MASK | PTADD_PTADD2_MASK; // LED pins
    PTBD = 0xFF; // LEDs off (active low)
    PTBDD = 0xFF; // LEDs on all pins

    // Configure ADC and prime records of peak and valley
    ADCCFG = ADCCFG_ADIV1_MASK | ADCCFG_MODE1_MASK; // Clock = bus/4 = 1MHz, 10-bit mode, short sample time
    APCTL1 = APCTL1_ADPC0_MASK; // Analog input on pin ADP0=PTAO
    ADCSC1 = SOUND_IN; // Start conversion
    while (ADCSC1_COCO == 0) {} // Wait for conversion to complete
    top = bottom = ADCR; // Prime peak and valley

    // Configure TPM to roll over and produce periodic ticks, no interrupts
    TPMSC = TPMSC_CLKSA_MASK; // Bus clock, no prescale
    TPMMOD = TIMER_COUNTS - 1; // Modulus (period is (TPMMOD+1))

    for (;;) {
        // Infinite tick loop paced by TPM
        while (TPMSC_TOF == 0) {} // Loop pacer
        TPMSC_TOF = 0; // Wait for timer to overflow
        ADCSC1 = SOUND_IN; // Acknowledge by clearing flag
        while (ADCSC1_COCO == 0) {} // Start conversion
        if (ADCR > top) {
            top = ADCR;
        } else if (ADCR < bottom) { // ...or valley
            bottom = ADCR;
    ```
The software has several defects. The attack time is too slow, so it misses short bursts of sound. The top LED on the display flickers with a constant level of sound because the algorithm for updating the display is poor.

### 9.4 Reports

I said that at the start but clearly need to repeat it: *An engineering report is not a story.* It should be an account of what you achieved, but not the steps along the way. Please do not write ‘First we did this, then Morag did that,…’. The example report on moodle should make this clear.

Several schematic drawings were illegible where they had been integrated into the body of the report. You must be careful when pasting graphics into Word, particularly on a PC. I deliberately asked for separate printouts, not integrated into the text of the report, because this is a common problem.
The PCB was the major cost in almost all cases. The bar graph display is surprisingly expensive because the LEDs are closely matched for brightness.

9.5 Laboratory books

These varied from the dismal to the excellent (as usual). Some were full of insight, so I could see the different ideas that you had considered; others had so little content that it was hard to believe that the owner had made any meaningful contribution to the team. Most appeared to have been used as you went along, which is critical. I penalised books heavily if they appeared to have been written up after the event (usually obvious from the neatness).

Your book should contain enough information that somebody could repeat your work. If you show measurements on a circuit, you need to draw the circuit precisely or the results are meaningless.

9.6 Conclusion

I was disappointed with the performance of my final product, shown in figure 7. It used a bar graph with 10 LEDs and should display a dynamic range of 60 dB in theory. In practice the bottom 3 or 4 LEDs remained alight permanently so the range was only about 40 dB. Eventually I read the data sheet for the microphone more closely and discovered that it has a signal-to-noise ratio of only 40 dB. This limits the range over which the system can operate. A few teams noticed this too – well done. Sadly it is impossible to display a range anywhere near 100 dB as some teams initially hoped.

The designs that used rectifiers (analogue peak detectors) could not detect low levels of sound because of the voltage dropped across the diode. A superior rectifier circuit is needed with this approach but it is better to avoid a rectifier completely.

Almost all of you put a lot of effort into this project and learnt a lot from it. Congratulations!