Design and Implementation of a Laser Level Detector

By

Geoff Cook

School of Information Technology and Electrical Engineering,

The University of Queensland.

Submitted for the Degree of
Bachelor of Engineering (Honours)
in the Computer Systems Engineering Stream

October 2002

54 Jude Street
Bracken Ridge, Q 4017.
Tel. (07) 3261 3961

October 18, 2002

The Head
School of Information Technology and Electrical Engineering
The University of Queensland
St Lucia, Q 4072

Dear Professor Kaplan,

In accordance with the requirements of the degree of Bachelor of Engineering in the Computer Systems Engineering stream, I present the following thesis entitled "Design and Implementation of a Laser Level Detector". This work was performed under the supervision of Dr. Peter Sutton.

I declare that the work submitted in this thesis is my own, except as acknowledged in the text and endnotes, and has not been previously submitted for a degree at the University of Queensland or any other institution.

Yours sincerely,

Geoff Cook.

Abstract

The project began with the concept of building an electronic reader for a laser level (height) measurer. This idea was expanded to embrace a variety of different uses throughout the construction industry, where low-precision alignment may be required as part of a construction process.

The device is designed to substitute for the 'human' (subject to error) element in reading the position of a laser beam on a scale. Instead, the detector automatically determines this position and displays it to the user.

Such a detector would only require low-resolution reading: similar to that which a human eye could perceive. Other specifications were developed to make this product as 'useable' as possible, such as the required length, the sampling or reading rate, the display requirements, current limits and a component budget. These limitations were imposed to mimic real-world design constraints.

The hardware concept for the sensors was quite simple: an array of detectors, with a digital output fed into a row of shift registers. The shift registers could be 'latched' to lock in the detector data, and then 'clocked' to feed the data back to a microcontroller. The controller module hardware involved a display to be multiplexed, a microcontroller (in-circuit programmable), a serial connection and a power supply.

The embedded software was responsible for coordinating, reading and interpreting the hardware. It would calculate the position of the laser beam using the data fed back by the shift registers. The display would be updated with the position reading, which was also sent to an optional connected PC via a serial connection. The PC software simply read the data and displayed it on the monitor.

The solution was the 'Laser Level Detector' – a device that reads the position of a laser beam on a ruler and displays its value.

Acknowledgements

There are many people that must be acknowledged for there guidance, help and input in the development of this project. Without their help the achievements I made would not have been so great. It has been a pleasure to work with all, and to learn from all.

Firstly, thanks goes to Dr. Peter Sutton for his guidance as supervisor. Your input has been greatly appreciated.

Thanks also to Ian Turner, Richard Cocks and Len Payne for the interest they have taken in this project. I didn't expect your help; it was greatly appreciated when it was received.

Thanks to Luke Pomery for being in the labs in the many late nights.. and early mornings. You made this whole thing legal (lab rules).

Thanks to Kieth Bell for your guidance in surface mount PCB soldering, and for the teams in the electronics and mechanical workshops.

To my parents, Steve and Paivi: your support, encouragement and personal guidance have been of paramount importance to my life.

Mark Nahuysen, Daniel Walls, Ian Turner and Candi Shore and all my brothers and sisters at BCF for your understanding, guidance and prayerful support over my university years.

My Lord and God, whose grace and enabling came in the times of greatest need, whose miraculous provision still astounds me, and whom I shall follow for the rest of my life.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 – Laser Alignment

There are many different methods of leveling and alignment in the industrial world. Perhaps one of the most versatile tools for alignment is the laser beam. It is used in a variety of complex (and simple) leveling/alignment solutions [1] [5]. One of the main reasons for its use is the tendency for laser light to travel in a straight line and not diverge (although divergence eventually becomes a problem).

1.2 – The Problem

The title 'Design and Implementation of a Laser Level Detector' is perhaps ambiguous. It does not refer to a level detector of 'flatness' but of 'height'. In the preliminary definitions for this problem, it was described in reference to the need for an accurate reading on a surveyor's level stick. This is a situation where a laser is directed along a certain level of height; shone onto a vertical 'ruler' which rests on the ground. The height of the laser beam on the ruler indicates the displacement of the unknown ground height from the laser beam height.

Although such a scenario was described in the initial stages of problem definition, the challenge should not be narrowly defined to such a unique application. More importantly, the general purposes of the device must be described in order that it may be utilized in a wider array of leveling/aligning applications.

Firstly, initial market research will show a number of high complexity and high investment products for laser alignment, which require significant capital investment. Not all circumstances demand such high precision, nor do they support such significant cost. These are often situations were a laser

beam can be read adequately by eye of a scale or a ruler. However, when considering the relatively slow and error-prone nature of human input, such a solution would not readily lend itself to the efficiencies of automation in the construction process.

Therefore, the problem could be defined as '*the need to find a low-cost, relatively low precision reader for a laser-ruler interface, to replace the human eye as an instrument in alignment*'.

1.3 – The Solution

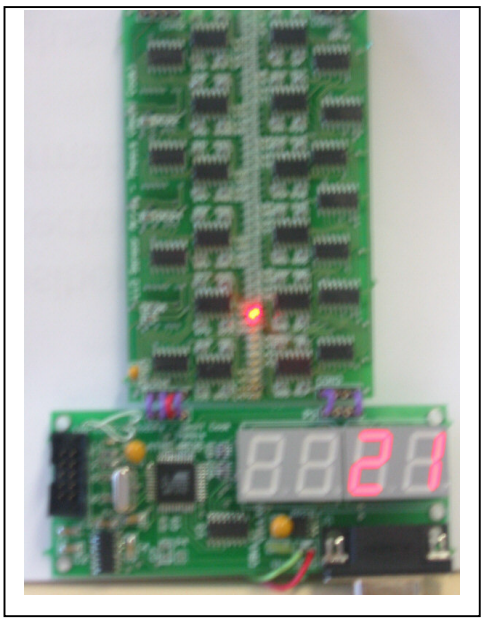


Figure 1: The solution

Figure 1 shows the final product developed. The position of the laser beam is shown on the display (21mm from the base of the sensor board). On the bottom right, a serial cable (9 pin D-type) connector can be seen. This will be used to send the information from the Laser Level Detector to an optional PC. When this occurs, the information will also be displayed on the PC monitor.

The information on the display is updated at a speed that would seem almost immediate to a human operator.

1.4 – Overview

This thesis will examine the design and implementation of a laser level detector. It will begin with a review of current technologies in optical sensing (*Chapter 2*). The nature of desired specifications will be developed in *Chapter 3*. How the solution was developed (and why) will be discussed in the following two chapters, relating to both the hardware and software implementations of the product. The product will then be evaluated in *Chapter 6*, and future

developments identified in *Chapter 7*. The final conclusions are found in *Chapter 8*.

Chapter 2: Review of Technologies

This chapter begins with a review of light detecting technologies (*Section 2.1*). The following section (*2.2*) describes a current industrial solution to this problem.

2.1 – Sensing Technologies

Three light sensing technologies are described below: the Position Sensing Detector (PSD), the Charge Coupled Device (CCD) and the simple Detector Diode. These technologies will be referred to in *Chapter 4*, when the possible solutions for this product are explored.

2.1.1 – Position Sensing Detectors (PSDs)

The PSD is an optical sensor that has an analogue output proportional to the position of a light beam's centroid on its surface [2]. These detectors are used in many applications with lasers. A PSD is quite a simple device, and is therefore reliable and hardy. They are available in one or two dimensions (a thin strip, or a wider surface). PSDs are manufactured in strips around 19mm long, and will return a result in a matter of nanoseconds. The output is highly accurate (to the order of nanometers). Detecting the centroid of light 'gravity' on its surface is the only useful function the PSD can perform. The output of the PSD is one continuous voltage waveform [4].

2.1.2 – Charge Coupled Devices (CCDs)

A CCD detector is essentially a grid of MOS diodes [3]. As the result on each of these diodes is read, the grid forms an image. Due to its complex nature, the CCD is more expensive, and more likely to fail than a PSD. It can, however, be used for a wider range of purposes [4]. As it takes time to form an image by scanning the diodes, the basic CCD is susceptible to error in a highly dynamic image capture. There are, however, CCDs built from CMOS diodes that can reduce these problems. A CCD can also be designed to determine the point of greatest light intensity on its surface.

2.1.3 – Laser Detector Diodes (LDDs)

Perhaps at the most elementary level of optical detection is the single LDD, which gives an output based on the optical flux entering its view. These diodes are extremely cheap, and readily available. The technology of these detectors is described below.

2.1.3.1 – Semiconductor Theory

A semiconductor is a material whose electrical properties are neither completely conductive, nor completely resistive [8]. Conductance of semiconductors will in fact increase with temperature. Such elements exist in group-IV of the periodic table; an example of which is silicon, and have an equal number of electrons and holes. By 'doping' a semiconductor substance with an element from an adjacent group in the periodic table, a substance can be formed that has a disproportionate number of electrons and holes. Doping a semiconductor with a group V element makes a substance with a larger number of electrons; this is known as an 'n-type' substance. Similarly, doping with a group III element makes a substance with a larger number of holes.

A semiconductor may be doped such that half is n-type, and the other is p-type. This is called a 'p-n junction'. Here electrons will diffuse from the n-type and neutralize holes in the p-type region. Thus, a potential barrier (known as a depletion layer) is created between the two regions. A voltage across such a semiconductor will either reduce the width of this depletion layer until it is overcome, at which point a large current will flow; or increase the width such that only a small 'reverse saturation current', made by minority carriers, can flow until a 'turnover' voltage is reached.

Considering conductance on the atomic level, it can be seen that there are two 'bands' separated by an energy gap. The valance band is at a low Fermi level (energy state), and the conductive band is at a higher Fermi level. Electro-luminance occurs when electrons drop from a higher band to a lower, giving off energy in the form of a photon (light). If the free electrons in the n-type

substrate are at a higher Fermi level to the available holes in the p-type substrate, a forward current through the diode (produced when the bias voltage is overcome by an external voltage, allowing the majority carriers to flow in a larger current) will produce light as the electrons fill holes in a lower energy level. This is the principle of a Light Emitting Diode (LED).

Photodiodes also use the property of electro-luminance, but in reverse. Incident photons will move electrons to higher Fermi levels.

A 'photoconductive' photodiode is especially designed so that the presence of light produces the energy to create electron-hole pairs. This will allow a current to flow if an external field is present.

A 'photovoltaic' photodiode generates a voltage, which converts to a current when connected as part of a circuit. The positive voltage will appear on the p-type substrate [10] [8].

2.1.3.2 – Using an LED as a photodiode

LEDs are used industrially in much larger numbers than photodiodes, allowing economies of scale to greatly reduce their cost. However, when considering the similar theory behind both photodiodes and LEDs, it should not come as a surprise that incident light shone on an LED will cause it to act like a photodiode, albeit without the benefits of design specificity.

However, if incident light is intense, such as the light provided by a laser, such benefits may not be required. Thus, LEDs could be used in 'reverse' in this application, greatly reducing the cost of sensors.

Theoretically, intense light on an LED p-n junction should cause electrons from the lower Fermi level p-type substrate to jump to the higher-level n-type substrate. Thus, the depletion layer would be widened. However, these extra

charges in each region are not part of the natural bias due to the p-n junction, and can therefore flow as current.

Another benefit to using LEDs is that there are a wider variety of different types available. An LED which produces a very specific wavelength range will obviously only act as a photodiode for incident light in that range.

2.2 – Existing Solutions

There are technological solutions available for the simple laser level detector.

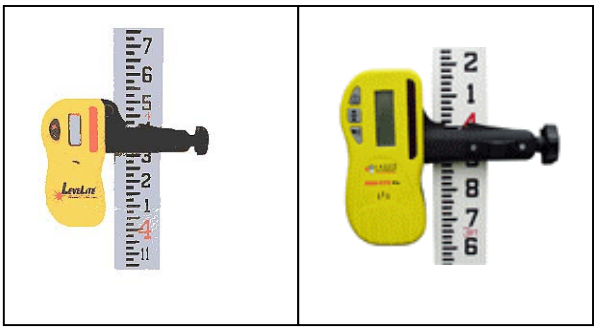


Figure 2: Laser Level Detectors

Two examples of these are shown in *Figure 2* (left) [6] [7]. These technologies are suitable for the situation described in the initial stages of problem definition (*Chapter 1*), but do not serve the broader set of purposes also desired. However, the concept is similar. A laser beam (in this case spinning in the horizontal plane) shines onto the ruler as well as the device. The device reads the position of the laser beam in the vertical direction. In these simple cases, however, the position of the beam is not displayed, but rather the relative position (above or below) is indicated until the desired level is reached. The devices shown in *Figure 2* are the 'Professional Detector Kit for Rotating Lasers' from LeveLite [7] and the Red-Eye Pro Rotating Laser Detector [6].

Other solutions exist for a variety of functions: leveling, alignment of rotating shafts, distance detection (one, two and three dimensional) and horizontal/vertical alignment. These tools are highly specialized to specific purposes.

Chapter 3: Defining the Problem

This section develops a specification of the problem, with a description both of the perfect dream product, and a more realistic product to be confronted in this thesis.

3.1 – Developing a Specification

Revising the problem described in Chapter 1, the device must replace the human eye as a tool for reading the position of a laser beam from a scale. This infers that there is already a situation where a laser beam is pointing to a scale (or ruler), or the product is designed for a similar situation that will exist in future. Therefore, there is a laser, a laser beam, and a ruler. Their interactions are shown in *Figure 3* (right). As the laser moves up and down, the laser beam moves with it. This movement is observable as a change in the position of a laser beam 'dot' on the scale or ruler. If laser properties are used in the pre-existing method, they will also be desired in the new solution. The solution to the problem will therefore involve replacing the scale or ruler in figure 3 with a device that automatically senses the position of a laser beam on its surface. This is the concept of the Laser Level Detector (LLD).

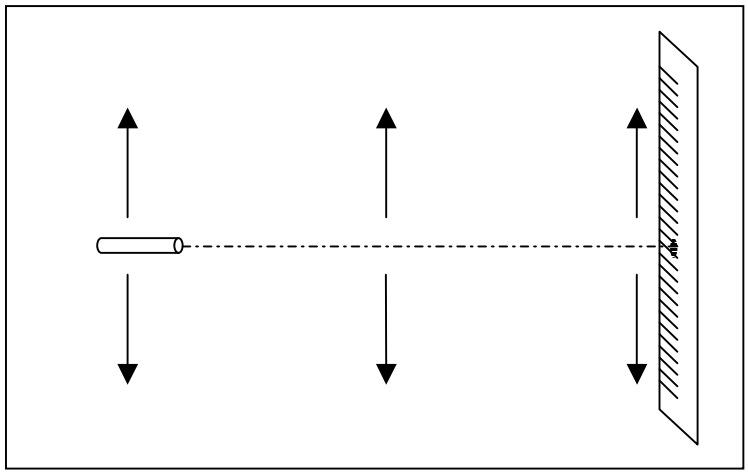


Figure 3: Laser Position Read From Scale

If the ruler can read the position of the laser beam, it will also need to convey this information. The benefit of the device may simply be to remove human error in reading from the scale; in this case, the value of the 'reading' must be displayed back to the human in a numerical (perhaps digital) form. This situation would arise where a laser beam diverges significantly, forcing the

human reader to interpolate the position of the beam's center. In such an instance, errors percentages could increase. Alternatively, if the function of the device is to replace the human element completely, it would need some form of communications protocol to 'talk' to the alternative user (which would be some sort of computer device, perhaps embedded). To accommodate both of these scenarios of use, the device will need both a display and a communications channel to a computer.

2.2 – Defining the Dream Product

The perfect product would replace the ruler unobtrusively. It would be easily constructed to be the same length as the ruler it replaces; it is assumed that the ruler to be replaced is 20cm in length, however an indeterminate length of up to 5m may be desirable. It will need to detect the position of a laser beam anywhere on its surface, and determine the relative position of the centroid of the beam with respect to the to the distance along the ruler ('y' in figure 4) and not the distance across ('x' in *Figure 4*).

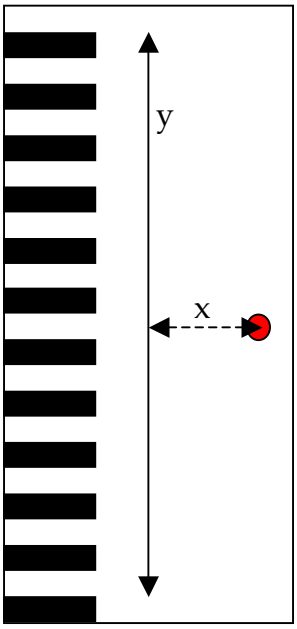


Figure 4: Ruler Diagram

The position of the vertical center of the beam would quickly (5ms) be calculated with high resolution (+/- 0.25mm) and presented on a display such that the information can be read from around 10m away. The power consumption of the device would be low, such that it could run on batteries for many hours (for example, two AA batteries lasting 10 operating hours). The device would be portable and easy to set up and use.

Information would be updated quickly and sent to a connected computer, perhaps wirelessly. This will allow possible automation of construction (where this alignment technique is used), with speeds greater than that which

a human could manage. All this would be done with low budget expenditure (under $100AU to manufacture).

2.3 – The Realistic Product

Unfortunately, the constraint of budget will result in a product, which does not have all of these 'dream' specifications, especially at this stage in the product life cycle. It is possible that continual improvement as technology advances would allow these specifications to be met over time. For now, however, a more realistic set of specifications must be developed whilst still allowing all of the functionality defined in *Section 2.1.*

Firstly, as this product is designed to replace the human eye in reading a laser displacement value from a ruler, the required resolution needs discussion. The human eye can quickly and accurately read measurements down to the order of millimeters, without any special equipment other than a ruler scale. Therefore, the resolution required for the reader would be quite the same: in the order of millimeters. The product should be designed such that its maximum error is at most one millimeter (that is, if it does read incorrectly, it is one millimeter either side of the true measurement).

The distance of the x variable in *Figure 4* (previous page) also requires consideration. How wide a region on the ruler should be readable? In the initial product, such a wide region could come at quite a cost. However, if a thin region were implemented, it would be possible to increase this width using optical techniques at some point in the future (see *Chapter 7: Future Developments*).

To keep the product affordable, a components/price budget for the prototype was set at $200AU. This price was determined in discussion with an industry representative, and was set to allow the product to be marketable.

Within this price range, a sensor module with a length of around 10cm should be obtainable, but a length of 20cm is desirable. A higher goal than these would see the production of a device with a design that could be easily expanded or shrunk to allow any length to be manufactured (up to around 5m would be useful).

The reading should be output onto a display that can be read from between 5 and 10m away, and the output should be sent to a computer (via the serial port).

The project should be designed to operate in diverse ambient lighting. For this reason, the device should have some form of filtering which allows only light of a certain wavelength range through, as narrowly around laser wavelength as possible. The wavelength of the laser used in development was 635nm.

The power consumption of the device should be sufficiently low for portable (battery) operation. Preferably, the device should be designed to operate for two or three hours on a single set of batteries.

Physically, the 'ruler' part of the device should be thin (in the order of 1cm) to allow it to be incorporated into a larger variety of machinery. A wider device may not be adequate for use in some low-space applications.

Finally, the device should update the information on the display regarding the sensor module at an acceptable rate to allow automation at sufficient mechanical speeds. If the alignment involves the fast movement of a particular machine, a faster read rate will allow the machine to move into position more accurately, more quickly. Therefore, the rate should be faster than 5Hz.

Chapter 4: Hardware Implementation

4.1 – Hardware Overview

This section describes how the hardware of the Laser Level Detector (LLD) was implemented. On a simple level, there are two major tasks involved in the hardware. Firstly, there is the 'ruler' hardware. *Chapter 3* described the need to engineer a ruler that automatically detects the laser beam on its surface. This will involve some form of sensory detection. Secondly, a control module will be required to interpret the data (in whatever format it may be) from the sensor module. It will be responsible for displaying the position of the laser beam and transmitting the relevant information to the (optional) connected computer (specifications defined in *Chapter 3*). *Figure 5* shows a block diagram of the functionality. At this stage, the sensor module is undefined, as its structure will depend on the detection technology to be used. The development of these two modules is described below.

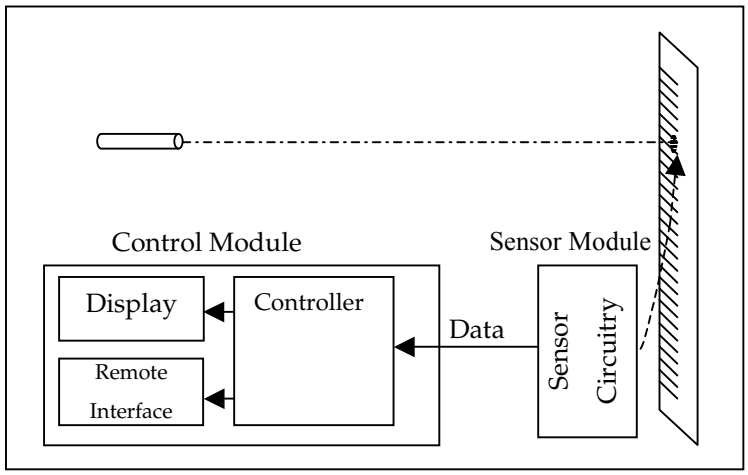


Figure 5: LLD Control Module Block Diagram

4.2 – Sensor Module

The sensor module is responsible for detecting the presence and position of a laser beam on its surface. This section begins with a discussion of the three technologies described in *Chapter 2*. One of these technologies is selected, and the supporting hardware is designed around it.

4.2.1 – Detecting the Laser Beam

Research into three alternative technologies for detecting the presence of a laser beam was conducted in *Chapter 2*. These alternatives were the use of Position Sensing Detectors (PSDs), Charge Coupled Devices (CCDs) and Laser Detector Diodes (LDDs). It is useful to keep in mind a number of the specifications outlined in *Chapter 2*. The beam will ideally be detected anywhere on the surface of the ruler. The ruler size dimension must be upheld. The design is assumed to be for a 20cm ruler, however an indeterminate length is desirable. The design must also detect the center of the beam with sufficient accuracy (+/- 1mm).

A PSD could be used in conjunction with optical prisms to measure the displacement of the laser beam, however this would not be the most desirable solution. Firstly, to have a prism 'custom made' would be quite expensive. Secondly, the prism would need to be calibrated in order to reduce any unnecessary error. The design would not be suitable for rough industrial treatment (able to be dropped), and would not be as small and neat as the proposed solution (using a prism would require a certain bulkiness which does not fulfil the design specifications for size). PSDs are quite expensive, and do not allow the device to be built at an indeterminate length: there is a finite limit to how far a prism can converge an incident light beam to a sensor. Finally, such an optical design would be outside the scope of this course.

A CCD is perhaps a better option than the PSD. It could be used in conjunction with image recognition software to determine the position of a light dot on the surface of a 'ruler'. However, there are also a number of drawbacks to this solution. Firstly, a CCD camera would need to be positioned somewhere near the ruler. This would limit any applications requiring portability. CCD cameras are also quite expensive: a CCD solution would not meet the budget requirements of the solution. Thirdly, the image resolution of a cheap CCD may not be accurate enough to determine the position of the centre of the laser beam, and like the PSD, a solution involving

a CCD could not be easily built to be independent of length. However, if the CCD was mounted together with the laser, its view input could move with the laser beam to any location on the ruler. If the CCD could learn to 'read' the inscriptions on the ruler, the position of the dot may be determined. The requirement of a very specific CCD camera (with a certain zoom level that allows the picture to be read with sufficient accuracy) is a drawback in terms of cost. This is a feasible solution apart from the inability to meet the budget requirement.

The remaining option is to use a row of detector diodes to detect the light. The precision requirement (+/- 1mm) could be achieved if the detectors were placed at 2mm intervals. Alternatively, if an analogue reading was made on diodes spaced at further intervals (assuming a 'wide' beam), the centre of the beam could be interpolated. The width of the beam at closer distances is narrow, however, disqualifies this method of determining the position of the beam accurately as it may fit between two diodes without being detected. A third option would involve placing a smaller array of diodes (or even a PSD) on a motorised arm that scans back and forth across the surface. This does not present a particularly versatile solution, as the length of the ruler surface is limited by physical constraints. Furthermore, this would not be a particularly stable or hardy solution, suitable for portability.

With the preceding arguments in mind, it would appear that a fixed array of closely spaced detector diodes would be the appropriate choice. It would be difficult, however, to produce a device of 'indeterminate length', unless the system was designed to allow individual sensor modules to be joined together to create a longer length. This would fulfil the portability requirement if a small number of modules were used, and also allow the product to be used in fixed situations where a longer length is desirable. This concept of design will be termed 'modularity', and is shown diagrammatically in *Figure 6* (following page).

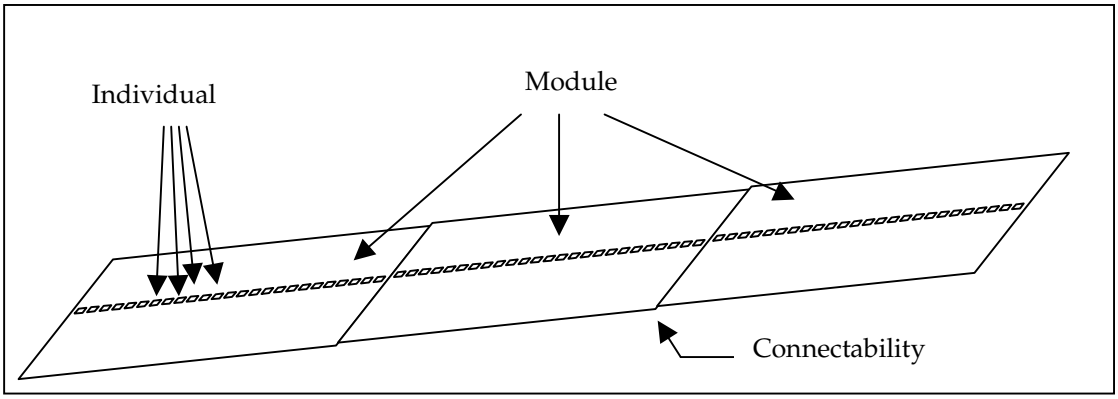


Figure 6: The Proposed Detecting Solution

4.2.1.3 – Obtaining a Digital Output

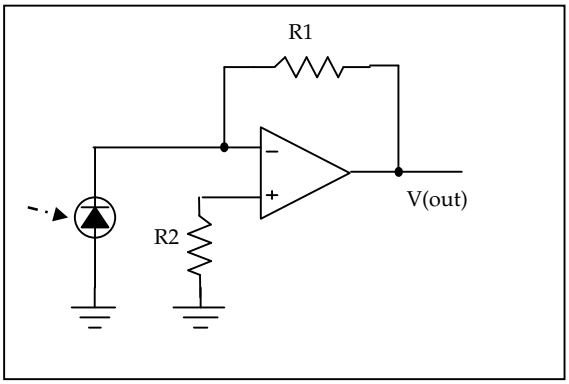


Figure 7: Current to Voltage Converter

Due to the small nature of the output of such photodiodes, the signal must be amplified. The incident light creates a reverse current through the circuit that would flow if a circuit were present; the photodiode acts as a current source in this situation. The signal could be amplified using a transistor or an op-amp. A transistor is both small and cheap; the voltage produced by the photodiode, however, is insufficient to overcome the 0.7V needed to turn the transistor on. Operational amplifiers are available cheaply and can be purchased in quad arrays. An ideal current to voltage converter circuit was obtained from the *Art of Electronics* [9]. This is shown in *Figure 7*.

If current of 'I' is induced on the photodiode (reverse current is considered positive here), then the output voltage (V(out)) will produce a level that brings the negative input of the operational amplifier to the same potential as the positive input, that is 0 (earth). Therefore, by Ohm's law:

$$\frac{V(out) - V(-ve)}{R1} = I$$

Therefore:

$$V(out) = I \cdot R1 + V(-ve) \qquad (V(\text{-}ve) = 0)$$

$$V(out) = I \cdot R1$$

Therefore, the output voltage is amplified by the value of the resistor R1. Due to the small current levels induced on the photodiode by a distant (diverged) laser beam, a resistance value for R1 was determined experimentally to be $10M\Omega$.

The operational amplifier selected was the single rail quad LM324. A single rail is required as the output is to be between 0V and 5V. The quad op amp is used to minimize the number of chips needed on the product (Four detectors will be interfaced through one chip). This output swings between 0 and 5V. However, a clean digital output is desirable, so this waveform was passed through the 74x14 inverting Schmitt trigger.

$10\,M\Omega$

Figure 8 (right) shows the circuitry required to produce a TTL/CMOS compatible signal dependant on the presence of a laser beam on the detector surface.

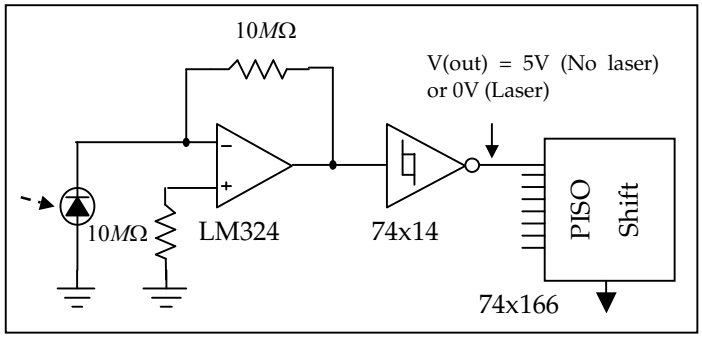


Figure 8: Detector interface circuitry

A Schmitt trigger is available in the standard 74x series: the 74x14 hex inverting Schmitt trigger.

4.2.2 – Interfacing to the Microcontroller

The circuitry described in *Section 4.2.1* gives a digital output showing whether a laser beam is present on each LED. Potentially, there are hundreds of

possible detector output values that must be passed to a microcontroller. Therefore, there must be some form of interfacing circuitry to pass the data from the detectors to the microcontroller. There were a number of possible solutions, including the use of shift registers, multiplexers or a large IO gate array. As there are budget requirements for this thesis, the large IO gate array was not considerable as an option. The Schmitt triggers and multiplexers both have reasonably simple implementations; however the most efficient design involves the use of shift registers. This will also allow the product to be modular, as outputs of the shift registers from one module can be passed easily into the inputs of the shift registers for the following module, enabling any control circuitry to interface a large number of modules.

Therefore, the interface to the microcontroller would involve an input from the shift registers, a latch output (to catch the data on all detectors simultaneously) and a clock output (to shift the data back to the microcontroller).

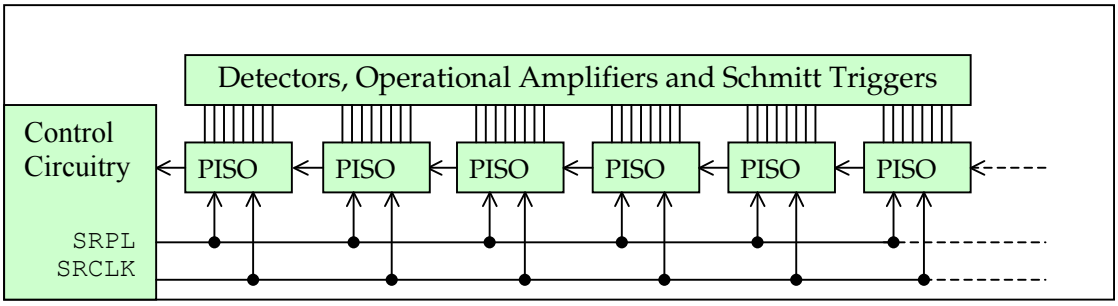


Figure 9: Complete Sensor Module

Figure 9 shows a block diagram of the sensor module. The detector, operational amplifier and Schmitt trigger circuitry for each detector cell was described in *Figure 8*. The quad input operational amplifier, hex Schmitt trigger and eight-input shift register provide a cell for 24 (lowest common multiplier of 4, 6 and 8) detectors. By placing one of these cells on each side of the diode array, a module of 48 detectors is built. To allow a maximum error of +/- one millimeter, these diodes can be spaced 2mm apart, creating a module that is 9.6cm long.

4.2.3 – Power and Decoupling

Power is supplied to the sensor module by the control module. The power required for each sensor module is shown in *Table 1*.

Single Sensor Module Power Consumption				
Device	**Function**	**Quantity**	**Current/Device (mA)**	**Total (mA)**
LM324	Op Amp	12	1.5	18
74LS14	Schmitt Trigger	8	3	24
74HC166	Shift Register	6	3	18
TOTAL				**60**

Table 1: Sensor Module Power Consumption

Due to the high-frequency nature of the digital circuitry, decoupling capacitors have been used on every chip. These capacitors are necessary because the thin PCB tracks cannot provide enough current to allow the circuitry to switch voltages at the required time. A local current source, such as a 150nF capacitor can provide the current required to perform a switch, allowing the rails to remain much closer to their 5V difference [9].

Bulk capacitors (10uF) are also placed on each module to smooth out that module's voltage rails [9].

4.3 – Controller Module

The controller module is the 'brain' of the LLD. It is responsible for coordinating the activities of the sensor module, interpreting the data received, displaying it and sending it to a PC via a serial connection. This functionality is described below.

4.3.1 – Microcontroller Choice

Table 2 (following page) shows a short list of available microcontrollers, and their respective features. It can be seen immediately that the PIC MCUs have a lower power consumption than the Atmel MCUs; however, there are a number of extra Atmel tools available at the University of Queensland that promote the Atmel as the microprocessor of choice. These involve in-

MCU	Pins	Clock	Flash	Ram	UART	Timers	Power @4MHz	Extra Features	Price
PIC16F876	28	20MHz	8K	368	yes	3+WDT	0.6mA		$22.08
PIC16F628	18	20MHz	2K	224	yes	3+WDT	2mA		$14.65
Atmel 90S8535	44	8MHz	8K	512	yes	3+WDT	6.4mA	AVR Studio 3.53 avrgcc compiler In-circuit programming dongle (STK200) Ponyprog programmer	$15.89
Atmel 90S8515	44	8MHz	8K	512	yes	2+WDT	3mA	AVR Studio 3.53 avrgcc compiler In-circuit programming dongle (STK200) Ponyprog programmer	$21.17

Table 2: Microcontroller Comparison

circuit programming circuit schematics, and a LPT1 connection dongle (STK200 compatible); software for development and compiling, and finally a device programmer which is compatible with these circuits.

The avrgcc compiler is also written in such a way that programs are easily ported between differing chips. The dongle connections and circuits (originally designed for the 8515) are portable to the 8535. Due to the lower cost of the 8535, it is the microcontroller of choice.

4.3.2 – Displaying the Data

According to the specifications in *Chapter 3*, the data must be displayed such that it can be read from several metres away. For this reason, 7-segment displays are preferred as a means of data representation. A liquid crystal display may be difficult to read, particularly in darker areas, and will often not have the required font size. The cost of a LCD is also significantly greater than that of a standard 7-segment display. The other option for displaying the data is to use the monitor of a PC. This would display the information more clearly, however, if a PC is not available or connected, the data must still be read. Therefore, the 7-segment displays were used.

The next design decision regarded the number of these displays to use. Using 3 digits will allow an LLD of up to 999mm (1m) long at a resolution of 1mm. This value may be sufficient for most applications, however it does perhaps limit the length of the ruler (there may be cases when a ruler is required to be 120cm long). Therefore, a fourth digit is desirable, even if it is not to be used.

The IO port A on the Atmel 90S8535 can source up to 20mA current; it is suitable for driving LED displays. Each individual LED (one segment of display) will require an average of 10mA to display brightly. To reduce the number of IO ports required on the device, these displays should be multiplexed. This involves turning each of the displays on individually while the others are off; allowing each of the four to share the same data inputs, but be multiplexed over time. These displays come in two types: common anode and common cathode.

A common cathode configuration is shown in *Figure 10* (below). Here, each input (A – G, DP) is connected high to turn 'on', while common is connected to ground. If 'common' is connected to 'high' rather than to ground, the display is turned 'off'. Therefore, each of the four displays can have connected inputs (A-G, DP) and be time multiplexed by setting the 'common' pin of each segment low, one at a time.

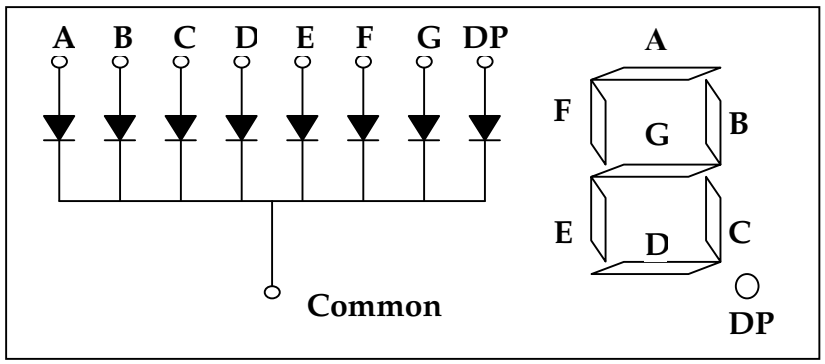


Figure 10: Common Cathode Configuration

An important consideration then arises: if the common supports the negative voltage connection for eight diodes, it must be able to sink eight times the current running through each diode. If the diodes average 10mA current each, and all eight inputs are high, this will result in a required sink current of

80mA. This is more than an individual IO pin of the microcontroller can sink, so it must come through another source.

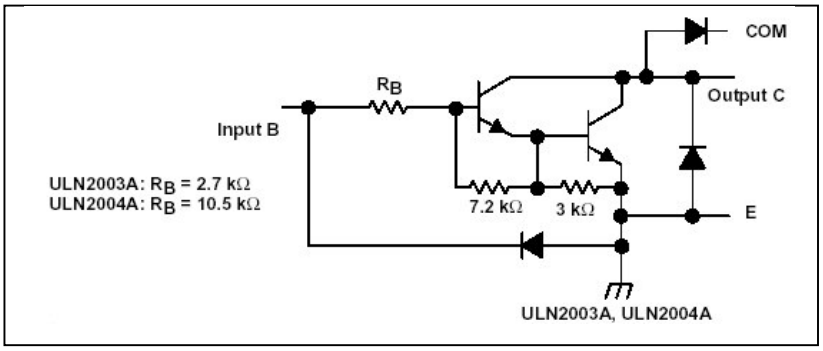


Figure 11: Darlington Driver Equivalent Circuit

The DS2003 is an array of high-current Darlington Drivers. These drivers are capable of delivering currents of up to 350mA with an input voltage of 1.35V. An equivalent circuit for a single driver is shown in figure 10 below. This will allow a small current (1.5mA) from the output of the microcontroller to drive the common cathode of a 7- segment display to a low value (0.7 V). The equivalent circuit is shown in *Figure 11* [15].

Com	Input	LED
low	high	on
low	low	off
high	X	off

Table 3: Segment truth table

An appropriate current of up to 20mA is supplied to each individual segment from the microcontroller, and the common cathode is driven low by an output of the microcontroller turning on a Darlington driver. Therefore, the following truth table shown in *Table 3* is the result.

4.3.3 – Serial Communications

The standard RS232 communications protocol will be used to transmit data between the LLD and the PC. The signal conversions required between TTL/CMOS and RS232 circuitry are easily performed using the MAX232 chip [12].

Peacock [16] describes the functions of the pins in a D9-Pin serial connection. Of interest are the receive data; transmit data and a ground signals. The Data Set Ready, Data Terminal Ready and Carrier Detect lines are used in a

modem serial link. The LLD link will be a 'null-modem' link, so these pins will be tied together. The Request To Send (which informs the PC's UART that device is ready to exchange data) and the clear to send (indicating that the device is ready to receive data) are usually tied together in a null modem link. However, they may be useful in this application. The RTS signal can be used by the LLD to identify the presence of a RS232 serial connection with a PC, and the CTS signal can be used by the PC to detect the presence of an LLD on the serial port.

The RTS and CTS signals have been passed through the MAX232 chip to allow such detection to take place.

4.3.4 – Power and Decoupling

A 5V voltage rail was used, allowing the prototype to have TTL/CMOS compatibility. The LD117 low-power fixed positive voltage regulator was chosen, giving a maximum current of 800mA.

Control Module Power Consumption				
Device	**Function**	**Quantity**	**Current/Device (mA)**	**Total (mA)**
AT90S8535	Atmel MCU	1	6.4	6.4
MAX232	Line Driver	1	4	4
HDSP-523	7-Seg Display	1	40	40
2003	Darlington Drivers	1	1	1
TOTAL				**51.4**

Table 4: Control Module Power Consumption

Table 4 shows the current usage of the control module. The controller module will use around 50mA with a standard loading on the display. The 800mA supply will therefore allow up to 12 sensor modules (at 60mA each, see *Table 1*) to be connected. This will be adequate for the product prototype.

The controller chips also use decoupling capacitors and a bulk capacitor. For further information regarding the decoupling of the MAX232 and the AT90S8535, consult their respective datasheets [12] [13].

4.3.5 – Programming the Microcontroller

The control module was also equipped with the ten-way box header and circuitry required by the dongle for in-circuit programming. Both the dongle and the header connection schematics were supplied by the University of Queensland [17]. The result of using these tools will be prototype that is in-circuit programmable. This will allow development of the software to be more time-efficient.

Final Schematics of the sensor and controller hardware are attached in *Appendix A*.

Chapter 5: Software Implementation

5.1 – Software Overview

The software can be divided into two categories: the firmware for the microcontroller, and the display software required to be present any connected computer. The former is discussed first. The role of this software is to control and integrate the functionality of the hardware to produce a working product.

5.2 – Microcontroller Software

The software was written for the Atmel AT90S8535 chip, using AVR Studio v3.53 and the avr-gcc compiler by GNU [18]. The device was programmed using the PonyProg2000 Serial Device Programmer v2.04n Beta. The main routine in the program was developed as a Finite State Machine. This was an obvious choice due to the number of clear states the LLD needs to move through in operation, such as latching the data, shifting it in, calculating the position and updating the display vector. There are, however, timer/interrupt functions, such as the multiplexing of the display, which occur outside of the operation of the state machine. This section contains a detailed description of the elements contributing to software development for the LLD.

The software itself was segmented into four parts. Firstly, serial.c and serial.h provided functions for initializing and sending data to the serial port. Display.c and display.h are responsible for initializing, refreshing and writing characters to the display. Data.c and data.h provide data storage, retrieval and processing functions for the detector diode information. They contain functions for initializing the data shift interrupts; as well as latching, shifting, storing and processing the data. These modules are all set up in a main initializing function. The modularity of this code makes switching between drivers for debugging quite simple.

5.2.1 – Finite State Machine Implementation

The first state identified in the diagram (*Figure 11*, below) is initialization. In this state the general microcontroller features are set up, such as IO port direction, interrupt masks, timers, serial communications (UART) and pulse width modulation (PWM).

In the second state, the number of sensor modules connected to the LLD is determined. This number will be displayed briefly.

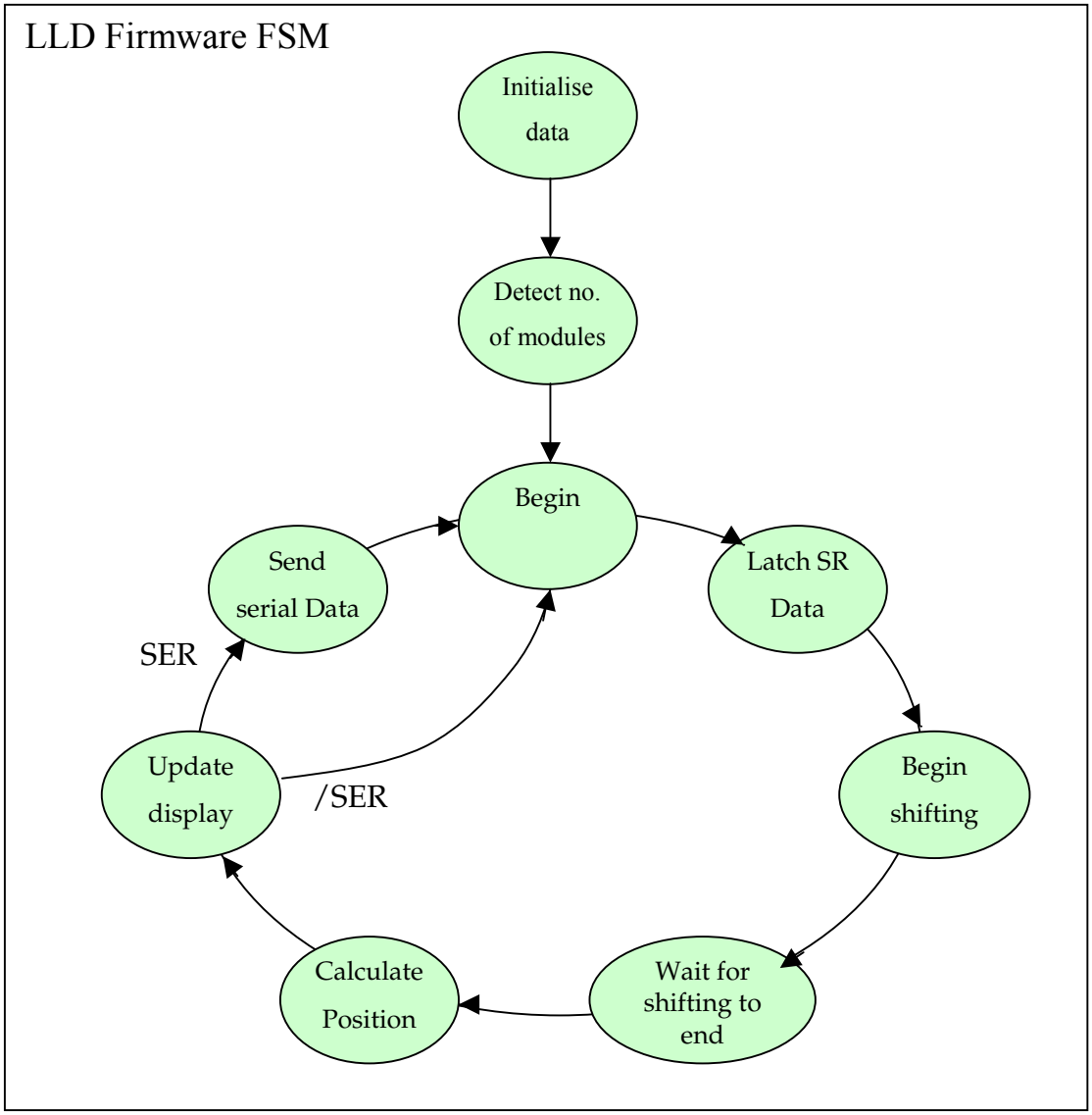


Figure 12: LLD firmware implemented as a Finite State Machine

Once the device has been set up, and the number of modules detected, it will enter the main operational loop (the conceptual state 'begin'). The device will

latch the detector data into the shift registers and shift it back to the microcontroller (using interrupts). Once this data has been shifted back, the position of the laser beam can be calculated. The display vector (which the display multiplexing interrupt will access) will be updated with this new information. If the device is connected serially to a PC (SER = 1), the serial data will be sent. The device will then latch the new detector data into the shift registers, and commence the process again.

5.2.2 – Detecting the Modules

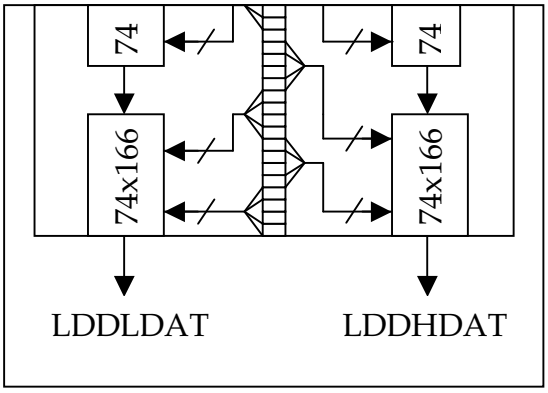


Figure 13: Sensor Module Outputs

As mentioned in the hardware development, the number of modules can be detected by shifting in data. The shift register inputs will be high if they are connected to a detector circuit that is not receiving laser light. *Figure 14* (following page) shows the timing operation of the sensor module. At the beginning, the data is clocked. LDDHDAT and LDDLDAT (the output of the shift registers, see *Figure 13*) are both high for each cycle if that particular corresponding circuit did not detect a beam. When the data has completed shifting in, LDDHDAT and LDDLDAT both drop low. If a distance greater than the length of a module (48 bits or 48 rising edges on SRCLK) is detected, than the last module has been reached. Therefore, if z consecutive zeros are counted before calculation takes place (z > 48), and n is the total number of bits counted before z consecutive zeros are detected, the number of modules can be calculated using the formula:

$$\text{Number of modules} = (\text{int})\frac{n-z}{z}$$

5.2.3 – Shifting in the Data

Shifting in the detector data is a relatively simple task. Firstly, the data must be latched. The shift registers selected have an active-low synchronous latch, so the parallel load pin (SRPL) is held low across a rising clock edge (SRCLK). On every subsequent rising edge of the clock, two bits of data from the shift registers are available (LDDHWRD and LDDLWRD). Four clock pulses will shift in one byte of data, the lower 4-bit word arriving via LDDLWRD and the higher by LDDHWRD (see *Figure 13*). The timing information for this operation can be seen in *Figure 14* below.

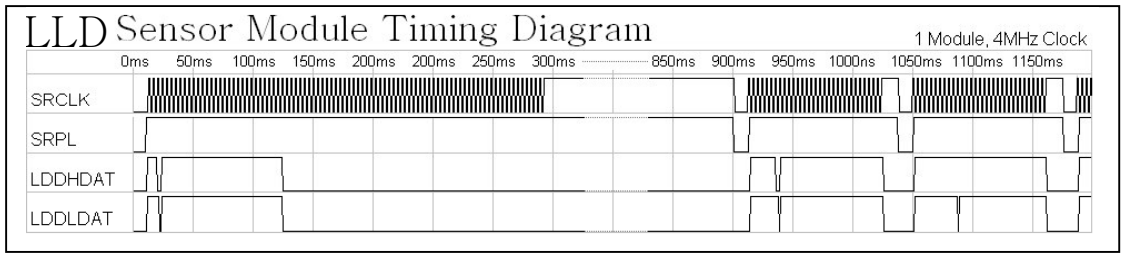


Figure 14: LLD Sensor Module Timing Diagram

The SRCLK signal was generated using the TIMER2 overflow interrupt. The clocking signal is therefore started/stopped by enabling/disabling this interrupt. When an overflow occurs, the interrupt is called and SRCLK signal is toggled. If the clock signal is a falling edge, the data is read from LDDLDAT and LDDHDAT. This allows time for the hardware to stabilize before it is read by the MCU. This transition can take up to 40ns, with a load capacitance of 50pF. Even with an 8MHz clock, this time is insignificant. The data is therefore clocked in on the falling edge as 'best-practice'; if the sensor module is clocked faster, and the time taken to service the data and service the interrupt becomes significant, the rising edge may be used to double the time available. The storage algorithm used for bit number n is as follows:

$$\text{Byte Number} = (\text{int})(n/8) \qquad \text{Bit Number (of byte)} = (n \cdot \text{mod}(8))$$

This fills each byte LSB first.

Due to the nature of the hardware design, if bit (n) is being shifted into LDDLWRD, bit (n + 4) is being shifted into LDDHWRD.

The shift begins (by enabling the interrupt) in the 'Begin Shifting Data' state of figure 11, and finishes at the end of the 'Wait' state.

5.2.4 – Calculating the Position

As the data is shifted in, it is stored bit by bit into an array. Although the hardware returns a 'one' when the laser beam is not on a diode, the software converts this into a 'zero'. Therefore, the stream of data may appear like so:

10000000-00000110-00000000-00000000-00000000-00000000

The storage algorithm pushes bits in least-significant bit first. This stream of data represents the sequence of bits.

00000001.0110000.00000000.00000000.00000000.00000000

Each bit value represents a certain distance from the origin (left). The diodes are placed every 2mm, and the first diode is centred 1mm from the edge of the sensor module PCB; the distance value given by diode n is:

$$\text{Distance(diode n)} = (2 \cdot n + 1)mm$$

The value of the center of the laser beam is then calculated by taking the average of these distances, over the number of diodes switched 'on'. If v is the logical state of the diode ('1' = on; '0' = off) then we have:

$$\text{Distance(Laser beam)} = \frac{\sum (2 \cdot n + 1) \cdot v}{\sum v}$$

5.2.5 – Multiplexing the Display

According to *Section 4.3.2* there are four seven-segment displays on the device. These are to be multiplexed to minimize the number of IO ports required on the microcontroller. The seven segments, from A to G, and the decimal point H are connected to port A (pins 0 to 7 respectively). Each individual segment is turned on by applying a high voltage to port C (C0..C3 turns on segments 1..4 respectively).

The timer0 overflow interrupt is used to multiplex the display. This will allow normal program operation to proceed while the display is refreshed. This timer is reloaded with a new value every interrupt to force the display frequency to around 60Hz. The duty cycle is obviously 0.25, as there are 4 displays. When each interrupt occurs, the display is turned off, the data on port A is changed, and the next display is turned on. The multiplexing of the word "HELO" (used in device initiation) is shown in *Figure 15*.

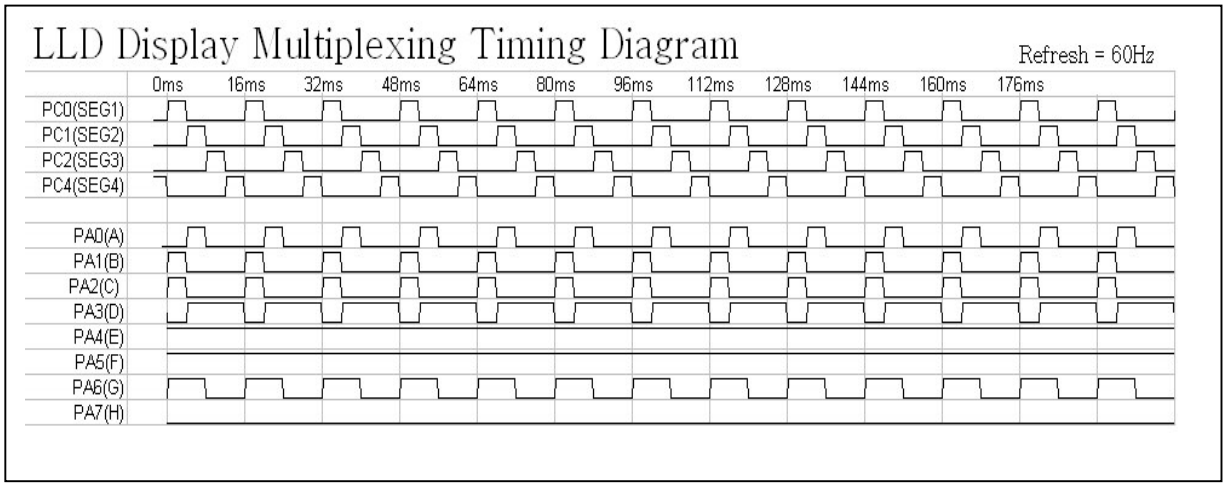


Figure 15: LLD Display Multiplexing Timing Diagram

The service time of this interrupt should not affect any of the other 'processes' running on the microcontroller (for example, the clocking of the shift registers) because the functionality does not require strict timing, apart from perhaps the UART, which is controlled in hardware.

5.2.6 – Serial Communications

Serial communication was implemented to allow the position of the laser beam to be displayed on a connected PC. The data received from the detector diodes was passed also to the serial port.

The software made use of the hardware UART on the AT90S8535.

5.2.7 – Run Length Encoding (RLE)

When the sensor module was being clocked at high frequencies it was found that the speed was limited by the speed of serial communications to the computer. The entire array of LED data was sent to the computer to allow the computer-based software to have full autonomy; the actual data read in from the detectors was transferred, not just the position calculated. This was to done to allow the computer-based software not only to display the data, but also to show the solution graphically, and perhaps even to guess how far the laser beam is away by any diversion discovered. It was decided that data compression should be used to remove the delay caused by serial communications.

Run length encoding is a compression algorithm, which substitutes a 'value' and a 'length' in place of a continuous stream (run) of similar characters [11] [14]. For example, if a particular passage had one character repeated ten times, RLE would replace this with a control character followed by a length value, and then the actual value. Looking at a simple level, this would reduce the size of the stream from ten characters to three (a control character, a length, and an actual value). The control character is necessary to identify to the algorithm that the following two characters are a length and a value, and therefore must be interpreted as such.

RLE would seem ideal for this situation, for we would expect that a laser beam will only turn a few adjacent detectors 'on', with the rest of them

remaining 'off'. Therefore, we would have a long stream of 'off', followed by a short stream of 'on', and finally another long stream of 'off' (if the beam was located in the center of the LLD). The region of 'on' we shall call a *dynamic* region, as we cannot guarantee that all the detectors will return an 'on'; the values may be changing between 'on' and 'off'. The two peripheral regions, where the laser beam is not shining, could be described as *static,* for we don't expect the value to change from 'off'.

Other forms of more complex decoding are available, such as 'deflation' and 'LZ77' [14], which identify runs of character patterns (as opposed to runs of single characters). However, due to the simplicity of the data to compress, such complex algorithms would actually be less efficient. Also, because we have extensive knowledge of what the data stream will look like, we can design the compression algorithm specifically to the data stream.

RLE is often used for ASCII data, in which there are a large number of characters, and one (or two) can be identified as the 'control' character. However, due to the format of the data in this situation (binary), there are only two signals – a logical '0' and a logical '1'. It is important to note here that in our situation, the *dynamic* region will be much smaller than the *static* region (assuming the LLD functions as expected). Therefore, it would seem allowable to expand a logical '0' to '00', and a logical '1' to '01', and have '1' as our control signal. Two questions must then be answered. Firstly, how long (how many bits) should the length field be, and secondly, what is the minimum efficient length of a stream to encode?

We know the LLD module consists of 48 detectors. We would therefore not expect the length to be greater than six bits long, allowing a length of 63. Therefore if the control field is one bit ('1'), the length field is six bits, and the value field is one bit, the length of one compression 'unit' is an octet (8 bits). This would mean a static (unchanging) stream 3 bits long (aaa) would be better expanded as 0a0a0a. A 4 bit static stream (bbbb) could either be

expanded as 0b0b0b0b or compressed as '1.000100.b', each option resulting in an 8-bit output. Therefore, the minimum efficient run size would be five. We can then optimize the working of the length field by saying '000000' equates to '5', '000001' equates to '6', and '111111' equates to '68'. Therefore the value in the length field will be 'run length – 5'.

Therefore, we would compress a stream such as:

11111111111111111111111111100101111111111111111

To:

1.010110.1 {control, 27 –5 = 22, value = 1 … 27 1's}
00.00.01.00 {0, 0, 1, 0 expanded}
1.001100.1 {control, 17 – 5 = 12, value = 1 … 17 1's}

This compresses 48 bits down to 24 – compression ratio of 2.

Due to the nature of the data to be compressed, it is possible to develop an algorithm even more efficient. We know that the runs will consist of a long stream of '1's. Therefore, a value of '1' could represent a stream of say 8 '1's, and a value of '0' could mean 'interpret the next 8 bits literally'. Therefore, the same bit stream would compress to:

1.1.1	{24 1's}
0.11100101	{interprets as 11100101}
1.1	{16 1's}

This compresses 48 bits down to 14 – compression ratio of 3.43. This second algorithm was designed for one module. However, remembering that the RLE maximum run length was 68 bits (compressed to 8 bits), this algorithm would be quite suited to larger applications, as eight 1's would represent a run of 64 1's. Therefore, the best algorithm to use would be the latter variation of RLE.

The implementation of RLE was built in two intermediate layers. The bottom layer was responsible for packing both individual bits and bytes into a series of bytes to be sent on the serial port. On top of this layer the encoder was built. As the data was clocked in from the shift registers it was passed both to local memory, and to the encoder.

The layered implementation described above is shown below in *Figure 16*.

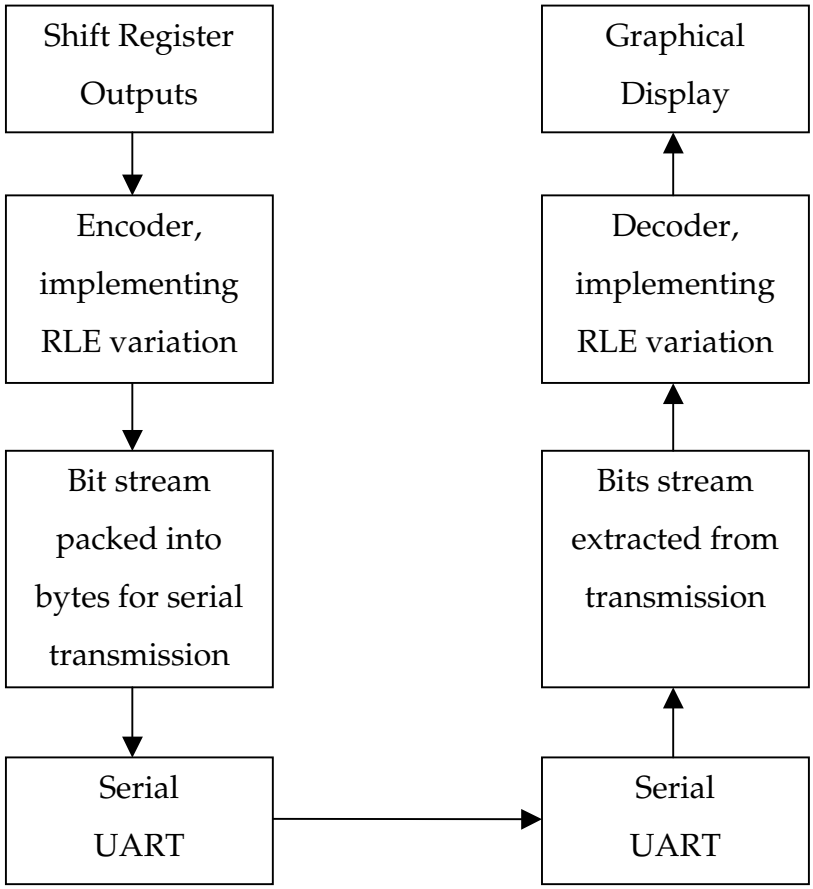


Figure 16: *Layered implementation of encoded serial transmission*

5.3 – PC Software

Simple PC software has been written to demonstrate the effective compressed serial communications. This software was developed in Microsoft Visual Basic 6.0. This programming language was chosen for its power to create Graphical User Interfaces (GUIs).

The PC software calculated the position of the laser beam in parallel with the microcontroller. The position of the laser beam was represented both numerically and graphically on the connected computer. As the information received through the serial port was encoded, a decoding algorithm was written.

The PC Software was specifically designed to be of use in the displaying of the product; parts of it are not particularly practical for users, yet it will draw their intention to it in the initial stages of interest.

Chapter 6: Product Evaluation

Each of the specifications derived in chapter 3 are discussed with reference to what was implemented in chapters 4 and 5.

6.1 – Specifications Match

Overall, the specifications for the design have been matched. Of greatest concern is the physical size due to the use of 6 AA batteries, and the precision of the device.

6.1.1 – Budget Requirement

The component cost of the laser level detector should be under $200AU. *Table 5* shows the budget breakdown.

Component Budget for the Laser Level Detector			
Item	**Price**	**Quantity**	**Total**
Sensors			
635nm Surface Mount LED	0.084	96	8.064
LM324N Quad Op Amp	0.65	24	15.6
74LS14M Hex Schmitt Trigger	0.625	16	10
74HCT166D PISO shift register (RS)	0.91	12	10.92
10M Resistor FA335-2419 0603	0.146	96	14.016
1M Quad Resistor Array	0.203	24	4.872
Decoupling Capacitors (150nF)	0.12	29	3.48
Bulk Capacitor	0.525	1	0.525
PCB @ 2.20per inch^2	2.2	9.12	20.064
PCB @ 2.20per inch^2	2.2	9.12	20.064
Display			
Dual 7-Segment Display Common Anode	5.18	2	10.36
2003 Darlington Driver (x7)	0.981	1	0.981
600R Quad Resistor Array	0.203	2	0.406
Serial Communications			
MAX232 Line Driver	3.78	1	3.78
100nF Capacitor	0.203	5	1.015
Main			
Atmel AT90S8535	15.89	1	15.89
Fixed Voltage Regulator 5V	2.59	1	2.59
10uF Tantulum Caps	0.488	2	0.976
10 Way box header	3.22	1	3.22
DB9 90d Femail Socket	2.81	1	2.81
4MHz Crystal	2.53	1	2.53
22pF Capacitors	0.109	2	0.218
Misc. Resistors	0.096	2	0.192
635nm Surface Mount LED	0.084	1	0.084
PCBs @ 2.20per inch^2	2.2	6.08	13.376
TOTAL			166.033

Table 5: Budget Breakdown

This price may initially seem deceiving. The price quoted for PCB manufacturing is dependant on the PCB being part of a bulk order. The components selected are individually purchased from suppliers who focus on delivery speed rather than price for product diversification. Therefore, this price could fluctuate to be more expensive or cheaper, depending on these two factors.

6.1.2 – Precision Requirement

Due to physical limitations of solder mounting and routing, the diodes could only be placed every two millimeters. By testing the device, and allowing a maximum diversion of 8mm, it was found that the center of the laser beam was at most +/- 1mm out. The laser was detected up to a divergence of around 13mm. Although this fulfills the minimum precision requirements it is unfortunate that a greater precision was not reached, as this would expand the set of possible uses of the product.

6.1.3 – Modularity Requirement

The requirement of modularity has been met. The sensor modules are designed such that they can easily be 'plugged' together, allowing many sensor modules to be connected to one control module. The control module will automatically detect the number of sensor modules. The sensor modules themselves are designed so that they can be placed directly adjacent to each other, allowing the 2mm distance between each detector diode to be continuous across the borders of the modules.

6.1.4 – Visibility Requirement

The visibility of the display was tested in the brightly lit thesis labs. The display could be read from a distance of 10m with ease, and depending on the eye-sight of the beholder, the distance could extend further from there.

6.1.5 – Power Consumption Requirement

One sensor module requires a current of 70mA to operate (experimentally determined). This is due to the large number of logic ICs on the board. For

example, each of the Schmitt trigger ICs draws a current of around 3mA in operation. The sensor board contains 26 ICs in total, giving the expected current requirement in the order of 70mA. A current budget for the sensor module can be seen in *Table 1* and for the control module in *Table 4*.

The total current usage for a 1-sensor module device is in the order of 110mA. The requirement for power consumption was that it should be sufficiently low to allow portable (battery) operation.

A standard Energizer AA alkaline battery has a nominal capacity of 2500mAh. With six of these batteries, a voltage of 9V will be supplied to the device. This will have a nominal capacity of 15000mAh. Per battery used, the device will require 16.7mA of current. This is equivalent to a load resistance of 90Ω. *Figure 18* shows the output voltage of a standard AA Energizer Eveready battery, with a constant resistance of 43Ω [19]. The device will continue to function correctly until the output voltage of the batteries is 5V. If six batteries are used, the output voltage of each battery is 0.83 volts. If it can be assumed that the service hours in *Figure 18* can be doubled to give the curve for a 90Ω resistance, the service hours would be around 190 hours (2 x 95hrs) at 4 hours / day. This is certainly sufficient.

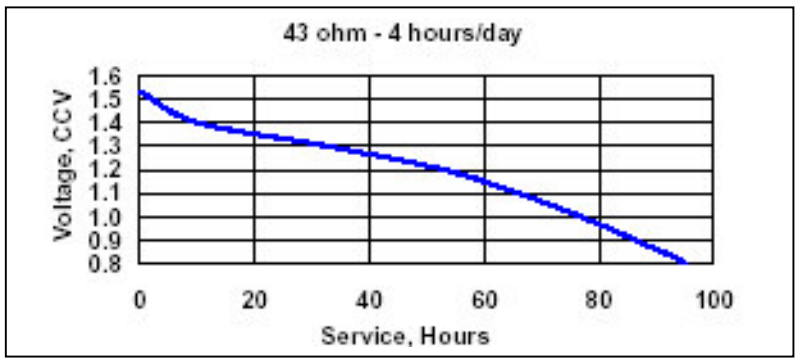


Figure 18: AA Eveready discharge

6.1.7 – Serial Link Requirement

The serial link with the PC was implemented successfully. This allows the information on the display of the LLD to be sent to a PC. The PC may be set up to log this information, or to display it.

6.1.8 – Read Rate Requirement

The device must be read and updated at a rate fast enough to deceive the human eye into thinking it was immediate. At the time of writing this thesis, the device was running with one module at a rate of 400Hz (2.5ms per sensor module scan). In this respect the specifications have been exceeded. It was believed that the implementation of a fast sensor module would allow the product to be used in applications of high-speed automation.

6.1.9 – Package Dimension Requirement

The design of the sensor PCB is quite thin, and fits within the package dimensions specified (thinner than 1cm). The control module will be a little thicker – due to the connectors and the batteries required.

6.2 – Personal Performance

The strengths and weaknesses of my personal performance are discussed below.

6.2.1 – Strengths

Throughout the course of this project, one of my major strengths has been managing my time so that the project gets done. There have been certain milestones, for example the first PCB run in semester 2, that I have set for myself and adhered to; this has resulted in the project remaining on schedule. These milestones have been reached by applying a constant work discipline.

Contingencies have been accounted for in my planning, allowing for recovery when things go wrong, such as a PCB which was made with an incorrect footprint. Because I had planned ahead, the mistake was recoverable.

Overall, my methodologies were sound and resulted in the development of a project which functions mostly according to the desired specifications. I was particularly strong at developing the solution once the path to that solution

had been specified. However, there were several holes in my performance. These are discussed below.

6.2.2 – Weaknesses

Firstly, in the design stage of the project I was far too quick to jump to a solution. An x-inefficiency is described as inefficiency in a system due to a lack of search. This stage of development was particularly poor for me; I would have benefited greatly if further research had been done into detection techniques. For example, there is an array of detector diodes already available in devices such as scanners, which could be reverse-engineered and reused for this application. These would have accuracy far greater than the system that I have developed.

Another weakness was the inconsistency of design being done with direct reference to the specifications. In many cases, the specifications were developed with the design in mind, instead of vica-versa. Also, considerations such as current usage were not particularly held in mind while designing the project, even though they were outlined in the initial specifications.

6.2.3 – New Skills Learnt

In completing this project, many new skills were learnt. I had to learn how to manage my time and effort to complete the project as required. Skills in surface mount PCB design and routing design were learnt with the complexities of the sensor module. There is a need for current analysis to make sure the current supplied can perform the function required (in the case of the display multiplexing). Experience was gained with Atmel development, and the project was designed to be in-circuit programmable. Knowledge was gleaned from a variety of different resources for design of the serial connection, controller circuitry, display, and sensor circuitry.

6.2.4 – Addressing Weaknesses

The weaknesses described in the previous section should be addressed. When starting my next project, it is imperative that I perform a broader and deeper

search before manufacturing the solution in my mind. As design goes on, a more formal approach should be taken to make sure specifications are being met. This will address the second point of weakness.

Chapter 7: Future Developments

This chapter examines possible future developments that would improve the product.

7.1 – Optical Enhancement of Design

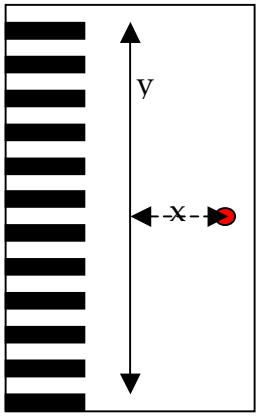


Figure 19: Ruler Diagram

The ideal specifications for the device in chapter 3 identify the need for a broader reception area. Currently, the device only recognizes a beam when it lands directly on the thin line of detectors running down the center of the board. Ideally, the distance of 'x' shown in the diagram (*Figure 19*, left) should be in the order of 1 or 2 centimeters. This could be achieved adding some simple optics to the front of the device, such as a converging lens. The x distances in *Figure 19* and *Figure 20* are the same. By adding the lens shown in *Figure 20*, any incident beams displaced from center of the board will be focused in onto the thin row of detectors.

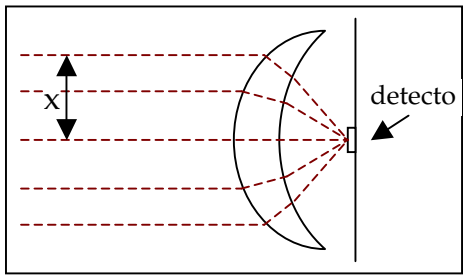


Figure 20: Optical Enhancement

7.2 – Using a Different Detector

The sensor module that has been developed as part of this thesis is quite similar in concept to the sensor module in a scanner device. The difference is that the scanner's sensors have been designed for a much higher resolution and a broader array of input colours. However, it could be possible to use the scanner's input detector array in this device, as the technology is quite similar.

7.3 – Current Limiting 'Sleep' Mode

A current limiting 'sleep' mode could be used for the device. The largest consumers of current on the device are the sensor modules. These are being constantly scanned (at quite a fast rate) even when a laser beam is not present. The device could put the sensor module to sleep (switch it off) and turn it back on again only to do another quick scan to see if there is a laser beam present. If there isn't it is turned back off; if there is, the device continues to scan at the fast rate. The sensor module could be re-awoken every second for a quick scan. A delay countdown is reset to the initial value every time a laser beam is detected, leaving the sensor device 'on' for only a certain time after the laser beam is no longer present.

7.4 – Wireless Link from Sensor to Controller

A wireless link from the sensor module to the controller module would allow the controller to be hand-held. This could also involve a wireless link from the controller to a PC, if one was necessary.

7.5 – Latch Stage in Circuitry

If the improvement in *Section 7.2* is not made, adding a flip-flop latching stage could enhance the current sensor circuitry. At present, a laser beam may enter and leave the sensor area while the data is being clocked in. This would present problems if the device were to be used with a spinning beam. The latches could be cleared immediately after the shift register inputs have been latched, and will capture and hold a detection until the shifting is complete. Performing this enhancement and using a spinning beam would render the detection width principle obsolete, as the spinning beam would always cross the center of the ruler.

Chapter 8: Conclusion

Initially, the project concept involved building an electronic reader for a laser level detector. The concept was expanded to encompass the detection of a laser beam in any low-precision one-dimensional alignment application. The device is designed to replace the 'human' element in reading the position of a laser beam on a scale. Instead, the scale/ruler automatically reads the position of an incident laser beam on its surface.

The device specifications laid out in *Chapter 3*. The resolution required, the length to be measured, the sampling rate and the display requirements were then determined. With these requirements and a review of current technologies in mind, the hardware and software of the device were developed.

The hardware concept for the sensors was quite simple – an array of detectors, with a digital output fed into a row of shift registers. The shift registers could be latched to lock in the data, and then clocked to feed the data back to a microcontroller. The controller module hardware involved a display to be multiplexed, a microcontroller (in-circuit programmable), a serial connection and a power supply.

The on-board software was responsible for coordinating, reading and interpreting the sensor hardware. It would calculate the position of the laser beam from the data fed back by the shift registers, and update the value to the display and serial connection. The PC software reads the data and displays it on the PC.

The specifications set out were met in the final project. In generally, the performance was good, however in retrospect some aspects of the product life cycle (particularly the research stage) could have been performed on a more detailed level.

Reference List:

1. Jatco (Aust.) PTY LTD. n.d., *Semiconductor laser applications development (Products)* Available: http://members.ozemail.com.au/~jatco/www/products/index.htm [2002, April 15]

2. Stenberg, L. n.d., *Why is the existence of position sensing detectors so important?* Available: http://www.sitek.se/section1.htm [2002, April 17]

3. The PSD School. N.d., *PSD vs CCD,* Available: http://www.sitek.se/psd_cdd.htm [2002, April 17]

4. On-Trak Photonics, Inc. n.d., *Application Notes: PSD vs CCD,* Available: http://www.on-trak.com/appnote2.html [2002, Mar 24]

5. Applied Resolution Technologies. n.d., *Product List,* Available: http://www.appliedresolution.com.au/newart/cnc/docs/prod.htm [2002, Mar 24]

6. Leica Geosystems GR LLC, 2002. *General Construction Lasers,* Available: http://www.laseralignment.com/general.html [2002, May 14]

7. Master Wholesale, Inc. n.d., *Pocket Laser Accessories*. Available: http://www.masterwholesale.com/levelite/cat15_2.html [2002, May 14]

8. Fisher, J.E. and Gatland, H.B., 1976. *Electronics from Theory into Practice*, 2nd Edition, Pergamon Press Ltd, Great Britain.

9. Horowitz, P. and Hill, W., 1980. *The Art of Electronics*.1st Edition, Cambridge University Press, New York.

10. Mattson, R.H., 1966. *Electronics*. 1st Edition, John Wiley & Sons, Inc., New York.

11. Wikipedia, 2002. *Run Length Encoding*, Available: http://www.wikipedia.com/wiki/Run-length_encoding [2002, September 18]

12. Maxim Integrated Products 1999. *+5V Powered, Multichannel RS232 Drivers/Receivers*. No. 19-4323; Revision 8 (11/99).

13. Atmel Corporation 2001. *8-bit AVR Microcontroller with 8K Bytes In-System Programmable Flash.AT90S8535*. Revision 1041H (11/01).

14. DataCompression Reference Center, 2000. *RLE – Run Length Encoding*, Available: http://www.rasip.fer.hr/research/compress/algorithms/fund/rl/index.html [2002, September 18]

15. Texas Instruments Inc., 1993. *ULN2001A, ULN2002A, ULN2003A, ULN2004A* D*arlington Transistor Arrays*. SLRS027 Revised 1993.

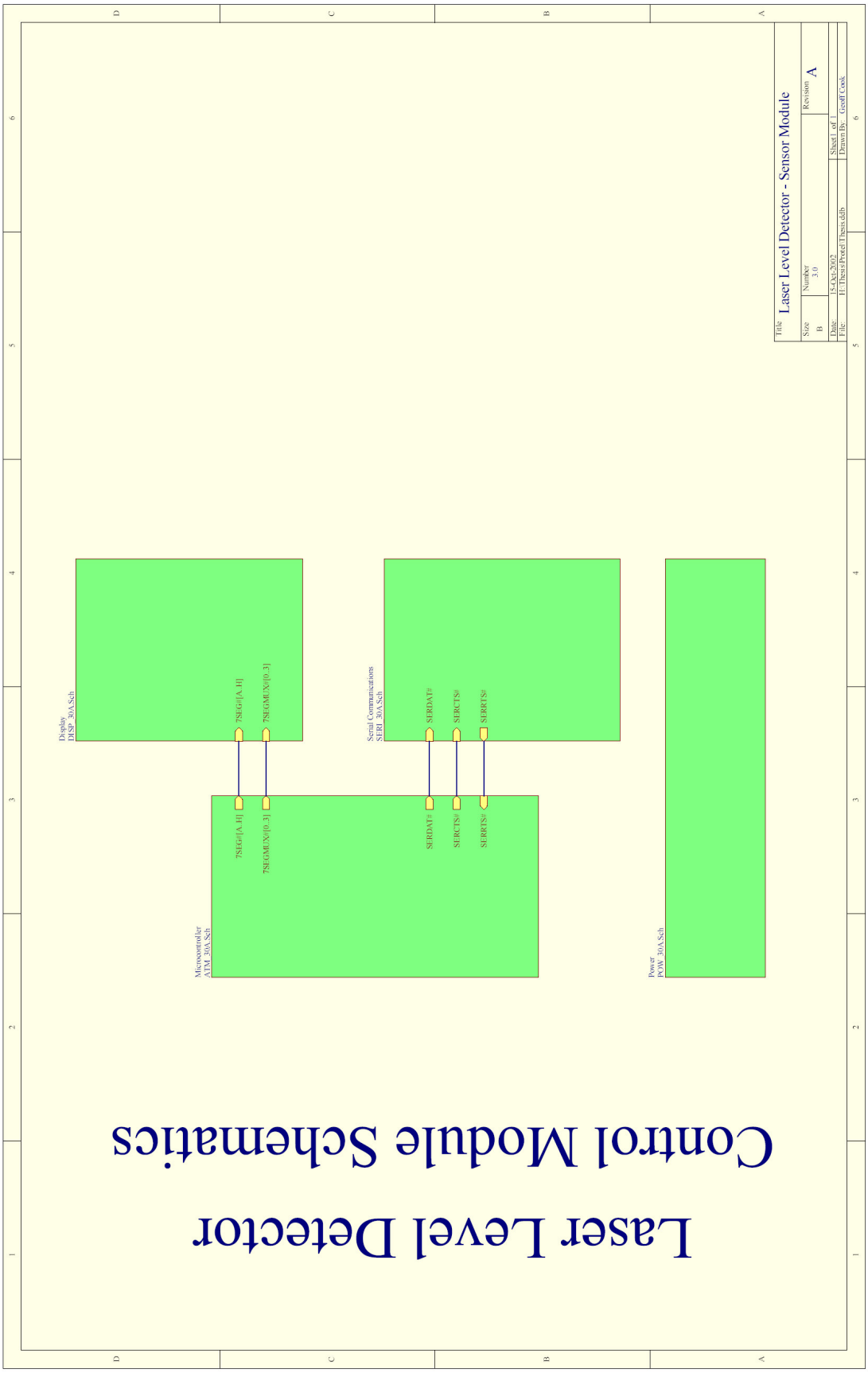
16. Peacock, C., 2001. *Interfacing the Serial/RS232 Port*. Available: http://www.beyondlogic.org/serial/serial1.htm [2002, July 7]

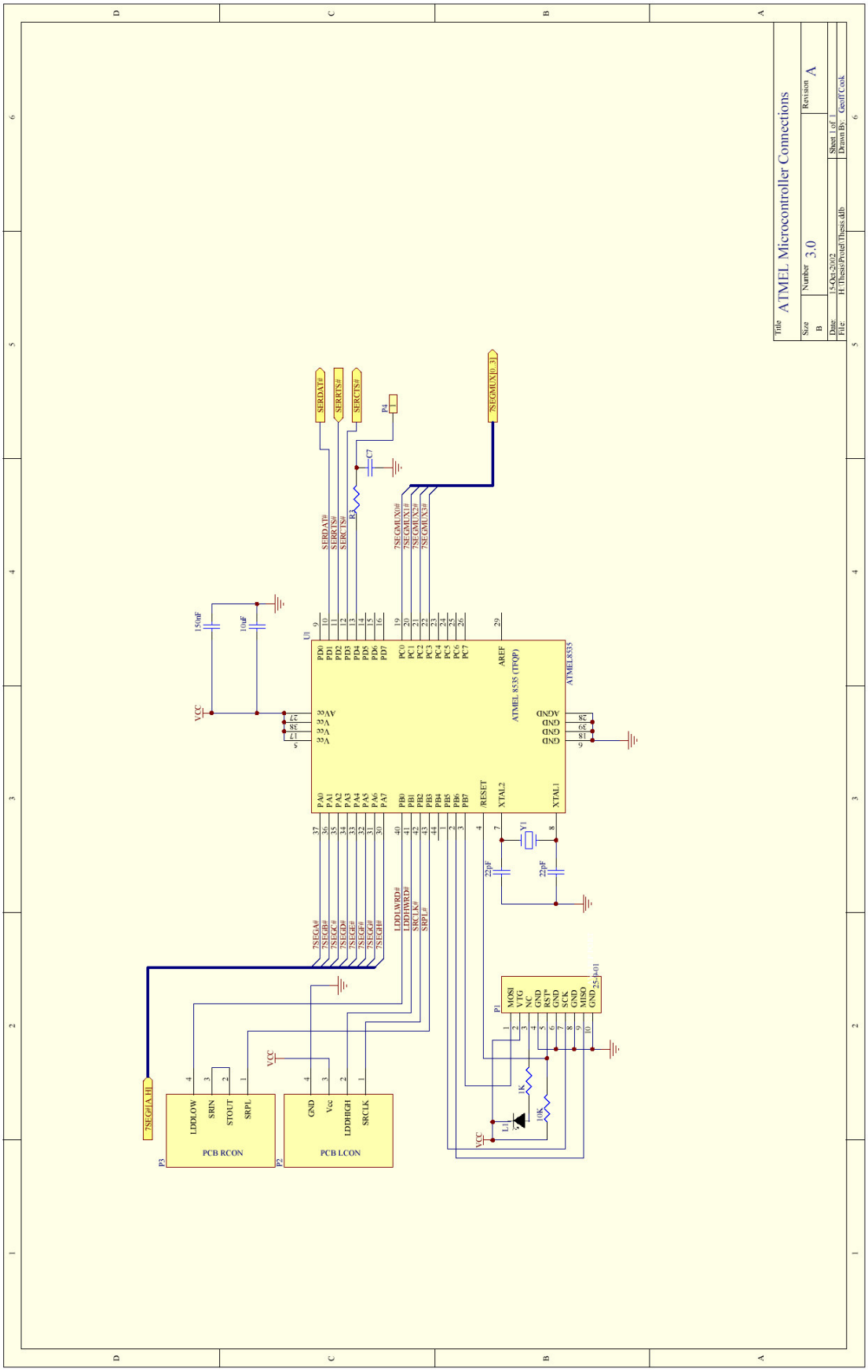
17. Payne, L. J., 2002. *AVR 8151 Project Board and Dongle Schematics*. Available: http://www.itee.uq.edu.au/~comp1300/pracs/Atmel_AVR_Resources/AVR_Project_Board.pdf [2002, September 3]

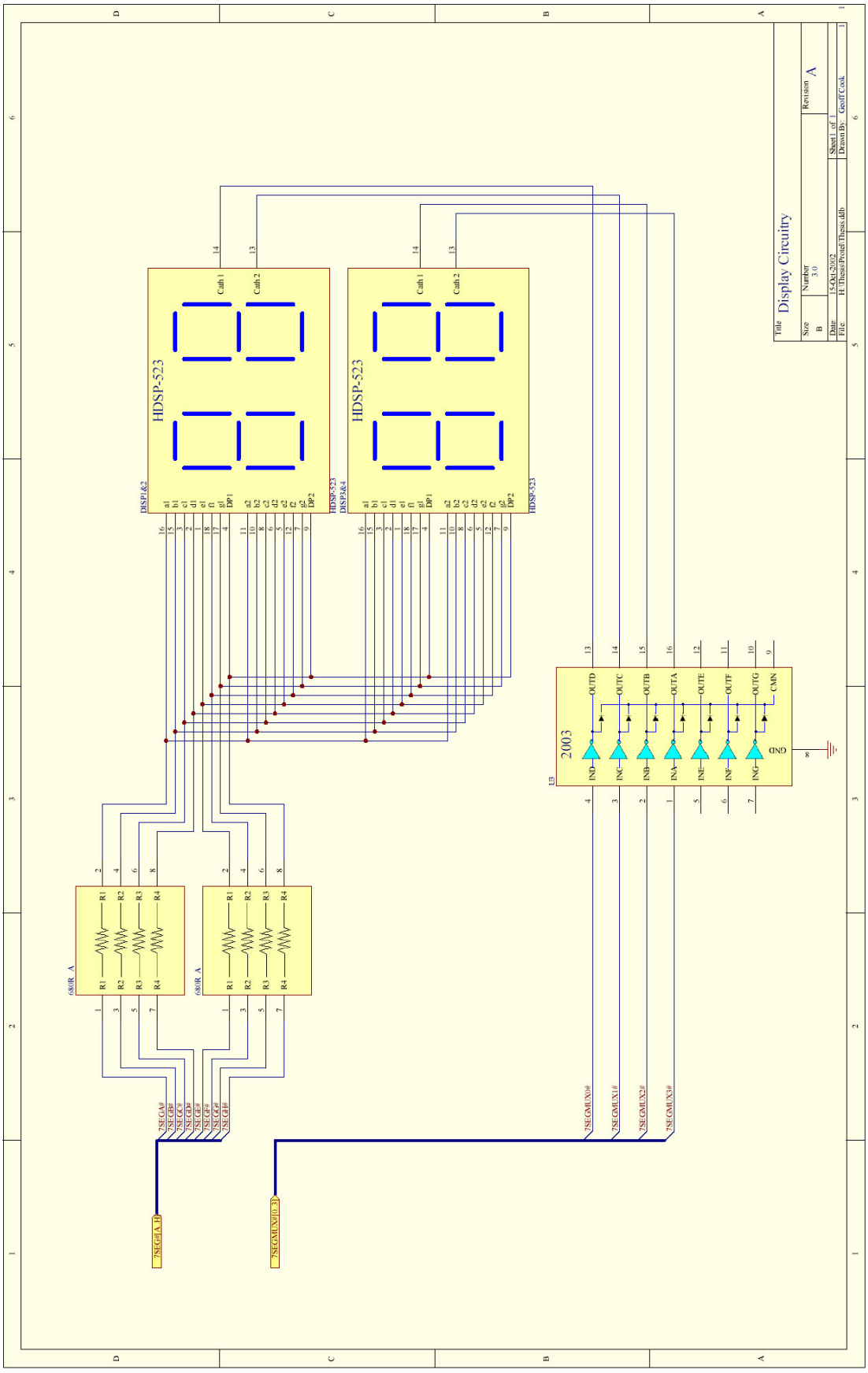
18. Free Software Foundation, 1997. *GNU Development Tools*.Available: http://www.itee.uq.edu.au/~comp1300/pracs/Atmel_AVR_Resources/_avr_gcc/man_avr-gcc.pdf [2002, October 15]

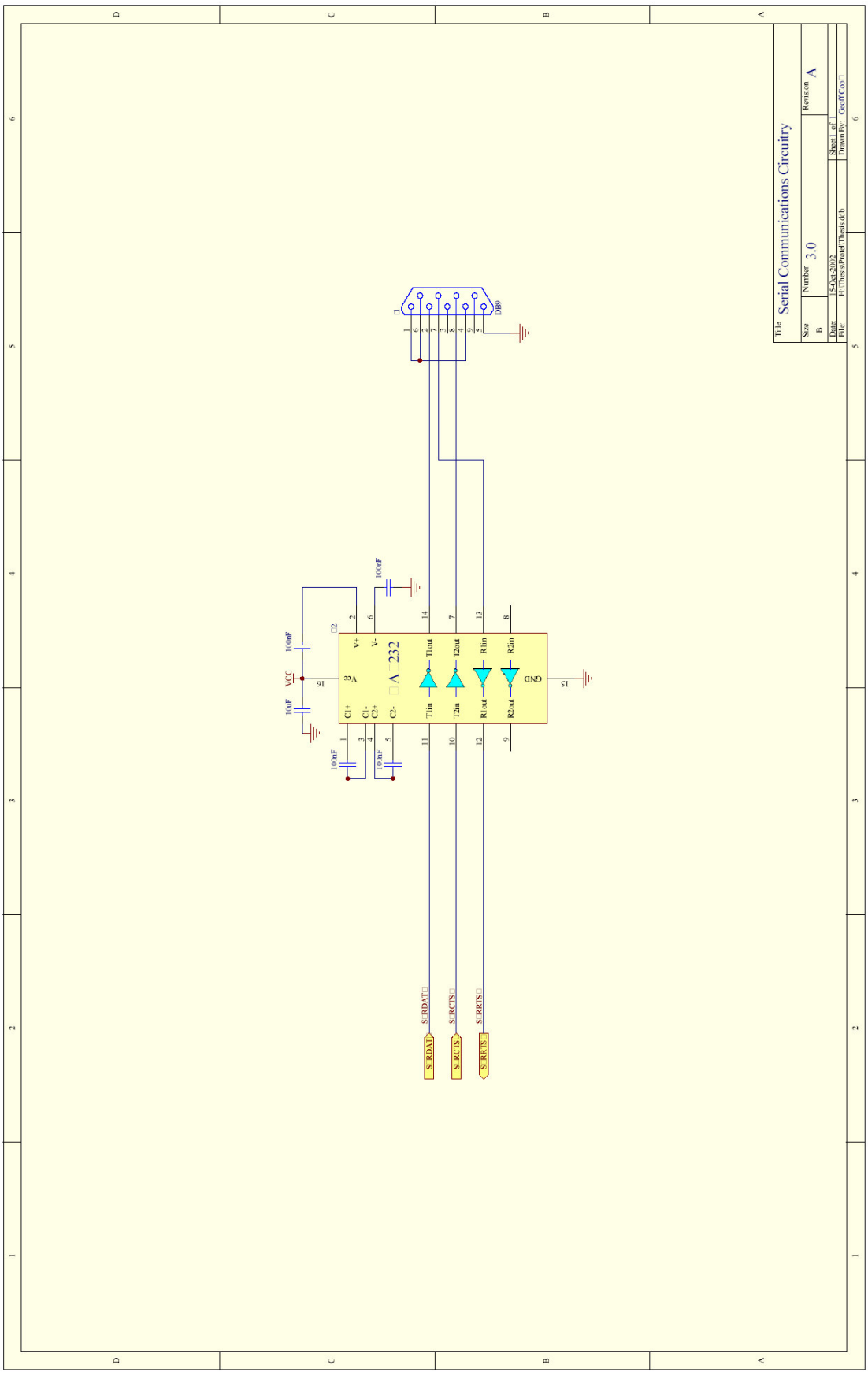
19. Eveready Battery Co. n.d., *Engineering Datasheet: Energizer No. A91.* Form No. EBC - 1189C

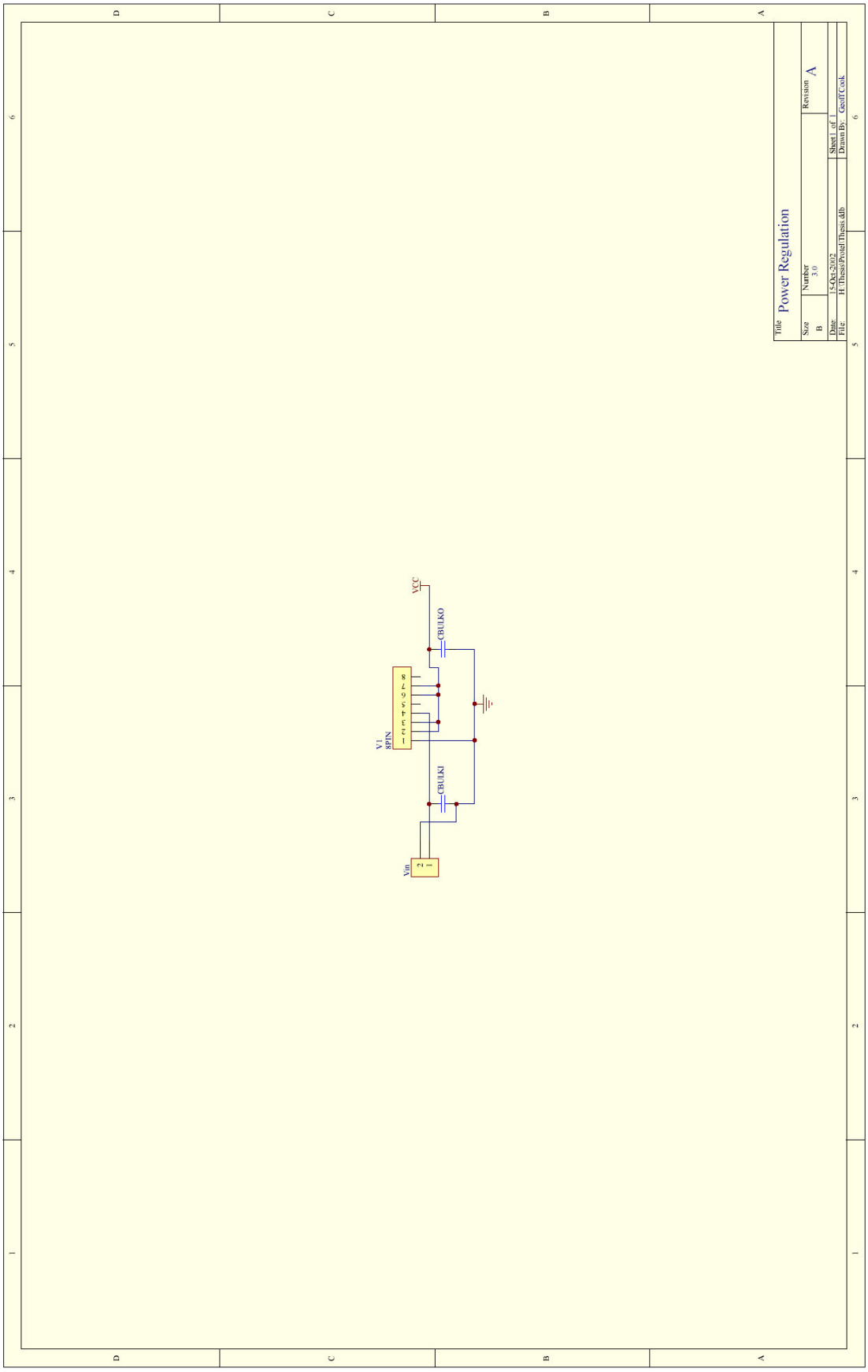
Appendix A: Hardware Schematics


Laser Level Detector - Sensor Module
Title
Size
B
Number
3.0
Revision
A
Date
15-Oct-2002
Sheet 1 of 1
File
H:\Thesis\Protel Thesis.ddb
Drawn By:
Geoff Cook
Display
DISP_30A.Sch
7SEG[A..H]
7SEGMUX[0..3]
Serial Communications
SERI_30A.Sch
SERDAT#
SERCTS#
SERRTS#
Microcontroller
ATM_30A.Sch
Power
POW_30A.Sch
Laser Level Detector
Control Module Schematics

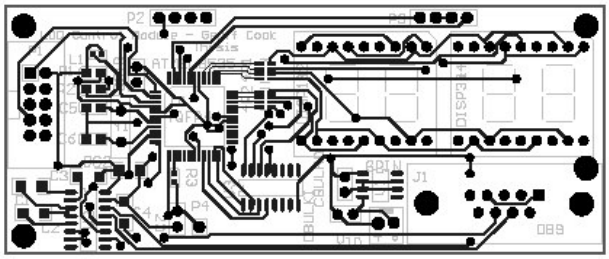

Title ATMEL Microcontroller Connections
Size B
Number 3.0
Revision A
Date: 15-Oct-2002
File: H:\Thesis\Pcad\Thesis.ddb
Sheet 1 of 1
Drawn By: Geoff Cook
ATMEL 8535 (TPQP)
ATMEL8535
PCB RCON
PCB LCON
SERDAT#
SERRTS#
SERCTS#
7SEGMUX[0..3]
LDD1WRD#
LDD0WRD#
SRCLK#
SRPL#
7SEGA#
7SEGB#
7SEGC#
7SEGD#
7SEGE#
7SEGF#
7SEGG#
7SEGH#
7SEG[A..H]
VCC
22pF
1K
10K
MOSI
VTG
NC
GND
RESET
GND
SCK
MISO
GND

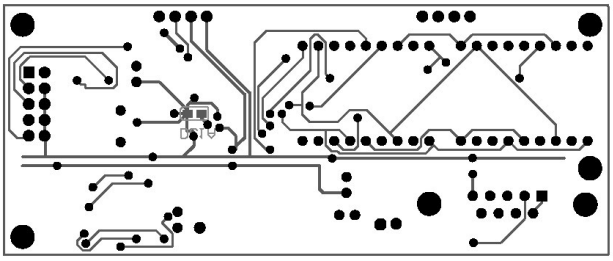

Display Circuitry
HDSP-523
HDSP-523
2003
68HR_A
68HR_A
7SEGMUX0#
7SEGMUX1#
7SEGMUX2#
7SEGMUX3#
7SEGMUX[0..3]

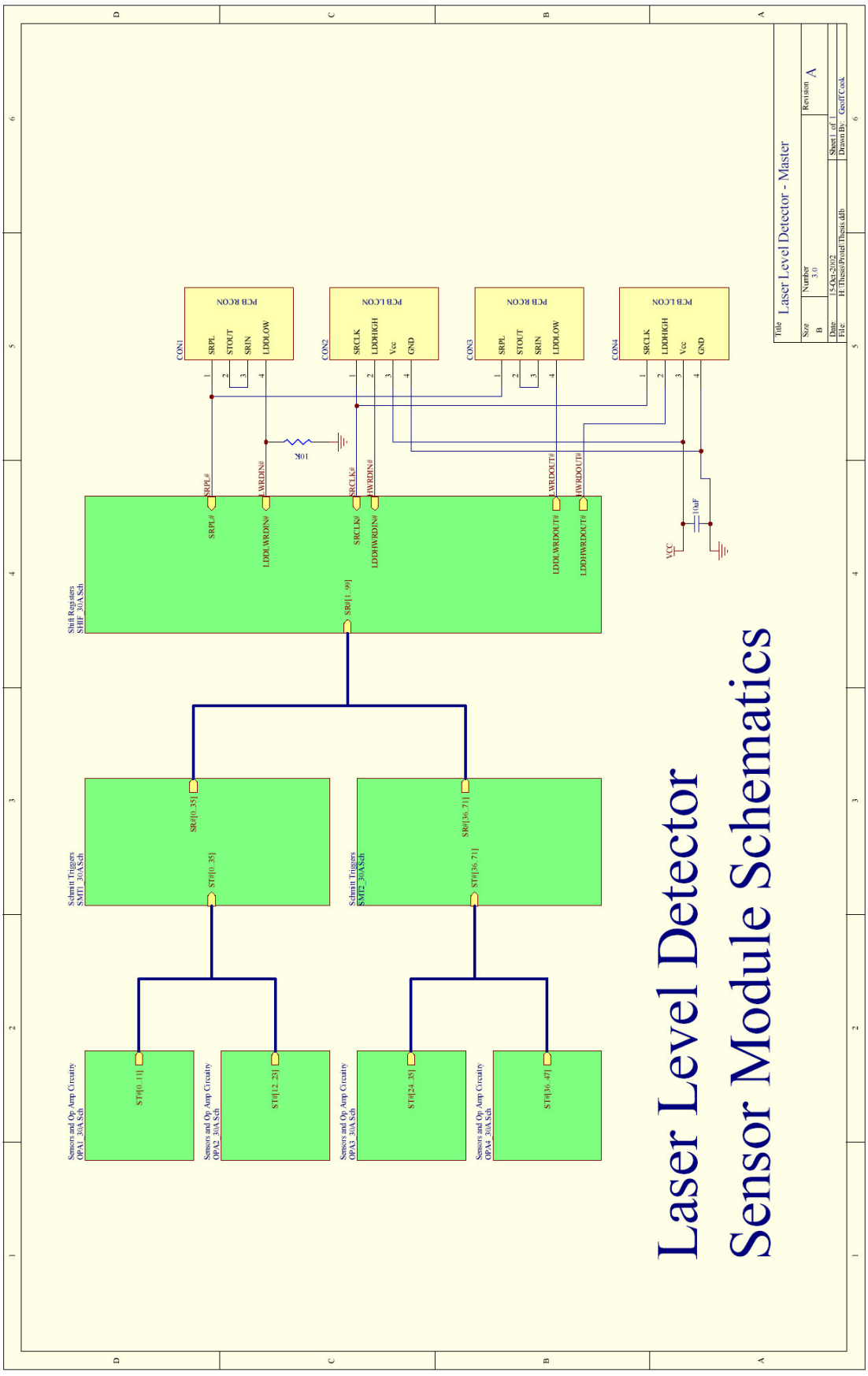

Title Serial Communications Circuitry
Size B
Number 3.0
Revision A
Date 15-Oct-2002
File: H:\Thesis\Pestal\Thesis.ddb
Sheet 1 of 1
Drawn By: Geoff Cook
DB9
MAX232
VCC
100nF
10uF
S_RDAT
S_RCTS
S_RRTS

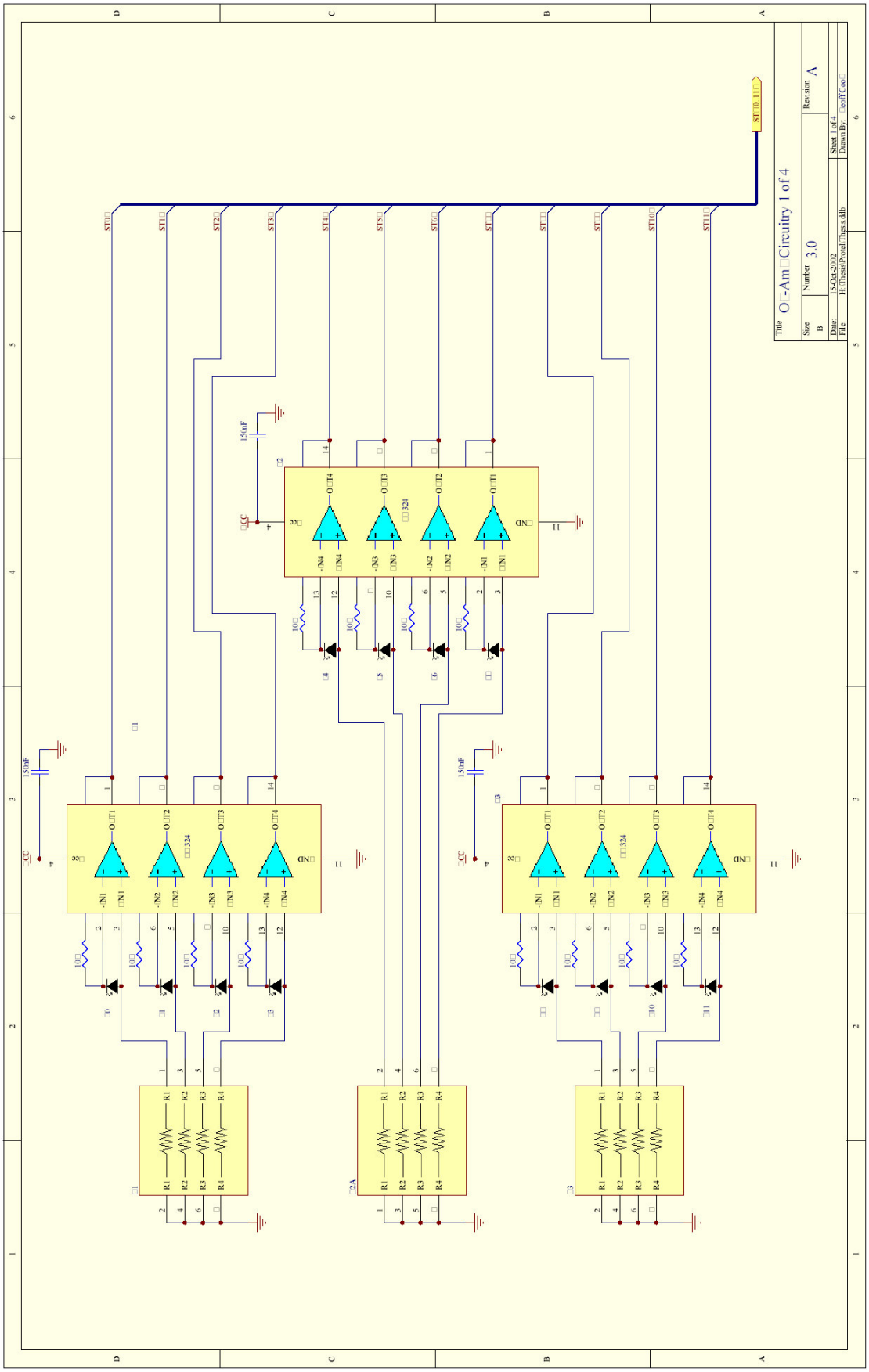

Power Regulation
Size
B
Number
3.0
Revision
A
Date:
15-Oct-2002
Sheet 1 of 1
File:
H:\Thesis\Pcad\Thesis.ddb
Drawn By:
Geoff Cook
VCC
CBULK0
CBULK1
V1
8PIN
Vin

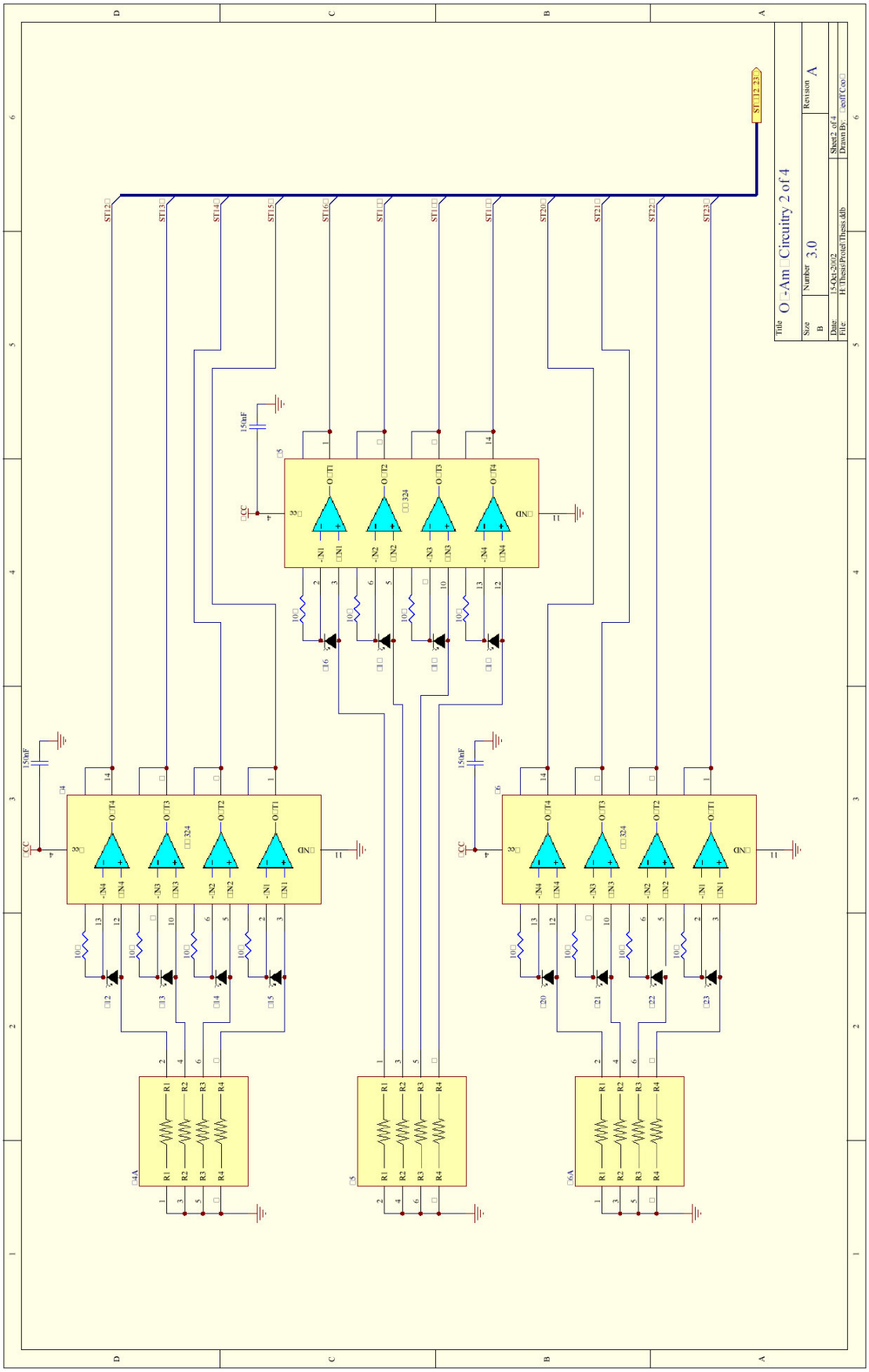
Control Module PCB

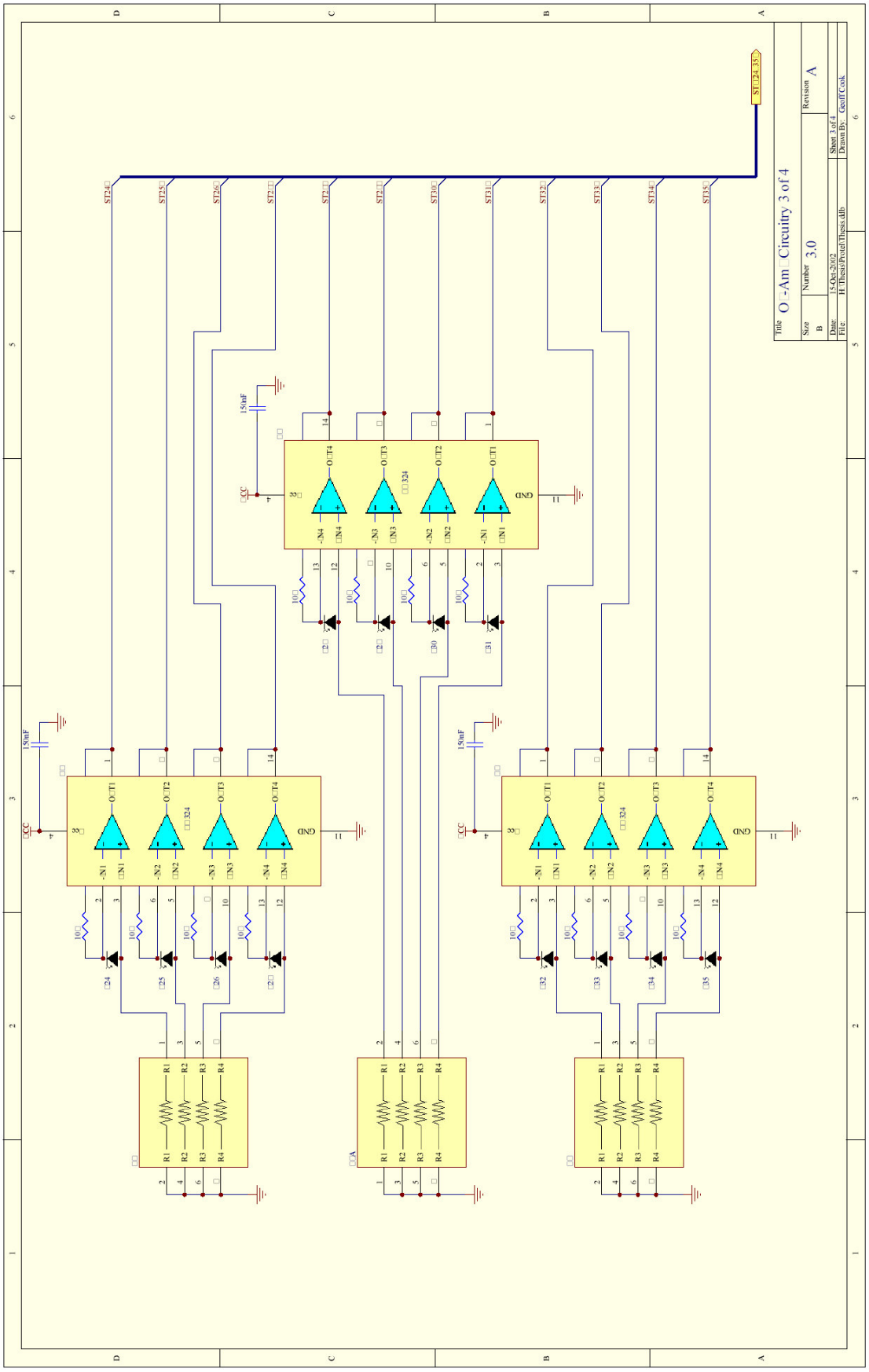


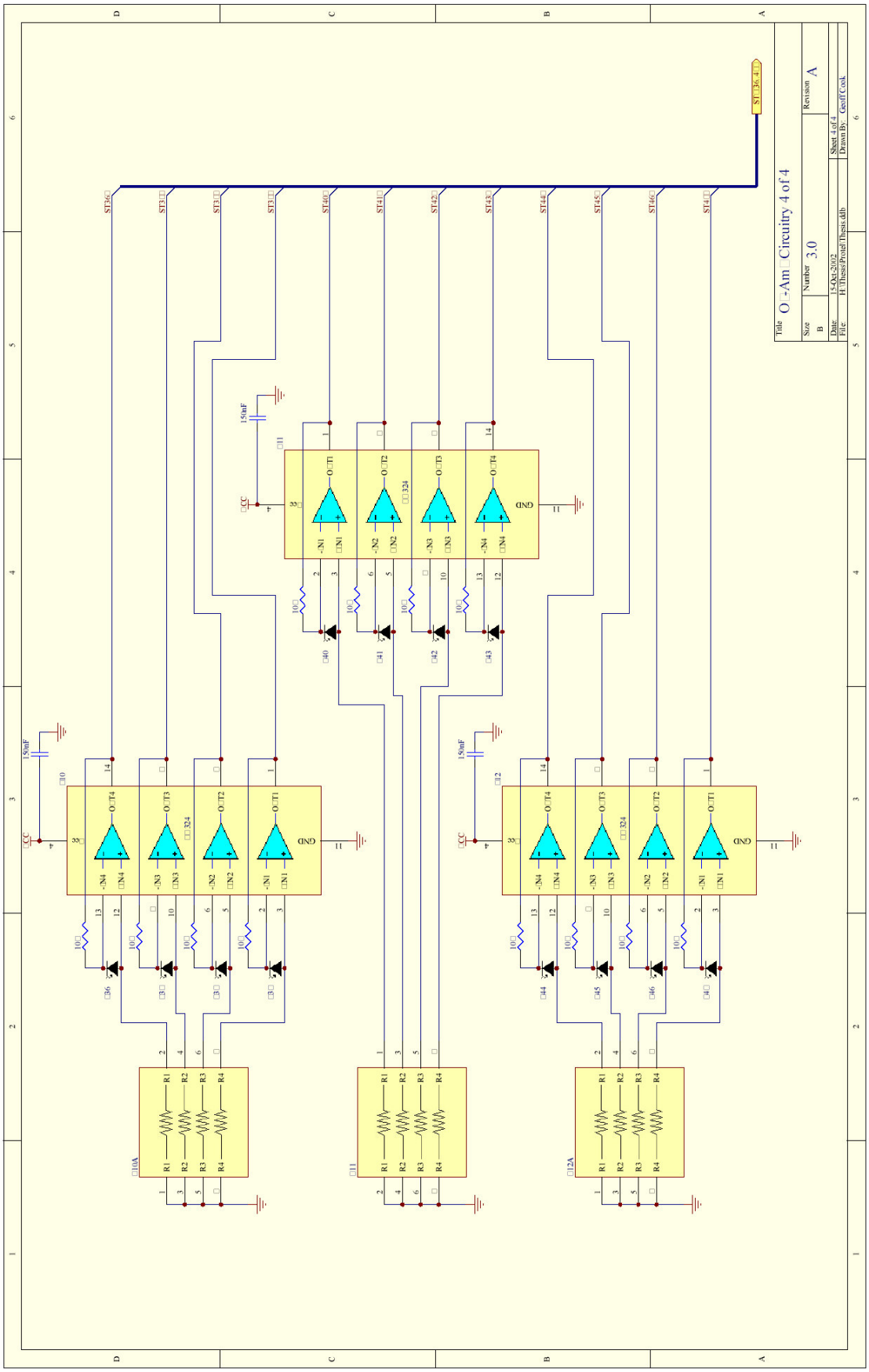


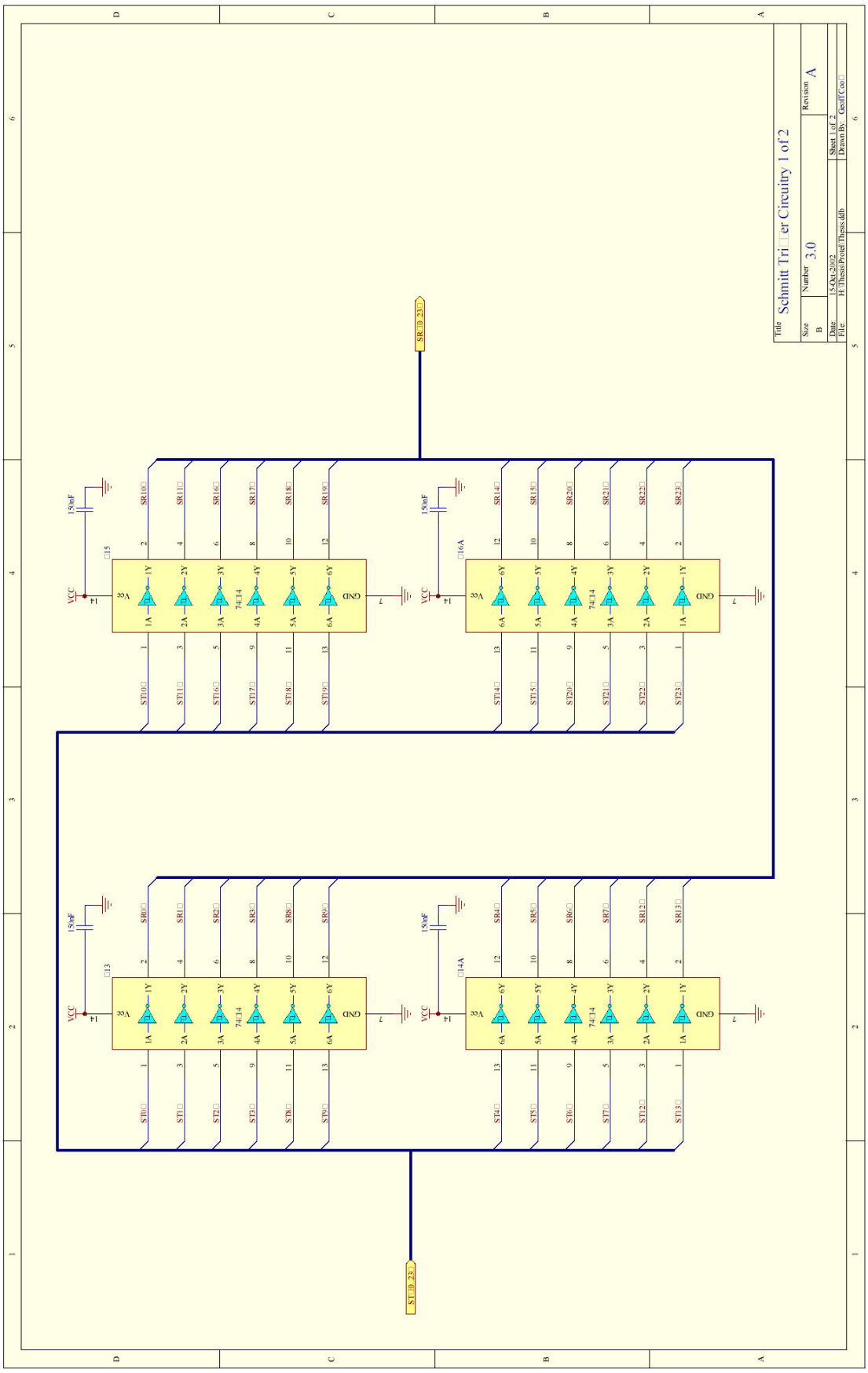

Laser Level Detector
Sensor Module Schematics
Laser Level Detector - Master
PCB RCON
PCB LCON
CON1
CON2
CON3
CON4
SRPL
STOUT
SRIN
LDDLOW
SRCLK
LDDHIGH
Vcc
GND
10K
VCC
10uF
SRPL#
LWRDIN#
SRCLK#
HWRDIN#
LWRDOUT#
HWRDOUT#
LDDLWRDIN#
LDDHWRDIN#
LDDLWRDOUT#
LDDHWRDOUT#
SR#[1..99]
Shift Registers
SHIF_30A.Sch
SR#[0..35]
ST#[0..35]
Schmit Triggers
SMT1_30A.Sch
SR#[36..71]
ST#[36..71]
Schmit Triggers
SMT2_30A.Sch
ST#[0..11]
ST#[12..23]
ST#[24..35]
ST#[36..47]
Sensors and Op Amp Circuitry
OPA1_30A.Sch
OPA2_30A.Sch
OPA3_30A.Sch
OPA4_30A.Sch
Size
B
Number
3.0
Revision
A
Date
15-Oct-2002
File
H:\Thesis\Pestal\Thesis.ddb
Sheet 1 of 1
Drawn By:
Geoff Cook

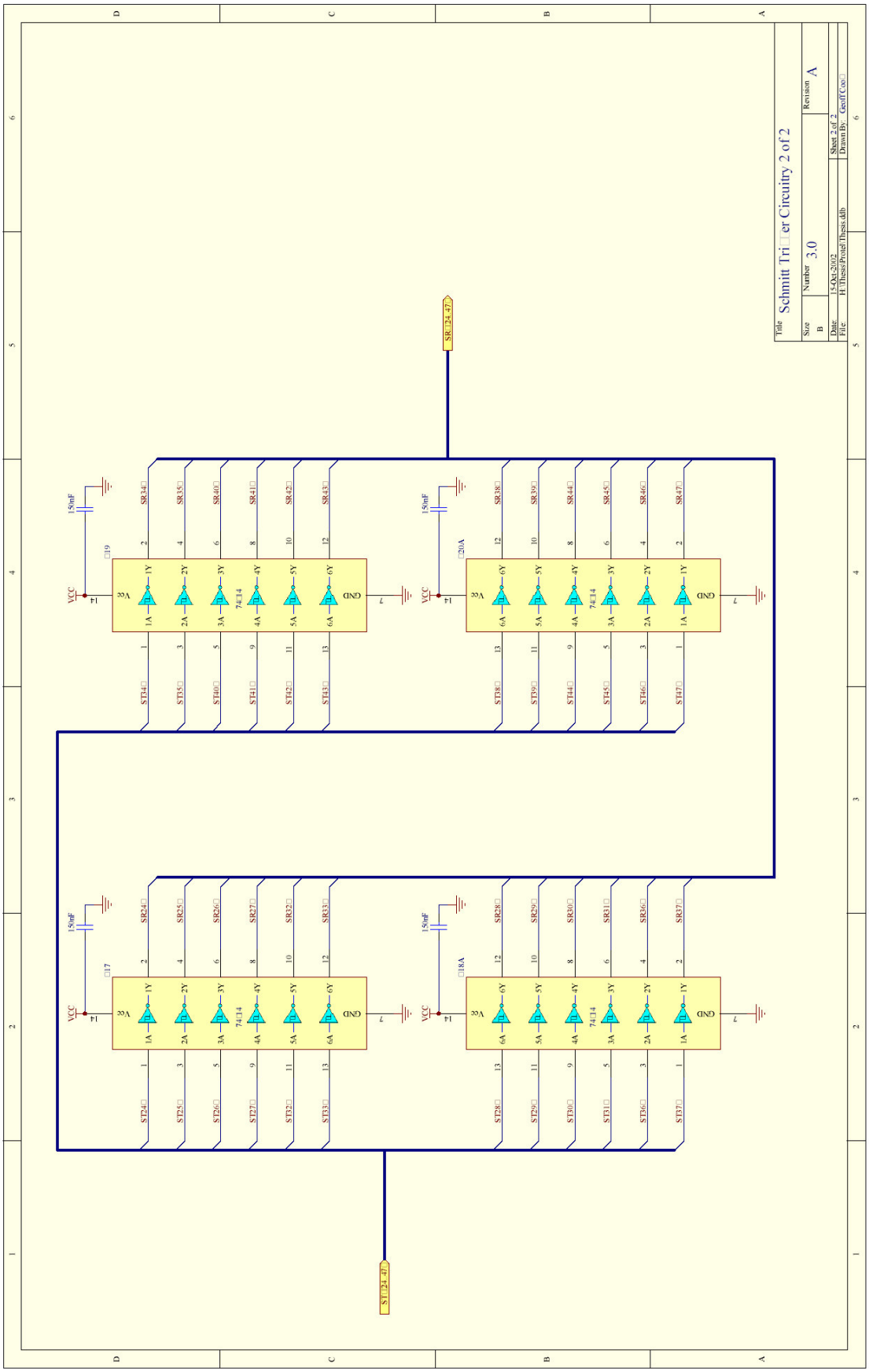

Title O-Am Circuitry 1 of 4
Size B
Number 3.0
Revision A
Date: 15-Oct-2002
File: H:\Thesis\Proto\Thesis.ddb
Sheet 1 of 4
Drawn By: Geoff Cook
ST[0..11]
150nF
LM324
10K

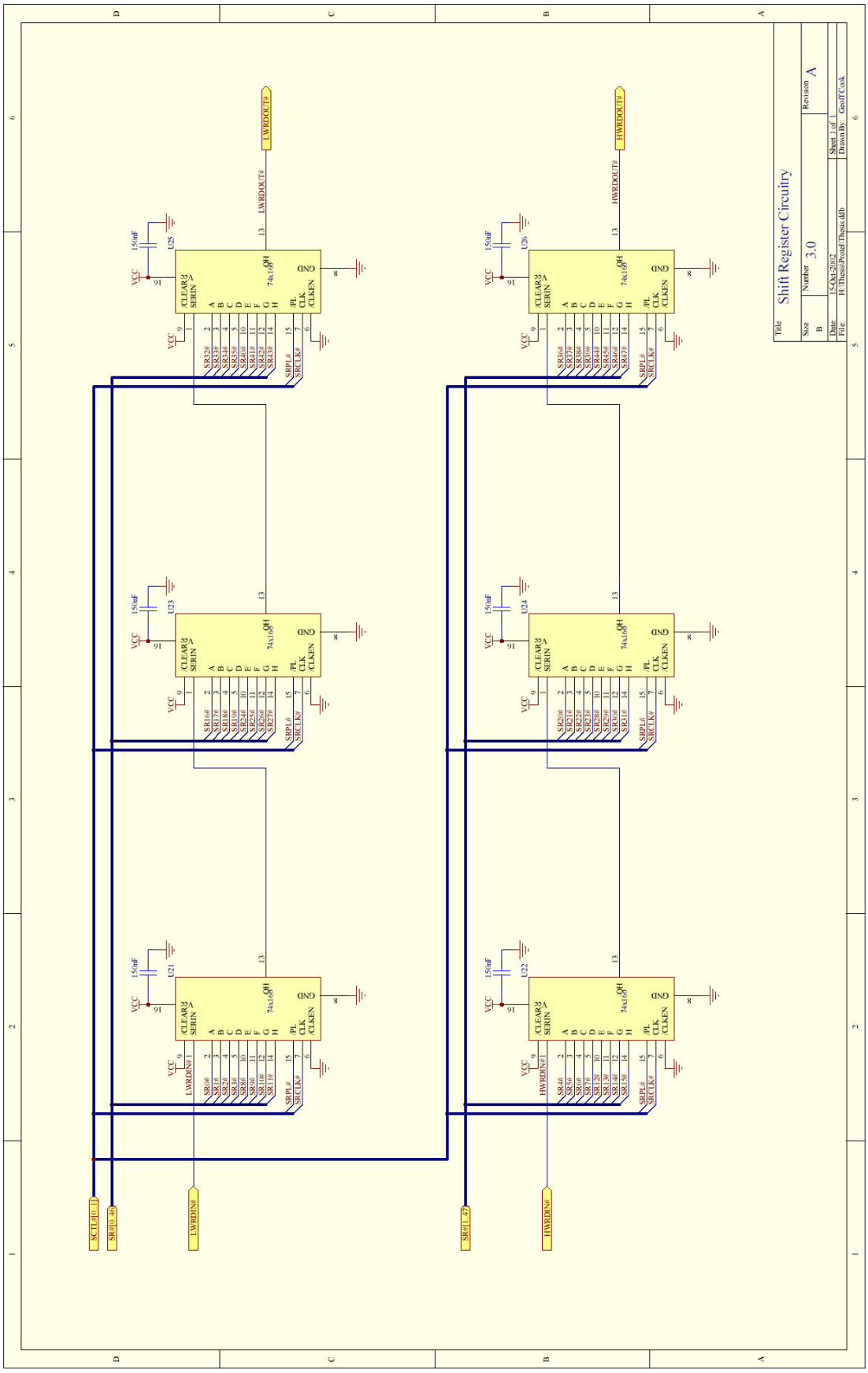

Title
O -Am Circuitry 2 of 4
Size
B
Number
3.0
Revision
A
Date:
14-Oct-2002
File:
H:\Thesis\Pcad\Thesis.ddb
Sheet3 of 4
Drawn By:
geoff Coo
ST[12,23]
150nF
LM324
10K


Title
O□-Am□Circuitry 3 of 4
Size
B
Number
3.0
Revision
A
Date:
15-Oct-2002
Sheet 3 of 4
File:
H:\Thesis\Postd\Thesis.ddb
Drawn By:
Geoff Cook
ST[24..35]
150nF
VCC
GND
O□T1
O□T2
O□T3
O□T4
-IN1
-IN2
-IN3
-IN4
10□
R1
R2
R3
R4

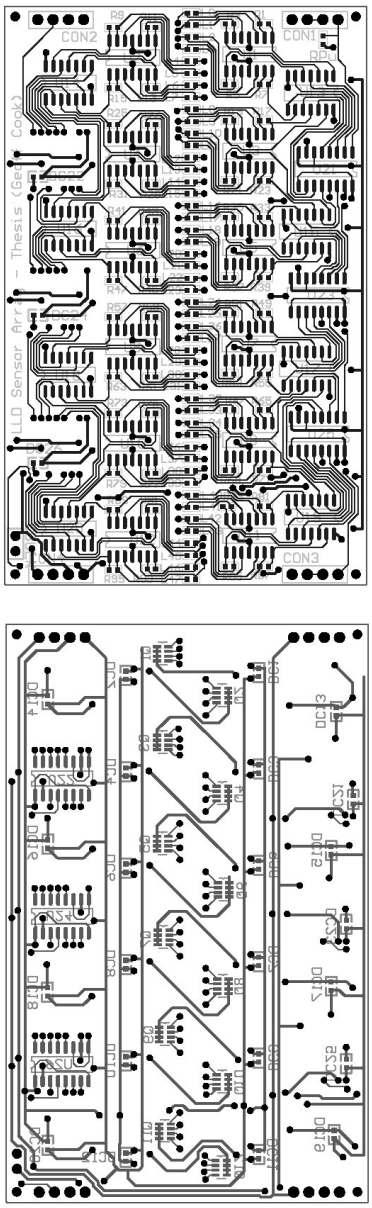

Title O -/Am_Circuitry 4 of 4
Size B
Number 3.0
Revision A
Date: 15-Oct-2002
File: H:\Thesis\Pcbelf\Thesis.ddb
Sheet 4 of 4
Drawn By: Geoff Cook
ST[36..41]
150nF
GND
CC


Schmitt Trigger Circuitry 1 of 2
Size B
Number 3.0
Revision A
Date: 15-Oct-2002
Sheet 1 of 2
File: H:\Thesis\Pcad\Thesis.ddb
Drawn By: Geoff Cook
SR[0..23]
ST[0..23]
74LS14
VCC
GND
150nF


Schmitt Tri er Circuitry 2 of 2
Size B
Number 3.0
Revision A
Date: 15-Oct-2002
Sheet 2 of 2
File: H:\Thesis\Proteff\Thesis.ddb
Drawn By: Geoff Coo
SR[24..47]
ST[24..47]
74F14
VCC
GND
15nF


Shift Register Circuitry
Number 3.0
Revision A
Sheet 1 of 1
Drawn By: Geoff Cook
Date: 15-Oct-2002
File: H:\Thesis\Protel\Thesis.ddb
LWRDOUT#
HWRDOUT#
U25
U26
U23
U24
U21
U22
74x168
VCC
GND
150nF
SCTL#[0..13]
SR#[0..46]
LWRDIN#
HWRDIN#

Sensor Module PCB



Appendix B: Software Segments

```
/*
 * FILE: lld.h - Laser Level Detector
 *
 *       Thesis of Geoff Cook, 2002
 *
 *       School of Information Technology and Electrical
 *       Engineering.
 *
 *       (Computer Systems Engineering)
 */

#include <interrupt.h>
#include <sig-avr.h>
#include <io.h>


#define MASK INT1       0x80
#define MASK INT0       0x40
#define MASK OCIE2      0x80
#define MASK TOIE2      0x40
#define MASK TICIE1     0x20
#define MASK OCIE1A     0x10
#define MASK OCIE1B     0x08
#define MASK TOIE1      0x04
#define MASK_TOIE0      0x01

/*
 * State machine for LLD
 */
volatile enum states {
    INITIALISE,     // initialise the device/data
    HELLO,          // display hello message on device
    DETECT,         // detect the number of modules
    BEGIN,          // LLDetection startes here!
    WAIT,           // Wait a certain time
    LATCH,          // Latch data into Shift Registers
    SHIFT,          // Shift Data into the MCU
    CALCULATE,      // Calculate the position of the centroid
    UPDATE,         // Update display vectors
    SEND            // send data to the serial port
} state = INITIALISE;



#include "serial.h"
#include "encode.h"
#include "timer.h"
#include "data.h"
#include "display.h"


void lld_init(void);


/*
 * lld_init -
 *
 * initialises the laser level detector
 *
 */
void lld_init(void)
{

    /* blank out interrupts */
    outp(0x00, TIMSK);
    outp(0x00, GIMSK);
    outp(0x00, SPCR);
    outp(0x00, UCR);
    outp(0x00, ADCSR);
    outp(0x80, ACSR);

    /* set directions on IO ports */
    outp(0x00, PORTA);
    outp(0xFF, DDRA);
    outp(0x00, PORTB);
    outp(0xFC, DDRB);
    outp(0x00, PORTC);
    outp(0xFF, DDRC);
    outp(0x00, PORTD);
    outp(0x1A, DDRD);
    outp(0x03, PINA);
```

```
        ENABLE_INTERRUPTS();

        /* initialise drivers */
        serial init();
        display init();
        timer init();
        data init();
        encode_init();
    }
```

```
/*
 * FILE: lld.c Laser Level Detector State Machine
 *
 *       Thesis of Geoff Cook, 2002
 *
 *       School of Information Technology and Electrical
 *       Engineering.
 *
 *       (Computer Systems Engineering)
 */

#include "lld.h"

int main (void) {

    for (;;) {
        switch (state) {

            /*
             * INITIALISE - initialises the device
             */
            case INITIALISE: {

                lld_init();

                state = HELLO;
                break;
            }

            /*
             * HELLO - displays "hello" message
             */
            case HELLO: {

                delay ms(500);
                display message("HELO");
                delay ms(1000);
                state = DETECT;

                break;
            }

            /*
             * DETECT - detects the number of modules and displays on 4x7seg
             */
            case DETECT: {

                data latch();
                BEGIN_DATA_SHIFT();

                while (state == DETECT);

                {
                    char number[2];
                    number[0] = (modules / 10) + 0x30;
                    number[1] = (modules % 10) + 0x30;

                }

                DISPLAY_REFRESH();

                display char('-');
                display char((modules / 10) + 0x30);
                display char((modules % 10) + 0x30);
                display char('-');
                delay ms(1000);
                state = BEGIN;
                break;
            }

            /*
             * BEGIN - LLD device function begins here (dummy state)
             */
            case BEGIN: {
                state = LATCH;
                break;
            }

            /*
             * LATCH - Latch detector data into shift registers
             */
            case LATCH: {
```

```
                data_latch();
                state = SHIFT;
                break;
            }

            /*
             * SHIFT - Shifts the data into the MCU
             */
            case SHIFT: {

                BEGIN_DATA_SHIFT();
                state = WAIT;
                break;

            }

            /*
             * WAIT - waits until shifting completes (state changed by interrupt routine)
             */
            case WAIT: {
                break;
            }

            /*
             * CALCULATE - calculates the position of the beam from the data
             */
            case CALCULATE: {

                current_value = 0;
                unsigned int position = 0;
                unsigned int number_on = 0;

                for (position = 0; position < (48 * modules); position++) {

                    unsigned short micro_position = position % 8;
                    unsigned short macro_position = position / 8;
                    unsigned short this_value = (data[macro_position] & (1 << micro_position))
                                                    >> micro_position;
                    if (this_value == 1) {
                        current_value = current_value + position + 1;
                        number_on++;
                    }

                }

                current_value = 2 * current_value / number_on;

                if ((current_value - ((int) current_value)) > 0.5)
                    current_value++;

                if (number_on == 0) current_value = 0;

                state = UPDATE;

                break;
            }

            /*
             * UPDATE - updates the display vectors
             */
            case UPDATE: {

                display_number((int) current_value);
                state = SEND;
                break;
            }

            /*
             * SEND - sends information to the serial port
             */
            case SEND: {
                state = BEGIN;
                if (SER_CLEAR_TO_SEND()) output_add_byte(0xAA);
                break;
            }
        }
    }
}
```

```
/*
 * FILE: serial.h - Serial driver for the laser level detector
 *
 *       Thesis of Geoff Cook, 2002
 *
 *       School of Information Technology and Electrical
 *       Engineering.
 *
 *       (Computer Systems Engineering)
 */

#define ENABLE_INTERRUPTS()     sei()
#define DISABLE_INTERRUPTS()    cli()

#define BAUD_19200          24
#define TXCIE               0x40
#define UDRIE               0x20
#define TXEN                0x08

#define DATA_DIRECTION_D    0x1A

#define SERDAT              0x02
#define SERRTS              0x04
#define SERCTS              0x08

#define OUT_CONTROL_READY   1
#define OUT_CONTROL_WAIT    0

#define DISABLE_DRIE()      outp(inp(UCR) & (0xFF - UDRIE), UCR)
#define ENABLE_DRIE()       outp(inp(UCR) | UDRIE, UCR)

#define SER_CLEAR_TO_SEND() ((inp(PIND) & SERCTS) == SERCTS)

void serial_init(void);
void serial_putch(unsigned short);

volatile unsigned short output_control;

/*
 * uart data register empty interrupt -
 */
SIGNAL(SIG_UART_DATA)
{
    DISABLE_DRIE();
}

/*
 * serial_init -
 *
 * initialises serial communications device driver
 */
void serial_init(void)
{
    /* set baud rate */
    outp(BAUD_19200, UBRR);

    /* set up interrupts */
    outp(UDRIE | TXEN, UCR);

    /* set up data direction of port d */
    outp(DATA_DIRECTION_D, DDRD);

    /* request to send */
    outp(SERCTS, PORTD);

    /* initialise controller */
    output_control = OUT_CONTROL_READY;

    DISABLE_DRIE();

}


/*
 * serial_putch -
 *
 * writes a character to the serial port
 */
void serial_putch(unsigned short the_value)
{
    outp(the_value, UDR);
}
```

1

```
/*
 * FILE: encode.h - run length encoding variation for lld.
 *
 *       Thesis of Geoff Cook, 2002
 *
 *       School of Information Technology and Electrical
 *       Engineering.
 *
 *       (Computer Systems Engineering)
 */

#define ENCODE_BUFFER_SIZE  3


unsigned short bit stream[ENCODE_BUFFER_SIZE];          // output bit stream
unsigned short bit counter = 0;
unsigned short byte counter = 0;
unsigned short consecutive_bytes;

unsigned short encode byte;                             // ecoder byte
unsigned short encode bit_counter;
unsigned short all_zeros;


/*
 * encode_init -
 *
 * initialises encoder
 */
void encode_init(void) {

    int i;

    encode_byte = 0;

    for (i = 0; i < ENCODE BUFFER SIZE; i++) {
        bit_stream[byte_counter] = 0;
    }

    all_zeros = 0;

}


/*
 * output_add_bit -
 *
 * Packs output bits LSB first into bytes to be sent to serial.
 * If a byte is full, it is sent to the serial port.
 */
void output_add_bit(unsigned short the_bit) {

    bit stream[byte counter] |= (the bit << bit_counter);
    bit_counter = (bit_counter + 1) % 8;

    if (bit counter == 0) {
        serial putch(bit stream[byte counter]);
        bit stream[byte counter] = 0;
        byte_counter = (byte_counter + 1) % ENCODE_BUFFER_SIZE;
    }

}


/*
 * output_add_byte -
 *
 * packs in bytes LSB first
 */
void output_add_byte(unsigned short the_byte) {

    short i;

    for (i = 0; i < 8; i++) {
        output_add_bit((unsigned short) (the_byte << bit_counter));
    }

}


/*
 * encode_low_bit -
```

```
 *
 * encodes the bit to the low word and sends it to the output stream
 * LOW BIT MUST BE WRITTEN FIRST!!
 *
 */
void encode_low_bit(unsigned short the_bit) {

    encode_byte |= (the_bit << encode_bit_counter);

    if (encode bit counter == 0) all zeros = 1;         // new byte
    if (the_bit != 0) all_zeros = 0;                    // not a zero!!

}


/*
 * encode_high_bit -
 *
 * encodes the bit to the high word and sends it to the output stream
 * LOW BIT MUST BE WRITTEN FIRST!!
 *
 * high bit written last - therefore complete byte is sent off in this function
 */
void encode_high_bit(unsigned short the_bit) {

    encode_byte |= (the_bit << (encode_bit_counter + 4));

    if (the_bit != 0) all_zeros = 0;                    // not a zero!!

    if (encode bit counter == 3) {                      // byte complete!
        if (all zeros == 1) {
            output_add_bit(0);                          // all zeros - out a zero
        } else {
            output add bit(1);                          // not all zeros - out a 1,
            output_add_byte(encode_byte);               // followed by byte
        }
        encode_byte = 0;
    }

    encode_bit_counter = (encode_bit_counter + 1) % 4;
}
```

```
/*
 * FILE: data.h - data input and storage for the laser level detector
 *
 *       Thesis of Geoff Cook, 2002
 *
 *       School of Information Technology and Electrical
 *       Engineering.
 *
 *       (Computer Systems Engineering)
 *
 */

/*
 * PCU PORTB bit masks
 */
#define MASK LDDLWRD        0x01
#define MASK LDDHWRD        0x02
#define MASK SRCLK          0x04
#define MASK_SRPL           0x08

/*
 * Modules
 */
#define MAX MODULES         20                          // Max Modules per LLD
#define DD_PER_MODULE       48                          // Detectors per podule

#define COUNT2_RELOAD       190                         // affects SRCLK speed

/*
 * enables/disables timer2 generated pulsing of SRCLK
 */
#define BEGIN DATA SHIFT()      outp((inp(TIMSK) | 0x40), TIMSK)
#define STOP_DATA_SHIFT()       outp((inp(TIMSK) & 0xBF), TIMSK)

/*
 * reads LDDLWRD and LDDHWRD
 */
#define LDDLWRD()               ((inp(PINB) & MASK LDDLWRD) == 0x00)
#define LDDHWRD()               (((inp(PINB) & MASK_LDDHWRD) >> 1) == 0x00)

/*
 * edge, toggle, set macros for SRCLK and SRPL
 */
#define SRCLK RISING()          ((inp(PORTB) & MASK SRCLK) == MASK SRCLK)
#define SRCLK SET HIGH()        outp(inp(PORTB) | MASK SRCLK, PORTB)
#define SRCLK TOGGLE()          if ((inp(PORTB) & MASK SRCLK) == MASK SRCLK)        \
                                    outp(inp(PORTB) & (0xFF - MASK SRCLK), PORTB);  \
                                else outp(inp(PORTB) | MASK SRCLK, PORTB)
#define SRPL SET LOW()          outp(inp(PORTB) & (0xFF - MASK SRPL), PORTB)
#define SRPL SET HIGH()         outp(inp(PORTB) | MASK_SRPL, PORTB)
#define INC_SHIFT_COUNTER()     shift_counter++


void read_data (void);

unsigned short bits zero = 0;                           // used for detecting number of modules
unsigned int bits counted = 0;                          // how many bits counted since latch
unsigned short modules = 0;                             // number of modules
unsigned short data ready = 0;                          // has data finished shifting?
unsigned char data[MAX MODULES * DD PER MODULE / 8];    // array to store data into
unsigned int current_value;                             // used in calculation of position

/*
 * this overflow will be used to shift in data
 *  - increments counter on rising edge
 *  - stores data on falling edge
 */
SIGNAL(SIG_OVERFLOW2)
{
    outp(COUNT2_RELOAD, TCNT2);

    SRCLK_TOGGLE();

    if (state == DETECT) {
        /*
         * Detecting the number of modules!!
         * Want 47 consecutive zeros to determine how
         * many modules are connected.
         */
        if (!SRCLK_RISING()) {

            if (LDDLWRD()) {
```

```
                bits_zero ++;
            } else {
                bits_zero = 0;
            }
            bits_counted++;

            if (bits_zero == 47) {
                /*
                 * 47 consecutive! Proceed to next state.
                 */
                modules = (int) ((bits_counted - 24) / 24);
                state = BEGIN;
            }
        }

    } else {

        if (!SRCLK_RISING()) {
            /*
             * Normal SRCLK operation. Each time store a new bit into the array.
             */

            unsigned short hmicro position = bits counted % 8;          //position within byte
            unsigned short hmacro position = bits counted / 8;          //position within array
            unsigned short lmicro position = (bits counted + 4) % 8;    //position within byte
            unsigned short lmacro_position = (bits_counted + 4) / 8;    //position within array

            /*
             * Stores data into the array using positions defined above.
             */
            if (LDDLWRD()) {
                data[lmacro position] = data[lmacro_position] | (1 << lmicro_position);
                encode_low_bit(1);
            } else {
                data[lmacro_position] = data[lmacro_position] & (0xFF - (1 << lmicro_position))
;
                encode_low_bit(0);
            }

            if (LDDHWRD()) {
                data[hmacro position] = data[hmacro_position] | (1 << hmicro_position);
                encode_high_bit(1);
            } else {
                data[hmacro_position] = data[hmacro_position] & (0xFF - (1 << hmicro_position))
;
                encode_high_bit(0);
            }


            bits_counted++;

            /*
             * Count bits 0 and 4, 1 and 5, 2 and 6, 3 and 7 simultaneously. Therefore
             * Jump 4 bits to count 8 and 12, 9 and 13, 10 and 14, 11 and 15
             *
             * 2 four-bit words are filled simultaneously
             */
            if ((bits_counted % 4) == 0) bits_counted = bits_counted + 4;

            /*
             * reached end of module!!
             */
            if (bits counted > (48 * modules)) {
                state = CALCULATE;
                STOP_DATA_SHIFT();
            }
        }
    }
}


void data_init(void) {
    /*
     * Set up Timer 2 to count and overflow
     */
    outp(0x01, TCCR2);

    /*
     * Get first interrupt ready
     */
    outp(COUNT2_RELOAD, TCNT2);
```

```
}

/*
 * data_latch -
 *
 * latches data into the shift registers (SRPL)
 */
void data_latch(void) {

    int i;

    /*
     * Toggle the latch
     */
    SRPL_SET_LOW();
    SRCLK_TOGGLE();

    bits_counted = 0;

    /*
     * Clear the data - occurs while latch is low
     */
    for (i = 0; i < (8 * MAX_MODULES); i++) data[i] = 0x00;

    /*
     * Toggle again
     */
    SRCLK_TOGGLE();
    SRPL_SET_HIGH();

    /*
     * First piece of data is available immediately - store!
     */
    if (LDDHWRD()) data[0] = 0x01;
    if (LDDLWRD()) data[0] = data[0] | 0x10;
    bits_counted++;
}
```

```
/*
 * FILE: timer.h - An approximate 'milliseconds' timer for the LLD
 *               - Assumes a 7.68MHz Clock
 *
 *       Thesis of Geoff Cook, 2002
 *
 *       School of Information Technology and Electrical
 *       Engineering.
 *
 *       (Computer Systems Engineering)
 */
#define     DIVISOR 1           0x01
#define     MS HIGH LOAD        0xE2
#define     MS LOW LOAD         0x00
#define     MASK TOIE1          0x04
#define     TOIE1 ENABLE()      outp(inp(TIMSK) | MASK TOIE1, TIMSK)
#define     TOIE1_DISABLE()     outp((inp(TIMSK) & (0xFF - MASK_TOIE1)), TIMSK)

#define     MS RELOAD()         outp(MS HIGH LOAD, TCNT1H);\
                                outp(MS_LOW_LOAD, TCNT1L)

volatile unsigned int milliseconds = 0;

/*
 * timer overflow interrupt 1
 *
 * executed every 1ms when interrupt is enabled.
 */
SIGNAL(SIG_OVERFLOW1) {

    if (milliseconds > 0) milliseconds--;

    MS_RELOAD();
}


/*
 * timer_init -
 *
 * initialises timer 1.
 */
void timer_init(void) {

    outp(0x00, TCCR1A);
    outp(DIVISOR 1, TCCR1B);
    outp(0x00, ICR1H);
    outp(0x00, ICR1L);

    TOIE1_DISABLE();
}

/*
 * delay_ms
 *
 * suspends for 'milliseconds' ms.
 */
void delay_ms(unsigned int time) {

    milliseconds = (unsigned int) (time);

    TOIE1_ENABLE();

    while (milliseconds > 0);

    TOIE1_DISABLE();
}
```