The Design of an Electric Fence Fault-Finder

A thesis submitted in partial fulfilment of the requirements for the degree of Master of Engineering in Computer Systems Engineering at Massey University, Albany, New Zealand

Glen McGillan

2009

Abstract

Electrified fencing is commonly used throughout the world to control animals with smaller and cheaper fence constructions than would otherwise be necessary with non-electrified wires. Typical installations have a long wire or wires starting from an electric fence energiser and then surrounding fields in various complex configurations. Faults on electric fences can be difficult to locate, with the average fence using tens of kilometres of wire.

Basic fault-finding tools allow an operator to read the peak fence voltage, requiring the user to decide whether a fault is present and to randomly search for the source of the problem. The focus of this thesis is to develop a device that reduces the time to locate faults on a fence by providing more information about the location and nature of a fault, and will point in the direction of the fault.

Acknowledgements

I would like to thank my supervisor Dr Tom Moir for his support and assistance during this research.

Thank you Craig Baguley for your inspiration, ideas and assistance throughout the project.

Mostly I am grateful to my wife Rusul for the motivation and encouragement to see this to the end.

Table of Contents

List of Figures

List of Tables

Symbols and Abbreviations

F	Frequency [s^{-1}]
Hz	Hertz [s^{-1}]
kHz	Kilohertz [s^{-1}]
MHz	Megahertz [s^{-1}]
E	Stored Energy [J]
L	Inductance [H]
H	Henry
mH	MilliHenry
C	Capacitance [F]
Q	Capacitor charge
F	Farad
pF	PicoFarad
nF	NanoFarad
µF	MicroFarad
G	Conductance
R	Resistance [Ω]
V	Voltage
V_{out}	Voltage Output
V_{in}	Voltage Input
mV	Millivolt
kV	Kilovolt
V_p	Peak Voltage
V_{RMS}	Voltage, Root Mean Square
I	Current [A]
I_{out}	Current Output [A]
I_{in}	Current Input [A]
A	Amp
µA	MicroAmp
mA	MilliAmp
I_p	Peak Current

t	Time [s]
s	Second
ms	MilliSecond
µs	MicroSecond
km	Kilometre
m	Metre
cm	Centimetre
h	Height [m]
r	Radius [m]
A	Area [m^2]

1

Chapter 1

Introduction

1.1 Research Outline

All over the world, electric fencing is used to control animals. When operating properly, electric fences are very effective; however, a single fault can disable an entire fence system. Because electric fences are designed to be a psychological rather than physical barrier, they will not contain animals while faulty.

Depending on the fault, it can take hours or even days to locate the problem, during which time stock may be lost, property damaged, or herds mixed.

This thesis aims to develop a tool that reduces the time required to locate electric fence faults, maximising the users productivity and minimising losses.

1.2 Initial Requirements and Objectives

This project was partially sponsored by a company that is attempting to gain access into the electric fencing market. At the time of writing this thesis, the company has not publicly declared their intention to do so, and has requested to remain anonymous.

The desired outcome of this project is a commercially ready fault-finding tool (referred to as the device or fault-finder throughout this thesis). Due to the competitive nature of the fencing market and a large choice of fencing tools already available, the device must be more desirable than similar offerings, have better or new features, and be distinctive to make it very marketable.

Electric fence fault-finding accessories vary greatly in price and usefulness. Simple neon indicators that flash with each fence pulse can be purchased for a few dollars, ranging up to monitoring systems with dozens of remote sensors that call a cell phone once an irregularity is discovered, costing thousands of dollars. The device created must exist between these two extremes, providing all the functionality required to quickly locate faults while cheap and simple enough to use that every farmer will be willing to purchase it.

The company has identified that a type of portable hand-held product that read both fence voltage and current are the single most useful tool available for fault-finding electric fences, and therefore a similar or better device is to be developed.

1.3 Thesis Scope

This thesis presents the development process of the hardware and software of an electric fence fault-finder that can measure voltages and currents present in a fence system and indicate the direction to a fault (when present). The outcome will be a commercially ready product that operates reliably and consistently, is cost effective, and suitable for mass production.

A search for similar products already on the market has revealed three major manufacturers. Paul Thompson conceived the original fault meter concept of incorporating voltage and current measurements to aid fault-finding in 1995, and later formed the company Pakton which now sells the design.[1] Later, both Tru-Test and the Gallagher Group released similar devices. Each device is shown in Figure 1-1 (from left to right – Gallagher SmartFix, Stafix Fence Compass, and the Pakton Electric Fence Power Probe II).

Figure 1-1: Similar competitor products

Some energisers, such as the Smart Power MX7500 from Gallagher (see Figure 1-2) monitor their own output and earth voltages, providing

warning when a possible problem is detected. These systems do not indicate all types of fault, such as a broken wire, and do not assist in locating the fault.

Figure 1-2: Smart Power MX7500 energiser with self-monitoring

Patent number 331366, titled “Fault Meter”, was filed in 1999 by the Gallagher Group and lapsed in 2001 after opposition by Tru-Test. The patented device is described as “A method of detecting a fault in an electrical system”,[2] and covers any device used to measure both voltage and current magnitude and polarity for the purpose of fault-finding.

Another patent (number 501475) titled “A Detector Device” was filed in 2000 by Tru-Test and is in force until 2020. The device is given as “A fault-finder device which has particular application with electric fencing”.[3] The patent does not attempt to cover the general principle of fault-finding via voltage and current measurements, but instead the method of using audible indications of magnitude, direction and fault status.

Several papers have been written that touch on the electrical effects of faults in an electric fence such as Hancock[4], McKay[5] and Thrimswithana[6]. A master’s thesis published in 1996 by Yit Moon Ho

of Waikato University titled “Design and Construction of a Pulse Current Meter”, describes the development of a fence current meter, and was the most similar prior work discovered.[7] The research presented formed the initial basis of the Gallagher SmartFix current sensing circuit, however, the technology researched in that thesis was not used in the final product.

The chapters of this thesis are arranged as follows:

Chapter 2 – Electric fencing history and practices are presented, focussing on energisers and common faults encountered in fencing systems.

Chapter 3 – Electrical characteristics of fence faults are examined in detail along with industry fault-finding techniques. Using this information, the requirements of the project hardware and software are determined.

Chapter 4 – Details of the design and development of the project hardware are given, including several prototypes that were created as part of the engineering process.

Chapter 5 – The software requirements and design specification are listed, along with the expected behaviour of each software component.

Chapter 6 – Results from testing of the final device on farms and in the workshop are provided, and a final specification made based upon these results.

Chapter 7 – The outcomes of the project are summarised and conclusions drawn. Suggestions for future improvements are recommended.

2

Chapter 2

Electric Fencing

2.1 Introduction

Electric fencing is used in many agricultural areas, as construction of electric fences can be much cheaper and faster than conventional fences (plain wire is used along with much lighter construction since the fence does not need to physically restrain animals). The risk of injury to livestock is lower compared to fences made of barbed wire or woven wire with large openings that can entangle the animals' feet.

The disadvantages of electric fencing include: the potential for the entire fence to be disabled due to a break in the conducting wire, shorting out to vegetation or other fence wires, power failure, forced disconnection due to the risk of fires starting by dry vegetation touching an electrified wire, lack of visibility, and the potential to shock an unsuspecting human passer-by.

New Zealand inventor Bill Gallagher developed one of the earliest electric fence units in 1936-1937. It was constructed from a meccano set and car ignition coil, and was used to protect his car from an itchy horse that loved to scratch itself, damaging the vehicle.[8] Around the

same time, American farmers were experimenting with connecting mains voltages directly to their fences. So many animals were killed by this practice that new laws were introduced that required all energisers be powered from batteries.

In 1962 a New Zealand inventor created the first “low impedance” energiser and the basic design has remained the same ever since. About once per second a capacitor charges to a preset voltage, then a thyristor rapidly discharges the capacitor through a transformer. The transformer steps up the output to a high voltage, which usually provides a very unpleasant or painful shock.

Electric fencing works by creating a "fear" barrier that uses a safe electric shock to deter animals from challenging the fence. When an animal touches the fence, a painful but safe shock is received (see Figure 2-1). The animal will remember the pain and will not attempt to break through the fence.

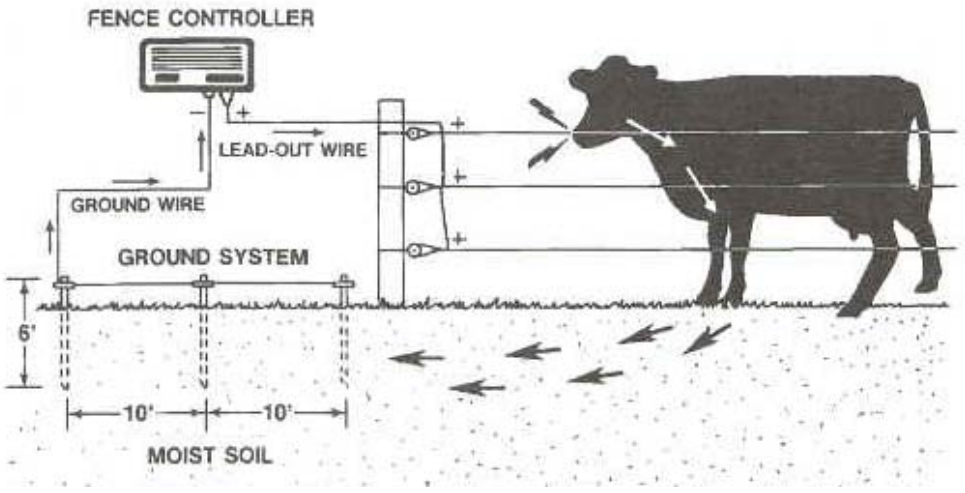


Figure 2-1: Animal completing the electric fence circuit path and receiving a shock[9]

In order for an animal to feel a shock, the voltage produced by the fence controller must be high enough to penetrate the animal's hair, hide, and hoof. Once the voltage is high enough to deliver a shock, current must

flow through the fence wire, then through the animal touching the fence and into the soil the animal is standing on. The current then travels through the moist soil back to the ground rods and through the ground wire that is attached to the fence controller's ground terminal. The circuit is completed and the animal feels the shock instantly (see Figure 2-2).

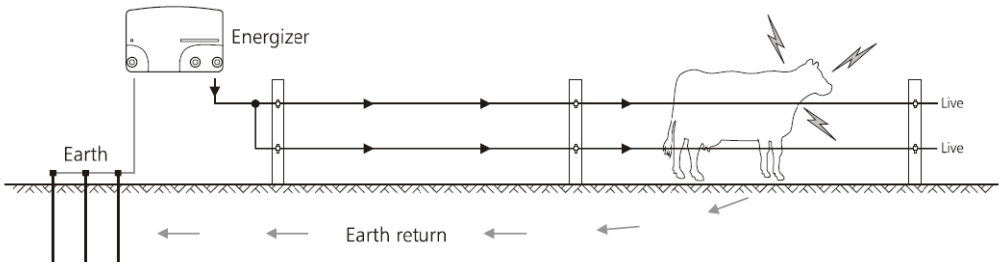


Figure 2-2: Typical "all live" electric fence[9]

When the ground return resistance is high, such as sandy soil, the animal may not receive a significant shock and an earth return wire is needed. In this case, the animal must touch both wires to receive a shock (see Figure 2-3).

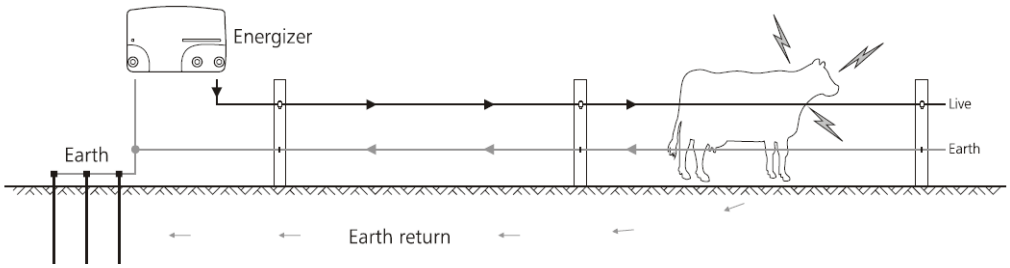


Figure 2-3: Electric fence installation using an earth return wire[9]

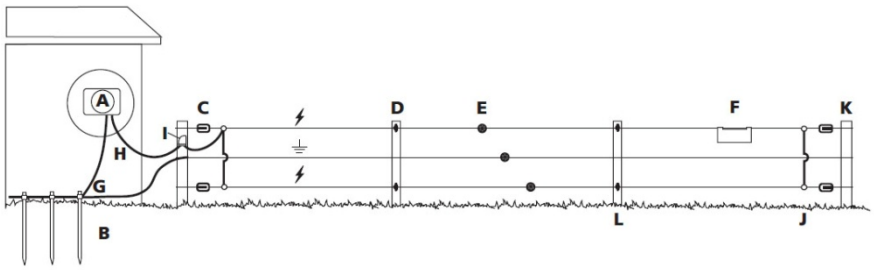
2.2 Fence Construction Types

There are many possible ways to construct an electric fence, beginning with a single wire offset from an existing fence to a large multi wire fence with all wires electrified.

The choice of fence construction is determined by the animal to be controlled, and the degree of certainty desired. A single wire fence will contain cows to a green pasture, but a seven-wire fence may be needed to separate a bull from those same cows. In India, reinforced chain-link electric fences are used to control elephants.

Fences may also be temporary, using flexible tape woven with steel wires and insulated posts. Often these fences are small and a battery-powered energiser is all that is needed.

The components and typical installation of an electric fence is shown below in Figure 2-4.



A	Energizer	D	Post insulator	G	Green	J	Joint clamps
B	Earth stake	E	Wire tightener	H	Red	K	Strain post
C	Strain insulator	F	Warning sign	I	Cut out switch	L	Line post

Figure 2-4: Fence construction typical components[10]

Some farmers believe that electric fence energisers are more effective when connected to the fence in reverse – the energiser negative terminal is connected to the live wires, and positive terminal to earth. This produces negative pulses on the fence that are otherwise identical to those found on an ordinary fence.

2.3 The Energiser

Electric fence energisers were first invented in the early 1930s and used a continuous high voltage DC or AC output that was current limited to 24mA for safety (below lethal threshold).[11] There were significant limitations with this design, since they failed to produce an effective shock during wet conditions or when high vegetation growth was present under the fence due to excessive leakage to ground.

Next generation energisers introduced in the late 1930s overcame these limitations by producing very short, high voltage, high current pulses. Shown in Figure 2-5, these were constructed with capacitive or inductive energy storage elements, along with mechanical switches and timing devices to periodically generate fence pulses. Over time, the switch contacts degraded, leading to irregular pulsing and failure.

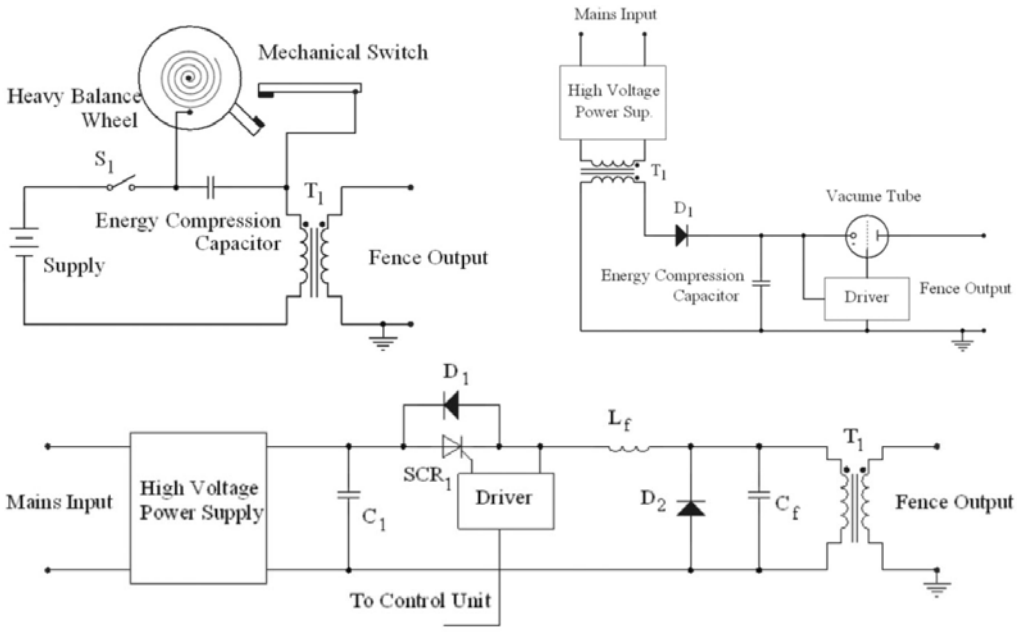


Figure 2-5: Second (top left), third (top right) and fourth generation (bottom) energiser designs[12]

A third generation of fence energisers, shown in Figure 2-5, replaced mechanical switches with vacuum discharge switches, and employed

electronic timing circuits. These suffered from poor performance on heavily loaded fences, as the series impedance of the vacuum discharge switch provided a high-impedance output to the fence.

A New Zealand scientist created the fourth and still current generation of energisers in the 1950s. Solid-state power devices charge capacitors, and then a silicon-controlled rectifier (SCR) discharges the capacitors through a step-up pulse transformer. This design provides long operational life, improved reliability, efficiency and temperature stability. Some modern energisers alter their output voltage / power to maintain a desired level of performance without providing excessive power.

2.3.1 Energiser Pulses

Safety standards place limits on energiser pulse characteristics to ensure people and animals are protected from lethal voltages and currents. There are three significant requirements relating to pulses:[13]

- The impulse repetition rate shall not exceed 1Hz
- The impulse duration of the impulse shall not exceed 10ms
- The energy/impulse in a 500Ω load shall not exceed 5 joules

The common electric fence energiser repetitively generates very short (50 - 500µs), high voltage (4 - 12kV), high power (1 – 5,000kW) pulses at a frequency of approximately 0.6 Hz. A typical example is shown in Figure 2-6.

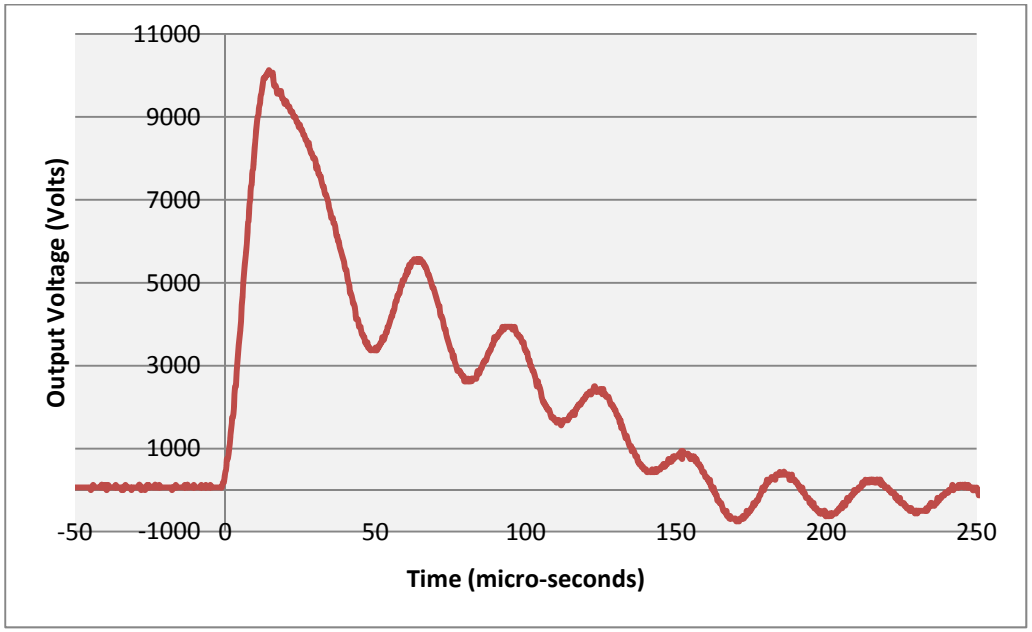


Figure 2-6: Typical unloaded energiser output pulse

2.3.2 Size of Energiser Required

Each electric fence installation is unique, making a definitive energiser selection guide impossible. Different fence materials, topologies, condition, construction style, and wire length can cause unexpected changes in energiser load. In addition, energiser performance with varying loads changes substantially between manufacturers.

It is generally accepted by professional electric fence installers that a good rule of thumb is that each joule of output energy can power 10,000 metres of fence wire. Therefore, a 12J energiser can realistically power about 120km of fence wire.

2.4 Common Electric Fence Faults

Faults in the fence reduce its effectiveness and may cause other problems, such as interference on telephone lines or internet connections.

Causes for faults include: [6, 9, 10, 14, 15]

- Vegetation growth touching the live wires
- Broken wires
- Cracked or burnt insulators
- Poor earthing
- Corroded metals / poor connections
- Over extended energiser
- Flat energiser battery
- Weather conditions – snow, rain, fog, strong winds, cold

2.4.1 Vegetation Growth

Vegetation under the fence will grow over time. Often animals will not eat plants from below the fence, allowing the plants to grow indefinitely, with many grasses and weeds easily reaching one metre in height (see Figure 2-7).

Plants typically have high moisture content, and in sufficient quantity provide a low impedance path to ground. This impedance can reach as low as 20Ω, effectively a dead short.[14] The problem is made worse during fog, rain and snow when the plants will hold additional water and provide even lower impedance to ground.

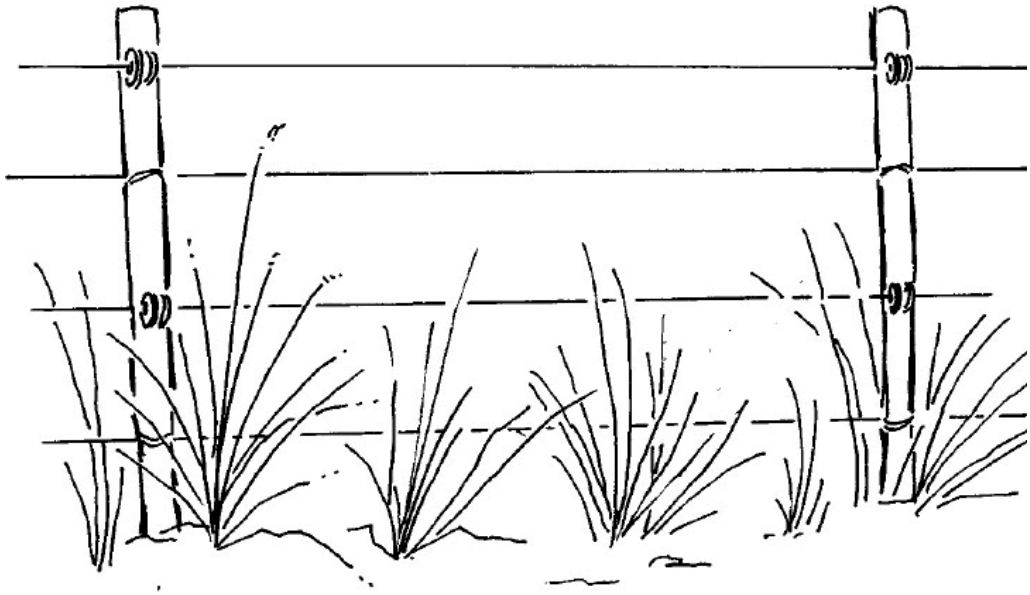


Figure 2-7: Vegetation growth in contact with live wires [15]

At any particular moment, the vegetation under a fence presents a constant impedance to ground. This attenuates the energiser pulse by loading the energiser output, reducing the voltage seen on the fence (see Figure 2-8).

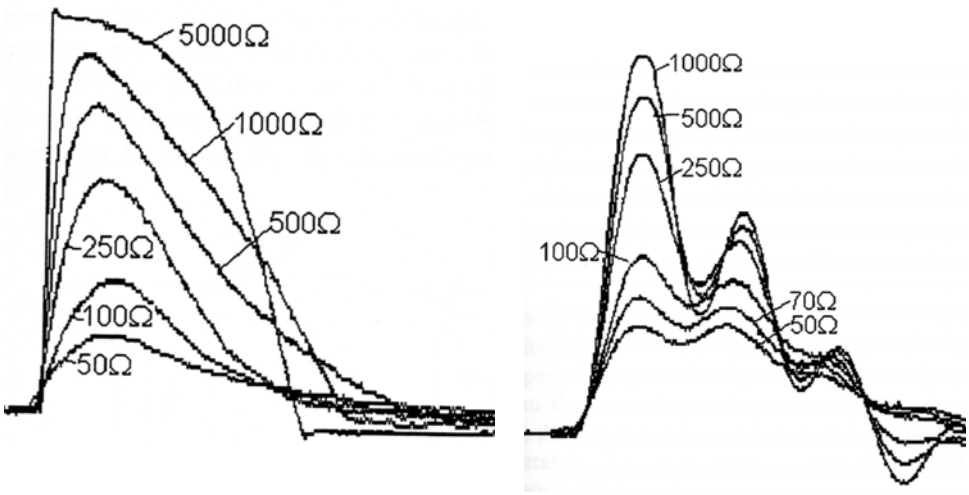


Figure 2-8: Small (0.1J) and large energiser (11J) output waveforms with increasing load[16]

2.4.2 Broken Wires

When a live wire breaks, all sections of the fence after the break are completely without power. In addition, the broken wire may fall across non-electrified wires (see Figure 2-9), causing unexpected shocks along the fence and grounding the live wire, heavily loading the energiser. This additional load could cause the entire fence to fail.

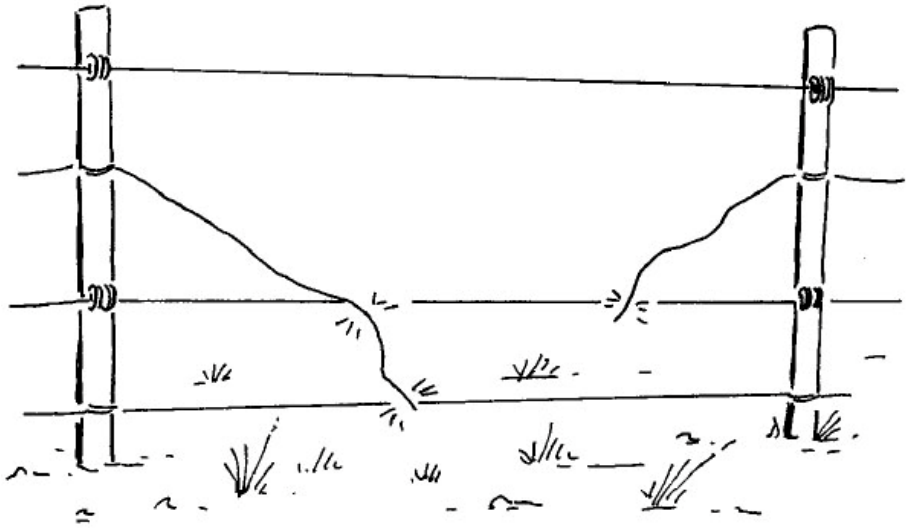


Figure 2-9: Broken wire shorting live wire[15]

If a non-electrified wire breaks and falls across one or more electrified wires, a similar effect will be seen as that in the above paragraph except the entire fence will still be electrified.

A broken wire can be detected by either an absence of voltage and current on the subsequent fence wires after the break (and normal voltage levels before the break), or a reduced voltage on the entire fence, particularly after the break.

2.4.3 Cracked or Burnt Insulators

Insulators prevent live electric fence wires coming into contact with fence posts and other non-electrified wires. Most fence posts are made from wood, and when wet have very low impedance to ground. Steel posts are very good conductors and require insulators. Some fences use plastic or fibreglass posts which are inherently insulating and do not require installation of insulators.

Typical modern insulators are moulded from UV stabilised plastic and provide several centimetres clearance from the wire to the post. Older insulators are made from a variety of materials including ceramic, porcelain and rubber.

If an insulator is cracked, chipped, deformed or otherwise damaged during installation or from subsequent wear, the clearance between the live wire and fence post is reduced. When reduced enough, a spark will jump from the wire to the post with each fence pulse, reducing the effectiveness of the fence. Additionally, with each spark carbon is deposited, laying a conductive path over the surface of the insulator that supports arcing earlier and more often.

It has been observed that bird droppings have a particularly high mineral content, and can significantly increase the distance across which a spark will form. Informal lab tests showed a 250% increase in spark length (from 10mm to 25mm with a $10kV_{peak}$ fence pulse). It is common for birds to sit on fence posts and leave droppings smeared down posts and insulators. Therefore, an insulator that is not damaged and has been properly installed can cause a fence to fail for several days, and then return to normal without a trace of what caused the original failure.

Another problem associated with insulators is wires sagging because the insulator supporting the wire has failed completely (see Figure 2-10). This can allow a live wire to come into contact with neighbouring wires, producing problems similar to those outlined for broken wires in 2.4.2.

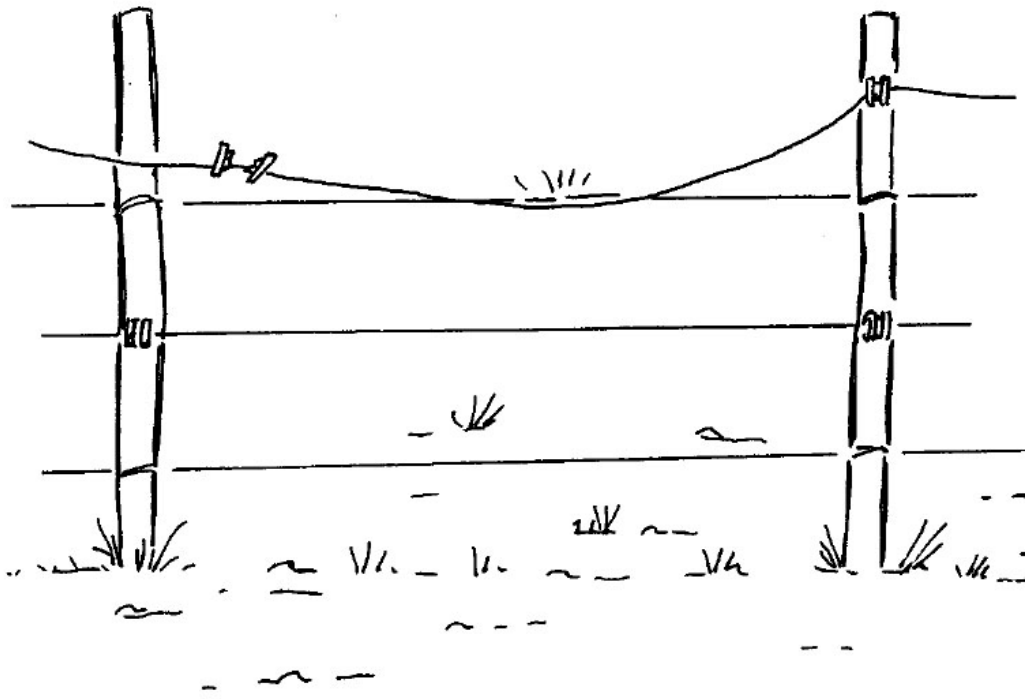


Figure 2-10: Broken insulator allowing live wire to sag[15]

A damaged insulator can present in several ways when attempting to locate the fault. The overall fence voltage may be lowered by a constant impedance to ground, or a section of the fence could be without fence pulses due to a short to ground. Finally, the fence pulse may arc to a post, causing reduced voltage on the fence and possibly increasing the amount of radio interference produced by the fence (due to rapid discharges as a result of arcing).

2.4.4 Poor Earthing

Inadequate earthing is the most common problem in electric fencing. The earth is half the circuit of a fencing installation, and must be as conductive as possible to produce an effective shock.

The size of the fence and the energiser used will determine how large an earth return system must be. Table 2-1 provides a guide to the number of earth stakes needed for a fence. The number of earth stakes will vary depending on the power of the energiser and the soil type: high-powered energizers need more stakes than low powered energisers; dry soils need more stakes than wet soils.

Energiser Size	Earth Stakes Required
Up to 15J	3 Stakes minimum
Up to 25J	5 Stakes minimum
Up to 35J	7 Stakes minimum

Table 2-1: Number of earth stakes required[10]

Weather often changes the efficiency of the earth return system, as the conductivity of the soil typically decreases as the soil dries. After a period of hot dry weather, a fence may become ineffective. Some countries such as Australia have a permanent earth problem, requiring a fence construction that includes earth return wires. If the earth return wire breaks, animals touching live wires will not receive a significant shock (see Figure 2-11).

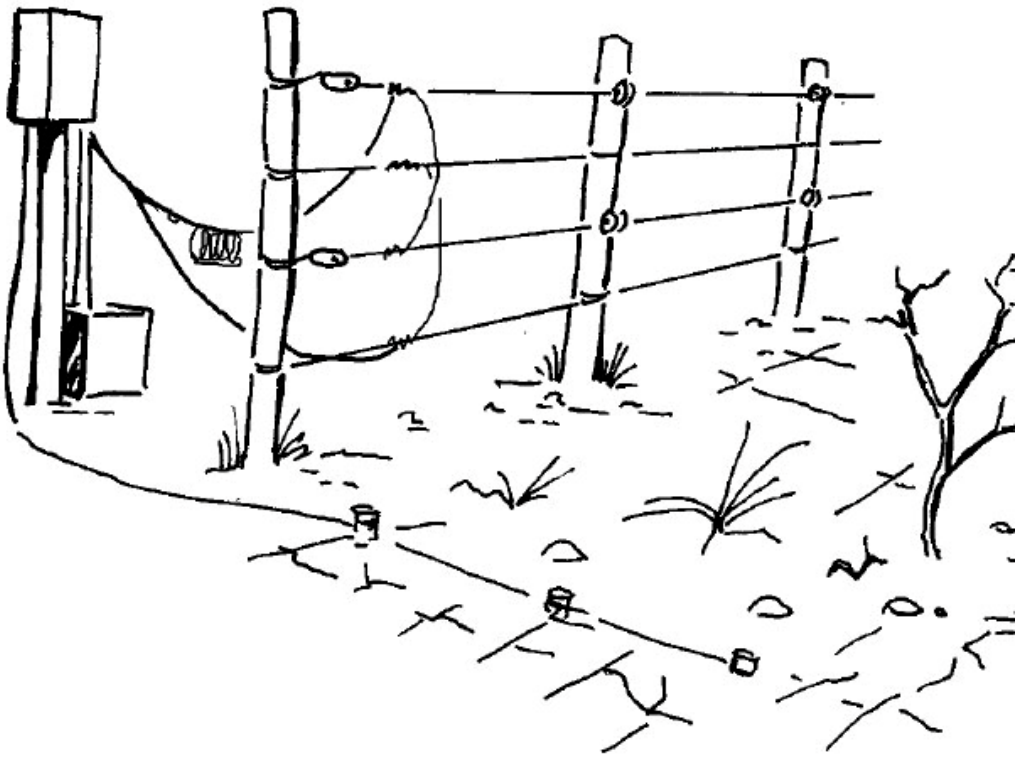


Figure 2-11: Insufficient earth return system[15]

If an earth return system has high impedance, a voltage of 0.3kV or greater[9] will be present with each fence pulse (enough to produce a shock when an earth stake is in contact with skin). Another symptom will be a low shock sensation when touching a live fence wire.

2.4.5 Corroded Metals / Poor Connections

Wires used in electric fence construction are made from mild or high tensile steel with a galvanised coating. Steel is used not only for its strength, but also because it is a good conductor. The disadvantage of steel is that due to its high iron content the wires are prone to rusting.

If an animal comes into contact with a rusty electrified wire, the rust will insulate the animal from the fence pulse and can greatly reduce the

effective shock that the animal feels. In addition, rusted wire joints are more likely to fail, producing a high-impedance junction between the two wires or localised arcing.

Certain types of knots are recommended when joining wires because they maximise surface area contact and are self-tightening when under tension. See Figure 2-12 for examples of bad knots for a guide to good knots to use.

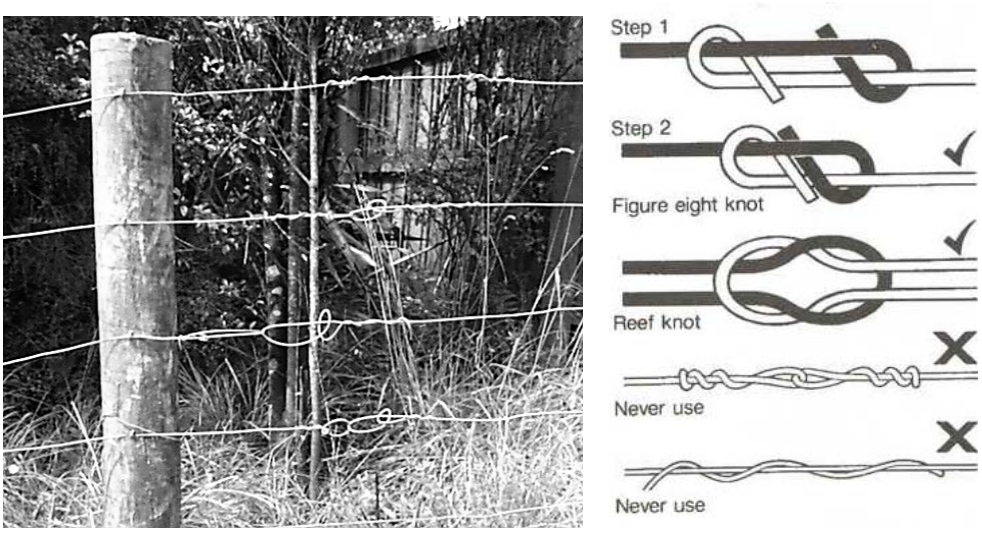


Figure 2-12: Electrical joints held together with tension (bad knots - left), good and bad joining knots guide (right)

Rusty wires are obvious because of their distinct colour and should always be replaced. Corroded joints are not as apparent and can be detected by a reduction or absence of voltage after the joint.

2.4.6 Over Extended Energiser

Most electric fence installations grow over time – old fences are upgraded / replaced, new paddocks are created, and existing fences extended. As the fence grows larger, it presents a greater load to the energiser. Eventually the energiser will not have enough power to

electrify the fence and either a larger energiser is required, or the fence can be divided in half and an additional energiser attached. To electrify a large farm as in Figure 2-13, several energisers must be used throughout the area.

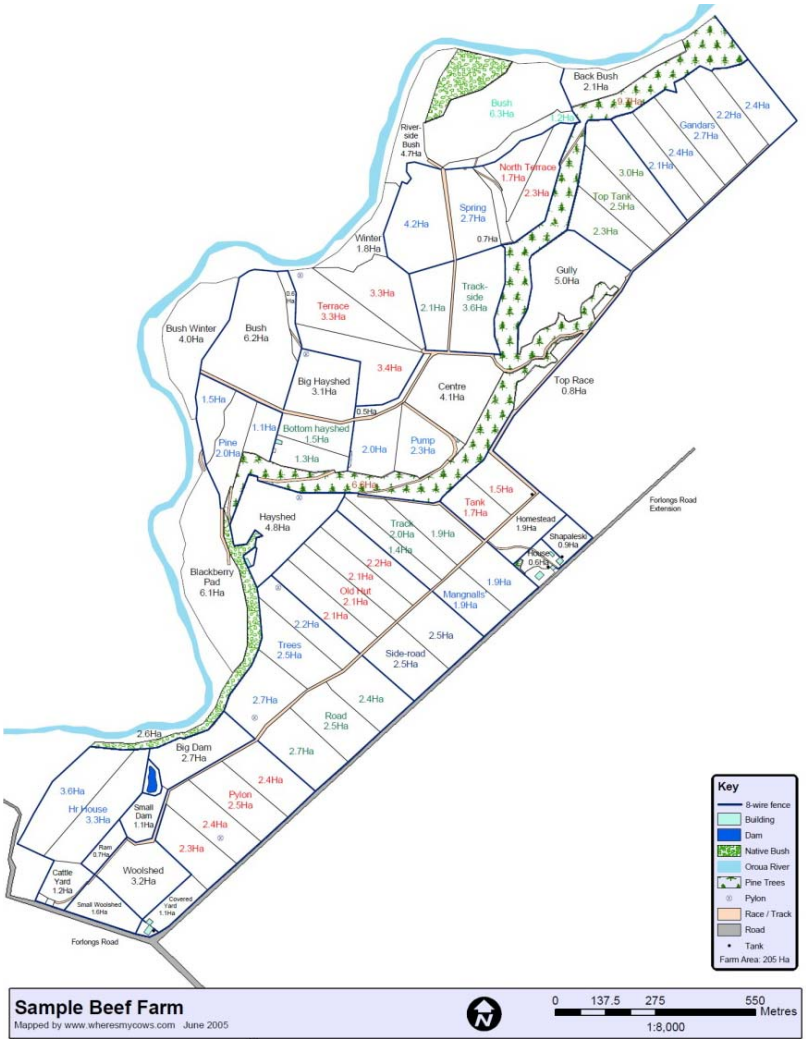


Figure 2-13: Aerial map of large farm

Sometimes an energiser that has powered a fence without problem for years is no longer able to maintain an adequate voltage. If none of the other problems discussed here are present, then it may be that the energiser itself is gradually failing, or its internal components are ageing and no longer operate at their designed efficiency.

The fence pulse voltage will be low when an energiser is over extended, while the current values will be normal or slightly lower than usual.

2.4.7 Flat Energiser Battery

As described earlier, energisers store energy in a capacitor then rapidly discharge that energy onto the fence through a transformer. When the energiser is operating normally, the charging circuit will be off for some of the energiser period (energiser pulses must be at least one second apart) because the capacitor is fully charged in less time than the energiser period. However, if the battery cannot supply sufficient current to fully charge the energisers capacitor(s), then the voltage of the energiser pulse is proportionally reduced (see Figure 2-14).

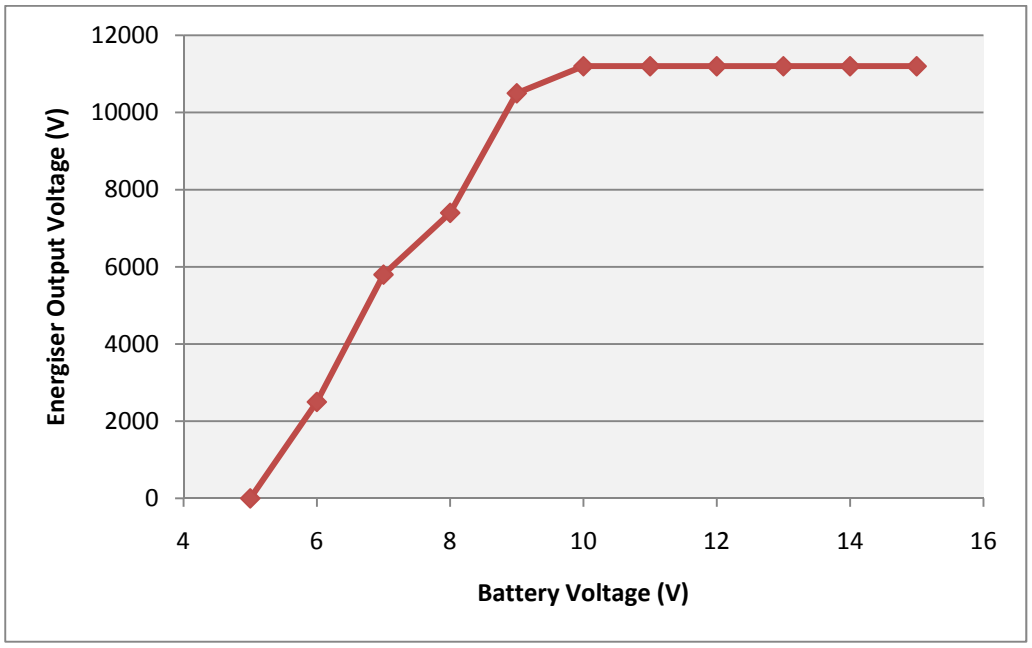


Figure 2-14: Energiser output voltage compared to battery voltage (O'Briens 52/36 battery energiser)

The amplitude of the energiser pulse gradually reduces as the battery discharges, until there is insufficient power to operate the energiser circuitry and the pulses cease altogether.

An energiser with a flat battery (if still operating) appears similar to an over extended energiser; however, the current flowing in the fence will be noticeably lower, especially as the energiser output voltage decreases.

2.4.8 Weather Conditions

Seasonal changes significantly affect the operation of an electric fence. Dry, hot weather can reduce the conductivity of the ground, creating a poor earthing situation as described in 2.4.4. Wet weather can increase the load on the fence through vegetation as per 2.4.1. Cold weather can increase the fence load; form snow and ice to short between live and earth wires, and even break wires or pull fences down (see Figure 2-15).

Figure 2-15: Ice build-up on fence wires

Storms can bring branches and other debris onto the fence, floods may short wires to earth. Wind can bring vegetation closer to the fence, causing intermittent periods of low fence voltage.

In most instances, extreme weather conditions will make an existing problem worse. A good example is an inadequate earthing system that works well in winter when the ground is wet, but poorly in late summer. Sometimes fences are permanently damaged by weather, such as a storm felled tree knocking over a fence section.

All the fence problems caused by weather can be found by searching for any of the other fault conditions already mentioned previously in sections 2.4.1 – 2.4.7.

3

Chapter 3

Fault-Finding Electric Fences

3.1 Introduction

The purpose of this chapter is to determine what electric fence parameters must be measured in order to determine whether a fence is in good working condition, or if repairs are needed. Electrical characteristics of each fault type discussed in chapter 2 will be examined, along with typical fault-finding methods currently used in electric fencing.

From the above information, a set of requirements will be determined that will guide initial development of the devices hardware and software.

3.2 Electrical Characteristics of Faults

The following section describes the electrical characteristics of each of the fault types previously outlined in section 2.4. It is important to know

the effect of each type of fault to determine what parameters the fault-finder must measure.

Fence voltage and current waveforms were measured in the presence of each type of fault using the same energiser. Figure 3-1 below shows the normal energiser voltage and current waveforms that are used as a reference in each fault condition, and was measured on a small test fence.

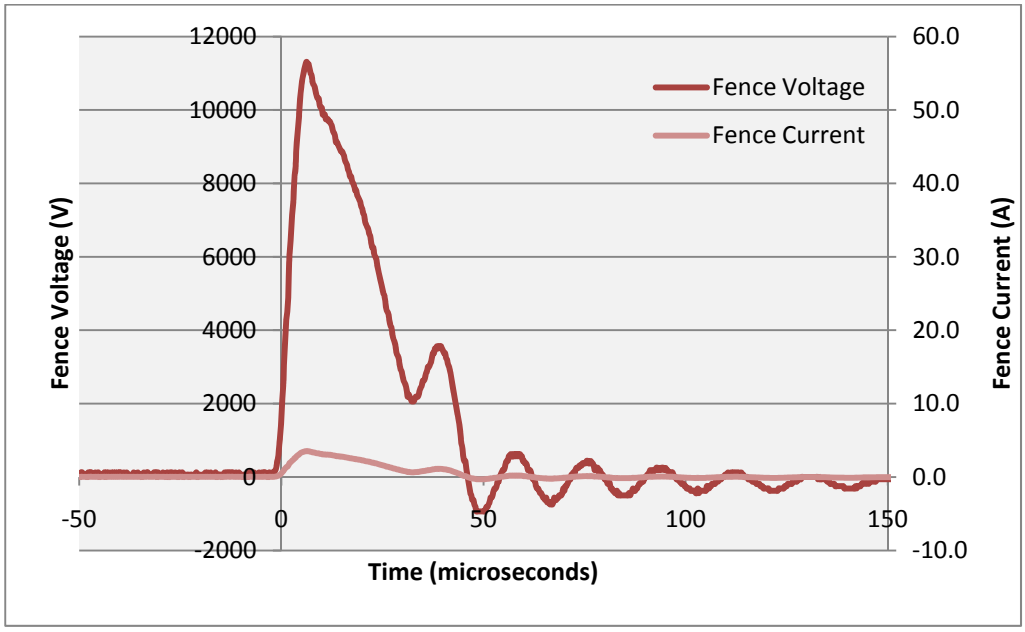


Figure 3-1: Normal energiser voltage and current waveform

3.2.1 Vegetation Growth Touching Live Wires

The exact impedance to ground of the vegetation under a fence changes unpredictably as plants grow, move in the wind, and die. In general, as the vegetation grows taller and more abundant, the impedance to ground reduces. This results in a gradual decay of fence voltage and increase in fence current (see Figure 3-2).

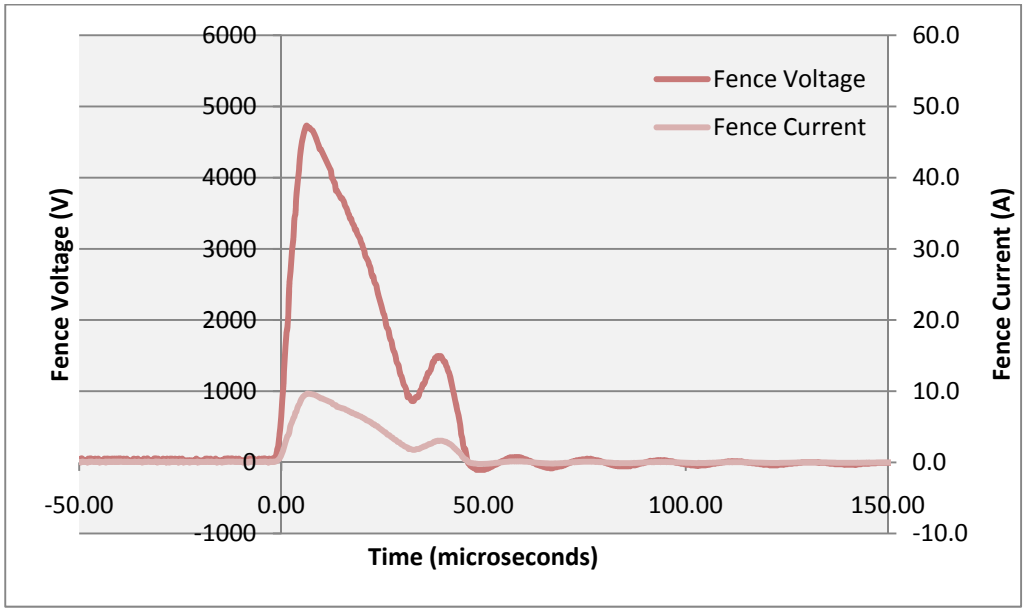


Figure 3-2: Fence voltage and current with heavy vegetation

3.2.2 Broken Wires

There are three broken wire scenarios: in each scenario, there is no voltage or current present after a broken live wire. However, the effect before the break is very different:

1. Open circuit live wire, not shorted – The fence voltage before the break will be operating at normal or higher voltage than usual, while normal or lower currents will be present.
2. Open circuit live wire, shorted to ground – Fence voltage will be very low or none at all, while current will be very high.
3. Open circuit live wire, shorted to other non-electrified wires – There will be lower than usual fence voltage and increased Radio Frequency Interference (RFI) as the fence pulse causes arcing from the non-insulated fence wires to ground. Fence current will be higher than usual.

3.2.3 Cracked or Burnt Insulators

Damaged insulators allow the formation of arcing. Once an arc is established, the impedance to ground seen by a fence pulse is very low, resulting in very low fence voltage and high current. The rapid discharge caused by an arc often generates a great deal of RFI.

The fence voltage may still reach its normal value before an arc is formed; however, the duration of the fence pulse is much shorter than normal after the formation of an arc (see Figure 3-3).

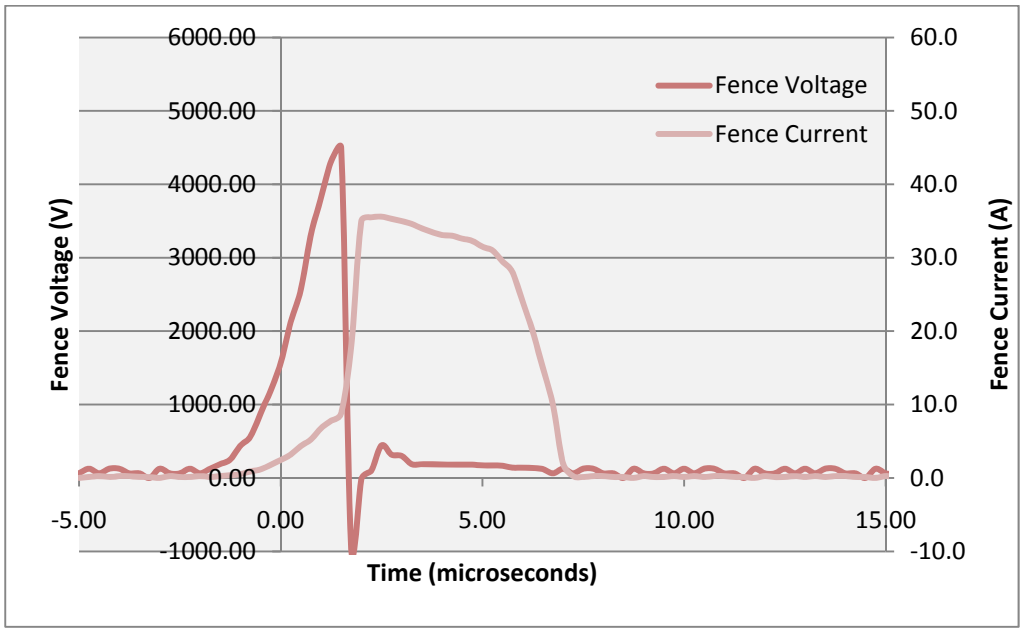


Figure 3-3: Fence voltage and current with arc discharge due to damaged insulator

3.2.4 Poor Earthing

If an electric fence system is poorly earthed, both the voltage and current readings will be normal. However, the shock produced when an animal comes into contact with the fence is greatly reduced.

In a good electric fence system, the ground impedance is as low as possible so that most of the fence voltage will appear across the animal's body (due to its higher impedance). As the earth return impedance increases, the fence itself drops more of the fence pulse voltage, until a point is reached where the ground impedance is much greater than that of the animal, resulting in only a small proportion of the fence pulse voltage appearing across the animal.

Fence installation guides state that the peak voltage measured at the earth system should not exceed 300V.[9, 10, 14] Any value above this indicates that the impedance is too high and either more earth rods are needed or an earth return wire fence construction should be used.

3.2.5 Corroded Metals / Poor Connections

Poor wire connections and corroded metals create high impedances in series with the fence wire. As with a poor earth return system, when an animal comes into contact with the fence only a small proportion of the fence voltage appears across the animal. Voltage and current readings drop immediately after a high-impedance joint.

If the fence wires are rusty, then the rust acts as insulator between the fence and animal. Rust is easily seen and the affected length of wire should be replaced immediately.

3.2.6 Over Extended Energiser

When an energiser is not powerful enough for the size of fence used, the measured voltage will steadily decrease as the distance from the energiser increases until the voltage is too low to be effective. For a

properly matched fence, the furthest point in the fence system should still have more than 3kV to maintain effective stock control.[4] In an over extended system, distant sections of the fence will have little or no voltage. Current readings will be normal or low, because the fence is not faulty and therefore has high impedance to ground (see Figure 3-4).

To test if the energiser is over extended, disconnect the energiser from the fence and measure its voltage directly at the terminals. If the voltage increases, then a more powerful energiser is required to operate the fence.

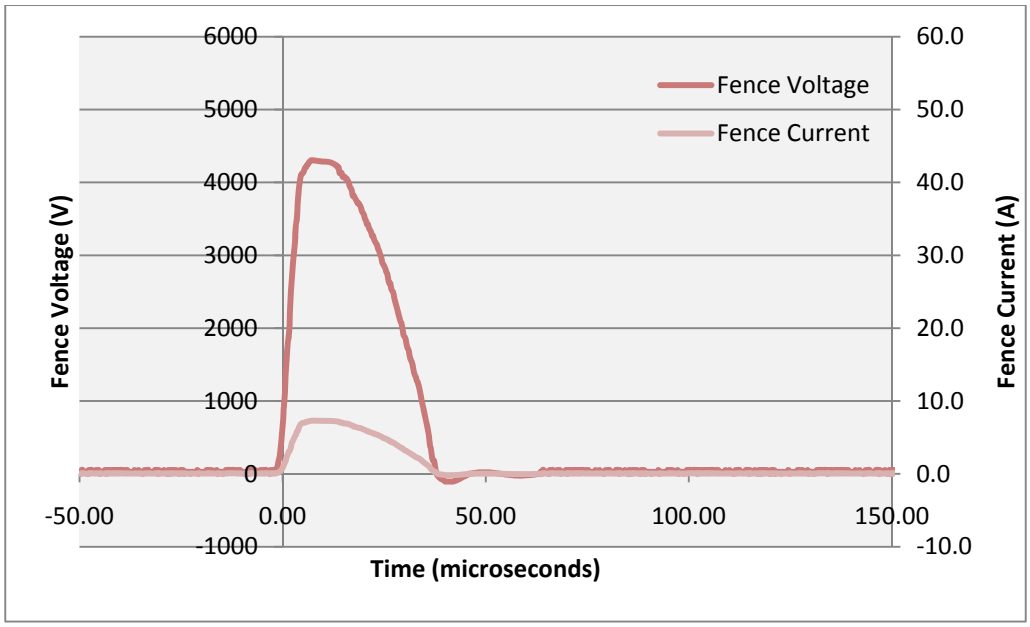


Figure 3-4: Fence pulse measured when energiser is over-extended

3.2.7 Flat Energiser Battery

If an energiser is operating from a flat battery, it will be unable to fully charge its storage capacitor(s) before generating a fence pulse. This will cause a reduction in both the peak voltage and energy content of the fence pulses. Eventually, the battery will be completely discharged

and the energiser will cease operation entirely. Some energisers will slow their output rate in an attempt to maintain the effectiveness of each pulse. Current readings will be lower than usual to correspond with the reduced energy content of each pulse.

Often the symptoms of this fault will appear identical to those of an over extended energiser as the output of the energiser reduces. A simple test of the energisers battery will determine the presence of this fault.

3.2.8 Weather Conditions

Extreme weather is often the cause of, or can magnify the effects of the other faults already mentioned. However, heavy rainfall, ice and snow can cause fence problems directly.

During heavy rain, water landing on the energiser support posts runs along the surface of the post until it reaches the ground. This can form a temporary low impedance path from live wires to ground. Snow piled up under a fence and ice formed directly on the fence can have the same effect. In each instance, the fence voltage is reduced and current increased.

3.2.9 Summary of Fault Effects

The following table (Table 3-1) summarises the effects of each fault type on the voltages and currents present in an electric fence.

Fault	Voltage	Current
Vegetation growth	Reduced	Increased
Broken wire, not shorted	Normal or increased, none past fault	Normal or reduced, none past fault
Broken wire, shorted to ground	Reduced	Increased
Broken wire, shorted to other wires	Reduced	Increased
Poor earthing	Normal	Normal
Poor connections / Corrosion	Normal, reduced past fault	Normal, reduced past fault
Over extended energiser	Reduced	Normal or reduced
Flat battery	Reduced or none	Normal or reduced
Weather conditions	Reduced	Increased

Table 3-1: Electrical effects of faults on electric fences

3.3 Fault-finding Techniques

A large number of guides are available that describe how to find electric fence faults. Each guide has the same principles – start from the energiser and work towards the ends of the fence. Supporting reasons for this approach are: [4, 9, 10, 14, 15]

- The energiser installation location is typically in the same place as the fault-finding equipment and farm transport (i.e. in a shed next to the farmers residence)
- If the voltage at the energiser is normal, the next component to test is the earth system (located nearby to the energiser)

- If the energiser is defective or over extended (very common in winter with high vegetation loading combined with rain), the voltage at the energiser will be low
- If fault-finding starts after a short to ground in the fence, there will not be sufficient voltage or current to be measured

Figure 3-5 is a typical flowchart guide, taken from the Stafix guide to electric fencing.[9] If the energiser voltage is normal, then the earth system is checked next. If that is also normal, then the fence itself must be checked for faults. Always check around gateways, branches in the fence and wire joins, as faults are likely in these areas. A fault will show up as an abnormally high reading. A sudden reduction in current between one point and the next indicates a fault between the two points. Test the fence periodically (approximately every kilometre, depending on the overall size of the fence). All wires in a junction are measured, and the highest current path should be followed. The readings will continue to drop until the fault is reached (or passed). After the fault, the readings should remain constant. It is very common to have more than a single fault, especially after bad storms and other bad weather.

If the voltage at the energiser is lower than usual, the energiser should be disconnected from the fence (leadout removed), and then tested again. If the voltage is still too low (i.e. below manufacturer's specifications) then the energiser is faulty or has a flat battery. If fault finding an adaptive output energiser, the user must be careful that the energiser is not working as designed in this scenario – consult the manufacturer's manual before commencing fault finding in this situation.

A typical fence is presented in Figure 3-6, both with and without the presence of a fault. When the fence is operating normally, the voltage is high (5kV in this example), and the current is relatively low, tapering off towards the end of the fence. If a short to ground is introduced halfway

along the fence, the voltage will drop and current readings become much higher. After the short, voltage and current readings drop to zero.

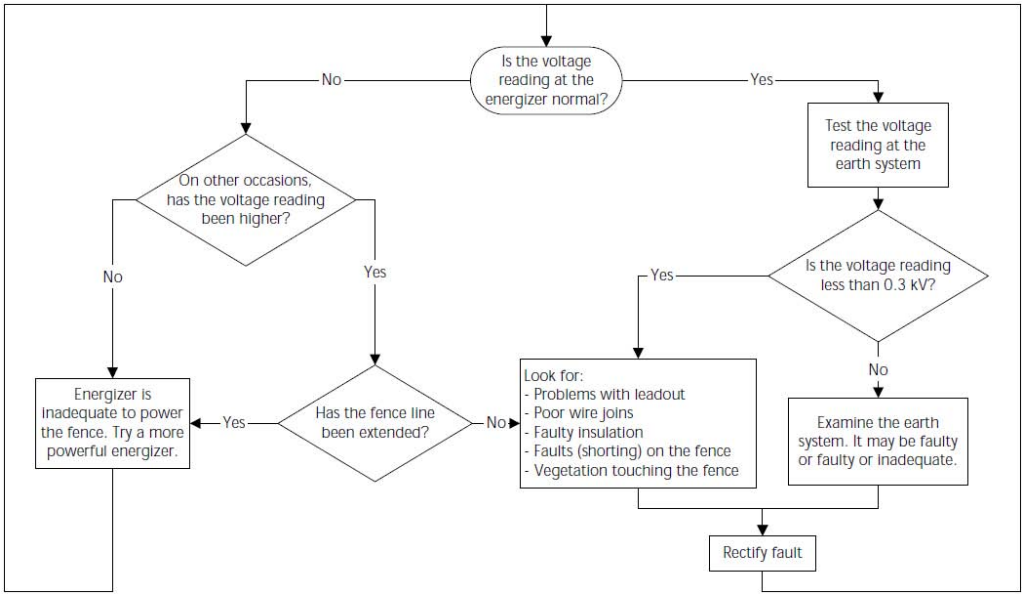


Figure 3-5: Electric fence fault-finding guide[9]

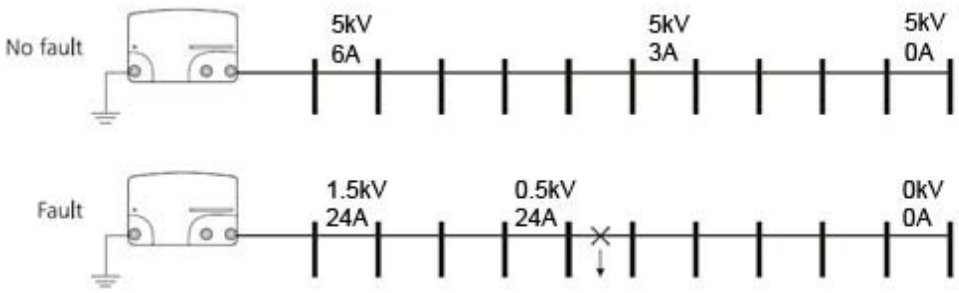


Figure 3-6: Single wire fence fault-finder readings with and without a fault[9]

Another example (Figure 3-7) shows a similar situation but on a multi-wire, earth return fence. When there is no fault, current readings are low. Readings taken after a live wire and earth return wire have been shorted together show increased current on both of the affected wires, while the other live wire shows reduced current.

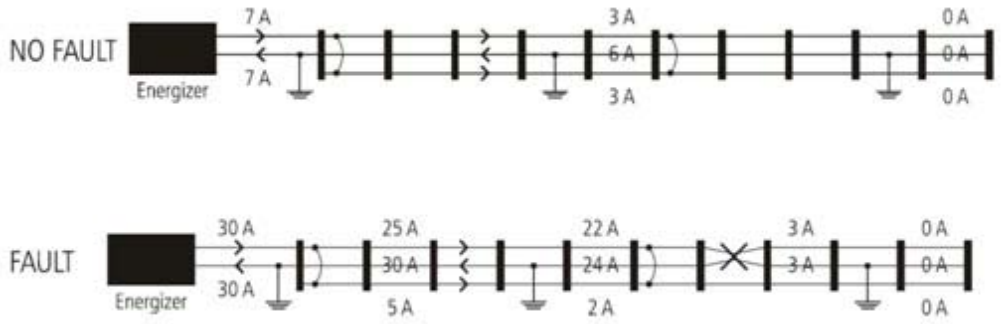


Figure 3-7: Earth return fence fault-finder readings with and without a fault[9]

3.4 Fault Meter Requirements

It is clear that electric fencing relies upon high voltages to form an effective barrier to animals; therefore, voltage should be used as the primary means to determine the presence or absence of faults. Some faults cannot be reliably detected by voltage alone (such as vegetation growth). In these situations, measuring the current in addition to voltage provides a much clearer indication of the problem.

Due to the very short pulses used in electric fencing, it is only necessary to measure the peak voltage and current values to ascertain the condition of the fence. The peak reading must be an absolute value taken from either the maximum positive or maximum negative reading. In addition, the device must detect when pulses have occurred to be able to measure the pulses and update the information presented to the user.

Because typical fault-finding techniques require the user to start at the energiser and work their way to the end of the fence, it is useful for any fault-finding device to indicate the correct direction for the user to travel. Calculating the direction to indicate is simple when the polarity of the

fence voltage and current is known. Note that this technique will indicate in reverse when used on earth return wires.

Often the user must remember previous voltage and current readings to determine if they are changing or are different to normal. Each reading may be taken several kilometres apart making it easy to forget specific readings, and it would be useful to the user if the fault-finding tool could store a history of recent measurements.

In summary, a useful fault-finding tool for electric fencing must possess each of the following capabilities:

- Detect fence pulses
- Measure peak fence voltage
- Determine voltage polarity
- Measure peak fence current
- Determine current polarity
- Remember previous readings

4

Chapter 4

Hardware Design

4.1 Introduction

This chapter presents the development of the hardware necessary to detect and measure the normal and fault conditions on an electric fence. These conditions were determined in chapter 3 by observing the characteristics of each fault and can be summarised as detecting pulses, and measuring both positive and negative peak voltages and currents.

In addition to detecting faults, the hardware must be suitable for use in a commercial product in that it is rugged, reliable, inexpensive to manufacture, produces consistent results across units, and conforms to current industry standards (such as removal of hazardous substances compliance).

Finally, for customers to want to purchase this device, it must be easy to use, water resistant, protect the user from fence voltages, fit in the users hand and pocket comfortably, simple to change the battery, and have a long battery operating life.

During development of the hardware, several prototypes were created and tested. The results of each prototype test have been included at the end of this chapter to complement the individual module descriptions presented in the chapter body.

The block diagram in Figure 4-1 below shows the anticipated hardware components and interactions necessary for the fault meter to fulfil its objectives. The development of each block is described in the body of this chapter.

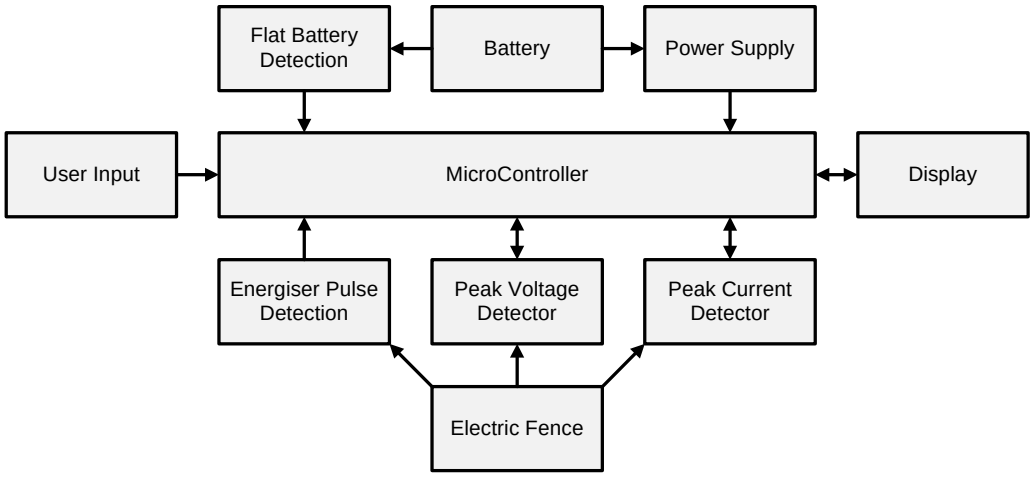


Figure 4-1: Anticipated hardware block diagram

4.2 Energiser Pulse Detection

In order to measure the peak voltage and current of a fence pulse, the device must first be able to detect the start of the pulse. The detection limits identified in chapter 3 are as follows:

- Must detect pulses with peak voltages as low as 0.1kV (for earth measurements)
- Must detect both positive and negative pulses

- Must be able to determine if pulse was positive or negative

Three supplementary requirements have been added to meet the needs of a commercially ready product:

- Must detect pulses reliably
- No false detections (drains battery)
- Low power consumption (pulse detection must always be active)

Several designs were evaluated, based on three technologies – transformer coupled, comparator detection, and passive resistive divider.

4.2.1 Integrated Peak and Pulse Detector

Most modern 8-bit microcontrollers contain one or more hardware based comparators as peripherals. Because the comparator output is connected directly to a physical pin as well as an internal interrupt, it is possible to use a single comparator as both a peak detect circuit (see Figure 4-2) and pulse detector.[17]

When a positive signal arrives at the comparator's non-inverting input, the comparator output follows by going high. An interrupt is generated at the same time that the output changes state, alerting the software that a pulse has arrived. The output will toggle high and low several times until the peak value of the pulse has passed, and an analogue representation of the peak voltage will be stored in capacitor C1. This voltage can be measured by another microcontroller peripheral, an Analogue to Digital Converter (ADC). The ADC will use the same pin as the comparator inverting input to read the voltage of C1 through resistor R3.

The circuit in Figure 4-2 was constructed on Vero board using a PIC16F630 microcontroller and formed the basis of the first prototype (see section 4.11.1). Pulses were detected accurately in each test.

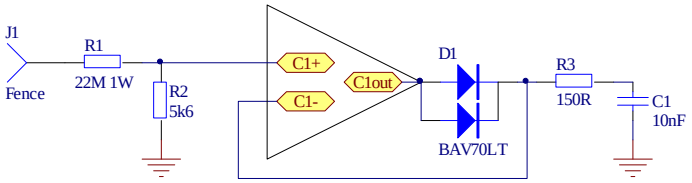


Figure 4-2: Integrated peak and pulse detector

Experimental observations showed that the output of the comparator was not accurate, since if the voltage stored in C1 was too high (due to overshoot) the comparator could not discharge the capacitor, resulting in frequent over-reading.

Another significant disadvantage is the need to maintain power to the comparator at all times. Even in low power mode, internal microcontroller comparators require at least 150µA, compared to less than 1µA microcontroller current without the comparator enabled.[18]

4.2.2 Step-Down Transformer

Voltages used in electric fencing are much higher than those typically used in modern electronics. Exposing standard passive electronic components directly to such voltages may lead to premature failure, drifting value tolerances and flashover. A step-down transformer is a reliable method to reduce fence voltages to an easily managed level, and can be used both for pulse detection and for peak value measurement (see Figure 4-3).

There are several design disadvantages associated with transformers. In general, they are bulky (especially at peak fencing voltages) and expensive. A suitable transformer must be custom designed for the unit to withstand high voltages and provide the exact division ratio needed. The inductance of the transformer will alter the shape of the fence pulse, and in some instances can delay detection of the pulse.

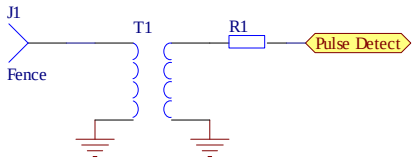


Figure 4-3: Step-down transformer pulse detector

4.2.3 Open Ended Resistor to Microcontroller

The simplest method to detect when a voltage is present is to feed that voltage directly into the microcontroller. I/O pins are protected from electro-static discharge (ESD) damage with internal bypass diodes, and as long as the current remains below 1mA, these diodes can form part of a divider network.[18] The circuit in Figure 4-4 utilises a microcontroller ESD protection diode at the end of a divider formed by R1, R2 and R5.

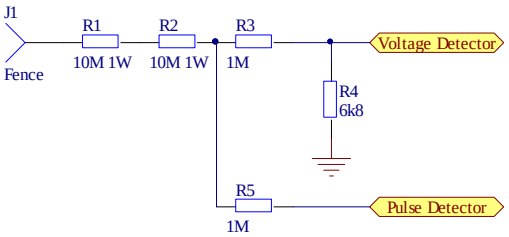


Figure 4-4: Open ended resistor pulse detector

There were several instances during testing of this design where the microcontroller pin latched high after a pulse. Further research showed that this is a common problem with ESD protected inputs.[17]

The detection threshold for this circuit is approximately the operating voltage of the microcontroller, plus one diode drop. Therefore, any voltages exceeding approximately 5V will trigger a positive detection. This level is too low for practical use, and caused many unwanted detections (such as when the top wire of the resistive divider encountered skin).

4.2.4 Zener Diode Clamped Resistive Divider

The open-ended resistor pulse detection method worked well except for two notable problems. The circuit in Figure 4-5 attempts to address both issues by placing a resistor (R6) at the bottom of the resistive divider to set the minimum detection voltage, and zener diode (D1) to prevent the microcontroller ESD protection diode latching.

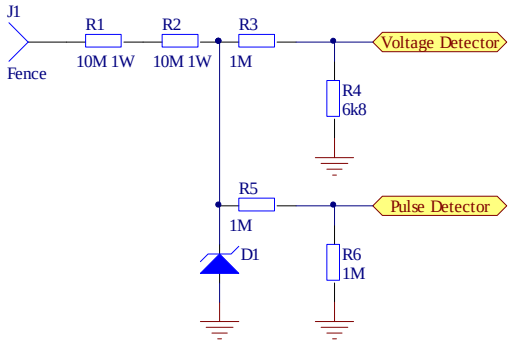


Figure 4-5: Zener diode clamped resistive divider pulse detector

In testing, the zener diode operation was too slow, resulting in the ESD protection diode conducting first and rendering the zener diode useless. Further diode selection was unnecessary since it was discovered that

using a full resistive divider lowered the voltage applied to the microcontroller pin sufficiently to prevent latching, solving the initial problem.

This circuit, like the previous one, suffers from false detections. The detections appeared to be random, occurring at any time, often when the unit was undisturbed. Eventually the cause was determined to be 50Hz noise from the mains distribution network coupling onto the fence wire and into the very sensitive and high impedance pulse detection input.

4.2.5 Transistor Buffered Resistive Divider

This circuit attempts to solve the false triggering problem inherent in the previous resistive divider circuit (Figure 4-5). As seen in Figure 4-6, the lower resistor value is reduced to provide a strong pull up/down, eliminating susceptibility to noise. To further isolate the microcontroller input pin from the pulse, transistors buffer the signal and provide a logic level output. Separate circuits detect positive and negative pulses, since the microcontroller can no longer use software to determine the polarity of the pulse.

Once constructed, testing showed that this circuit did not detect pulses every time. However, there were not any false detections either. Further investigation revealed that the transistor was not turning on fast enough, and that to accurately detect fence pulses the transistor requirements must be characterised and a specific model specially selected. This option is risky in a commercial environment as manufacturers often change or discontinue components, and relying on the manufacturer could lead to a future large-scale failure and subsequent recall of the product.

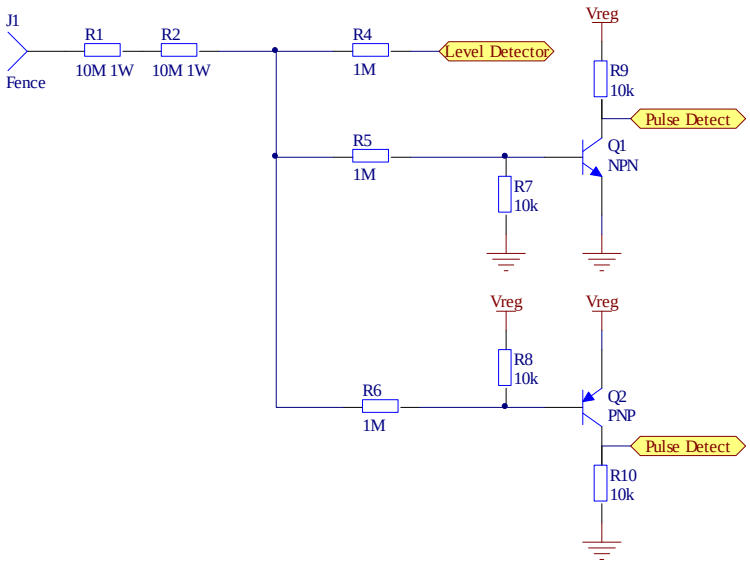


Figure 4-6: Transistor buffered resistive divider pulse detector

4.2.6 Capacitively Coupled Resistive Divider

Figure 4-7: Capacitively coupled resistive divider pulse detector

Figure 4-7 shows another variation of the zener clamped resistive divider from Figure 4-5. Capacitors C1 and C2 have been inserted into the resistive divider to create a high-pass filter, removing low frequency signals from the pulse detector. Separate positive and negative pulse detection circuits are needed because the duration of the signal output to the microcontroller is much shorter than previous circuits, and do not allow the software enough time to determine the pulse polarity.

This circuit was selected after testing produced excellent results – no false detections, and 100% of both positive and negative fence pulses were detected. It also has the advantages of zero power consumption, cheap, simple, reliable, small size (mostly surface mountable components), and ability to detect pulses as small as 100V.

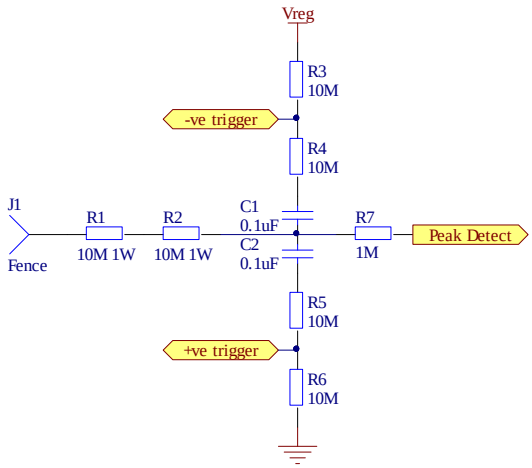


Figure 4-7: Capacitively coupled resistive divider pulse detector

4.3 Voltage Peak Detector

It was determined in chapter 3 that both positive and negative peak voltage readings are necessary due to variations in the way that farmers connect energisers to the fence. The operating range of the detector must start from 100V and continue until at least 12kV, while maintaining a reasonable degree of accuracy.

To save power, the peak detect circuit will be powered down while the device is idle. Therefore, the peak detector must be able to turn on fast enough to measure the first detected pulse. If achieved, this will allow the peak detector to be turned off between pulses to save even more power.

A number of circuit options were trialled, each of which are detailed below.

4.3.1 Direct Sampling by High-Speed ADC

A sufficiently fast ADC is capable of taking multiple readings during a fence pulse. If each reading were stored successively, once the pulse is complete, the peak voltage can be determined in software. This allows the sampling network to be constructed from all passive components (except the ADC, which is available as a microcontroller peripheral – see Figure 4-8).

The advantages of direct sampling include low component count, able to use cheap passive components, very fast start up time, and readings are supplied with a known accuracy.

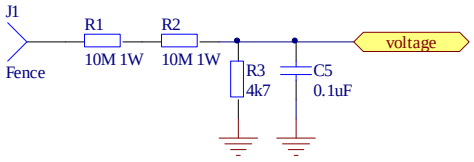


Figure 4-8: Sampling circuit for direct pulse measurement by high-speed ADC

To capture the peak value of a fence pulse, the ADC must take samples approximately every microsecond, requiring an ADC capable of at least one million samples per second. At the time this research was carried out, microcontrollers and external ADCs capable of this sampling rate were prohibitively expensive for this product.

4.3.2 Peak Detect Using Comparator

Comparators are commonly available as internal microcontroller peripherals, allowing the creation of very cheap and flexible peak detector circuits. When used as a unity gain buffer to charge a storage capacitor, the comparator output will go high while the input voltage is

higher than the voltage stored in the capacitor. A diode ensures that when the comparator output is low the capacitor is not discharged.

The peak detector circuit shown in Figure 4-9 has been introduced in section 4.2.1 as an integrated pulse detector and peak detector circuit. As stated earlier, experimental results showed that this circuit did not provide accurate results due to overshoot issues.

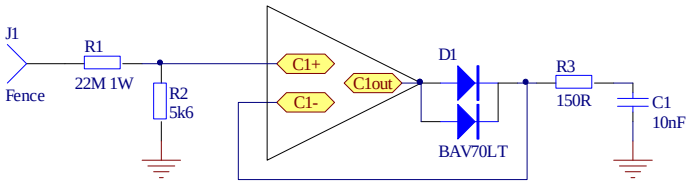


Figure 4-9: Comparator based peak detect circuit

4.3.3 Peak Detect Using Two Single Rail Op-Amps

Op-amps are commonly used in peak detect circuits because of their fast operation (when the appropriate model is selected) and linear output. A simple non-inverting unity gain amplifier is used as a positive peak detector, and an inverting unity gain amplifier creates a negative peak detector. An ultra-fast diode is placed between each amplifier output and storage capacitor to convert the amplifiers into peak detector circuits. Both peak detect circuits work in parallel for each pulse, allowing the software to determine which value to use.

The use of an op-amp instead of a comparator greatly reduces overshoot in the storage capacitor because the op-amp has an analogue output proportional to the input, instead of a logic output.

Figure 4-10 provides the schematic of the positive and negative peak detector circuits. R1, R2, R7 and R8 lower the fence voltage to a level

suitable for electronics. The maximum peak detect value of either circuit is the supply voltage minus one diode drop. With a 3.3V supply, the maximum input voltage is 2.6V. To measure at least 12kV, the resistance of R8 must be less than 4,550Ω (a standard value of 3k9Ω was used).

$$\boldsymbol{R8 = \frac{2.6V}{12kV} \times 21M\Omega} \qquad \textbf{[4-1]}$$

R10 and R20 isolate the op-amp inputs from each other, and form part of the unity gain selection. R13 and R23 are bleed resistors, designed to allow the stored voltage in C10 and C20 to dissipate before the arrival of the next fence pulse. A time constant of 100ms was chosen to allow enough time to set and read the stored voltage (energiser pulses are required to be shorter than 10ms), but short enough to have fully dissipated before the next pulse (at least one second apart).[13, 19]

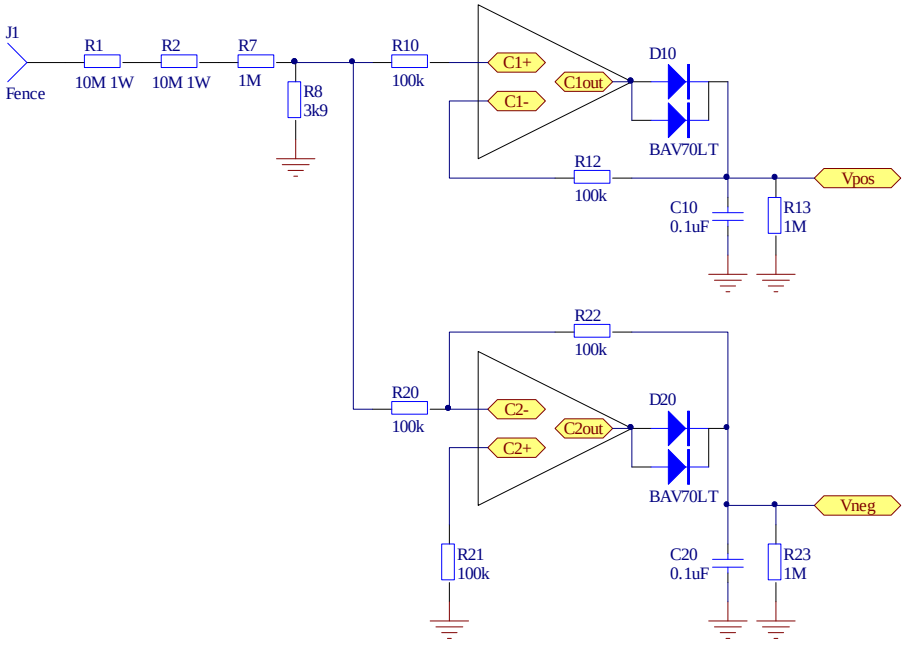


Figure 4-10: Positive and negative pulse peak detect circuit

Selection of an appropriate op-amp is critical to the operation of this circuit. To accurately capture the peak value of a fast-changing fence

pulse, the op-amp must have a high bandwidth and fast slew-rate. A high slew-rate is also important for fast turn-on because the peak detector can quickly react to full-scale inputs.[19]

For the inverting amplifier to work, the op-amp must be capable of operating with inputs below ground (otherwise to capture a negative going pulse the amplifier input would have to be pulled up to its power supply.[19]

Many op-amps were evaluated, and eventually the OPA2374 single rail dual op-amp was selected. Its 6.5MHz bandwidth and 5V/µs slew-rate allow zero to full-scale readings in less than 1µs. The device also has rail-to-rail input and output, allowing inputs of up to 200mV below ground.[20]

4.4 Current Peak Detector

The specifications for this circuit are almost identical to those of the voltage peak detectors. The only difference between them is that this circuit must measure positive and negative peak current. A simple choke is used as a magnetic pickup, positioned against the wire so that current flow through the fence wire couples into the choke winding. The choke is a reliable and cheap device, and is reasonably linear.

As can be seen in Figure 4-11, R30 is a low-value resistor connected across the choke (L1). Whenever current is induced into L1, a proportional voltage appears across R30. A typical induced current waveform is provided in Figure 4-12. An energiser connected to a 2kΩ load produced a peak voltage of 10kV. The resulting peak current of 5A induced a voltage across R30 of 120mV.

Discussions with electric fencing professionals and lab experiments indicate that the highest expected current is 50A (a typical fault-free fence has 2-3A). Using the maximum output value of 2.6V obtained in section 4.3.3, this allows the input to be scaled up by a factor of two.

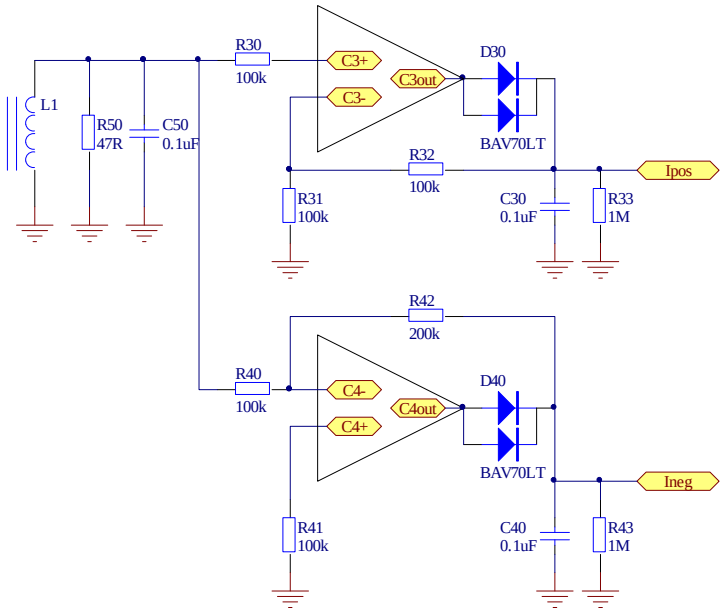


Figure 4-11: Current peak detector circuits

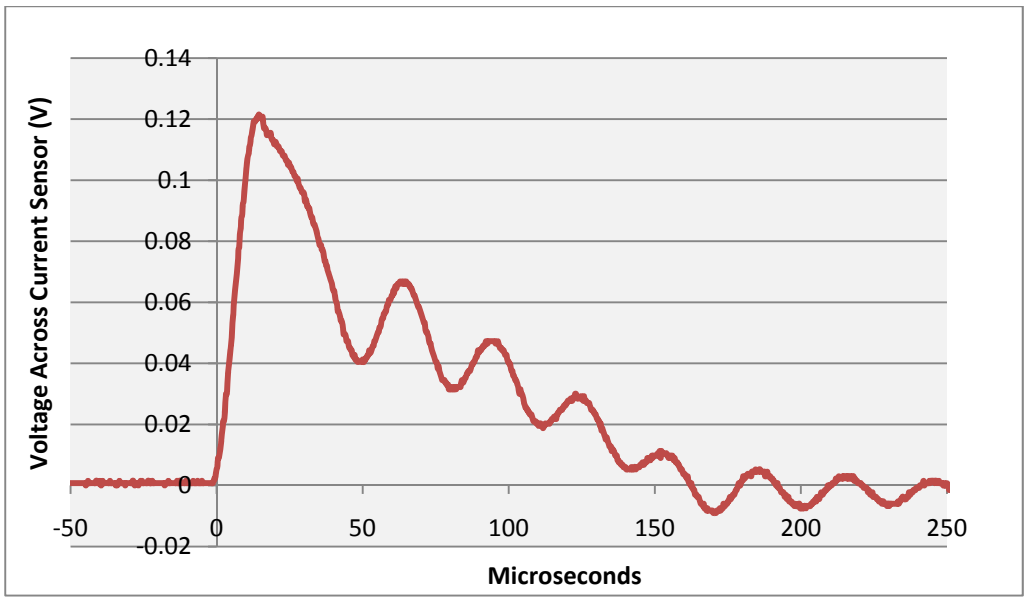


Figure 4-12: Moderately loaded energiser current pickup waveform (10kV peak output, 2kΩ load)

Resistor value selection for the non-inverting amplifier is as follows:

$$\boldsymbol{Gain = 1 + \frac{R32}{R31}} \quad \textbf{[4-2]}$$

$$\boldsymbol{= 1 + \frac{100k\Omega}{100k\Omega}} \quad \textbf{[4-3]}$$

$$\boldsymbol{= 2} \quad \textbf{[4-4]}$$

Standard values of 100kΩ are used to provide an input impedance greater than ten times higher than the resistive divider used to sample the fence voltage, and are an appropriate value according to the op-amp datasheet.[20]

The inverting amplifier values are also selected to produce a gain of two:

$$\boldsymbol{Gain = \frac{-R42}{R40}} \quad \textbf{[4-5]}$$

$$\boldsymbol{= \frac{-200k\Omega}{100k\Omega}} \quad \textbf{[4-6]}$$

$$\boldsymbol{= -2} \quad \textbf{[4-7]}$$

Output diodes (D30, D40), storage capacitors (C30, C40), and bleed resistors (R33, R43) have been used in exactly the same way as described in section 4.3.3.

4.5 User Interface

There are two parts to the user interface. The first is a display to indicate voltage and current readings, direction to fault and battery status. The display must be easily viewed in full sunlight and it would be

advantageous if visible at night. Ideally, all the data will all fit onto the display simultaneously and be large enough to view at full arms length.

The second part of the interface is a button that turns on the device, and can toggle the display to show previous readings for comparison. When a fence pulse is detected, the device will automatically turn on, making it possible to use the device without having to press the button. The most useful function of the button will be to check that the battery is not flat.

4.5.1 Display Selection

Four display options were evaluated: a 2-digit LCD, a 2x12 character LCD with backlight, a custom designed LCD, and a 1.5” colour OLED display. Each option had their own advantages that are discussed in the next section.

4.5.1.1 2-Digit LCD

The simple transflective LCD with two digits and full stop shown in Figure 4-13 is a very low cost option. The drive signals are simple; however, 17 microcontroller output pins (see Figure 4-14) are required to provide the correct AC drive waveform. A pair of LEDs surface mounted under the display would provide enough light to make the display visible at night. Each digit is large and easily viewed from a distance. Finally, the LCD is extremely low power.

A serious shortcoming of the LCD is that only two digits are available, restricting displayed accuracy and requires alternating the display from

current to voltage. In addition, LEDs or something similar is necessary to display direction to the fault.

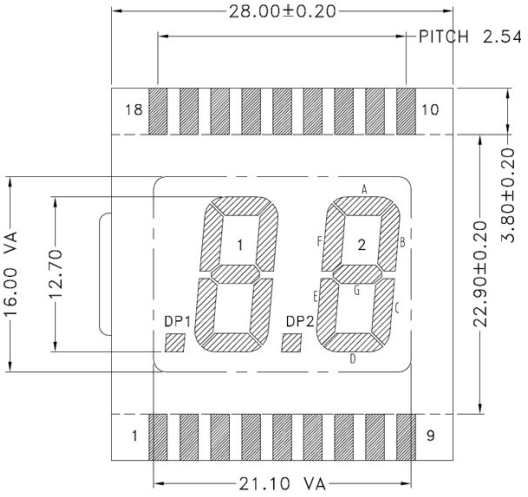


Figure 4-13: 2-Digit LCD physical appearance

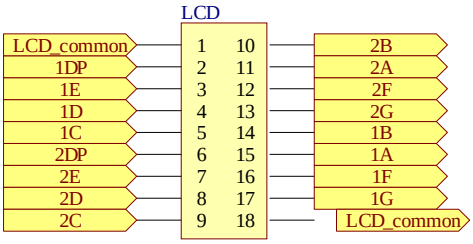


Figure 4-14: 2-Digit LCD electrical connection

This display was not used due to the need to alternate the display between readings, and because it potentially made the product appear to be of lower quality compared to other fault finding tools.

4.5.1.2 2x12 Character LCD

Character LCDs are commonly used where a variety of data is to be displayed (see Figure 4-15). There are many software libraries available

and writing code to drive the display is very simple. Simple animations can be implemented as well as custom graphical icons.

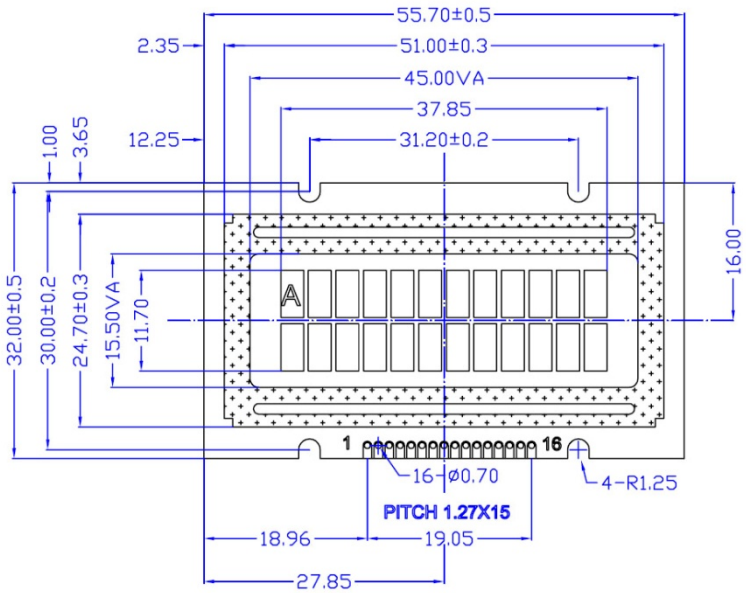


Figure 4-15: 2x12 Character LCD physical appearance

The LCD has a built-in LED based backlight that can operate from 3.3V, and only requires 10 microcontroller I/O pins (including power and contrast – see Figure 4-16).

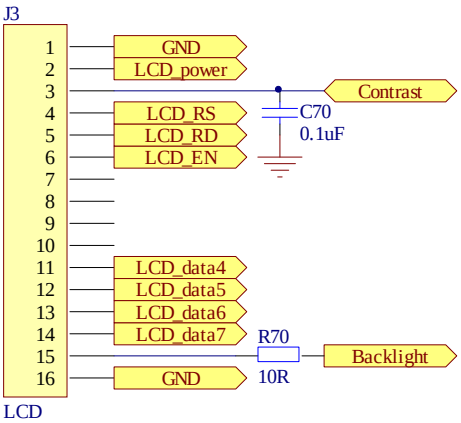


Figure 4-16: 2x12 Character LCD electrical connection

Disadvantages of this display are the width of the LCD – at 51mm wide it must either be mounted sideways in the product so that the user must rotate the device to read results, or make the plastic casing too wide to be comfortably held in the palm of a hand. In addition, each character is only 5.5mm high, making them difficult to see at arm's length.

This display was abandoned because the characters were too difficult to read, and turning the device sideways to read produced very negative feedback from customers.

4.5.1.3 Custom Designed LCD

A custom-made LCD would overcome the disadvantages of both the 2-digit and 2x16 LCDs. The shape, size of characters, and number of visible characters can be selected (see Figure 4-17).

The advantages of low power consumption, relatively simple drive software, and visibility of displayed data would be retained, and at the same time, the disadvantages of difficult size and insufficient display data are overcome.

Custom panels must be ordered in large quantities to become affordable, and at the time of this research, a minimum order of 2,000 units must be placed. In addition, the delivery time from placing an order is three or more months making this option unsuitable for a product that is not anticipated to sell in large volumes (initially).

Figure 4-17: Custom LCD possible appearance

If a lot of data is to be displayed on the LCD, then there will be a large number of pins or a complicated multi-level drive waveform required. Either option would necessitate a larger, more expensive microcontroller than could be otherwise used (pin count is the leading cost factor in 8-bit microcontrollers).

One other disadvantage is a lack of design flexibility. If a change were desired in the display, the entire panel would have to be redesigned and any existing stock would have to be cleared first.

This option was not selected because of the large number of panels that would have to be ordered each purchase, requiring a manufacturer to hold a large value of stock before manufacturing.

4.5.1.4 1.5” OLED Display

Organic Light Emitting Diode (OLED) displays are widely used in portable applications where small size and efficiency are important. A backlight is not required since each pixel is self-illuminating, giving the display an overall thickness of 1.55mm. The OLED under consideration has a viewable area of 1.5” diagonal, or 28.9mm square and an overall panel size of 36mm square (see Figure 4-18).

The display is 128 x 128 pixels, and each pixel is capable of 24-bit colour. When driven at 15V the display seen easily in full day light, which is helped by a wide viewing angle of 178^{o}. Because this is a graphical display, the user interface software will be much more complicated and 15 I/O pins are required from the microcontroller (see Figure 4-19). However, this also allows for company logos in full colour, animation and flexible output options such as alternate languages.

A 15V power supply capable of up to 30mA is required for the OLED display to function, as well as several biasing components and capacitors. This is the most expensive option at almost ten times the cost of the simple 2-digit LCD.

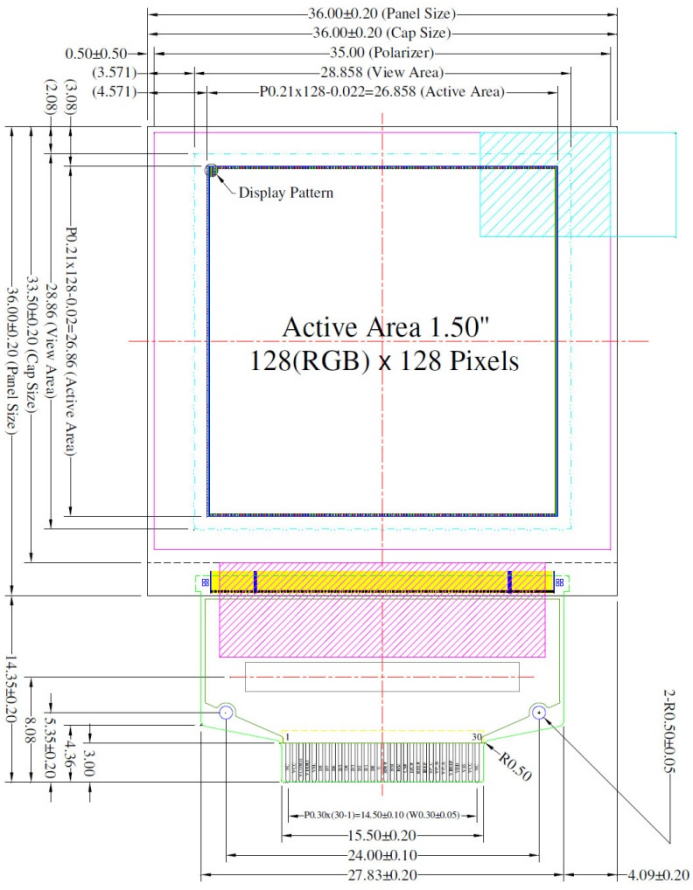


Figure 4-18: OLED display physical appearance

This display was chosen for the device despite its cost and complexity as it provides a modern professional appearance to the product, has huge flexibility of colour, layout, font and size, animations for direction to fault, full-colour company logo and graphical representation of previous readings all on the display simultaneously.

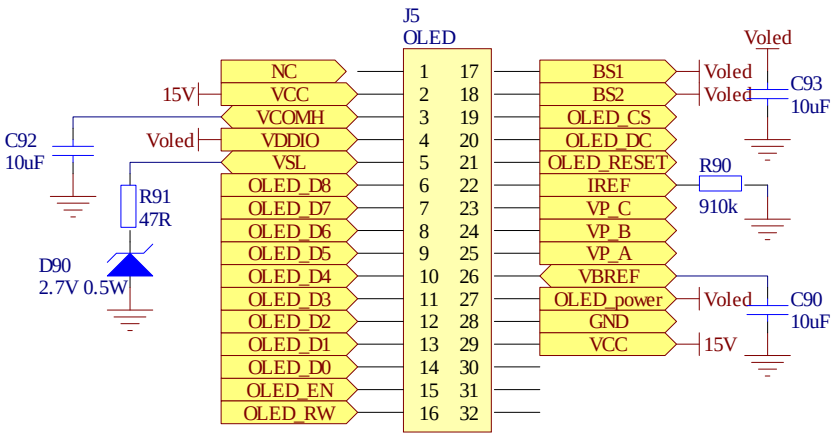


Figure 4-19: OLED display electrical connection

4.5.2 User Input Selection

Even though the device automatically turns on when fence pulses are present, there are times when the user may want to check that the battery is not flat or view previous readings. If desirable, advanced functions such as language selection could be selected by holding down a button or pressing in a coded sequence. To achieve all these goals, only a single button is necessary.

Pressure activated buttons introduce extra cost and potential for failure to a product. The button must be protected by a waterproof membrane and mechanically limited so that applying excessive force will not destroy the button. Designing the plastic case to attain these protective measures takes additional time at the design stage and often adds cost

to the manufacturing process. In an attempt to eliminate these costs, a capacitive touch sensor was investigated along with the traditional push button.

4.5.2.1 Capacitive Touch Sensor

Sensors such as the QT113 are capable of sensing the presence of body contact through over a centimetre of solid material such as glass or plastic.[21] This would allow the device to sense a touch through the plastic casing, eliminating the need for moving parts (see Figure 4-20), flexible water seals and additional tooling costs. A simple decal or raised feature on the plastic casing would indicate where to touch.

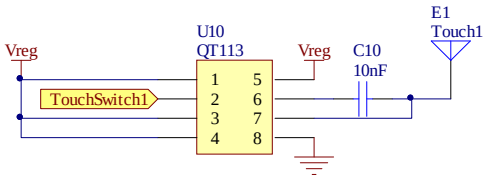


Figure 4-20: Capacitive touch sensor electrical connection

This system showed great promise as a way to remove the difficulties and failures traditionally associated with push buttons. During testing, the touch sensor worked very well while the device was in use and when placed on a bench or in a box, but produced a large number of false detections when carried in a pocket or on a belt (proximity to the body was regularly detected). These regular detections would cause excessive battery use, and if advanced features such as language selection are employed, the device could inadvertently change settings.

4.5.2.2 Push Button

Since the capacitive touch sensor proved to be unsuitable, a push button was the next best option. They have several advantages over touch sensors including zero power consumption when idle, positive tactile feedback when pressed, and are very cheap (see Figure 4-21 below).

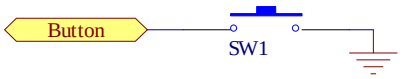


Figure 4-21: Push button electrical connection

Only a single button was used due to the simple interface – the first press turns on the device and subsequent presses toggle between the newest reading and historical readings.

4.6 Microcontroller

A microcontroller is used for this device as it is the only practical method for controlling the acquisition and display of fence / fault data. There are a huge variety of microcontrollers and a large selection of peripherals available. The list below contains the minimum feature set needed:

- Low power consumption when idle / sleeping
- Analogue to digital converter with 5 or more inputs
- Interrupt driven pin-change / edge detection
- Embedded C language compliant tools
- At least 15 I/O pins, preferably 28 I/O pins
- 5MHz or faster internal oscillator
- Readily available

Two microcontrollers from Atmel were selected as possible candidates – the ATmega48P and the ATmega324P. The Atmel 8-bit ATmega range has the industry's lowest power consumption when idle and sleeping, very low price, are widely used throughout the world, and fulfilled every list item above.

4.6.1 ATmega48P

This is a 32-pin TQFP packaged microcontroller available in program memory sizes of 4kB, 8kB, 16kB and 32kB, providing flexibility and a clear migration path if the devices complexity were to grow. It has 23 I/O lines, 8 ADC inputs, priority driven interrupt system, an 8MHz internal oscillator, multiple design tools, available in any quantity desired, and consumes less than 1uA in sleep.

With this processor, the OLED must be connected using a serial bus to reduce the number of I/O pins required (see Figure 4-22). There are two drawbacks to using a serial bus – the first is that communication to the OLED is slow relative to a parallel bus connection. Data is clocked at half the clock speed, producing a bit-rate of (8Mhz / 2) / 8 bits = 0.5Mbits / second compared to 8MHz * 8 = 64Mbits / second for parallel. A full screen refresh requires 128 rows * 128 columns * 24 bits per pixel = 393,216 bits. Therefore, a serial connection update would take about 800ms, compared to 6ms for parallel. The slower refresh rate would prevent the use of animation.

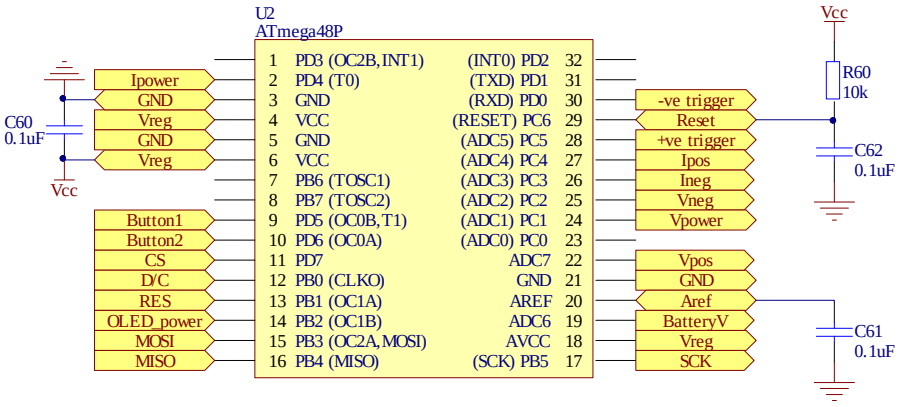


Figure 4-22: ATmega48P electrical connection

The second drawback is that the serial bus only supports write-only operation. This requires that the microcontroller uses delays instead of the more reliable polling method, and removes the option of read-modify-write operations that allow dynamic screen updates. If dynamic updates are required (i.e. animation and / or partial screen updates) then a screen buffer must be set up in microcontroller RAM. The amount of RAM necessary is 128 rows * 128 columns * 3 bytes per pixel = 49,152kB. Currently there are not any low-cost microcontrollers with that much available RAM (the ATmega48P has 0.5kB), and consequently the display would have to be completely redrawn with each update.

The combination of these two drawbacks provides enough reason to use a slightly more expensive microcontroller from the same family with more I/O pins.

4.6.2 ATmega324P

This microcontroller is very similar to the ATmega48P, with more peripherals and a 44-pin TQFP package. It is available in program

memory sizes of 16kB, 32kB, 64kB and 128kB and has 32 I/O pins. Power consumption and interrupts are the same as the ATmega48P.

With plenty of available I/O pins, the OLED can be connected in 8-bit parallel mode as per Figure 4-23. This allows fast screen updates, read-modify-write operation, and polled registers for maximum reliability.

The microcontroller requires a minimum of external components – three capacitors for power supply decoupling, one capacitor to stabilise the internal ADC voltage reference, and a pull-up resistor with bypass capacitor for the reset line. A six-pin header is included in the design to allow in-circuit programming and debugging (see Figure 4-24).

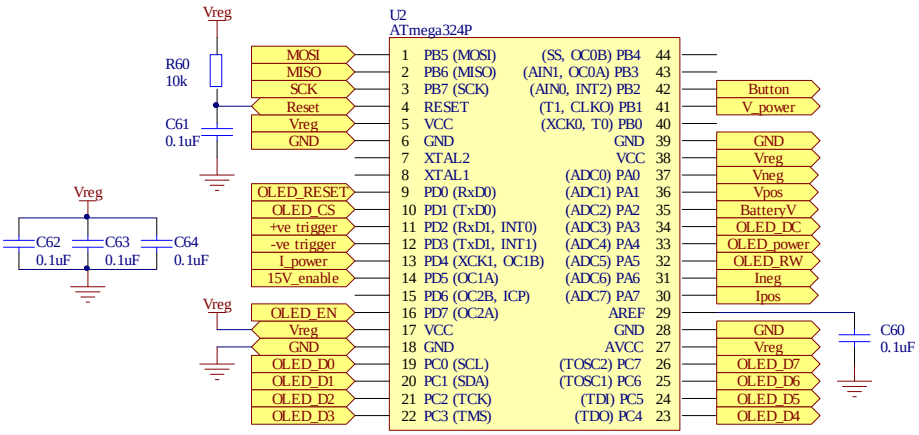


Figure 4-23: ATmega324P electrical connection

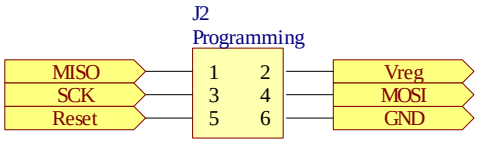


Figure 4-24: ATmega in-circuit programming header

4.7 Power Supply

The most commonly used power source for small portable farming equipment is the 9V battery. For this reason and because of its relatively large capacity when compared to other battery configurations of comparable size, a single 9V battery was chosen.

The circuit requires a +3.3VDC power supply for the microcontroller and DC-DC converter. The microcontroller supplies power to the voltage and current peak detection circuits and OLED display logic to allow these circuits to be turned off when not needed. A +15VDC power supply is needed for the OLED driver circuitry.

Current consumption values for each circuit component are given in Table 4-1 below.

Power Supply	Component	Current (mA)
+3.3V regulator	Microcontroller, normal	4.8
	Microcontroller, idle	1.3
	Microcontroller, sleep	0.005
+3.3V regulator	Pulse detector	0
+3.3V regulator	DC-DC boost converter	0.03
+3.3V microcontroller	Voltage peak detector	0.75
+3.3V microcontroller	Current peak detector	0.75
+3.3V microcontroller	OLED logic	1
+15V	OLED display, 50% on	20
	OLED display, 100% on	30

Table 4-1: Current consumption values

4.7.1 15V Power Supply Design

The OLED driver circuit is the only component that uses this power supply. The datasheet states that the OLED panel has a maximum current consumption of 30mA (all pixels on, white, full brightness).[22]

Two circuit designs were considered:

a. A voltage doubler producing 18V from the 9V battery then regulated down to 15V, and
b. A DC-DC boost converter outputting 15V.

4.7.1.1 Voltage Doubler and Regulator Circuit

This circuit is a very low cost method to provide a regulated 15V supply (see circuit schematic in Figure 4-25). Two transistors (Q70 and Q72) are alternately switched on, providing an alternating drive voltage through C10. The diodes D70 and D71 are configured as a voltage doubler, producing 18V at the input of U70, which is a simple low cost 15V linear regulator.

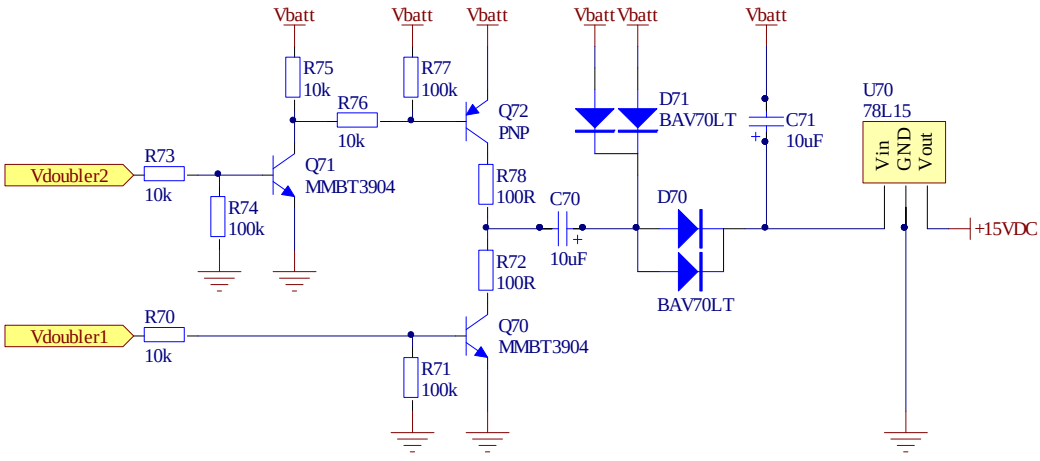


Figure 4-25: Voltage doubler and regulator circuit schematic

To operate, the microcontroller must supply two square waves of opposite polarity (Q70 and Q72 must never be on at the same time). This circuit was prototyped and showed promise when driving a resistive load. However, when used to power the OLED it was found that the circuit was very inefficient, delivering only 40% of the power consumed to the OLED, therefore requiring up to 125mA from the battery to produce 30mA at 15V.

Another problem encountered when disabling the circuit was that the battery voltage (minus two diode drops) appears at the output of the regulator, allowing 6mA to sink through the disabled OLED and reverse biasing C71. These problems are inherent in the circuit and significant enough to justify abandoning this design.

4.7.1.2 DC-DC Boost Converter Circuit

DC-DC boost circuits are frequently used to supply OLED displays due to their small footprint, high efficiency and low standby current. Although this design is almost twice the cost of the voltage doubler circuit, it has many advantages including simple on/off drive input, soft-start, undervoltage lockout and thermal shutdown. The circuit schematic is given in Figure 4-26.

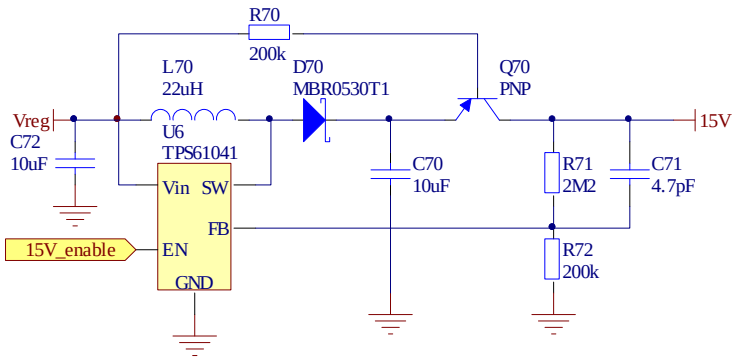


Figure 4-26: DC-DC boost converter circuit schematic

The TPS61041 DC-DC boost converter controller was selected for its low switching peak current (250mA), adjustable output, relatively low cost, easy availability, small physical size and high operating frequency allowing the use of smaller, cheaper external components.[23] Battery datasheets show that minimising peak currents can maximise the useful life of a battery.[24]

The minimum value of inductor L70 can be estimated from the formula:

$$\boldsymbol{Lmin} = \frac{\mathbf{2}\times \boldsymbol{Iload_{max}}\times(\boldsymbol{Vout}-\boldsymbol{Vin})}{\boldsymbol{\eta}\times \boldsymbol{fSmax}\times \boldsymbol{Ip^2}} \quad \textbf{[4-8]}$$

Where:

Ip = 250mA (peak switching current of converter)

fSmax = 1MHz (maximum switching frequency)

η = 85% (expected efficiency from datasheet)

$$\boldsymbol{Lmin} = \frac{\mathbf{2}\times \mathbf{30}\boldsymbol{mA}\times(\mathbf{15}\boldsymbol{V}-\mathbf{3.3}\boldsymbol{V})}{\mathbf{85\%}\times \mathbf{1}\boldsymbol{Mhz}\times(\mathbf{250}\boldsymbol{mA})^2} \quad \textbf{[4-9]}$$

$$= \mathbf{13.2\mu H} \quad \textbf{[4-10]}$$

This gives a minimum value for L70 of 13.2uH. Standard values available are 15uH and 22uH. The latter value was selected since it is widely available from multiple suppliers in the same size and price as the former.

Setting the output voltage is calculated as:

$$\boldsymbol{Vout} = \mathbf{1.233}\boldsymbol{V} \times \left(\mathbf{1} + \frac{\boldsymbol{R71}}{\boldsymbol{R72}}\right) \quad \textbf{[4-11]}$$

Where:

R71 = recommended value of 2M2 (from datasheet[23])

To obtain an output voltage of 15V, R72 should be 197kΩ. The closest standard value of 200kΩ produces an acceptable output of 14.8V.

Finally, the value of C71 must be calculated from:

$$\boldsymbol{C = \frac{1}{2\pi \times \frac{fS}{20} \times R71}} \quad \textbf{[4-12]}$$

Where:

fS = switching frequency of the converter at the nominal load current

$$\boldsymbol{fS(load) = \frac{2 \times Iload \times (Vout - Vin + Vd)}{Ip^2 \times L}} \quad \textbf{[4-13]}$$

$$\boldsymbol{= \frac{2 \times 20mA \times (15V - 3.3V + 0.3V)}{(250mA)^2 \times 22uH}} \quad \textbf{[4-14]}$$

$$\boldsymbol{= 349kHz} \quad \textbf{[4-15]}$$

Vd = rectifier forward voltage drop

Iload = expected nominal load current

$$\boldsymbol{C = \frac{1}{2\pi \times \frac{349kHz}{20} \times 2M2\Omega}} \quad \textbf{[4-16]}$$

$$\boldsymbol{= \frac{1}{2\pi \times \frac{350kHz}{20} \times 2M2\Omega}} \quad \textbf{[4-17]}$$

$$\boldsymbol{= 4.1pF} \quad \textbf{[4-18]}$$

The closest standard value of 4.7pF was selected. All remaining component values were chosen from the datasheet recommended parts list.

Components R70 and Q70 form a disconnect circuit to prevent the input voltage appearing at the output when the converter is disabled, preventing current flow through the OLED driver. Q70 must be rated to supply at least 50mA constant current, such as a BC856.

4.7.2 3.3V Power Supply Design

The 3.3V power rail is used by all parts of the circuit, including as a reference for analogue readings. It is therefore important that this supply is accurate and reliable under all load conditions.

In normal use, the device spends most of its time in a sleep state to minimise power consumption and therefore maximise battery life. To extend battery life as long as possible this regulator must have a very low quiescent current. Finally, the regulator must be able to supply the DC-DC boost converter with peak currents of 250mA.

After reviewing several voltage regulators, only one model satisfied every requirement: the LP2985 series of regulators. The LP2985-3.3V is a 3.3V fixed regulator with low noise, over current and temperature protection, low drop-out operation, low component count, and very stable accurate output.[25]

A 10µF ceramic capacitor is connected to the regulator output as recommended in the component datasheet (see Figure 4-27).[25]

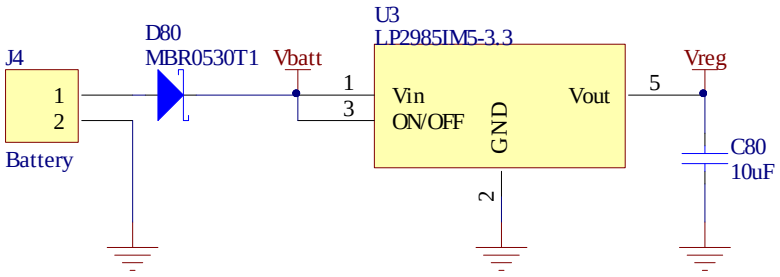


Figure 4-27: 3.3V regulator circuit schematic

4.7.2.1 Reverse Battery Connection Protection

During field trials, a prototype was taken to several countries. Upon arrival in the U.S.A., customs took the prototype away for inspection and attempted to connect the battery in reverse. This caused the 3.3V regulator to burn, destroying the prototype. Although the unit was eventually repaired, this demonstrated the need to protect against reverse connection in the final device.

There are two protective devices that would operate without permanent damage to the device and/or protective circuit: a series diode and a self-resetting fuse.

A series diode will prevent current flow if the battery is connected in reverse, protecting the circuit. When the battery is properly connected, current will flow through the diode and the circuit will operate normally.

A self-resetting fuse will operate if too much current flows (this will occur when the battery is reversed), creating an open circuit and protecting the circuit. The disadvantage of both options is that power supply efficiency is reduced.

A schottkey diode is used to minimise the forward voltage drop, therefore minimising the loss of efficiency. The impact on the circuit is calculated below.

$$\boldsymbol{Imax(no\ protection) = \frac{Pmax}{Vin}} \quad \textbf{[4-19]}$$

$$= \frac{\mathbf{0.5}\boldsymbol{W}}{\mathbf{9}\boldsymbol{V}} \quad \textbf{[4-20]}$$

$$= \mathbf{55.6}\boldsymbol{mA} \quad \textbf{[4-21]}$$

When the device is operating at full load (all peripherals active, OLED display all white) without reverse protection the battery current is 55.6mA.

$$\boldsymbol{Imax(diode\ protection) = \frac{Pmax}{Vin - Vf}} \quad \textbf{[4-22]}$$

$$= \frac{\mathbf{0.5}\boldsymbol{W}}{\mathbf{9}\boldsymbol{V} - \mathbf{0.3}\boldsymbol{V}} \quad \textbf{[4-23]}$$

$$= \mathbf{57.4}\boldsymbol{mA} \quad \textbf{[4-24]}$$

$$\boldsymbol{\eta(loss) = 1 - \frac{Imax(diode\ protection)}{Imax(no\ protection)}} \quad \textbf{[4-25]}$$

$$= \mathbf{1} - \frac{\mathbf{57.4}\boldsymbol{mA}}{\mathbf{55.6}\boldsymbol{mA}} \quad \textbf{[4-26]}$$

$$= \mathbf{3.2}\% \quad \textbf{[4-27]}$$

Placing a diode in series with the battery increases battery current to 57.4mA for the same device power consumption, resulting in a 3.2% loss of efficiency.

A self-resetting fuse with 300mA trip current (smallest standard value available greater than the peak current requirements of the circuit) has an operating resistance of 11Ω (from component datasheet[26]). The impact of this on the circuit is calculated below:

$$\boldsymbol{Pmax(fuse) = Pmax - (Imax)^2 \times Rfuse} \quad \textbf{[4-28]}$$

$$\boldsymbol{= 0.5W - (55.6mA)^2 \times 11\Omega} \quad \textbf{[4-29]}$$

$$\boldsymbol{= 34mW} \quad \textbf{[4-30]}$$

$$\boldsymbol{\eta(loss) = \frac{Pmax(fuse)}{Pmax}} \quad \textbf{[4-31]}$$

$$\boldsymbol{= \frac{34mW}{500mW}} \quad \textbf{[4-32]}$$

$$\boldsymbol{= 6.8\%} \quad \textbf{[4-33]}$$

From the results of the calculations, a series diode clearly has less impact on the efficiency of the device, and prevents excessive current rather than reacts to its presence. Therefore, introducing a reverse connection protection diode will lower the power supply efficiency by up to 3.2% but this loss is acceptable when compared to the alternative: destruction of the device.

4.7.3 Flat Battery Detection

Some farms are very large with significant travel times to potential fault sites, and therefore it is important that the user knows when the battery must be replaced to prevent frustration and wasted time, having to return back to a workshop just to replace the battery. If the battery is flat, the display can be used to notify users.

The typical 9V battery is considered flat when the terminal voltage drops to 6.0V (see Figure 4-28).[24] The terminal voltage will gradually decay over time rather than drop off suddenly, and since the device can theoretically operate down to 3.4V terminal voltage (due to the use a low dropout regulator), setting a flat battery indication threshold of 6.0V will give the user sufficient notice to replace the battery before the device stops operating.

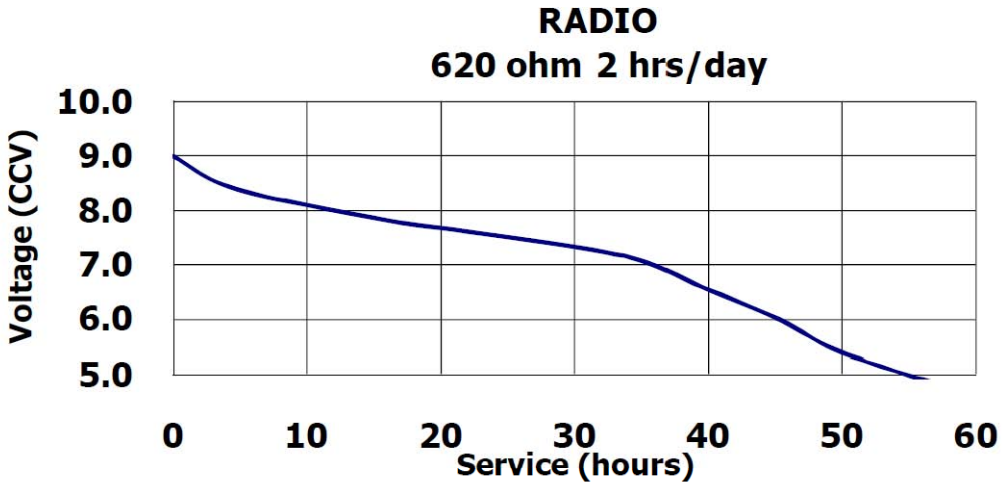


Figure 4-28: 9V battery discharge curve[24]

To enable the microcontroller ADC to read the battery terminal voltage it must be reduced to below 1.1V (ADC internal voltage reference level). A simple resistive divider can be used to scale the battery voltage down, however this will constantly consume current, requiring the use of large resistance values to minimise power wastage.

The microcontroller datasheet specifies that the maximum recommended impedance to ground for an ADC channel is 10kΩ.[18]

$$\boldsymbol{Idetection} = \frac{\boldsymbol{Vin}}{(\boldsymbol{R80}+\boldsymbol{R81})} \quad \textbf{[4-34]}$$

$$= \frac{\boldsymbol{9V}}{(\boldsymbol{100k\Omega}+\boldsymbol{10k\Omega})} \quad \textbf{[4-35]}$$

$$= 82uA \quad \textbf{[4-36]}$$

When the device is idle, the microcontroller, DC-DC converter and 3.3V regulator collectively consume approximately 2.5µA. It is desirable that the resistive divider only add minimal current consumption to this total. To lower the consumption of 82µA to a suitable level under 1µA it must be reduced by a factor of 100. To achieve this, the resistor values were substituted with standard values of 10MΩ and 1MΩ.

$$Idetection = \frac{9V}{(10M\Omega + 1M\Omega)} \quad \textbf{[4-37]}$$

$$= 0.82uA \quad \textbf{[4-38]}$$

This brings the total current consumption in sleep mode to approximately 3.3uA, well within acceptable limits (a 600mAh battery will provide over 20 years operation).

Through software design and multiple concurrent ADC readings, the accuracy of the microcontroller analogue readings can be maintained, allowing the use of a 1MΩ resistor to ground (see Figure 4-29).

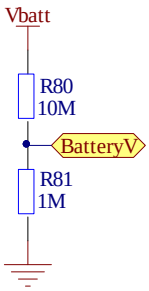


Figure 4-29: Flat battery detection circuit schematic

4.8 Earth Reference

When fault-finding, it is usually more important to obtain an indication of whether a fault is present and which direction to move, rather than making a highly accurate voltage and current reading. An accurate reading requires two connections to the fence – one to a live wire, and another to ground or an earth-return wire. Connection to the live wire is simple: the wire is at a comfortable height, and the device need only be placed so that it touches the wire. Making an earth connection is more difficult, necessitating that a probe is pushed into the ground.

Because of the rapid rise and fall times present in energiser pulses, it is possible to utilise the operator's body as a capacitive ground reference. The capacitance present to ground varies greatly with clothing, footwear, weather conditions and how the device is held. To maximise capacitive coupling to the operator, the bottom of the devices case (the area of the case closest to the human body) can be lined with a conductive film. Tests were carried out with both earth configurations, the results of which are detailed in chapter 6.

4.9 PCB Layout

The printed circuit board (PCB) layout must be carefully considered when high voltages and fast changing signals are present. The outline of the PCB is determined by the desired case shape, which for this product must comfortably fit into the average sized hand. In addition, the OLED display must be correctly mounted and the user-input button located in an appropriate spot. It is desirable to make the PCB as small as possible without compromising the manufacturability of the board to minimise the PCB purchase price.

Wherever possible, surface mount components are used to allow automated manufacture (through-hole parts must be hand placed and cost approximately four times more in labour charges per component).

The current sensor and voltage divider input must be located at the top of the PCB to allow interaction with the electric fence. A large gap will be placed around the high-voltage components to increase the arc-over creapage distance.

Because of the rapid rise and fall times possible in some of the circuitry, a large ground plane will fill all unused areas of the PCB (except around the high-voltage components). The ground plane is connected to the capacitive and metallic earth pickups that will be a part of the casing.

Finally, the PCB must have suitable mounting holes to ensure that it is held securely inside the plastic case. See section 4.11.6 later in this chapter for details of the PCB designs and physical constructions created during development of the device.

4.10 Case Design

A final case design was not completed for this project because it is the author's belief that such design work for a commercial product is a specialist area that must be completed by tool designers. However, a sample design was created using the SolidWorks 2005 CAD package (see Figure 4-30).

Important features of the case (apart from style and ergonomic fit) are:

- A rib that creates an internal compartment around the high-voltage resistive divider network used to sample the fence

voltage. This will prevent the formation of arcing that could jump the divider and damage circuit components

- A removable battery compartment that allows replacement of the 9V battery without using a screwdriver
- Mounting to secure the OLED display in position (it is connected to the PCB via a flexible ribbon)
- Lens to cover the OLED display, protecting it from water and impact
- Flexible section to allow use of the push button
- Metal insert at the top to connect the fence to the high voltage divider on the PCB
- Metal insert to enable connection of an earth lead
- Flexible grommet placed between the two halves of the moulding to make the case water resistant

ABS plastic is a suitable choice for the case moulding material – it is flexible, strong, easy to mould, low cost, and has a translucent option to allow its use as a lens to protect the OLED display. A flexible plastic such as santoprene can be used to make the grommet and button cover (from a single moulding).

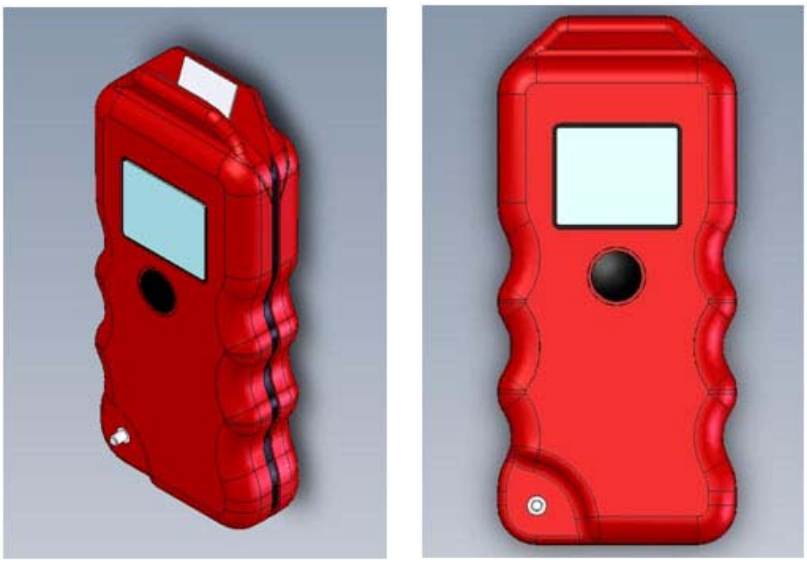


Figure 4-30: Possible case design

4.11 Prototypes Constructed

During development of suitable hardware for this project, a total of six prototypes were constructed. The first two prototypes were intended as proof of concept devices, and were used to identify important parameters and discover any unanticipated problems. In addition, having built physical devices enabled real-world field tests as well as early customer trials.

The details of each prototype are discussed in this section, along with advantages and limitations observed.

4.11.1 First Prototype

The primary goal of this prototype was to determine the operating conditions that the electronic circuitry would be subjected to, and to identify any issues surrounding the use of a microcontroller in such an environment. For these reasons, a very simple display was created from a row of LEDs to ensure that the software would be as simple as possible. In addition, the power supply is a pair of AAA batteries, removing the need for a voltage regulator and supplementary components.

The pulse and peak detection circuit described in section 4.2.1 is used, along with an 8-bit microcontroller (PIC16F630) with internal comparator peripheral. The circuit was constructed on vero board as seen in Figure 4-31

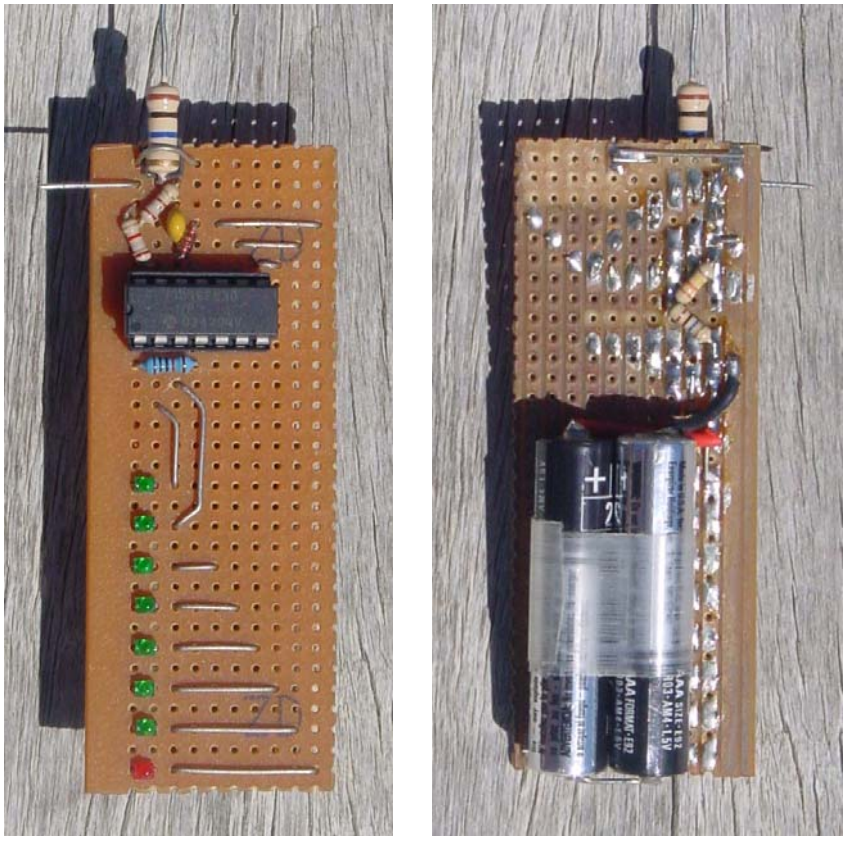


Figure 4-31: First prototype construction

Because this prototype is a proof of concept construction, only positive voltages will be detected and measured (see schematic in Figure 4-32). A small test program was written to detect pulses via interrupt and read the final stored peak voltage using the microcontrollers ADC. Finally, the peak voltage is displayed on the LEDs with the red LED indicating 3kV, and each subsequent LED another 1kV up to 10kV.

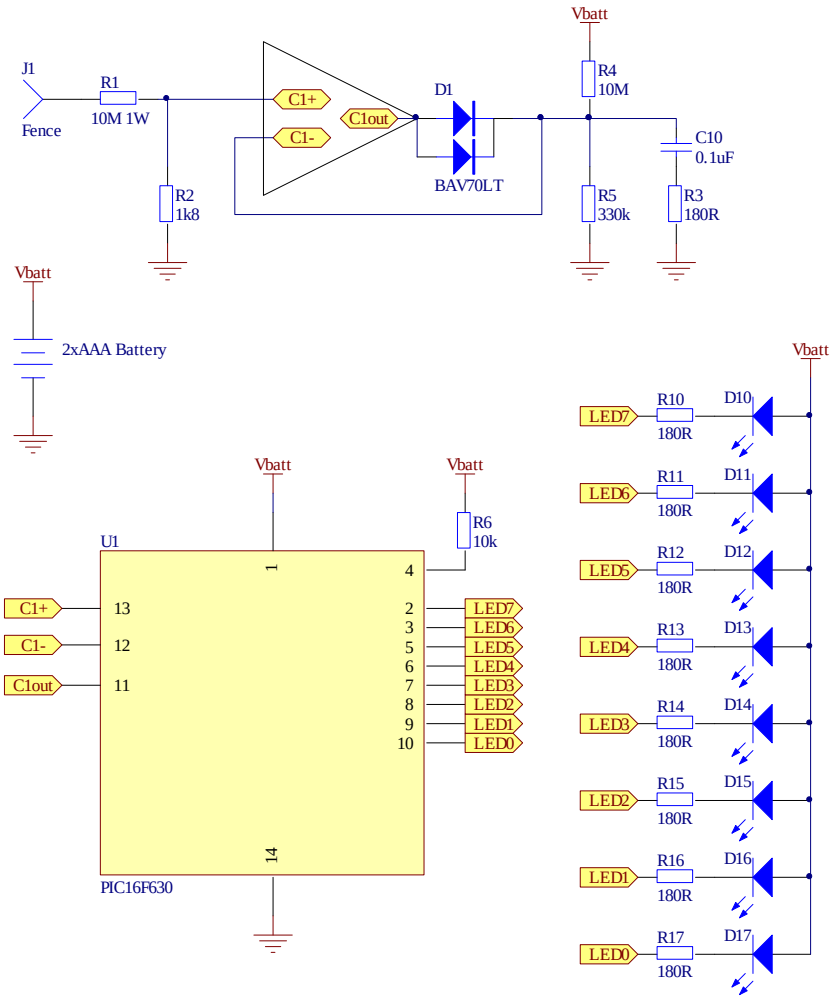


Figure 4-32: First prototype schematic

4.11.1.1 First Prototype Trial Results

Once construction was complete and the software written, an energiser and 500Ω load were connected to the prototype. The prototype did not appear to be able to detect the fence pulses. After further investigation, it was found that the rapidly changing fence pulse was inducing noise onto the microcontroller reset pin, causing the micro to reset with each pulse. A 0.1µF bypass capacitor fitted from the reset pin to ground (as recommended in the microcontroller datasheet[27]) solved the problem.

Although the 10MΩ 1W resistor R1 is electrically rated to withstand pulse voltages in excess of 10kV, the fence pulse would occasionally form an arc over the body of the resistor. Each time this happened, the microcontroller was subjected to the full fence voltage and subsequently destroyed. A simple solution to this problem is to use two resistors, halving the voltage across each resistor and therefore doubling the arc distance.

The accuracy of the stored voltage was not very good, and it was found that the pulse rise time had a strong influence on the measured result. Further investigation revealed that the slew rate of the comparator was not fast enough to track short, fast pulses. In addition, because the output of the comparator is either logic high or low, the stored voltage commonly overshot the desired value. Additionally, the specifications of the microcontroller comparator are more likely to change than a discrete part, increasing the risk that the device would have to be redesigned in a year or two.

If the comparator slew rate issue is resolved, another serious issue makes the usefulness of this design limited - current 8-bit microcontrollers do not have more than one or two internal comparators.

4.11.2 Second Prototype

This circuit, like the first prototype presented in 4.11.1, is a proof of concept design whose purpose is to determine if using a high speed ADC to directly sample both voltage and current readings has sufficient benefit to justify the high implementation cost of the design. A PCB design was created in Protel and hand-soldered (see Figure 4-33).

To be able to read both positive and negative voltage and current values, both sampling networks must be biased mid-way between power and ground. This allows the ADC to measure deviations above and below the mid-point reading. Pulse detection is via a capacitive coupled resistive divider similar to the design in section 4.2.6, and interrupt driven software routines (see schematic in Figure 4-34).

A 16-bit microcontroller from Microchip Ltd (PIC24HJ256GP610) has been selected for this design because of its 2Msps ADC with DMA controller that allows continuous analogue sampling at full speed. A 3.3V regulator is required to power this microcontroller and provide a reference for the ADC.

Results are displayed on a standard 16x2 character LCD (done for simplicity of development).

4.11.2.1 Second Prototype Trial Results

Initially fence pulse detection was very unreliable, making results difficult to obtain. Experimentation showed that the input impedance of the microcontroller pin was loading the pulse detection network, and the addition of a 10MΩ resistor between C1 and the microcontroller positive pulse detection input (another resistor was placed between C2 and the negative pulse detection input) resolved the issue. This modified pulse detection design proved very reliable and is presented in section 4.2.6 as the circuit used in the final design.

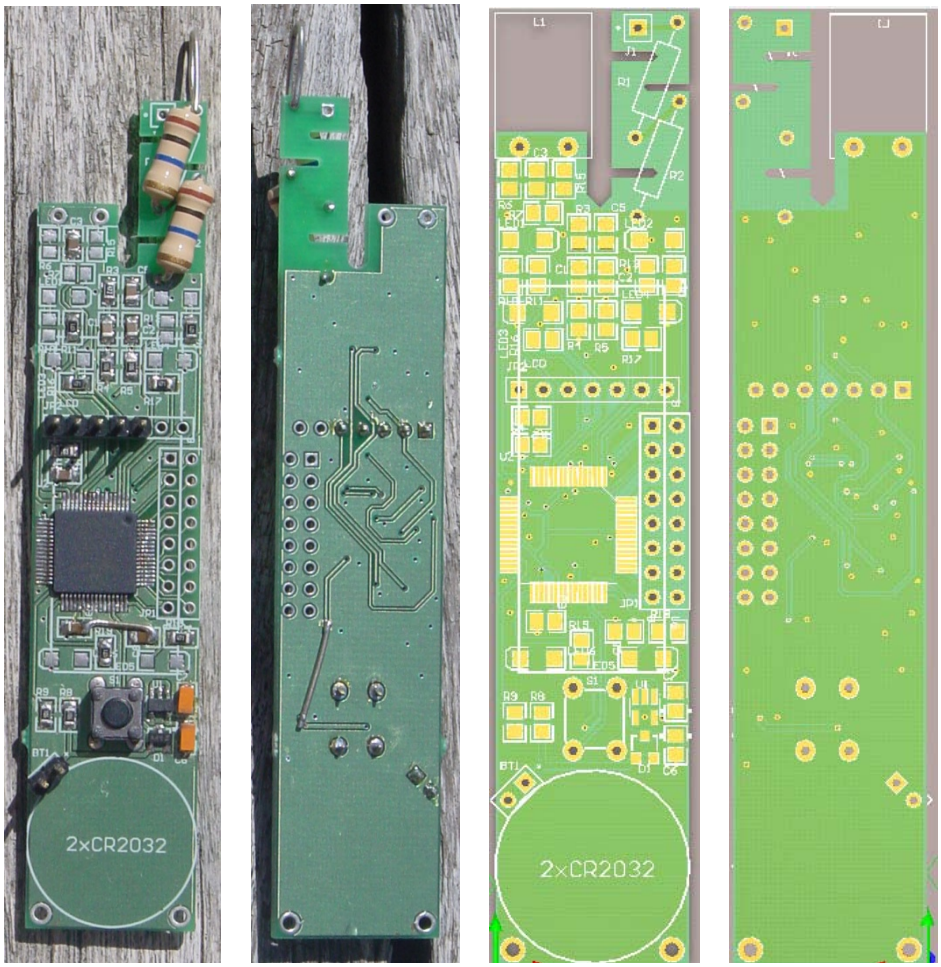


Figure 4-33: Second prototype construction and PCB representation

A simple program was written to capture continuous analogue readings from both the voltage and current inputs for 10ms after detection of a pulse. The ADC alternated between each channel, allowing a sampling rate of 1Msps each, producing 10,000 samples from each channel every pulse. The software processes the raw data with a moving window filter 20 samples wide to find the maximum peak reading. The filter prevents a single spurious reading from influencing the results.

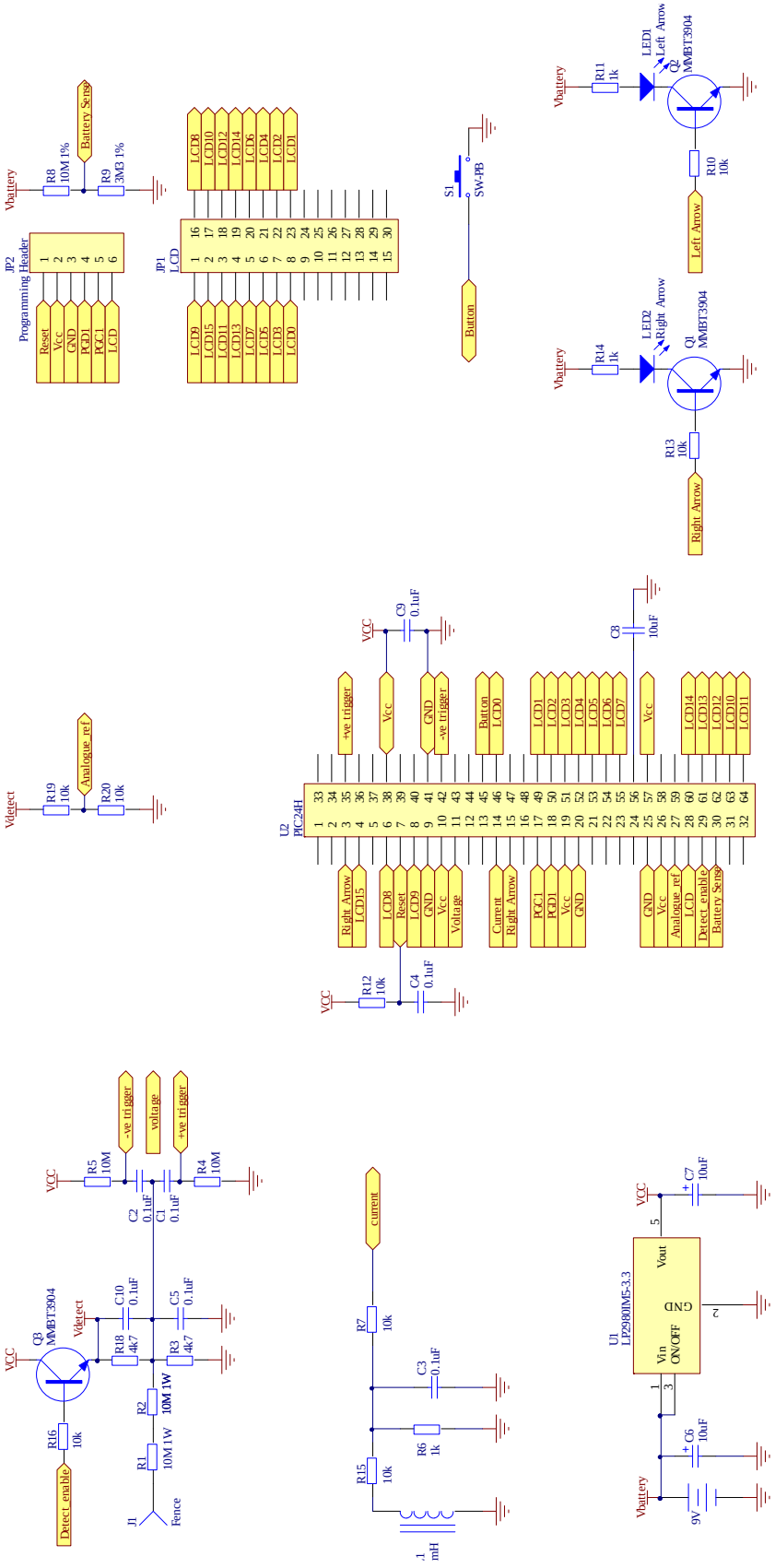


Figure 4-34: Second prototype schematic

It was observed that the microcontroller consumed 85mA while the ADC was active, which would be a significant drain on the battery and would require substantial power management.

Measurements taken from the analogue readings provided reasonably accurate results, however the software requires much more work before being able to provide consistent and reliable readings.

When this design is compared to an op-amp peak detector circuit, the high power consumption, expensive microcontroller, and additional software complexity of a high-speed ADC design make it a poor choice for a cost-competitive fault-meter design.

4.11.3 Third Prototype

This circuit is the first design that incorporates four op-amp based peak detect circuits to measure voltage and current. A 16x2 character LCD has again been used for simplicity of design to allow measurement results to be displayed in real time.

A more compact PCB shape was chosen to allow room inside a hand-held plastic casing for a 9V battery (see Figure 4-35 and Figure 4-36 below). A cut-out was placed in the board to prevent the high fence voltages from arcing into the logic circuitry. Plastic ribbing in the case design will create an internal partition inside the case to further isolate the high voltage section from the logic section.

The modified pulse detection circuit from the previous prototype has been included in this design, connected to a different microcontroller (ATmega48P) that offers good performance and very low power consumption with minimum cost (see circuit schematic in Figure 4-37).

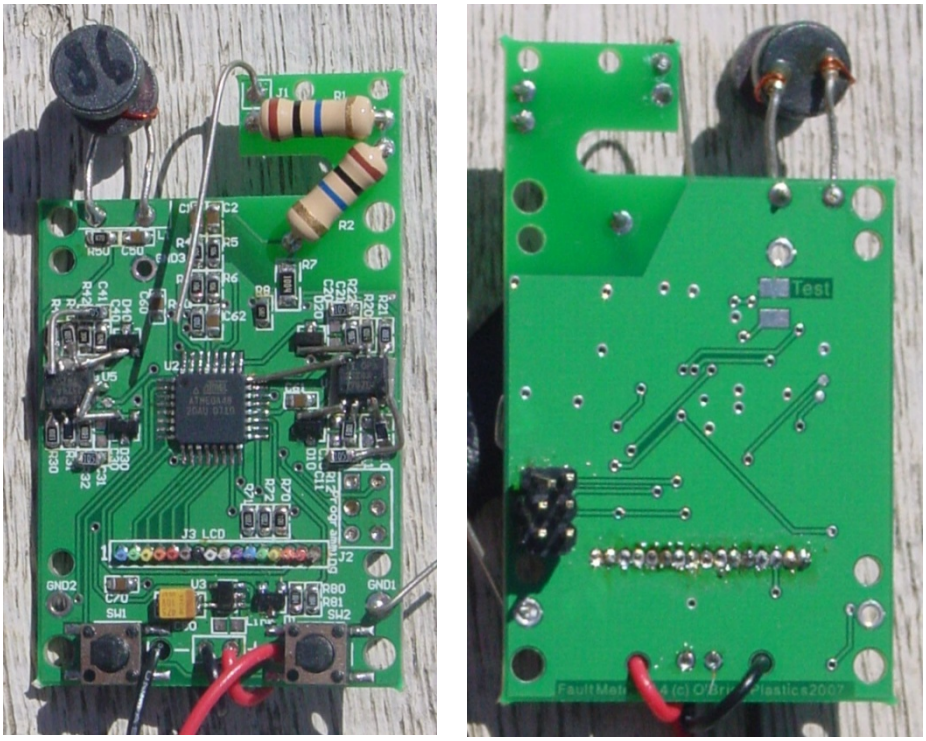


Figure 4-35: Third prototype construction

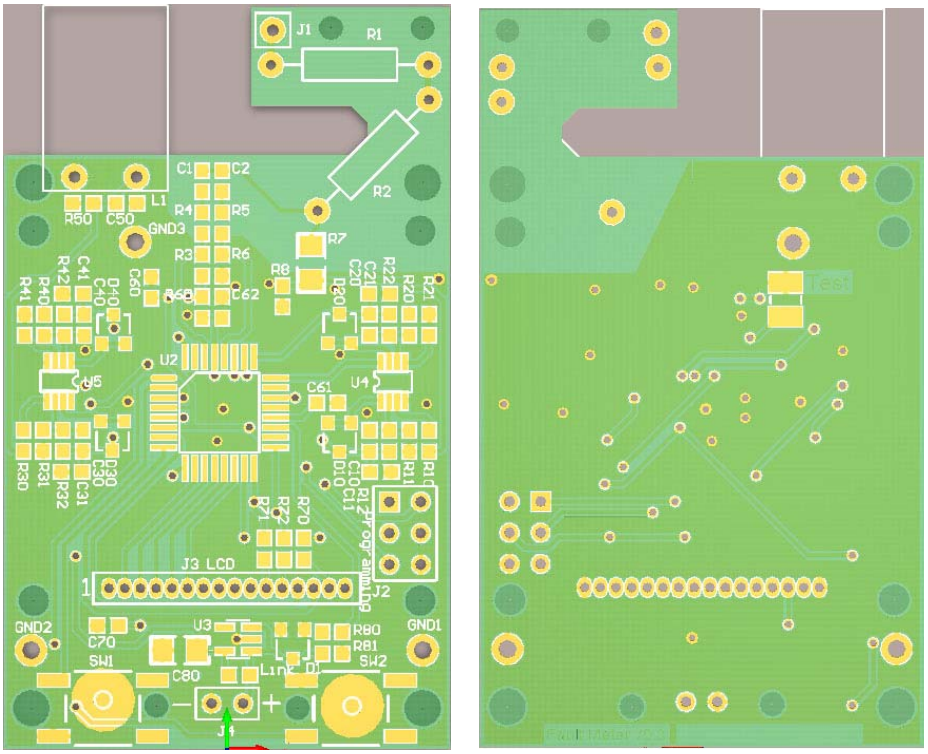


Figure 4-36: Third prototype PCB representation

A voltage regulator has been used again to allow this circuit to operate from a 9V battery, and to provide an accurate ADC reference. The ADC in the ATmega48P microcontroller can sample at 13kSps,[18] or one sample every 77µs. Sampling all four analogue inputs therefore takes at least 308µs, longer than most energiser pulses. Protection against reverse battery connection was added in the form of a diode to ground before the voltage regulator, designed to shunt current away from the regulator (which contains a reverse protection diode that passes damaging negative voltages directly to the microcontroller).[25]

Power to all peripherals is supplied directly from the microcontroller I/O pins, allowing full software control of power management.

4.11.3.1 Third Prototype Trial Results

Pulse detection worked reliably and consistently, as it did for the previous prototype (once modified). Interrupt generation in the microcontroller occurred without false indications or noise triggering.

The voltage and current peak detect circuits worked well, but as with the first prototype (see section 4.11.1), the rise time of fence pulses influenced the accuracy of the results. Speed up capacitors were used in various values in an attempt to compensate the op-amp (a pair of LMV824M op-amps) circuits with minimal success.

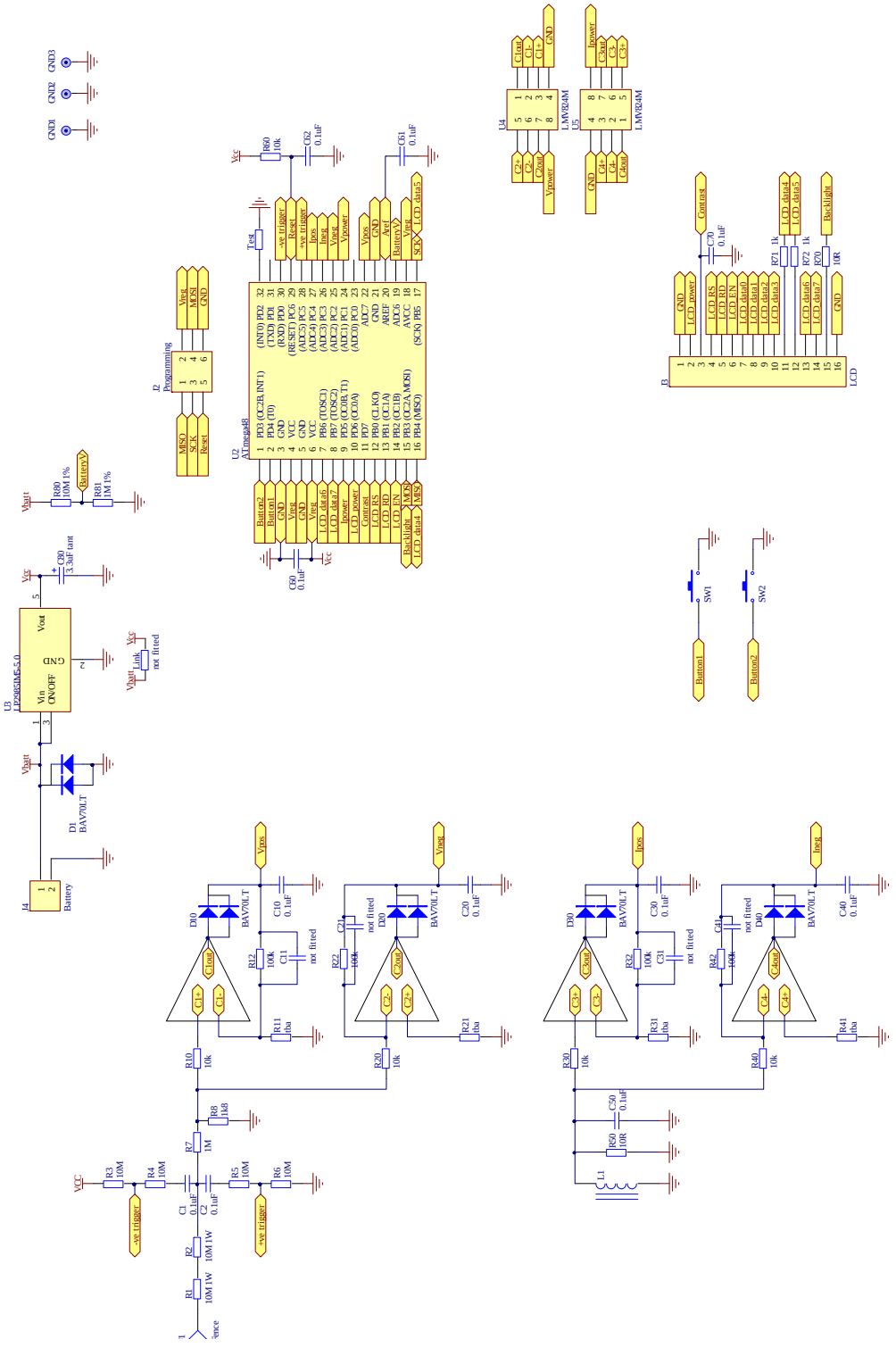


Figure 4-37: Third prototype schematic

4.11.4 Fourth Prototype

The primary purpose of this prototype was to evaluate the OLED display that was selected in section 4.5.1, and to provide a development board that would allow software development on the user interface design.

The PCB from the previous prototype was modified to remove connections for a 16x2 character LCD and new connections and support components added for the OLED display (see Figure 4-38 and Figure 4-39). A 30-pin Zero-Insertion Force (ZIF) socket was used to connect to the OLED.

Due to the limited number of microcontroller I/O pins available, the OLED display was connected using its Serial Peripheral Interface (SPI). The high LED driver voltage was supplied directly from the battery (see schematic in Figure 4-40).

4.11.4.1 Fourth Prototype Trial Results

The OLED display was bright enough to provide good readability indoors and on overcast days, but not bright enough to be easily read in direct sunlight. A makeshift 15V battery was created from 10 AAA batteries and substituted for the 9V battery. This dramatically increased display brightness, making it clearly visible in all lighting conditions.

Software routines were developed to operate the OLED and display sample screen shots (bitmap files were created for this purpose using Adobe Photoshop). The screen refresh times between sample screen displays was very noticeable (800ms to display), necessitating that a faster connection method is used for the OLED.

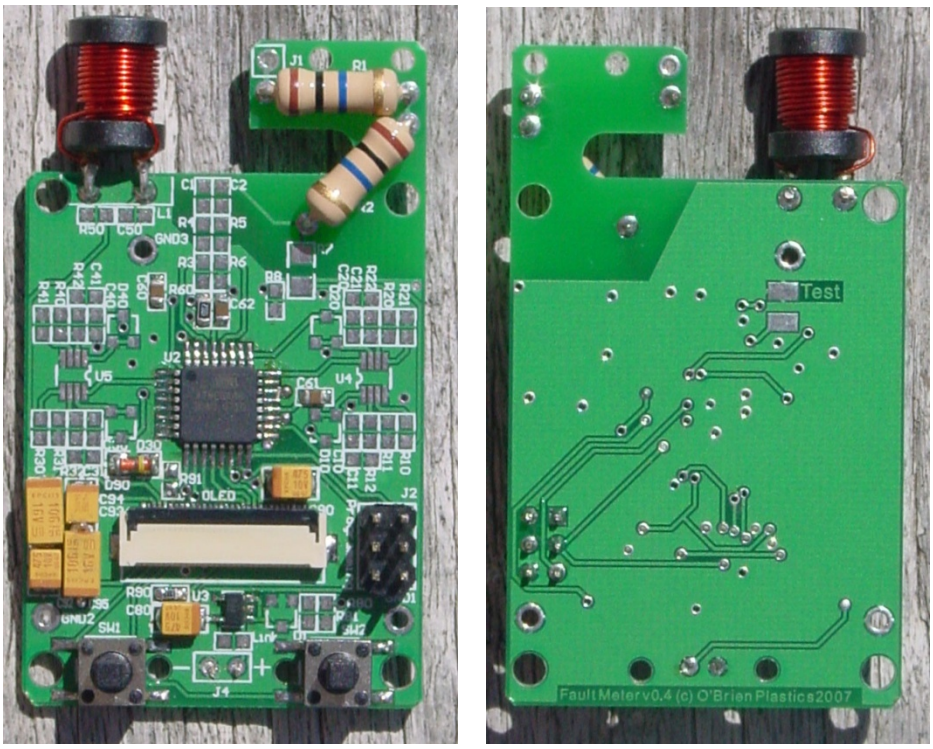


Figure 4-38: Fourth prototype construction

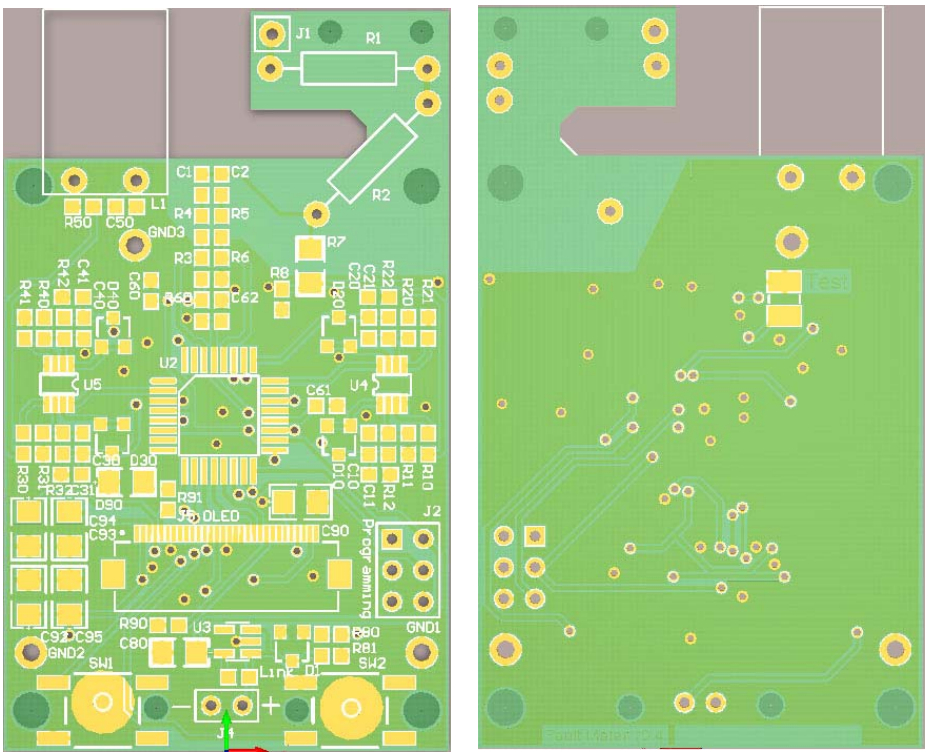


Figure 4-39: Fourth prototype PCB representation

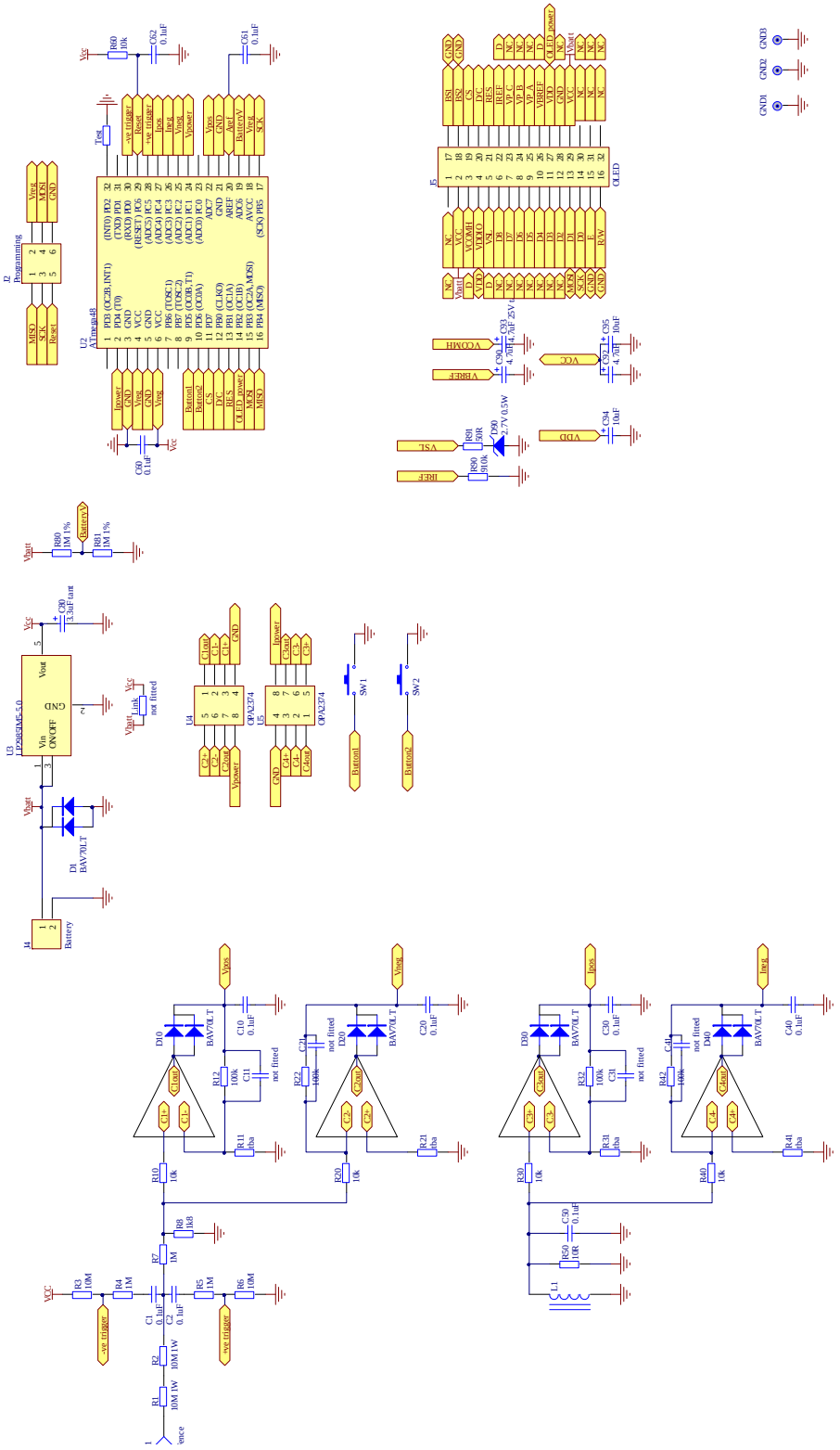


Figure 4-40: Fourth prototype schematic

4.11.5 Fifth Prototype

Necessary changes determined from prototypes three and four were implemented in this prototype. The peak detect circuits use the OPA2374 op-amp with a modified footprint from the previous op-amp (see Figure 4-41 and Figure 4-42), and the speed-up capacitors have been removed from the feedback network of each detector. Bleed resistors were placed across each of the peak detector storage capacitors (R13, R23, R33, R43, C10, C20, C30 and C40 in Figure 4-43) to ensure that they are discharged when each pulse arrives to prevent previous pulses reading affecting the current reading.

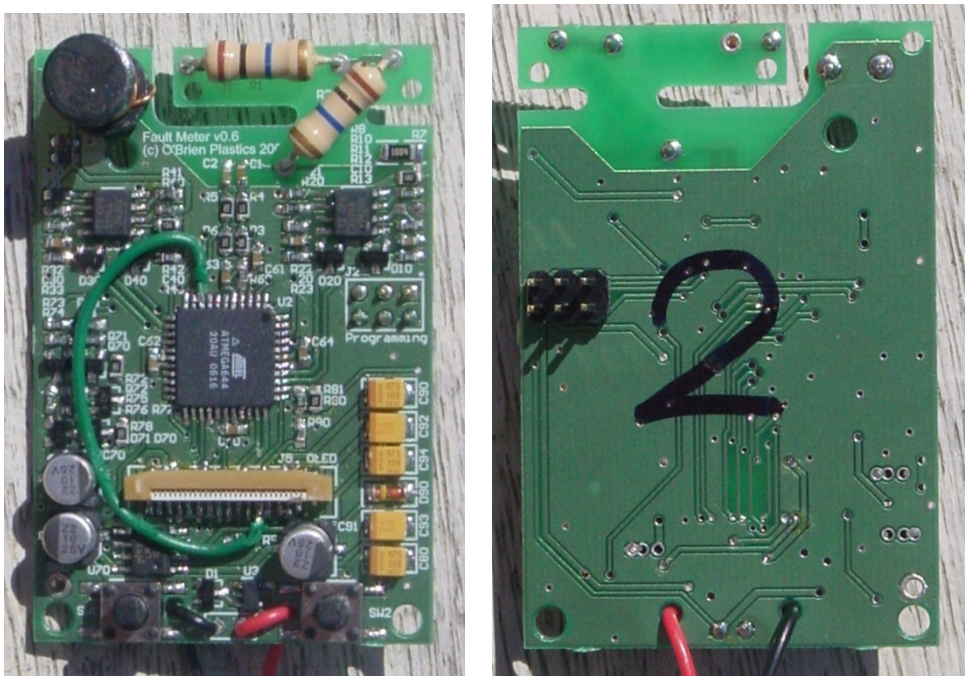


Figure 4-41: Fifth prototype construction

A new microcontroller (ATmega324P) with more I/O pins was used in place of the ATmega48P. Both microcontrollers contain the same core and instruction set, however the ATmega324P has additional internal peripherals and 12 extra I/O. This allows the OLED display to be connected using its parallel communication bus, increasing data throughput 64 times over the previously used SPI bus.

The OLED display ZIF connector was substituted with a vertical ZIF header, allowing better placement of the display when inserted into a plastic casing.

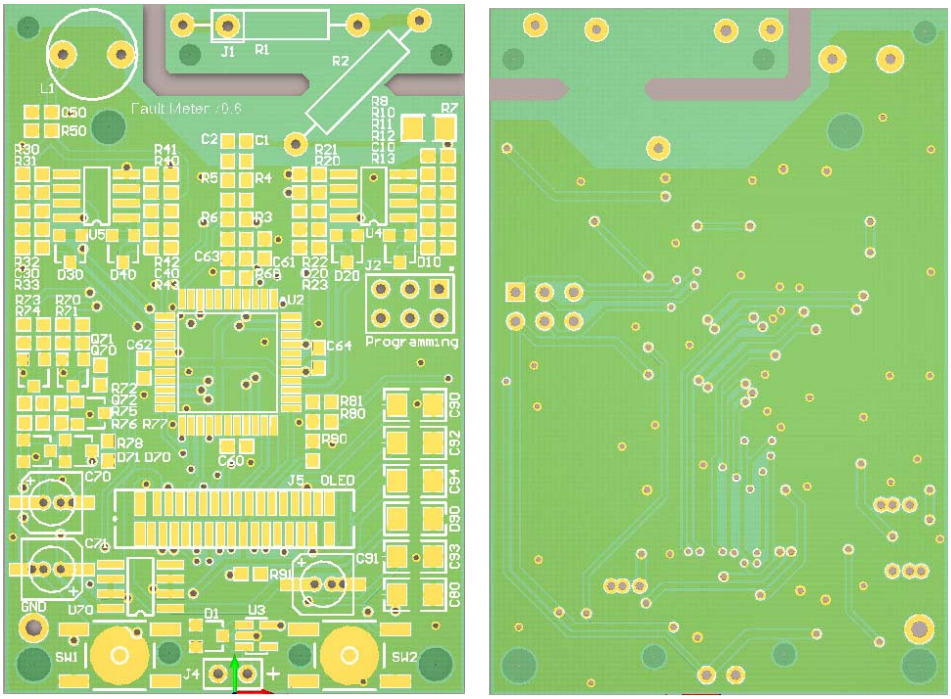


Figure 4-42: Fifth prototype PCB representation

A 15V power supply has been added which utilises a voltage doubler circuit and 15V linear regulator. This supply is to ensure that the OLED display is bright at all times, even when the battery is nearly flat.

Finally, the current sensor orientation was altered from horizontal to vertical orientation. Originally, the case design would have provided a wire guide that required the user to place the fence wire on top of the device; therefore, the sensor had a horizontal orientation. Further thought about case design has revealed that it is safer to trap the fence wire at the top end of the case, so that if the user slips they are not likely to come into contact with the fence wire.

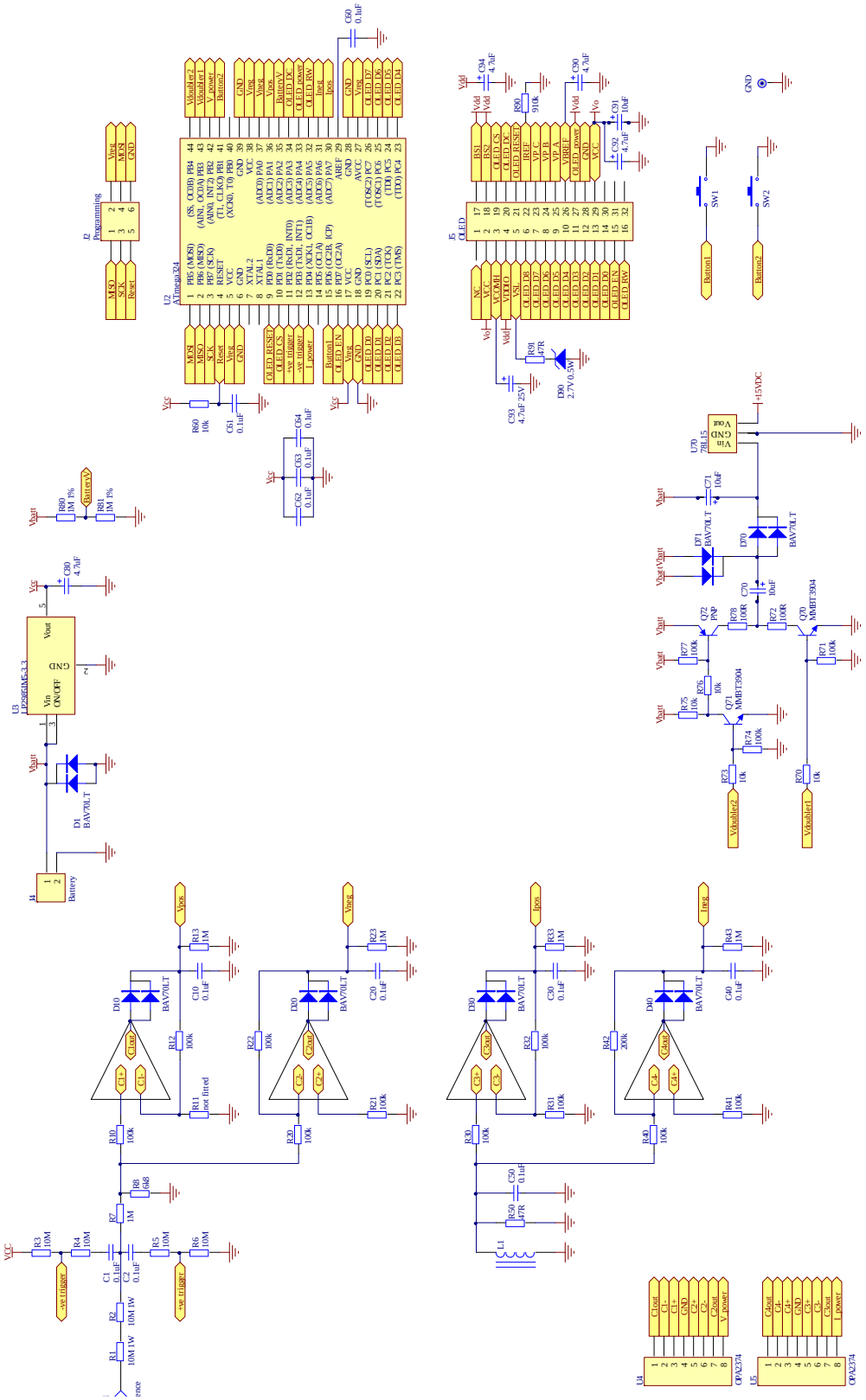


Figure 4-43: Fifth prototype schematic

4.11.5.1 Fifth Prototype Trial Results

The revised peak detect circuits based on the OPA2374 op-amp worked very well, accurately tracking fence voltages and currents over a wide range.

This prototype was taken to several countries around the world, and was destroyed by customs in America when the battery was connected in reverse. Subsequently, the input protection circuit must be changed to a non-destructive type.

OLED brightness is very good, and remains at its maximum level until the battery terminal voltage falls to 8V. The brightness then lowers proportionally as the battery voltage decays. The 15V power supply efficiency was found to be poor when the OLED power consumption was compared to battery current (around 40% efficiency).

When the 15V OLED power is disabled (device is in low power mode), the circuit consumes 6mA. A path exists through D70 and D71 from the battery to the voltage regulator U70. This regulator consumes power at all times when supplied with an input voltage. The combination of low efficiency and the requirement to redesign the circuit to correct the high power consumption while disabled made this design obsolete.

The vertical OLED connector places the display in a much better location without placing any strain on the ribbon cable.

4.11.6 Final Prototype

Only relatively minor changes were made to the previous prototype, apart from a new 15V power supply design. The new design uses a DC-DC converter controller and a small number of supporting components to produce 15V from the regulated 3.3V supply.

Solder mask in the bottom left corner of the PCB has been left off to allow electrical connection to an earth stud on the top of the case, and to a large conductive film on the inside bottom of the case (capacitive earth reference) – see Figure 4-44 and Figure 4-45.

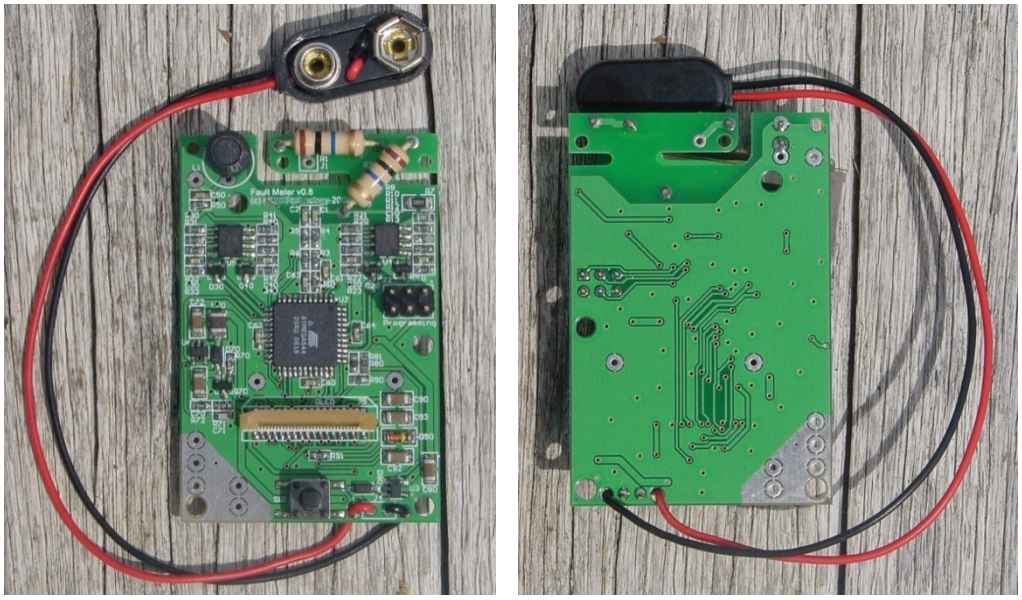


Figure 4-44: Final prototype construction (without display)

The reverse battery protection diode (D80) has been moved from across the battery, to a series connection with the 3.3V regulator, preventing destruction of the device when a battery is connected in reverse (see Figure 4-46).

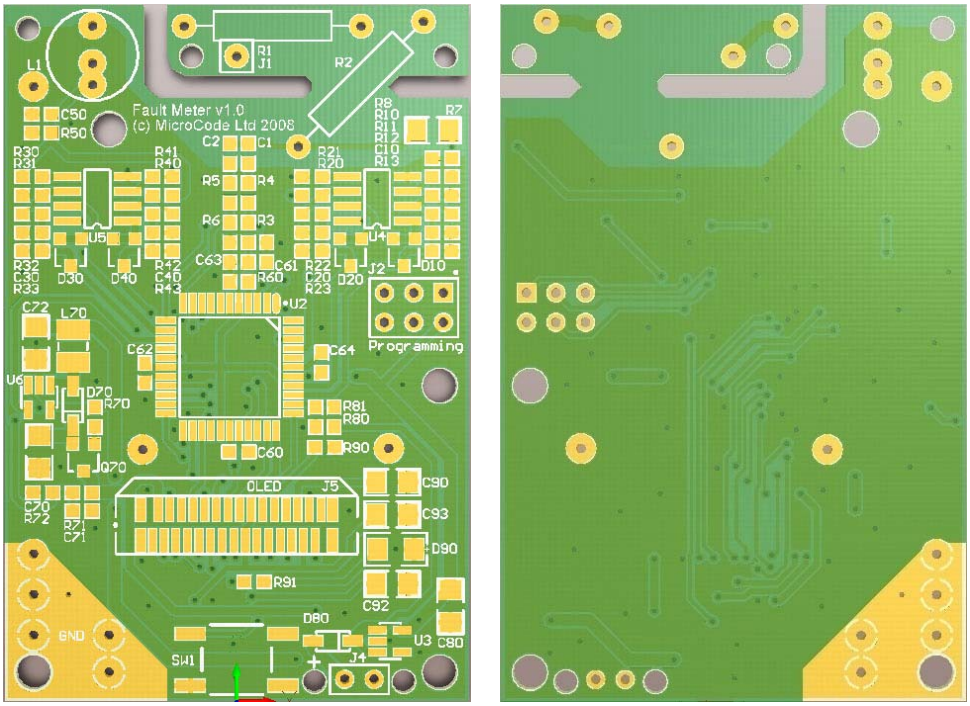


Figure 4-45: Final prototype PCB layout

4.11.6.1 Final Prototype Trial Results

Connecting a battery in reverse no longer has any affect on the circuit (D80 is reverse biased, preventing any current flow in the circuit).

The 15V power supply has an operating efficiency of over 80%, and draws zero current when disabled. Testing shows that full OLED brightness is maintained until the input voltage falls below 3.5V, however a typical 9V battery will stop providing sufficient current before its terminal voltage.

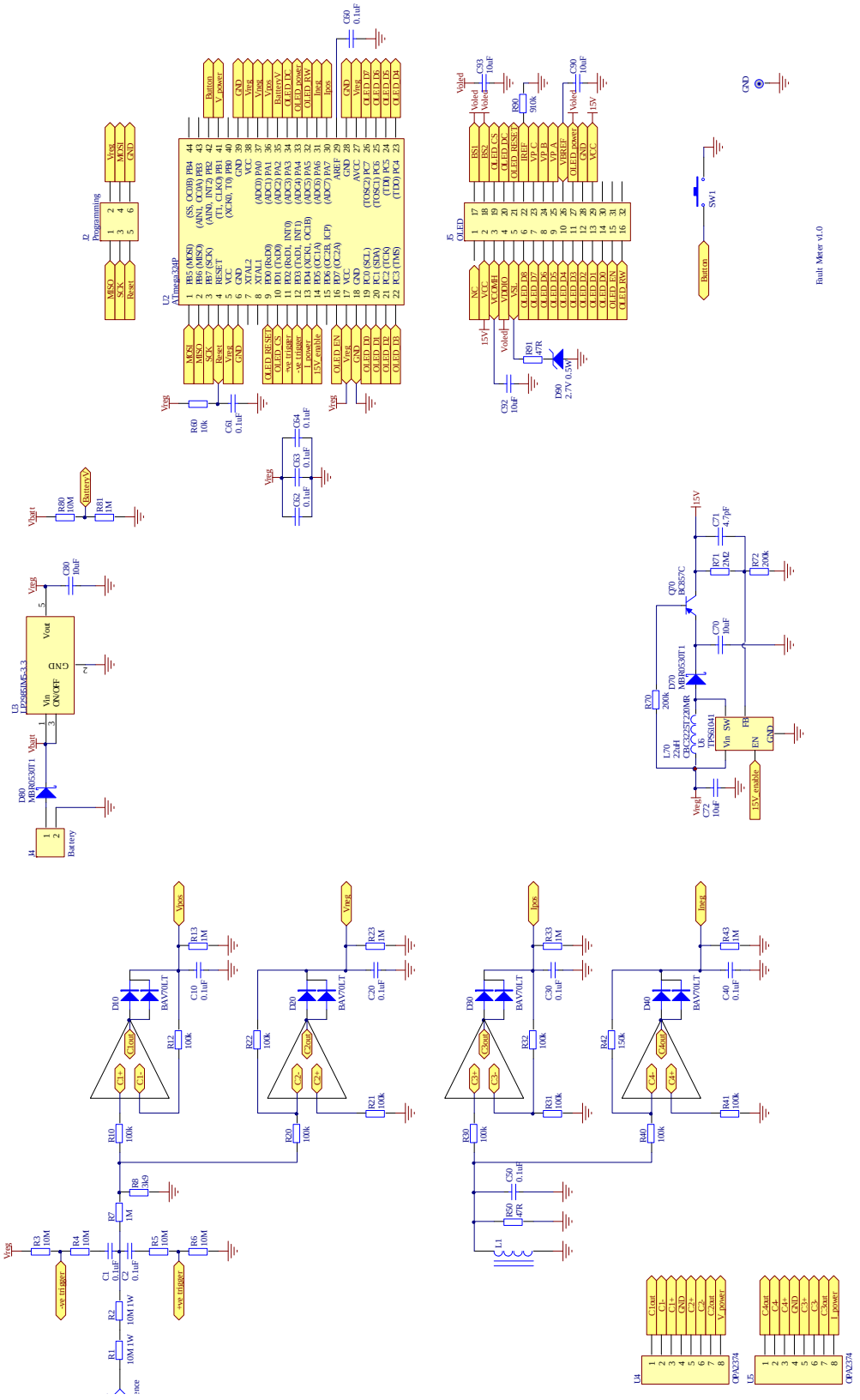


Figure 4-46: Final prototype schematic

4.12 Summary

Several changes were necessary to the anticipated hardware block diagram in Figure 4-1 to accommodate obstacles encountered during development. The final hardware block diagram is given below in Figure 4-47. Changes include separate positive and negative peak detection circuits for both voltage and current, and a separate power supply for the OLED display.

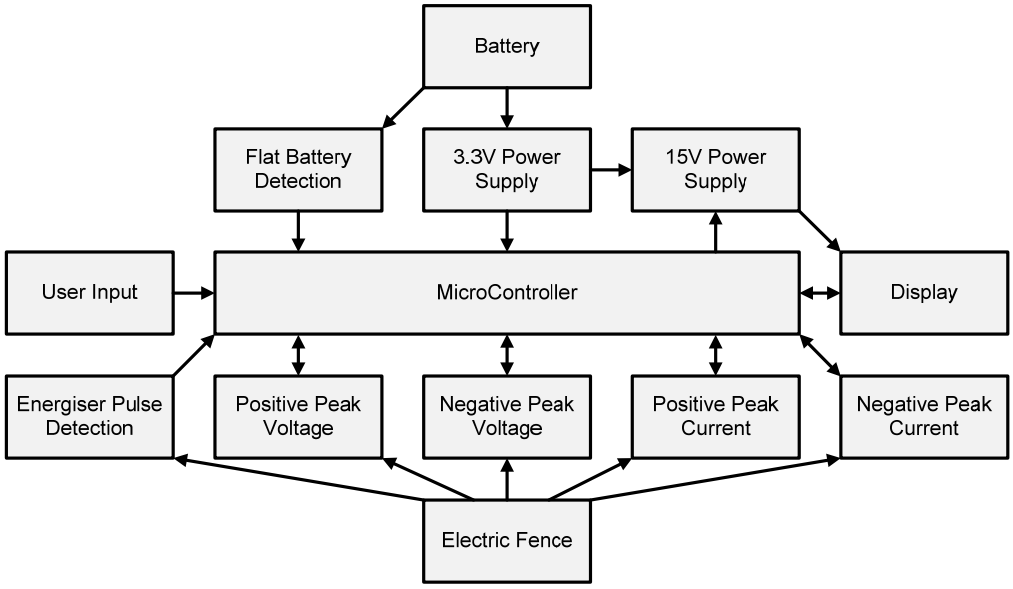


Figure 4-47: Hardware block diagram

The final prototype satisfies all requirements that were determined in Chapter 3, as well as those in section 4.1. Fence pulses are reliably detected, peak voltage and current values are accurately measured, power consumption is very low when the device is idle, the display is user friendly, and all components used are standard, readily available commercial parts. Figure 4-48 shows the final electronics design fully assembled with battery and OLED display attached.

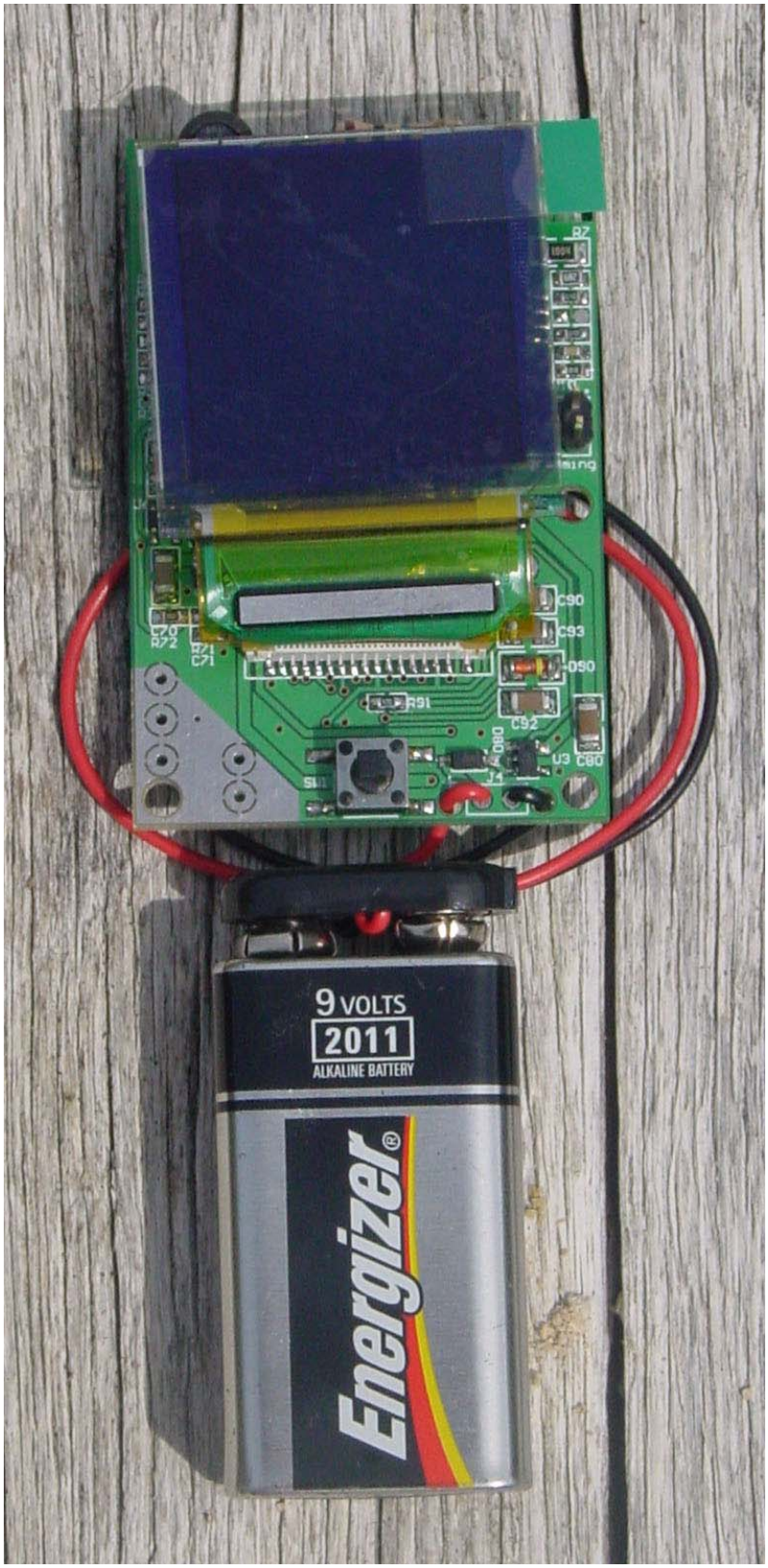


Figure 4-48: Final prototype construction (fully assembled)

5

Chapter 5

Software Design

5.1 Introduction

The primary function of the microcontroller in this design is to co-ordinate the hardware modules to properly control and time the measurement of electric fence pulses.

Each hardware module provides either an input to, or an output from the software (see Figure 4-47), necessitating that the microcontroller carries out all control, measurement and interface operations. The software requirements are largely determined by the hardware design and sequence of events that occurs before, during and after an electric fence pulse.

Electric fence pulses last at most 10ms as mandated by international standards[13], and can be as short as several microseconds. To reliably detect and measure every pulse the software must be able to react rapidly to events, requiring that interrupts be used for detection of events. This allows a simple state machine model in the main

foreground loop that responds to event data generated by interrupt routines.

Another important function performed in software is power management. To be a commercial success the single 9V battery powering the device must last as long as possible. Since the device spends most of its time in a dormant state, the software must disable all unnecessary peripherals. In addition, power must be rapidly reapplied when an electric fence pulse is detected in order to capture it.

As discussed in section 4.7.3, the software must take multiple analogue readings concurrently until a steady measurement value reached to ensure accurate measurements. This method applies to all analogue sources.

Finally, the graphical display is dynamically driven with image data generated as required, including colour and data location. Additional features such as a company logo and historical data are also displayed.

Figure 5-1 shows an outline of the software architecture used. It can be seen that some functions will run continuously, while others are only needed while the device is operating. Detection of an electric fence pulse or button press will place the software into the running state, while no events for thirty seconds will cause the resumption of the sleep state. The following sections go into each software routine in more detail.

5.2 Initialisation

The microcontroller is initialised to a known state after every reset. During this routine, the source of the reset is determined, since knowing the cause of the reset can indicate whether a problem exists in the

software during development, detect if the battery was disconnected and/or replaced, or even if a hardware failure is developing. At the end of this routine the device is ready to operate and will transition to the running mode.[28]

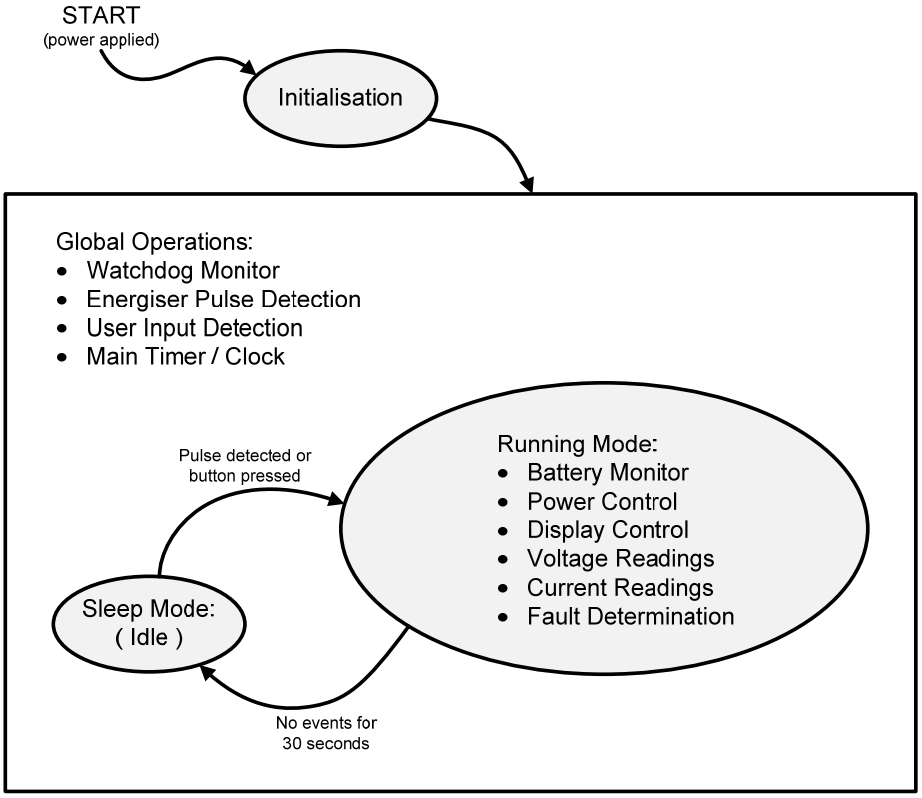


Figure 5-1: Outline of software architecture

5.3 Global Operations

Several software components must operate continuously over the life of the unit. These functions are:

- Watchdog Monitor
- Energiser Pulse Detection
- User Input Detection
- Main Timer / Clock

5.3.1 Watchdog Monitor

To improve software reliability an independent timer known as a watchdog is used. Its purpose is to prevent lockups and erratic behaviour by resetting the microcontroller if the watchdog timer is not regularly reset by the application main loop.[28]

In the ATmega324P microcontroller, a hardware-based watchdog is available with its own oscillator and registers. A specific timed sequence is necessary to set the watchdog (to prevent accidental changes), but once set only a simple periodic “reset watchdog” command is necessary.

A timeout period of eight seconds (the longest available) was selected to minimise power consumption while sleeping, and will be fast enough since the software is not required to perform any safety related tasks.

5.3.2 Energiser Pulse Detection

An energiser pulse could occur at any time, requiring that detection is done asynchronously using interrupts (polling would be too slow). More than one pulse detection is likely to occur from a single energiser pulse (a typical pulse has both positive and negative voltage swings, and can contain significant ringing).

When a pulse is first detected, the time of detection and pulse polarity is recorded. Pulse detection interrupts are then disabled for 10ms to provide a blanking period to prevent multiple detections from a single pulse (a single pulse may have several positive and negative edges). During the interrupt, power is applied to the voltage and current peak detect circuits to minimise the delay before they become active.

A flag is set to notify the main loop that an event has occurred and requires further processing.

5.3.3 User Input Detection

As with energiser pulse detections, user button presses could occur at any time and must be detected asynchronously using an interrupt. Mechanical buttons require debouncing to prevent multiple detections from a single press – this can be done using a simple delay to check that the button is still pressed a short time after the initial detection.

If the device is in sleep mode then a button press will turn on the device to the last displayed screen. If already in running mode, then a button press will toggle between the most recent fence reading and a historical display of the last several readings.

5.3.4 Main Timer / Clock

This module provides a clock with millisecond resolution to allow timing and co-ordination of events. Long time periods are measured for logging data, such as:

- Time spent sleeping / running (lifetime count)
- Number of button presses (lifetime count)
- Maximum voltage detected
- Lowest operating battery voltage registered
- Number of times battery has been replaced (lifetime count)
- Number of unexpected resets (lifetime count)

These logs are invaluable when a unit is returned for repair to determine whether the device has been abused, or if the software detected any faults, and during development for debugging.

5.4 Sleep Mode

This is the off state of the device. When sleeping, everything except the microcontroller and 3.3V regulator are disabled. All microcontroller internal peripherals apart from the watchdog and pin change interrupts are powered down, reducing the overall current consumption to almost nothing (measured value of 3.5uA).

Only global operations are running in this mode, allowing the devices battery to last a very long time when the unit is not in use (up to 20 years using manufacturers' capacity rating[24] and assuming an infinite battery shelf life). Figure 5-2 shows a state diagram of the sleep mode.

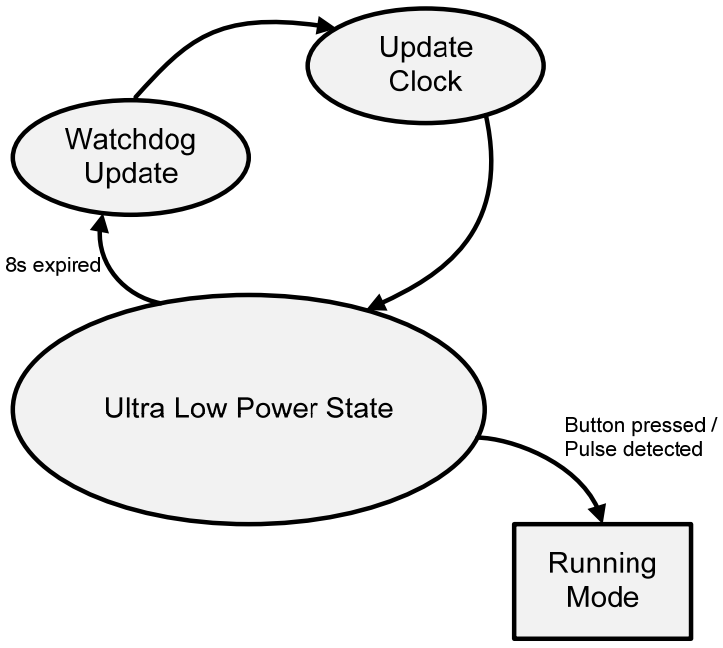


Figure 5-2: Sleep mode event states

5.5 Running Mode

This is the on state of the device. In this mode, all parts of the hardware are fully powered and operational. The purpose of this mode is to detect and display electric fence data collected, indicating to the user whether a fault is present, and if so, which direction to the fault. At the same time, the status of the internal battery is checked and the user notified if it requires changing.

The device will only enter the running mode if the user presses the button, an energiser pulse detection occurs or the microcontroller resets unexpectedly.

While in the running mode, several software modules run to achieve the output described above (see Figure 5-3).

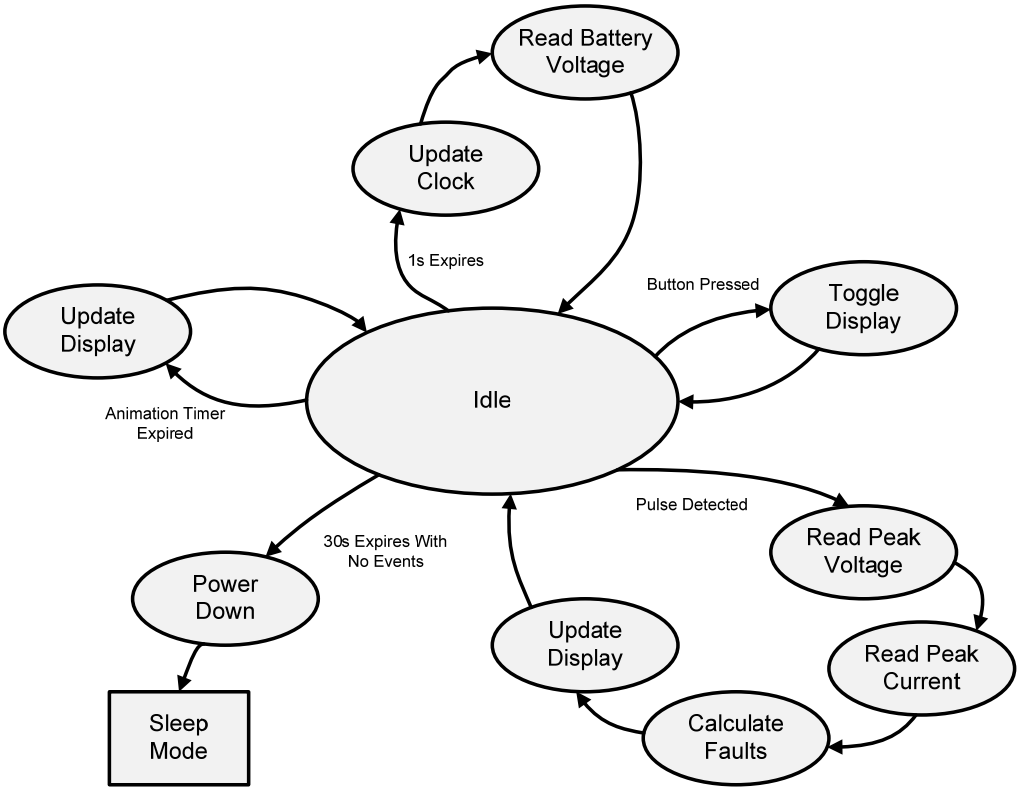


Figure 5-3: Running mode event states

5.5.1 Battery Monitor

The battery monitor uses the microcontrollers ADC with fixed internal reference to read the existing terminal voltage of the battery. As described in section 4.7.3, the value of the resistor to ground used to sample the battery voltage is 100 times higher than recommended by the ADC datasheet[18]. Because of this, to prevent inaccurate readings it is necessary to take continuous readings and maintain a running average until several consecutive readings are very similar.

This method allows poor readings to be rejected and ensures that the result always reflects the actual voltage on the ADC input pin. The readings must be taken continuously to allow the ADC sample hold circuit to charge to a steady state.

Readings are only taken while the device is in running mode. There are two reasons - the first is that when in sleep mode there is not any method of signalling a flat battery (the display is off). The second reason is that the battery terminal voltage is higher when unloaded, so the device must be fully powered to get a true indication of the charge remaining in the battery.

5.5.2 Power Control

It is important for the device to operate correctly that the hardware modules and microcontroller peripherals are powered at the appropriate times, and that power is removed in the sequence that produces the lowest current consumption while in sleep mode.

When transitioning from sleep to running mode, power is applied to all modules immediately (except the 15V power supply) to ensure that the

device can measure the first detected pulse. If the device was woken from sleep by detection of a pulse, then the peak detect circuits will already be powered. The 15V power supply is started after the OLED logic has been initialised to prevent damaging the OLED, as per the manufacturers' datasheet. [22, 29]

In order to achieve minimum power consumption during sleep mode, it is necessary to follow a shut down sequence otherwise microcontroller peripherals may remain active.[18] In addition, the OLED display can be damaged or its life shortened if not correctly shut down.[30]

The shutdown sequence is as follows:

- Remove power from voltage and current peak detect circuits
- Disable 15V power supply
- Send shut down command to OLED display logic
- Set ADC to use external voltage reference (disables internal reference)
- Disable all interrupts except pin change interrupts
- Set unwanted peripheral configuration registers to off
- Set power down registers to disable power to peripherals
- Wait one second
- Remove power from OLED display logic
- Enter power down sleep mode

5.5.3 Display Control

The majority of the microcontroller's resources and processing time are consumed by the OLED display control and image generation routines. The display has 128 x 128 pixels, and each pixel has 8-bit colour for red,

green and blue. Therefore, the data transferred for each display frame is 49,152 bytes (not including control commands).

There are two major display control areas – configuring / controlling the OLED logic, and generating images for display.

5.5.3.1 OLED Configuration and Control

The OLED datasheet[22] specifies a specific initialization sequence to follow after applying power. In addition to this, each display update requires control data before and after the display data (such as x and y co-ordinates, clear screen, etc).[29] These routines are available from the manufacturer's datasheet and application note, requiring only minor modification for this project.

5.5.3.2 OLED Display Image Generation

The OLED display is a graphical panel capable of up to 16.7 million colours.[31] Each frame of the display must be generated dynamically by the microcontroller and then sent one byte at a time to the OLED.

Some aspects of the display are fixed at compile time of the software, such as font (see Figure 5-4), character size, special characters (see Figure 5-5) and a company logo. Others are calculated according to the data to be displayed, such as character location, arrow direction and font colour (green = good fence, red = problem on fence).

Figure 5-4: Selected font

Figure 5-5: Special characters

The OLED logic will display an image continuously once received, and therefore it is only necessary to update the display if animation is present or a new image is ready. Figure 5-6 shows a sample screenshot with good voltage reading indicated in green and high fault current indicated in red.

Figure 5-6: Sample screen

At the time of writing this thesis, only voltage, current and direction to fault is displayed. Because the display is entirely graphical, it is possible to introduce fault descriptions and even pictures of likely faults for the user to search for. Such advanced features would require additional study and extensive field trials to implement and are beyond the scope of this thesis.

5.5.4 Voltage and Current Readings

After each energiser fence pulse, the voltage and current peak detect circuits will have stored their respective values as a voltage in a small capacitor.

It is important that the analogue values are read as soon as possible to prevent inaccuracy due to the stored capacitor voltage decaying, but also that the voltages are not read before the pulse is complete.

The current electric fence energiser standard[13] sets the maximum energiser pulse duration at 10ms, therefore the peak voltage and current readings will be measured 10ms after the pulse is first detected.

For reliability, each ADC reading will be performed several times and compared to a running average, requiring several similar readings before being accepted as an accurate ADC reading.

5.5.5 Fault Determination

In addition to fault-finding, fence testers are frequently used to verify proper operation of an electric fence. If the fence is in good condition then only a small peak current will be detected. Based on the voltage and current readings, the device must decide whether a fault is present, and if so, which direction to indicate to the user.

To produce an effective electric fence, it is generally accepted that the fence voltage must be 3,000V or greater.[9, 10, 14] Therefore, displayed voltage readings above 3kV are green and readings below are red.

Current readings are less predictable than voltage readings and can vary significantly between fences. It is important that the user is familiar with the normal operating conditions of the fence to judge whether a current reading is too high or low. Higher than normal current readings indicate short circuit faults, lower than normal current readings with the same voltage usually indicate a poor connection or broken wire.[9]

The direction of the fault is determined using the polarity of the current and voltage readings (as detected by the voltage and current pulse detect circuits). The following formula is used:

$$\boldsymbol{Fault\ Direction = Voltage\ Polarity \times Current\ Polarity} \quad \textbf{[5-1]}$$

A positive result indicates the fault is to the left, and a negative result to the right.

5.6 Software Language and Compiler

The software will be written in the Embedded C language. The microcontroller used has 64kB program space and 4kB RAM, so resources are plentiful. Using this environment allows for portable and easy to maintain code to be written quickly, and debugged using a variety of simulation tools.[32-34]

Borland C++ and CodeVision AVR compilers were used to develop the software. Borland allowed fast, computer based checking of the code, while CodeVision AVR produced compiled code that is compatible with the microcontroller.

6

Chapter 6

Experimental Results

6.1 Introduction

All readings presented in this chapter are based on measurements taken using the final prototype (completed unit) presented in section 4.11.6, and the following test equipment:

- Agilent 34401A Digital Multi-Meter (DMM)
- Fluke 123 oscilloscope
- Agilent HP34300A 40kV high voltage probe
- Fluke 177 DMM
- Benchtop 0-30V, 3A variable power supply
- High voltage load box (1Ω to 10kΩ in 1Ω increments)
- O'Briens 52/36 battery powered energiser
- 0.1Ω precision resistor

The device under test was powered by a single Energizer alkaline 9V battery at all times (except for battery voltage accuracy readings, where the benchtop power supply was used).

6.2 Device Verification

Each component shown in the hardware block diagram (Figure 4-47) was verified during development and again once the final prototype unit was completed with operational software. The final results from each module are presented in this section.

6.2.1 Flat Battery Detection

Each battery voltage measurement taken by the microcontroller is the result of several ADC readings. The software performs consecutive ADC readings of the battery voltage divider analogue input until the last five readings taken are all within 1% of their combined average. This method reduces the effects of noise, and compensates for the high input impedance of the sampling resistive divider as described in section 4.7.3.

The microcontroller's internal 1.1V bandgap voltage reference is used for all battery analogue measurements, as it is the only reference that is completely independent from the battery voltage.

The measurements presented in Table 6-1 were taken while the device was fully powered and displaying a regular screen (with the addition of the current ADC value). The battery was substituted for an adjustable DC power supply. The "calculated battery voltage" column in Table 6-1 is derived from the ADC value using the following equation:

$$V_{battery} = \frac{ADC\ value \times ADC\ reference \times (R_{top} + R_{bottom})}{Number\ of\ ADC\ steps \times R_{bottom}} \quad \textbf{[6-1]}$$

$$= \frac{\boldsymbol{ADC\ value} \times \mathbf{1.1}\boldsymbol{V} \times (\mathbf{1}\boldsymbol{M}\Omega + \mathbf{100}\boldsymbol{k}\Omega)}{\mathbf{1024} \times \mathbf{100}\boldsymbol{k}\Omega} \quad \textbf{[6-2]}$$

$$= \frac{\boldsymbol{ADC\ value}}{\mathbf{84.63}\boldsymbol{V}^{-1}} \quad \textbf{[6-3]}$$

Where:

ADC value = reading from microcontroller ADC

ADC reference = internal microcontroller reference voltage used to R_{top} = resistor value of top resistor in battery voltage divider network (1MΩ)

R_{bottom} = resistor value of bottom resistor in battery voltage divider network (100kΩ)

Number of ADC steps = maximum number of ADC values (10-bit ADC, therefore 1024 values)

Battery Voltage (V)	ADC value (no units)	Calculated Battery Voltage (V)	Percentage Error (%)
3.01	258	3.05	+1.3
4.03	334	3.95	-2.0
5.01	421	4.97	-0.8
6.03	508	6.00	-0.5
7.02	598	7.07	+0.7
8.00	685	8.09	+1.1
9.01	762	9.00	-0.1
10.03	851	10.06	+0.3

Table 6-1: Battery voltage ADC readings

The percentage error column in Table 6-1 shows that the worst error was 2%, the best error 0.1%, and the average error 0.85%. This

deviation from the actual value is likely to be caused by the high impedance of the analogue input, however, the results are well within the accuracy requirements necessary to determine if the battery is flat. ADC accuracy is specified as +/-2 LSB, or 0.2%.[18]

Figure 6-1 graphs the actual battery voltage against the calculated voltage, along with an ideal straight-line value. It can be seen from this graph that the measurement accuracy appears small enough to be ignored.

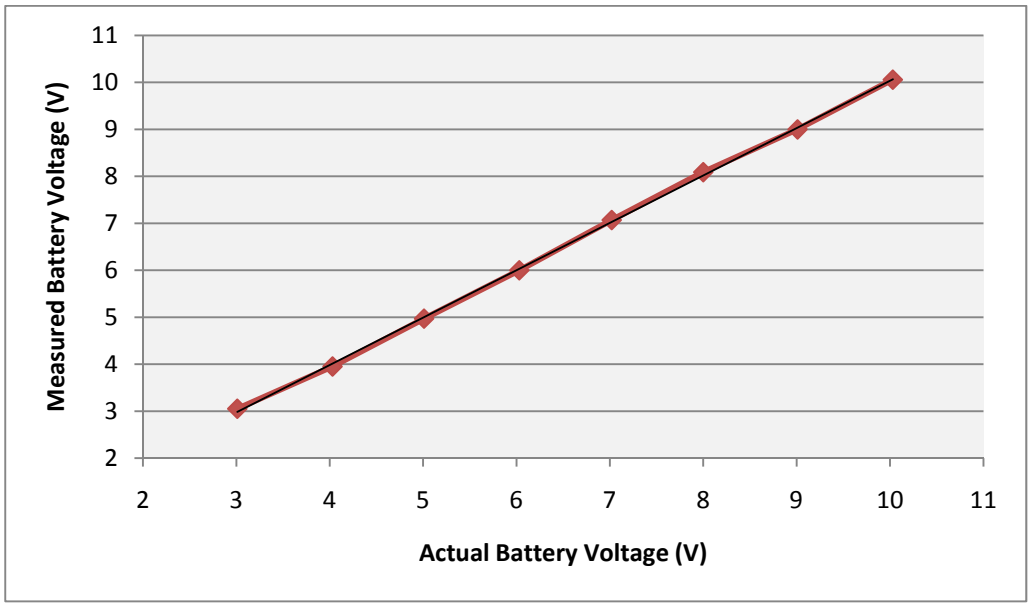


Figure 6-1: Battery voltage readings vs. Actual value

The lowest battery voltage that the device will operate from was measured as 2.5V. At this voltage, the microcontroller is guaranteed to operate reliably at 8MHz, and the 15V power supply is maintained at 14.5V. Below 2.5V, the display remains at full brightness but the microcontroller ceases to function (outside of minimum voltage for 8MHz operation[18]).

6.2.2 Power Supplies

Voltage and current measurements were taken for both power supplies over a range of loads. Battery current was also measured where applicable.

6.2.2.1 Current Measurements

Current readings were taken using a DMM in series with the 9V battery, 15V power supply, and 3.3V power supply. These readings are included below in Table 6-2.

Measuring the 3.3V and 15V supply current required cutting PCB tracks and soldering wires to either side of the cut to allow connection of the DMM. Each reading was taken using 300ms averaging to ensure only average values were recorded.

Device Status	3.3V Supply Current (mA)	15V Supply Current (mA)	9V Battery Current (mA)
Idle (powered down)	0.008	0	0.003
Active, blank screen	7.52	0	3.45
Active, normal display	117.82	20.3	54.01
Active, all-white display	172.69	30.1	79.15

Table 6-2: Power supply current consumption measurements

From these measurements, the expected operational life of a typical 9V battery can be estimated. Ultimately, battery life depends on the usage profile, as an infrequently used device will use less power than the same device used for hours a day. All life estimates in Table 6-3 have been based on a 620mAh battery capacity (taken from Energizer 9V battery datasheet[24]).

Usage Profile	Device on time	Estimated Battery Life (days)
Not used	Always off	8,611
Infrequently used	1 hour per week	79.6
Moderately used	3 hours per week	26.7
Frequently used	1 hour per day	11.5
Professional usage	4 hours per day	3
Constant usage	Always on	0.5

Table 6-3: Estimated battery operating life

6.2.2.2 Voltage Measurements

Readings were taken using a DMM connected across the negative supply terminal and regulator output, while the input voltage was supplied by a benchtop supply. Table 6-4 below contains the measurement results.

Battery Voltage (V)	3.3V Power Supply (V)	15V Power Supply (V)
3.01	2.627	14.643
4.03	3.295	14.959
5.01	3.293	14.958
6.03	3.292	14.959
7.02	3.291	14.957
8.00	3.285	14.960
9.01	3.282	14.959
10.03	3.281	14.959

Table 6-4: Power supply voltages compared to battery voltage

The 15V power supply maintained a constant voltage over the entire range of input voltages, until the supply voltage went below the 3.3V regulator dropout level.

The 3.3V power supply output voltage increased slightly with decreasing battery voltage. From 10V to 4V, the supply voltage increased 0.43% (see Figure 6-2). Because this supply is used as a reference for the ADC when measuring peak voltage and current, variations of the supply have a direct impact on the accuracy of the device.

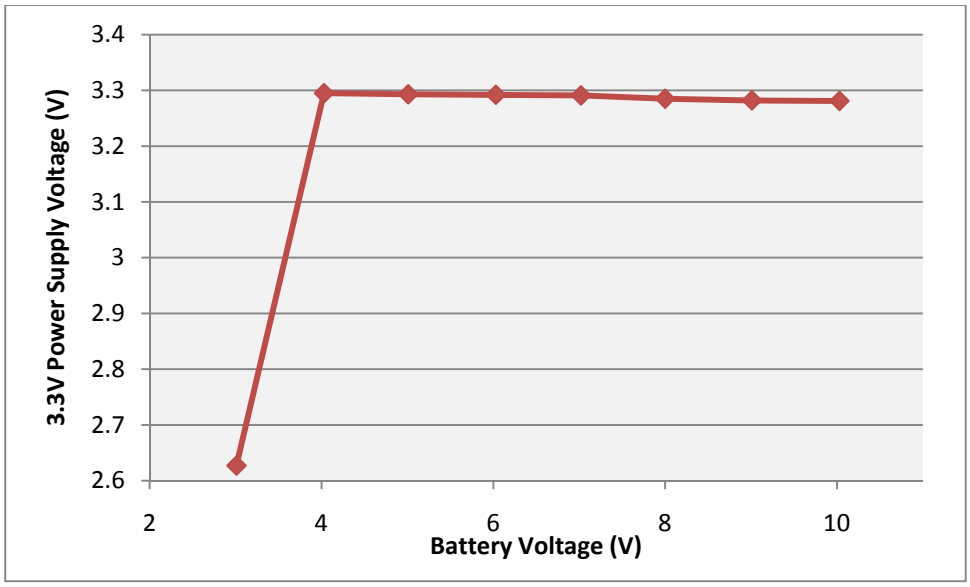


Figure 6-2: 3.3V power supply output with varying battery voltage

6.2.3 User Input

Button presses are detected and debounced in software. An interrupt is generated each time the microcontroller pin is pulled low by the button (a weak pull-up, equivalent to approximately 100kΩ, is supplied internally by the microcontroller).[18] The software debounces the button by checking that the input is still low 40ms after the button press

is first detected. See Figure 6-3 for a typical bouncing button contact waveform.

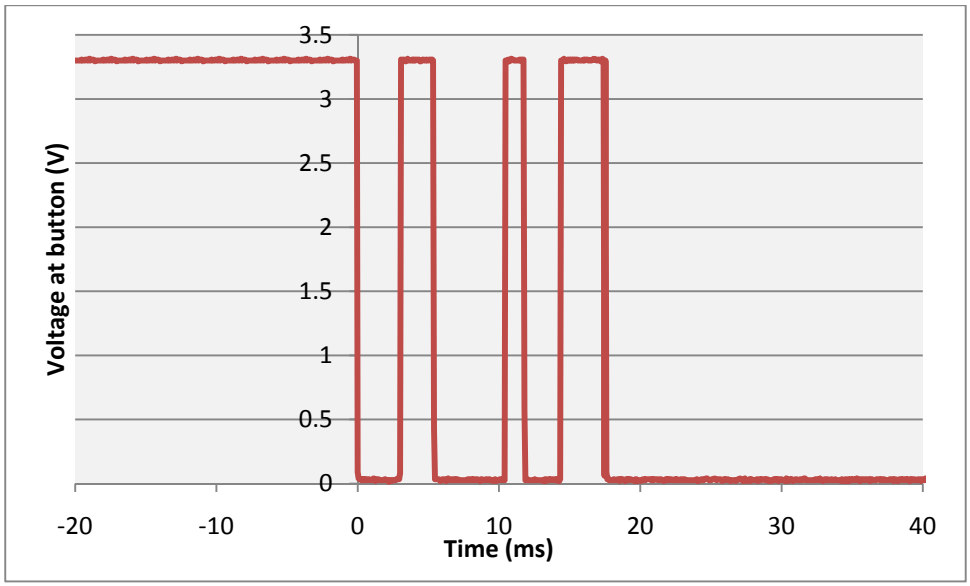


Figure 6-3: Voltage across a bouncing button contact

Each button press was detected, both when the device was on and off. Short duration noise introduced manually with a frequency generator did not generate a false button detection (even though many interrupts were produced).

6.2.4 OLED Display

Due to its visual nature, the display is one of the easiest components to confirm correct function. A short software routine was written that continuously generated random images, writing each image to the OLED controller (including updating all appropriate control registers). After writing an image, the display buffer would be read back and compared to the original image. Any data errors were logged and stored in EEPROM.

In field trials, the fault meter was left connected to an operating energiser for seven days (168 hours). At the end of the test not a single corrupted data event had been logged, indicating that communications between the microcontroller and the OLED display are stable and noise tolerant.

6.2.5 Peak Detect Circuits

This section presents measurements using the device of positive and negative voltages and currents generated by an electric fence energiser connected to a high voltage load box and 0.1Ω series resistor.

Each current measurement (fault-finder) was taken from the same fixed distance as would be present in the plastic case. Because the current sensor measures magnetic flux, distance from the fence wire influences the magnitude of the reading. Fence current readings were calculated by examining the voltage waveform across the 0.1Ω resistor.

Readings were taken at a variety of fence loads to test a range of voltage and current values, and are listed in Table 6-5.

6.2.5.1 Power-Up Time of Peak Detect Circuits

Figure 6-4 shows the time taken for the positive voltage peak detect circuit to reach a stored voltage of 560mV after power is applied to the op-amp (time = 0µs).

It can be seen from the graph that the storage capacitor reaches 90% of its final value after 0.75µs, and 100% after 1.6 µs. Since energiser pulses have a typical rise time in excess of 10µs,[4, 14] this response

time is adequate to capture all but the shortest pulses from the device's off state.

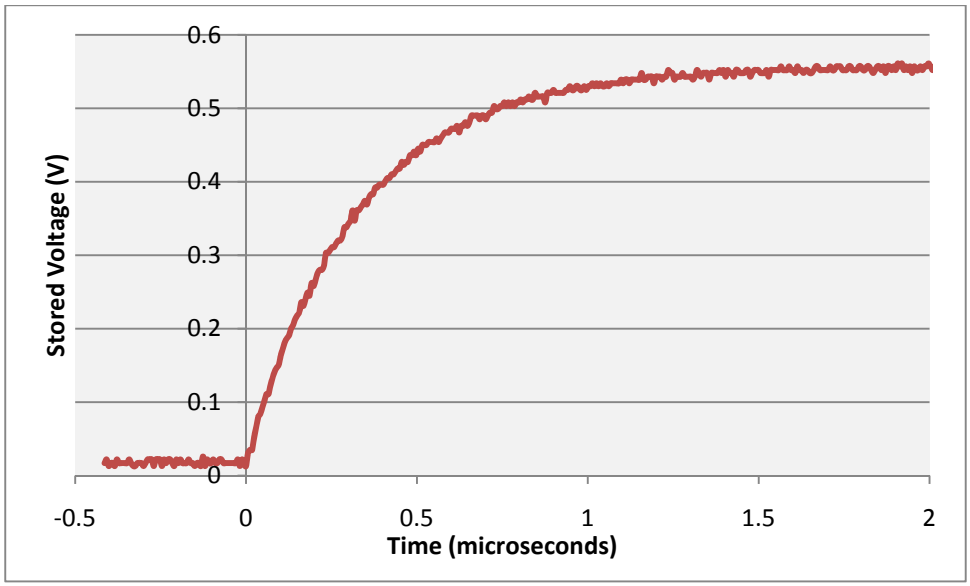


Figure 6-4: Peak detector step response to an input when power is applied

When the device is woken from sleep by a pulse, measurements can be made immediately. Other similar competitor devices must already be awake to make a measurement, and will only turn on when the first pulse is detected.

6.2.5.2 Measurement Accuracy

The measurements in Table 6-5 have been plotted in Figure 6-5 through Figure 6-8. Each graph has a trend line which was used to determine the equation to convert the raw ADC value from each circuit into a voltage value. It can be seen that readings taken without an earth connection are approximately 10% lower than when an earth connection is present.

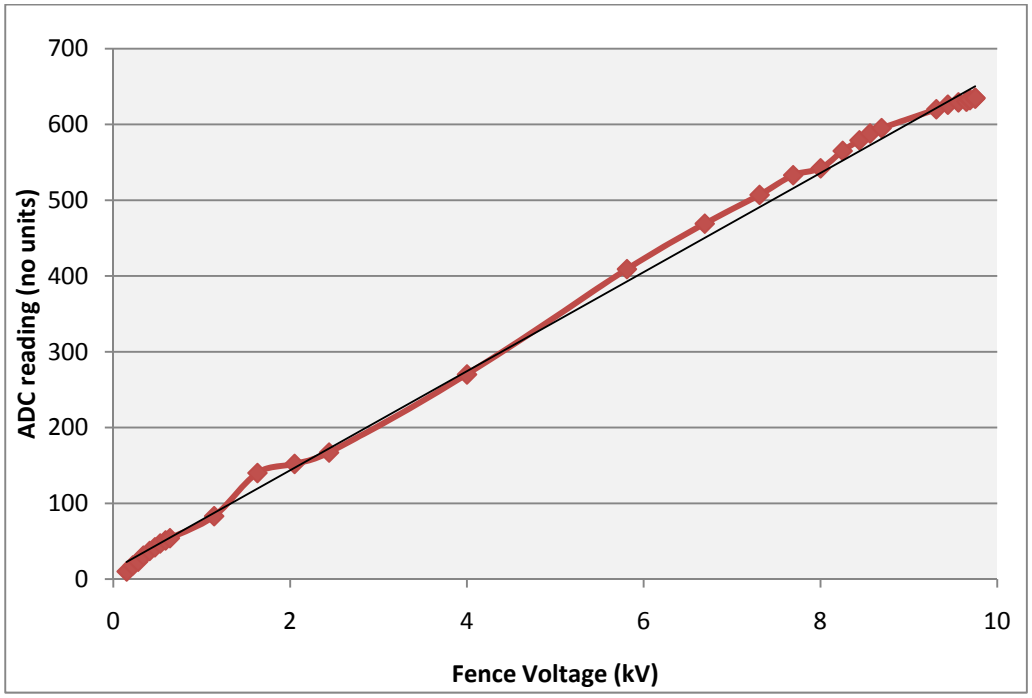


Figure 6-5: Positive voltage, ground lead connected

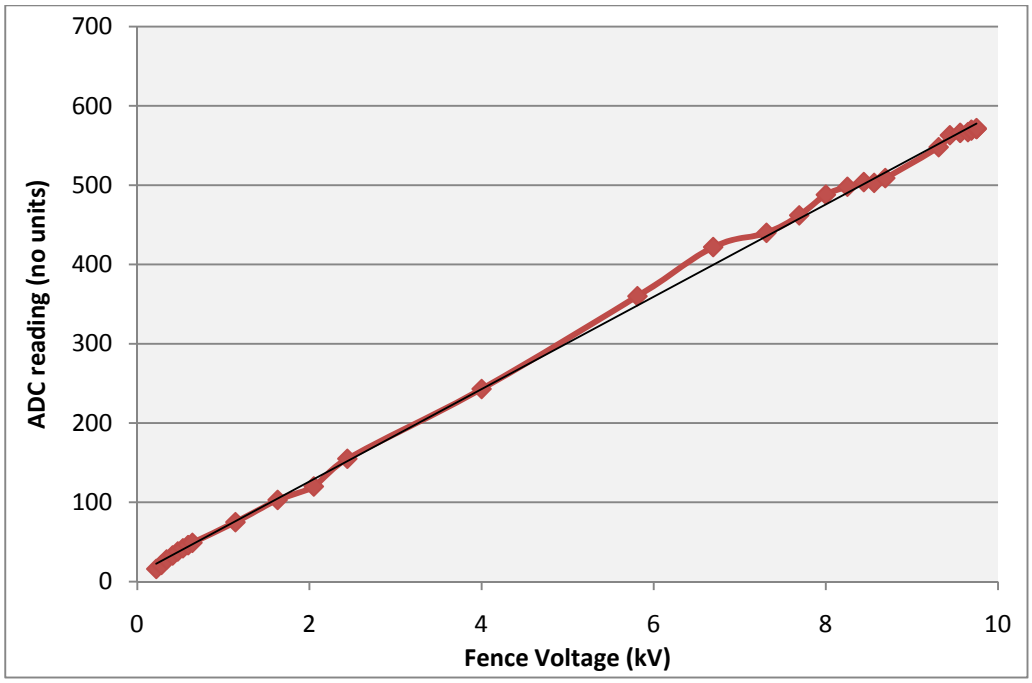


Figure 6-6: Positive voltage, no ground lead

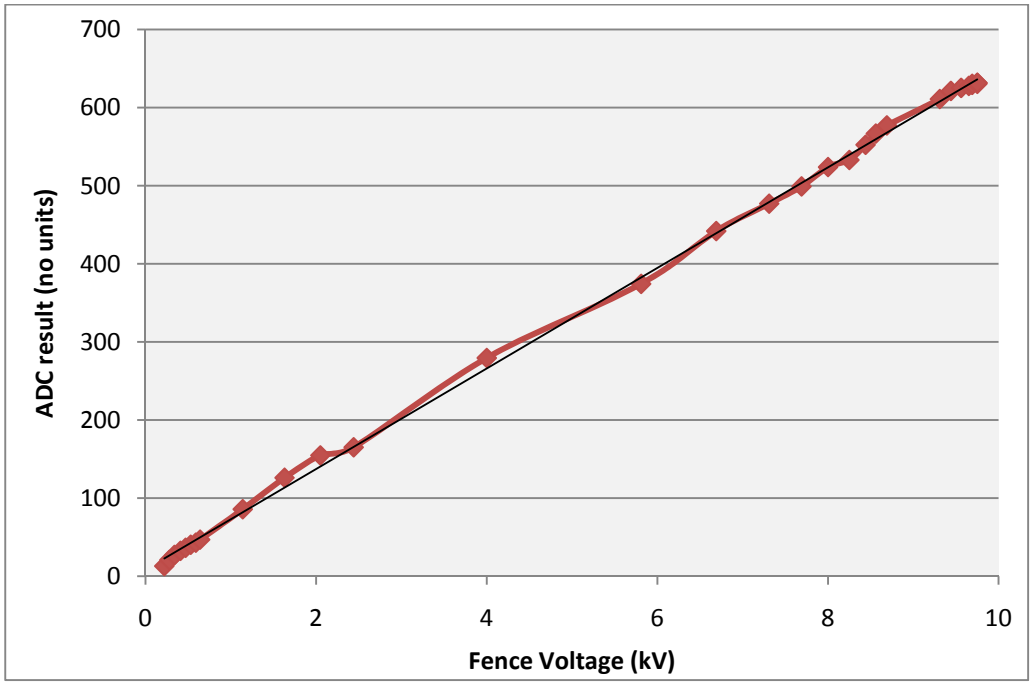


Figure 6-7: Negative voltage, ground lead connected

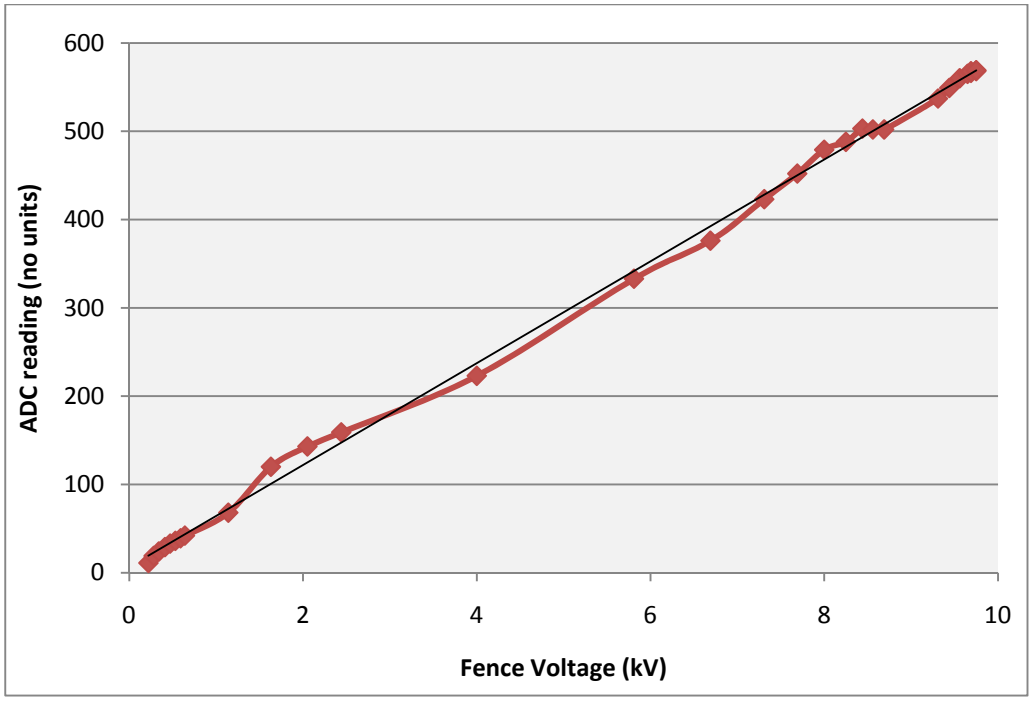


Figure 6-8: Negative voltage, no ground lead

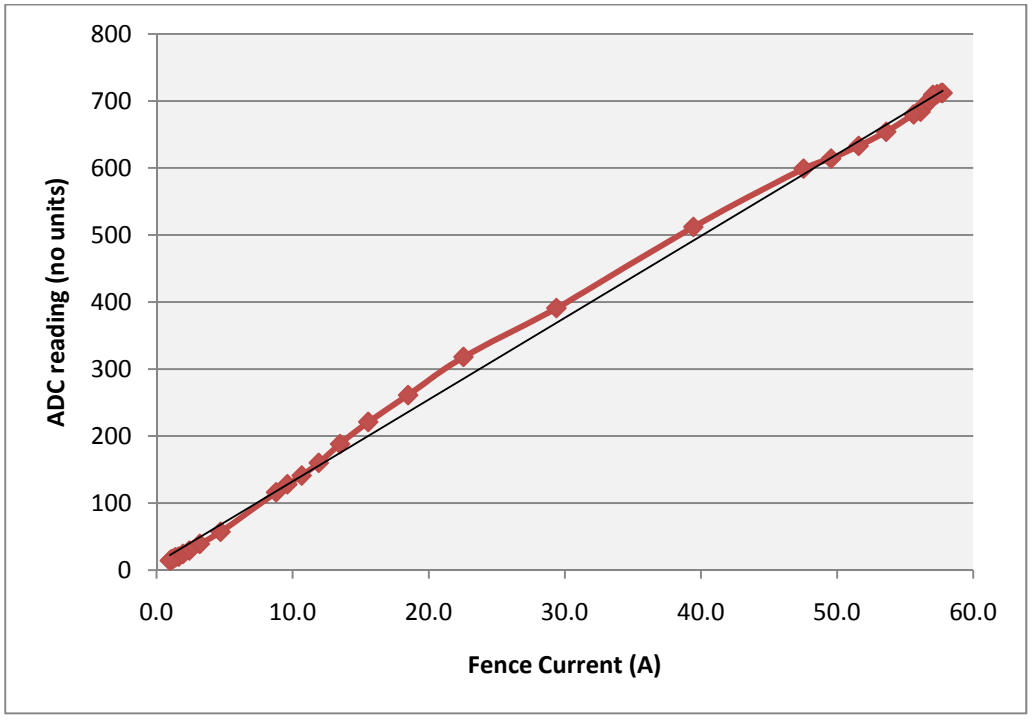


Figure 6-9: Positive current, ground lead connected

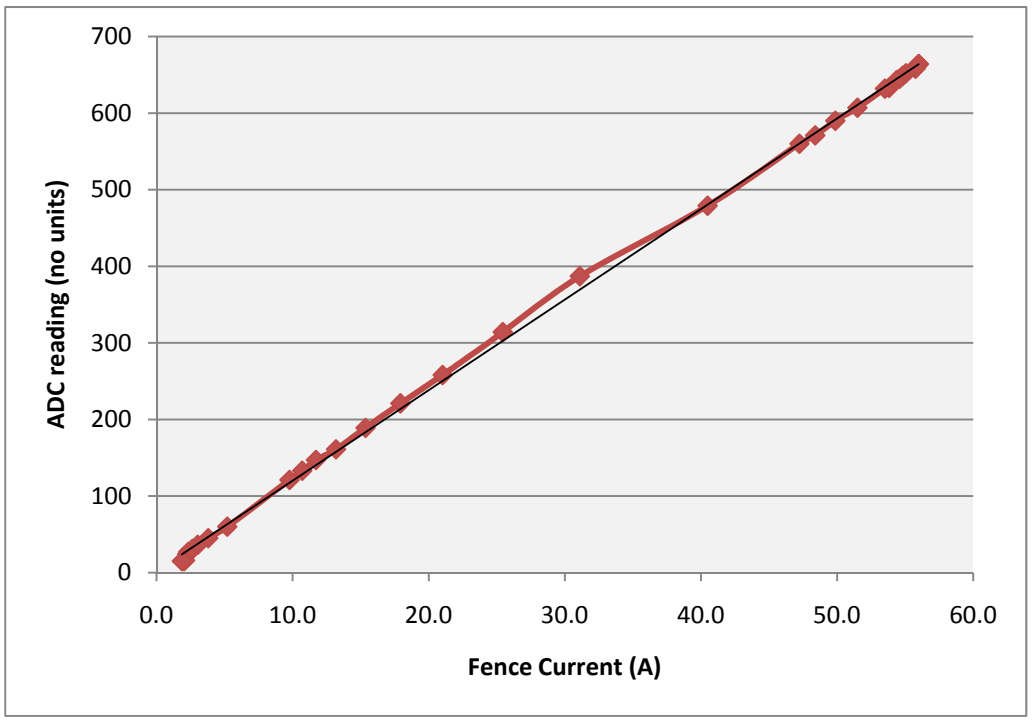


Figure 6-10: Positive current, no ground lead

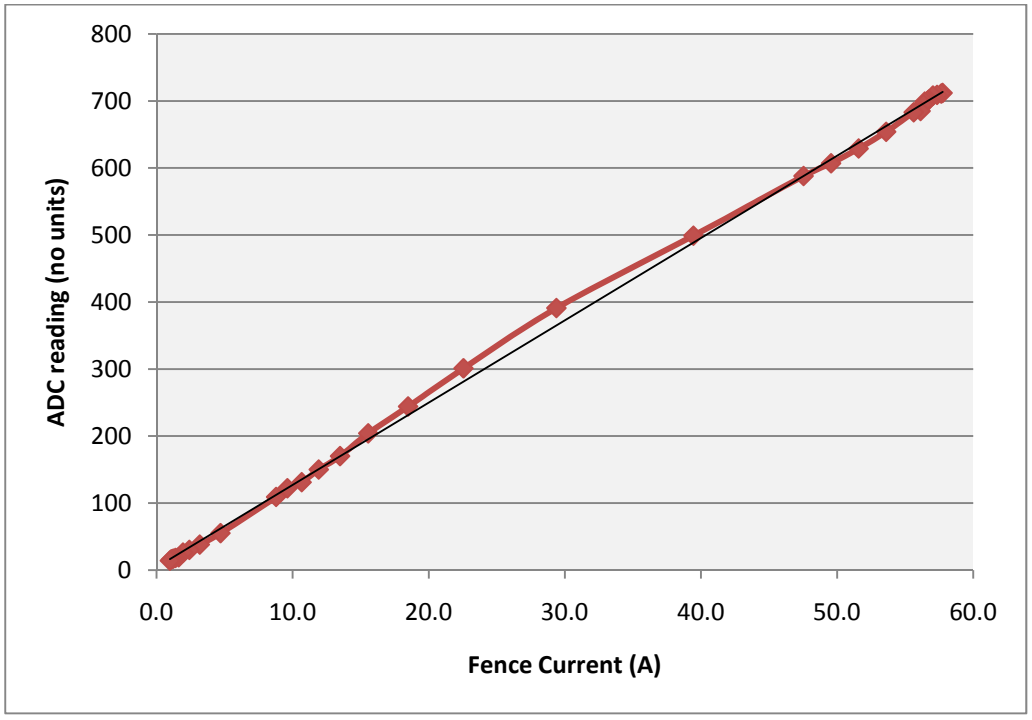


Figure 6-11: Negative current, ground lead connected

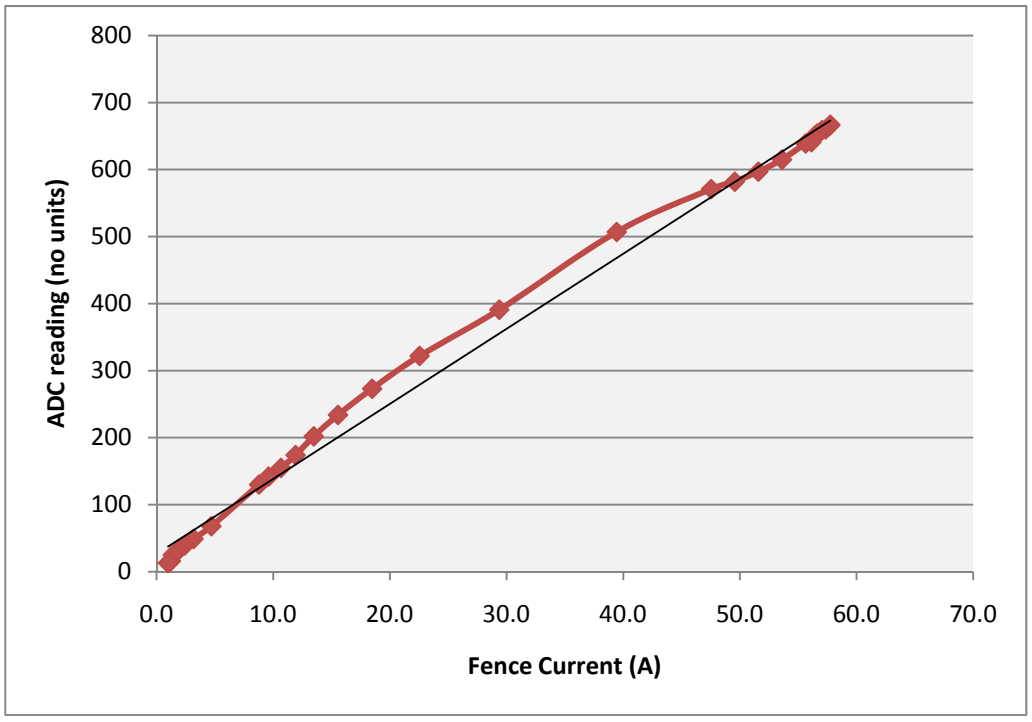


Figure 6-12: Negative current, no ground lead

Fence Load (kΩ)	Fence Voltage (kV)	Fence Current (A)	Positive Voltage, Grounded (ADC)	Positive Current, Grounded (ADC)	Positive Voltage, No Ground (ADC)	Positive Current, No Ground (ADC)	Negative Voltage, Grounded (ADC)	Negative Current, Grounded (ADC)	Negative Voltage, No Ground (ADC)	Negative Current, No Ground (ADC)
0.001	0.08	57.7	-	712	-	664	-	712	-	667
0.002	0.15	57.6	10	712	-	664	-	711	-	665
0.003	0.22	57.3	18	710	16	661	13	709	11	660
0.004	0.28	57.0	23	709	21	658	21	708	19	659
0.005	0.34	56.7	31	700	28	652	27	702	24	654
0.006	0.41	56.6	37	697	33	649	33	700	29	654
0.007	0.47	56.4	42	694	38	645	36	699	33	651
0.008	0.53	56.3	47	691	42	643	40	695	36	648
0.009	0.59	56.1	51	684	46	633	43	685	39	641
0.01	0.64	55.6	54	680	49	632	47	683	42	639
0.02	1.14	53.6	83	654	75	607	86	654	68	615
0.03	1.63	51.6	140	633	103	590	126	629	120	597
0.04	2.05	49.6	152	614	120	571	155	607	143	582
0.05	2.44	47.5	167	599	155	560	165	588	159	571
0.1	4.00	39.4	270	512	243	479	280	499	223	507
0.2	5.81	29.4	409	391	360	387	374	391	333	391
0.3	6.69	22.6	469	318	422	314	442	301	376	322
0.4	7.31	18.5	507	261	440	258	477	244	423	273
0.5	7.69	15.6	533	221	462	221	499	204	452	234
0.6	8.00	13.5	542	188	488	189	524	170	479	202
0.7	8.25	11.9	565	160	498	161	533	150	488	174
0.8	8.44	10.7	579	141	504	147	553	131	503	155
0.9	8.56	9.6	588	128	503	133	567	122	502	142
1.0	8.69	8.8	595	116	509	121	577	109	502	130
2.0	9.31	4.7	620	57	548	60	611	55	537	68
3.0	9.44	3.2	626	39	563	45	621	38	549	49
4.0	9.56	2.4	629	29	566	36	625	30	560	39
5.0	9.65	2.0	630	24	567	31	628	26	565	33
6.0	9.69	1.6	632	20	569	27	629	19	566	28
7.0	9.69	1.4	633	19	570	24	631	18	568	25
8.0	9.75	1.2	634	17	571	16	631	17	568	16
9.0	9.75	1.1	635	16	571	15	632	16	569	14
10.0	9.75	1.0	635	14	572	15	632	14	569	13

Table 6-5: Peak detect circuit readings

Accuracy of voltage readings taken below 2kV fence voltage tends to be much lower than readings above. Any fence voltages below 3kV are considered ineffective for stock control, therefore it has been assumed that lower accuracy readings can be ignored for determining overall voltage accuracy. All accuracy values have been calculated from the tables contained in appendix A.

	Average Accuracy (%)
Positive voltage, with ground connection	1.91%
Positive voltage, no ground connection	-9.49%
Negative voltage, with ground connection	0.86%
Negative voltage, no ground connection	-10.38%
Positive current, with ground connection	-2.43%
Positive current, no ground connection	-1.26%
Negative current, with ground connection	2.74%
Negative current, no ground connection	9.07%

Table 6-6: Peak detect circuit average accuracy values

The data suggests that both voltage peak detect circuits have a linear response, and an average accuracy of better than 2%. Accuracy of readings below 2kV varied greatly, however, this is not a significant problem because any reading below 3kV indicates that the fence requires immediate attention. Individual reading accuracy varied from the straight trend line by more than 2% (as seen in Figure 6-5 and

Figure 6-7), therefore an absolute accuracy figure of 10% is a better indication of measurement accuracy.

As stated earlier, voltage readings taken without an earth probe (using the capacitive ground feature) are approximately 10% lower than the same readings taken with an earth probe. Because the device is likely to be used without a ground lead 99% of the time (fault-finding requires relative readings, not absolute values), the algorithm to calculate voltage could be optimised to make these readings accurate, at the expense of absolute reading accuracy.

Current readings were less accurate than voltage readings, largely because the measurements are slightly non-linear. Accuracy was similar for measurements taken both with and without an earth lead. Calculated current values could be made more precise by using a polynomial equation instead of the straight line method employed.

6.2.6 Energiser Pulse Detection

Each energiser pulse was correctly detected during testing. The device was connected to an operating energiser for several days. Special software counted the number and polarity of pulses detected, and the energiser also had special software to log the number of pulses generated. The test was carried out with positive and negative pulses, and both with a ground connection and without. The results of each test are given in Table 6-7.

The minimum fence voltage detected in every test was 0.22kV – lower voltage pulses can be detected when a ground lead is connected.

It must be noted that the energiser pulse detection is very sensitive, and can detect fence pulses from a distance of several centimetres from the wire (no metallic contact).

	Pulses Reported	Pulses Detected
Positive Voltage, With Ground	57,619	57,619
Positive Voltage, No Ground	54,812	54,812
Negative Voltage, With Ground	55,980	55,980
Negative Voltage, No Ground	58,136	58,136

Table 6-7: Energiser pulse detection count

6.3 Device Specifications

Specifications for the final device were selected based on the measurements taken in section 6.2.5. These specifications are compared to those of similar products (where available – the Stafix Fence Compass documentation does not give any specifications) in Table 6-8.

	Accuracy (%)	Min. Voltage (kV)	Max. Voltage (kV)	Min. Current (A)	Max. Current (A)
Device Prototype	+/- 10	0.2	15+	1	60+
Gallagher SmartFix	Not Given	0.2	14	1	35
Pakton Power Probe	+/- 10	0.2	9.9	1	25

Table 6-8: Final prototype declared specifications

6.4 Manufactured Cost

The final estimated build cost of the device is NZ$48.23. Competitor devices retail between $180 and $250, which dictates a wholesale price of approximately half, or $90 to $125. This device allows enough margin for sales and support costs to make this a viable commercial product. A cost breakdown is given below in Table 6-9. See appendix B for a costed Bill of Materials (BOM).

Cost Item	Cost (NZ$)
Components and bare PCB	28.73
Electronic manufacture	7.50
Plastic case	6.85
Final assembly and test	4.15
Packaging, manual	1.00
TOTAL	48.23

Table 6-9: Manufacturing cost breakdown

6.5 Field Trial

The device was taken to a farm for field testing in a real world environment. Readings were taken at several fixed points along the length of the fence, both with and without faults. A 500Ω high voltage resistor was used to simulate a fault to ground, and was placed in different locations for testing. For one test, a 0Ω fault was placed on the fence, and another test used two separate 500Ω faults simultaneously. The total fence length is approximately 2.1km, however only the first kilometre was been used for practical reasons. Table 6-10 through Table 6-15 show the readings produced by the device (fault locations are highlighted on each table).

Distance from Energiser (m)	Voltage (kV)	Current (A)
5	10.3	2
100	10.3	2
200	10.2	2
300	10.1	2
400	10.1	2
500	10.0	2
600	9.9	2
700	9.8	2
800	9.8	1
900	9.7	1
1,000	9.7	1

Table 6-10: Test fence, with no faults

Distance from Energiser (m)	Voltage (kV)	Current (A)
5	9.4	10
100	8.9	10
200	8.3	10
300	7.8	9
400	7.2	9
500	6.7	9
600	6.2	9
700	5.6	8
800	5.1	8
900	4.5	8
1,000	4.0	8

Table 6-11: Test fence, 500Ω fault to ground at 1,000m

Distance from Energiser (m)	Voltage (kV)	Current (A)
5	9.3	11
100	8.4	11
200	7.5	11
300	6.6	10
400	5.7	9
500	5.7	1
600	5.6	1
700	5.5	1
800	5.3	1
900	5.3	1
1,000	5.3	0

Table 6-12: Test fence, 500Ω fault to ground at 400m

Distance from Energiser (m)	Voltage (kV)	Current (A)
5	9.0	13
100	7.8	13
200	6.3	13
300	5.9	1
400	5.9	1
500	5.8	1
600	5.7	1
700	5.6	0
800	5.6	0
900	5.5	0
1,000	5.4	0

Table 6-13: Test fence, 500Ω fault to ground at 200m

Distance from Energiser (m)	Voltage (kV)	Current (A)
5	5.7	22
100	5.6	22
200	5.4	22
300	5.4	21
400	4.8	7
500	3.9	6
600	3.2	6
700	3.1	1
800	2.9	1
900	2.9	0
1,000	2.8	0

Table 6-14: Test fence, 500Ω fault to ground at 300m and 600m

Distance from Energiser (m)	Voltage (kV)	Current (A)
5	1.3	32
100	1.1	32
200	0.8	32
300	0.5	31
400	0.2	31
500	0.1	31
600	No reading	No reading
700	No reading	No reading
800	No reading	No reading
900	No reading	No reading
1,000	No reading	No reading

Table 6-15: Test fence, 0Ω fault to ground at 500m

From the results it can be seen that in each scenario the presence of a fault is easily detected, especially when the expected voltage and current at the energiser is known (given in Table 6-10). Locating each fault was a simple matter of taking readings, starting at the energiser and following the direction displayed on the fault-finder (always points away from the energiser).

One notable exception to the fault-finding method is when a fence with one or more earth return wires is being tested. Because earth return wires are connected to the negative terminal of an energiser, the current flow is opposite to that on live wires. Therefore, the user will read a low voltage (and possibly a high current depending on the fault, and fence condition), and the arrow will point away from the fault. It is up to the user in this situation to know which wires are earth returns, and which are live wires.

Another observation during field-testing was that when measurements are taken on the other side of a fault (i.e. the fault is between the device and the energiser), the direction arrow will continue to point away from the energiser, and subsequently away from the fault. Because the current reading falls to a much lower level when past the fault, the operator must remember previous readings (or use the software memory feature to browse past values) to know that they have gone past the fault. A solution to this problem may be to stop displaying the fault direction arrow if the current falls below a threshold (such as 3 Amps).

6.6 Fault Finding Speed Test

The main purpose of the device is to reduce the time taken to find faults. To determine whether this goal was successful, the device was used to

find an existing fault on an electric fence. The farm size was 20 acres, and the energiser used was a medium sized mains powered unit. The fence had two live wires (0.7m and 1.0 metres above ground) and no earth return wires in a typical trunk based layout – the pulse was distributed along a trunk fence path and each paddock branched from the central line. The fence was not controlling animals and was obviously needed repair. The fault was discovered to be a cracked insulator that still held the fence wire in place, but was allowing the fence pulse to arc to ground via the post.

The farm owner searched for the fault using a standard electric fence voltmeter. This tool was of limited use because the voltage was generally low throughout the entire fence system. After 153 minutes of searching, the fault was found about 700 metres away from the energiser on the main trunk wire. The fault was discovered by the sound of the arc with each pulse.

The author used the device to find the same fault (without knowing the location or nature of the fault). It was located after only 16 minutes by starting at the energiser and following the fault-finder's arrow until the current reading dropped significantly. After backtracking slightly, it was determined that the high current ceased after a specific fence post and visual inspection revealed the fault.

Although this example involves a particularly difficult fault to locate visually, it is typical problem, especially with older fences. Using the device to locate the fault reduced the time to find by almost 90%.

7

Chapter 7

Conclusion and Suggestions for Future Work

7.1 Conclusion

The goal of this thesis was to develop a commercially ready electric fence fault-finding tool that reduces the time taken to find fencing problems. The device created satisfies every aspect of this goal, providing voltage and current readings along with fault direction that allows fast location of faults.

The electronic design uses common commercial parts, and is not reliant on the special characteristics of a specific manufacturer's part. All that is required to make this product ready for sale is the manufacture of plastic injection moulding tools. Price estimates have been supplied at around NZ$65,000 to complete four tools – top and bottom case, grommet, and lens.

This device can be used with all types of fence construction and energisers. Both positive and negative voltages are measured, along with positive and negative current values, and positive and negative

edge pulse detection. The device circuitry is fast enough to turn on automatically when a pulse is detected, and provide measurements of the pulse that woke the device.

One shortcoming of the device is its battery life. Typical use will yield one month's life from a single alkaline 9V battery. Batteries are available with twice the capacity, but these are expensive. Rechargeable batteries are recommended, however these provide lower capacities than typical dry cells.

Field trials of the device yielded excellent results, allowing faults that were not apparent to visual inspection to be found quickly and without difficulty. In an example presented in section 6.6, it took 16 minutes to find a fence fault while using the device, compared to 153 minutes without.

7.2 Suggestions for Future Work

None of the field trials of this device were carried out using an adaptive output energiser. Additional tests should be performed to ensure that this design will work under every condition present in typical farming scenarios.

Microcontrollers incorporating high-speed ADC circuits have recently become readily available at low cost, with most IC manufacturers offering a range of 8-bit and 16-bit microcontrollers incorporating 500ksps to 2Msps analogue sample rates, priced below NZ$5. These microcontrollers typically contain more advanced mathematics support, providing native 32-bit arithmetic and floating point support.

Taking direct analogue readings of fence voltage and current allows an oscilloscope like snapshot of each fence pulse. Signal processing could be used to provide the following:

- Energy content of the fence pulse – can provide a better indication of fence effectiveness and shockability.
- Estimation of fault type based on characteristic waveform – could inform user that the fault is likely to be an arcing insulator, shorted wire, open circuit, etc.
- Calculation of a “fence health” index (i.e. a rating from zero to ten) – the user can more easily remember a single number and record how the performance of their fence is changing with time and / or fault resolution.

The battery life of the product is not very long. One possible solution is to incorporate a solar panel and charging circuitry into the device. Since the typical user will only need a few hours operation per week, the battery will always maintain full charge while outside.

When fault-finding at night, the device display is bright and clear, but the fence wire can be difficult to see. Placing an LED flashlight on the end of the fault meter facing the wire will allow the user to see what they are doing when taking readings, and even use the device as a flashlight when walking between test points.

Finally, because the device under-reads voltage levels by approximately 10% when using a capacitive ground reference (ground lead is not used), the overall product accuracy could be improved by detecting when a ground lead is not in use and compensate the readings accordingly.

References

1. Pakton Ltd. *Pakton Corporate Website, www.pakton.com.au/pte0111.html*. 2009.

2. Gallagher Group Ltd., *Fault Meter*. 1999: New Zealand.

3. Tru-Test Ltd., *A Detector Device*. 2000: New Zealand.

4. Hancock, A.T., *Electric Fences, Modelling the Electrical Characteristics and Behaviour of Electric Fence Systems*. 1991, MSc (Tech) Thesis, University of Waikato.

5. McKay, T.A., *Fence Line Communications*. 2001, University of Waikato.

6. Thrimawithana, D.J. and U.K. Madawala, *Modeling Pulse Reflections due to Multiple Discontinuities on Electric Fence Structures*, in *IEEE International Conference on Industrial Technology*. 2008: Chengdu, Peoples Republic of China. p. 1999-2004.

7. Ho, Y.M., *Design and Construction of a Current Pulse Meter*. 1996, University of Waikato.

8. Wikipedia, *Electric Fence*. 2009.

9. Stafix Ltd., *Electric Fencing Manual*. 2006.

10. Gallagher Group Ltd., *Power Fence Systems Manual*. 2008.

11. Thrimawithana, D.J. and U.K. Madawala, *Pulse Propagation Along Multi-Wire Electric Fences.* IET Science, Measurement and Technology, 2008. **2**(5): p. 349-358.

12. Thrimawithana, D.J. and U.K. Madawala, *Pulse Propagation Along Single-Wire Electric Fences.* IEEE Transactions on Power Delivery, 2008. **23**(4): p. 2302-2309.

13. Council of Standards; Australia and New Zealand, *Household and Similar Electrical Appliances - Safety - Particular Requirements for Electric Fence Energisers*, in *AS/NZS 60335.2.76:2003/Amdt 1:2006*. 2006.

14. McCutchan, J., *Electric Fence Design Principles*. 1980: University of Melbourne.

15. Action, *Maintaining Electric Fences*. 1999: World Wide Fund for Nature.

16. Adamson, P.R., *The Design of an Electric Fence Monitoring System*. 1996, Massey University: Palmerston North.

17. Boylestad, R.L. and L. Nashelsky, *Electronic Devices and Circuit Theory*. 10th ed. 2009: Prentice Hall.

18. Atmel, *ATmega324P/V Microcontroller Data Sheet*. 2007.

19. Jung, W.G., *IC Op-Amp Cookbook*. 1986: Prentice Hall.

20. Texas Instruments, *OPA2374 Datasheet*. 2004.

21. Quantum Technologies, *Capacitive Touch Sensor QT113 Product Datasheet*. 2006.

22. Univision Technology, *OLED Display Module Product Specification*. 2005.

23. Texas Instruments, *TPS61040 Low-Power DC/DC Boost Converter*. 2007.

24. Energizer, *Energizer 522 Product Datasheet*. 2004.

25. National Semiconductor, *LP2985 Micropower 150mA Low-Noise Ultra-Low Dropout Regulator Data Sheet*. 2000.

26. Bourns, *PTC Resettable Fuses Data Sheet*. 2005.

27. MicroChip Inc, *PIC16F630 Product Datasheet.* 2007.

28. Di Jasio, L., *Programming 16-bit Microcontrollers in C*. 2007: Newnes.

29. Univision Technology, *UG-2828GDEAF01 Application Note*. 2006.

30. Solomon Systech, *SSD1339 Advance Information.* 2006.

31. Osram, *Calgary 4 Bits OLED Module Application Instruction* in *AN018*. 2006.

32. Borland International, *Borland C++ Users Guide*. 1996.

33. Borland International, *Borland C++ Programmer's Guide.* 1996.

34. House, R., *Beginning With C*. 1994: Nelson ITP.

Appendix A:
Peak Detect
Measurement Data

Fence Voltage (kV)	Positive Reading (Grounded)	Calculated voltage (kV)	% Error
0.08	-	-	-
0.15	10	0.08	-49.03%
0.22	18	0.20	-9.65%
0.28	23	0.28	-1.70%
0.34	31	0.40	16.93%
0.41	37	0.49	19.34%
0.47	42	0.57	20.37%
0.53	47	0.64	21.17%
0.59	51	0.70	19.21%
0.64	54	0.75	17.07%
1.14	83	1.19	4.62%
1.63	140	2.06	26.64%
2.05	152	2.25	9.64%
2.44	167	2.48	1.52%
4	270	4.05	1.30%
5.81	409	6.18	6.32%
6.69	469	7.09	6.05%
7.31	507	7.68	5.00%
7.69	533	8.07	4.99%
8	542	8.21	2.64%
8.25	565	8.56	3.79%
8.44	579	8.78	3.99%
8.56	588	8.91	4.14%
8.69	595	9.02	3.81%
9.31	620	9.40	1.01%
9.44	626	9.50	0.59%
9.56	629	9.54	-0.20%
9.65	630	9.56	-0.97%
9.69	632	9.59	-1.06%
9.69	633	9.60	-0.90%
9.75	634	9.62	-1.36%
9.75	635	9.63	-1.20%
9.75	635	9.63	-1.20%
		Average Error	**4.15%**
		Average Error (Above 2kV)	**1.91%**

Table A-1: Positive voltage peak detection accuracy, with ground lead

Fence Voltage (kV)	Positive Reading (No Ground)	Calculated voltage (kV)	% Error
0.08	-	-	-
0.15	-	-	-
0.22	16	0.17	-23.55%
0.28	21	0.24	-12.63%
0.34	28	0.35	3.44%
0.41	33	0.43	4.42%
0.47	38	0.50	7.36%
0.53	42	0.57	6.75%
0.59	46	0.63	6.26%
0.64	49	0.67	5.12%
1.14	75	1.07	-6.11%
1.63	103	1.50	-8.07%
2.05	120	1.76	-14.22%
2.44	155	2.29	-6.00%
4	243	3.64	-9.02%
5.81	360	5.43	-6.57%
6.69	422	6.38	-4.69%
7.31	440	6.65	-9.01%
7.69	462	6.99	-9.13%
8	488	7.39	-7.68%
8.25	498	7.54	-8.63%
8.44	504	7.63	-9.60%
8.56	503	7.61	-11.04%
8.69	509	7.71	-11.32%
9.31	548	8.30	-10.82%
9.44	563	8.53	-9.62%
9.56	566	8.58	-10.27%
9.65	567	8.59	-10.95%
9.69	569	8.62	-11.00%
9.69	570	8.64	-10.84%
9.75	571	8.65	-11.24%
9.75	571	8.65	-11.24%
9.75	572	8.67	-11.08%
		Average Error	**-7.13%**
		Average Error (Above 2kV)	**-9.49%**

Table A-2: Positive voltage peak detection accuracy, no ground lead

Fence Voltage (kV)	Negative Reading (Grounded)	Calculated voltage (kV)	% Error
0.08	-	-	-
0.15	-	-	-
0.22	13	0.12	-43.52%
0.28	21	0.25	-11.25%
0.34	27	0.34	0.50%
0.41	33	0.43	6.07%
0.47	36	0.48	2.44%
0.53	40	0.54	2.57%
0.59	43	0.59	0.03%
0.64	47	0.65	1.92%
1.14	86	1.26	10.35%
1.63	126	1.88	15.29%
2.05	155	2.33	13.64%
2.44	165	2.49	1.84%
4	280	4.27	6.78%
5.81	374	5.73	-1.36%
6.69	442	6.79	1.45%
7.31	477	7.33	0.28%
7.69	499	7.67	-0.23%
8	524	8.06	0.76%
8.25	533	8.20	-0.60%
8.44	553	8.51	0.84%
8.56	567	8.73	1.97%
8.69	577	8.88	2.23%
9.31	611	9.41	1.10%
9.44	621	9.57	1.35%
9.56	625	9.63	0.73%
9.65	628	9.68	0.27%
9.69	629	9.69	0.02%
9.69	631	9.72	0.34%
9.75	631	9.72	-0.28%
9.75	632	9.74	-0.12%
9.75	632	9.74	-0.12%
		Average Error	**0.49%**
		Average Error (Above 2kV)	**0.86%**

Table A-3: Negative voltage peak detection accuracy, with ground lead

Fence Voltage (kV)	Negative Reading (No Ground)	Calculated voltage (kV)	% Error
0.08	-	-	-
0.15	-	-	-
0.22	11	0.09	-57.65%
0.28	19	0.22	-22.36%
0.34	24	0.30	-13.23%
0.41	29	0.37	-9.10%
0.47	33	0.43	-7.49%
0.53	36	0.48	-9.18%
0.59	39	0.53	-10.52%
0.64	42	0.57	-10.23%
1.14	68	0.98	-14.19%
1.63	120	1.88	15.27%
2.05	143	2.19	6.80%
2.44	159	2.72	11.37%
4	223	3.39	-15.37%
5.81	333	5.09	-12.34%
6.69	376	5.76	-13.89%
7.31	423	6.49	-11.21%
7.69	452	6.94	-9.74%
8	479	7.36	-8.00%
8.25	488	7.50	-9.09%
8.44	503	7.73	-8.38%
8.56	502	7.72	-9.84%
8.69	502	7.72	-11.19%
9.31	537	8.26	-11.27%
9.44	549	8.45	-10.52%
9.56	560	8.62	-9.85%
9.65	565	8.70	-9.89%
9.69	566	8.71	-10.10%
9.69	568	8.74	-9.78%
9.75	568	8.74	-10.34%
9.75	569	8.76	-10.18%
9.75	569	8.76	-10.18%
		Average Error	**-10.38%**
		Average Error (Above 2kV)	**-9.49%**

Table A-4: Negative voltage peak detection accuracy, no ground lead

Fence Current (A)	Positive Reading (Grounded)	Calculated Current (A)	% Error
57.7	712	57.90	0.27%
57.6	712	57.90	0.44%
57.3	710	57.74	0.69%
57.0	709	57.66	1.08%
56.7	700	56.92	0.32%
56.6	697	56.67	0.07%
56.4	694	56.43	-0.01%
56.3	691	56.18	-0.27%
56.1	684	55.61	-0.93%
55.6	680	55.28	-0.62%
53.6	654	53.15	-0.84%
51.6	633	51.43	-0.28%
49.6	614	49.88	0.65%
47.5	599	48.65	2.34%
39.4	512	41.52	5.27%
29.4	391	31.61	7.60%
22.6	318	25.63	13.66%
18.5	261	20.97	13.44%
15.6	221	17.69	13.73%
13.5	188	14.99	11.14%
11.9	160	12.69	6.50%
10.7	141	11.14	4.39%
9.6	128	10.07	4.73%
8.8	116	9.09	3.44%
4.7	57	4.26	-9.54%
3.2	39	2.78	-12.50%
2.4	29	1.97	-18.68%
2.0	24	1.56	-20.28%
1.6	20	1.23	-24.79%
1.4	19	1.15	-18.10%
1.2	17	0.98	-20.27%
1.1	16	0.90	-17.77%
1.0	14	0.74	-25.25%
		Average Accuracy	**-2.43%**

Table A-5: Positive current peak detection accuracy, with ground lead

Fence Current (A)	Positive Reading (No Ground)	Calculated Current (A)	% Error
57.7	664	53.97	-6.54%
57.6	664	53.97	-6.38%
57.3	661	53.73	-6.31%
57.0	658	53.48	-6.24%
56.7	652	52.99	-6.61%
56.6	649	52.74	-6.87%
56.4	645	52.42	-7.12%
56.3	643	52.25	-7.24%
56.1	633	51.43	-8.37%
55.6	632	51.35	-7.68%
53.6	607	49.30	-8.02%
51.6	590	47.91	-7.11%
49.6	571	46.36	-6.46%
47.5	560	45.45	-4.37%
39.4	479	38.82	-1.58%
29.4	387	31.29	6.49%
22.6	314	25.31	12.21%
18.5	258	20.72	12.11%
15.6	221	17.69	13.73%
13.5	189	15.07	11.75%
11.9	161	12.78	7.19%
10.7	147	11.63	9.00%
9.6	133	10.48	8.98%
8.8	121	9.50	8.10%
4.7	60	4.50	-4.32%
3.2	45	3.28	2.94%
2.4	36	2.56	5.86%
2.0	31	2.13	9.09%
1.6	27	1.80	10.31%
1.4	24	1.56	11.15%
1.2	16	0.90	-26.91%
1.1	15	0.82	-25.25%
1.0	15	0.82	-16.94%
		Average Accuracy	**-1.26%**

Table A-6: Positive current peak detection accuracy, no ground lead

Fence Current (A)	Negative Reading (Grounded)	Calculated Current (A)	% Error
57.7	712	57.89	0.24%
57.6	711	57.80	0.27%
57.3	709	57.64	0.52%
57.0	708	57.56	0.91%
56.7	702	57.07	0.59%
56.6	700	56.91	0.48%
56.4	699	56.83	0.70%
56.3	695	56.50	0.30%
56.1	685	55.69	-0.78%
55.6	683	55.53	-0.17%
53.6	654	53.17	-0.81%
51.6	629	51.14	-0.86%
49.6	607	49.35	-0.42%
47.5	588	47.80	0.57%
39.4	499	40.57	2.85%
29.4	391	31.79	8.20%
22.6	301	24.47	8.50%
18.5	244	19.84	7.33%
15.6	204	16.59	6.63%
13.5	170	13.82	2.49%
11.9	150	12.20	2.31%
10.7	131	10.65	-0.18%
9.6	122	9.92	3.11%
8.8	109	8.86	0.83%
4.7	55	4.47	-5.02%
3.2	38	3.09	-2.92%
2.4	30	2.44	0.90%
2.0	26	2.11	8.29%
1.6	19	1.54	-5.43%
1.4	18	1.46	4.53%
1.2	17	1.38	12.13%
1.1	16	1.30	18.73%
1.0	14	1.14	15.43%
		Average Accuracy	**2.74%**

Table A-7: Negative current peak detection accuracy, with ground lead

Fence Current (A)	Negative Reading (No Ground)	Calculated Current (A)	% Error
57.7	667	54.23	-6.10%
57.6	665	54.07	-6.21%
57.3	660	53.66	-6.43%
57.0	659	53.58	-6.07%
56.7	654	53.17	-6.29%
56.6	654	53.17	-6.12%
56.4	651	52.93	-6.21%
56.3	648	52.68	-6.48%
56.1	641	52.11	-7.16%
55.6	639	51.95	-6.60%
53.6	615	50.00	-6.72%
51.6	597	48.54	-5.90%
49.6	582	47.32	-4.52%
47.5	571	46.42	-2.34%
39.4	507	41.22	4.50%
29.4	391	31.79	8.20%
22.6	322	26.18	16.07%
18.5	273	22.20	20.09%
15.6	234	19.02	22.31%
13.5	202	16.42	21.79%
11.9	174	14.15	18.68%
10.7	155	12.60	18.10%
9.6	142	11.54	20.02%
8.8	130	10.57	20.26%
4.7	68	5.53	17.43%
3.2	49	3.98	25.18%
2.4	39	3.17	31.18%
2.0	33	2.68	37.45%
1.6	28	2.28	39.37%
1.4	25	2.03	45.18%
1.2	16	1.30	5.53%
1.1	14	1.14	3.89%
1.0	13	1.06	7.18%
		Average Accuracy	**9.07%**

Table A-8: Negative current peak detection accuracy, no ground lead

Appendix B:

Costed Bill of Materials

Value	Qty	Comment	Cost (ea)	Total Cost
0.1uF	12	Capacitor	$0.00	$0.05
100k	10	Resistor	$0.00	$0.04
10k	1	Resistor	$0.00	$0.00
10M	5	Resistor	$0.00	$0.02
10M 1W	2	Resistor	$0.10	$0.20
10uF, 35V	6	SMD capacitor	$0.10	$0.60
150k	1	Resistor	$0.00	$0.00
1M	5	Resistor	$0.00	$0.02
1M	1	Resistor	$0.00	$0.00
2.7V 0.5W	1	Zener Diode	$0.03	$0.03
200k	2	Resistor	$0.00	$0.01
22uH	1	SMD inductor	$0.20	$0.20
2M2	1	Resistor	$0.00	$0.00
4.7pF	1	Capacitor	$0.10	$0.10
47R	2	Resistor	$0.00	$0.01
6k8	1	Resistor	$0.00	$0.00
910k	1	Resistor	$0.00	$0.00
ATmega324P-10AU	1	ATmega 164-324-644P/V	$4.15	$4.15
Battery Lead	1	Wired	$0.30	$0.30
BAV70LT	4	Diode	$0.01	$0.04
BC857C or equiv	1	PNP Bipolar Transistor	$0.01	$0.01
Inductor Iron	1	Magnetic-Core Inductor	$0.20	$0.20
LP2985IM5-3.3	1	Voltage Regulator	$0.54	$0.54
MBR0530T1	2	Schottkey diode	$0.10	$0.20
OLED	1	MOLEX 30way vertical FPC	$0.84	$0.84
OLED display	1	1.5" colour display	$15.00	$15.00
OPA2374	2	Op-Amp	$1.20	$2.40
PCB	1	Bare circuit board	$2.00	$2.00
Programming	1	Header, 3-Pin, Dual row	$0.10	$0.10
Tactile Switch	1	Switch	$0.15	$0.15
TPS61041	1	DC/DC boost converter	$1.50	$1.50

Manufacture	$7.50
Total	**$36.23**

Appendix C:

Software Listings

First Prototype Software Listing

```
#include "C:\\_my documents\\uFence\\uFence_Auto.h"
void LoadOscCal();

//
// Main function
//
void main()
{
  OPTION_REG&=0x7f;             // Pull ups on PORTB by default
 #if HASOSCCAL==1
  LoadOscCal();
 #endif

 // Initialisation Code
 ANSEL=0;
 CMCON=7;

 // End Initialisation Code
 PORTC|=0xff;
 PORTA|=0xff;
 TRISA&=0xcb;
 TRISC&=0xc0;
#if !InitialValue8
 Out8Port&=~(1<<Out8Bit); // Port Driver
#endif // Port Driver
 // Port Driver
#if !InitialValue7
 Out7Port&=~(1<<Out7Bit); // Port Driver
#endif // Port Driver
 // Port Driver
#if !InitialValue6
 Out6Port&=~(1<<Out6Bit); // Port Driver
#endif // Port Driver
 // Port Driver
#if !InitialValue5
 Out5Port&=~(1<<Out5Bit); // Port Driver
#endif // Port Driver
 // Port Driver
#if !InitialValue4
 Out4Port&=~(1<<Out4Bit); // Port Driver
#endif // Port Driver
 // Port Driver
#if !InitialValue3
 Out3Port&=~(1<<Out3Bit); // Port Driver
```

```
#endif // Port Driver
 // Port Driver
#if !InitialValue2
 Out2Port&=~(1<<Out2Bit); // Port Driver
#endif // Port Driver
 // Port Driver
#if !InitialValue1
 Out1Port&=~(1<<Out1Bit); // Port Driver
#endif // Port Driver
 // Port Driver
#if !InitialValue0
 Out0Port&=~(1<<Out0Bit); // Port Driver
#endif // Port Driver
 // Port Driver

 UserInitialise();

 // Finally enable interrupts

 INTCON|=(1<<GIE);

 // Main Loop

 while(1)
 {                                          // Loop forever
  UserLoop();
  #ifdef WatchDogUsed
   #pragma asmline clrwdt                   ; Clear watchdog timer if used
  #endif
 }
}

//
// Interrupts
//
const int QuickInt=1;

void Interrupt()
{
 // Priority Interrupts first
 // Other Interrupts
 #pragma asm
  SETPCLATH UserInterrupt,-1  ; Set PCLATH
  goto UserInterrupt          ; goto user interrupt
 UserIntReturn::              ; Return to here after user routine
  clrf STATUS                 ; Clear RP0/RP1
  SETPCLATH IntRet,-1         ; and set PCLATH back to return address
 #pragma asmend
}

//
// Load oscillator calibration value on reset for 14 bit processors
//
#if HASOSCCAL==1
#pragma asm
module "LoadOscCal"
LoadOscCal::
```

```
SETPCLATH OVRet,LoadOscCal
call OVRet
bsf STATUS,RP0
movwf OSCCAL
bcf STATUS,RP0
MRET 0

endmodule

module "LoadOscCal" forced absolute _TOPROM+1
 OVRet::
retlw 0xff
endmodule
#pragma asmend
#endif


#include "C:\\_my documents\\uFence\\uFence_Auto.h"

unsigned char detect=0;

const int NormalInt=1;
void UserInterrupt()
{
  PIR1=0x00;              //} clear interrupt
  detect=CMCON;           //}
  detect=1;
  #asmline SETPCLATH UserIntReturn,-1  ; SETPCLATH for interrupt routine
  #asmline goto UserIntReturn          ; Assembler - go back to interrupt routine
}

void delay()
{
  unsigned int dec=1905;          //} 40ms delay
  while(--dec);                   //}
}

//*****************************************************************************
void UserInitialise()
{
  CMCON=0x01;             // comparator enabled for peak detect
  VRCON=0x00;             // Vref off to save power
  PIE1=0x08;              //} enable comparator interrupt (used to wake device from
sleep)
  INTCON|=0x40;           //}
}

//*****************************************************************************
void UserLoop()
{
  unsigned long timer;
  unsigned char lights;

  while(1) {
    if(!detect) {                 //}
      #asmline sleep;             //} wait in low power mode
    }                             //}
```

```
   PIE1=0x00;                    // disable comparator interrupts
   VRCON=0xA1;                   // set voltage reference for low threshold (<=3kV) -
placed here to allow settling time

   lights=0;                 //}
   while(lights<100) [9] wait until the comparator is off for 1ms before proceeding
     lights++;               //}  (ensures that the peak detect cycle is complete before
     if(CMCON&0x40) {        //}   reading the stored value)
       lights=0;             //}
     }                       //}
   }

   CMCON=0x04;           // comparator set for measuring peak voltage stored on
capacitor
   lights=0;                 // reset for new cycle

   while(!(CMCON&0x40) && lights<=6) {          // measure peak level
     VRCON++;              //} increment to the next detection threshold
     lights++;             //}
     timer++;              // delay to provide enough settling time between voltage
reference changes
   }

   VRCON=0x00;           // Vref off to save power

   LED1=0;  delay();     // ramp LEDs on up to highest level measured
   if(lights>=1) {  LED2=0;  delay();  }
   if(lights>=2) {  LED3=0;  delay();  }
   if(lights>=3) {  LED4=0;  delay();  }
   if(lights>=4) {  LED5=0;  delay();  }
   if(lights>=5) {  LED6=0;  delay();  }
   if(lights>=6) {  LED7=0;  delay();  }
   if(lights>=7) {  LED8=0;  delay();  }

   delay();        //}
   delay();        //} wait 200ms
   delay();        //}
   delay();        //}

   switch(lights) {                  // ramp LEDs off, leaving the highest LED on
     case 0:  break;
     case 7:  LED7=1;  delay();
     case 6:  LED6=1;  delay();
     case 5:  LED5=1;  delay();
     case 4:  LED4=1;  delay();
     case 3:  LED3=1;  delay();
     case 2:  LED2=1;  delay();
     case 1:  LED1=1;  delay();
   }

   CMCON=0x01;           // comparator enabled for peak detect
   PIR1=0x00;            //}
   detect=0;             //} re-enable comparator interrupt
   PIE1=0x08;            //}

   timer=49020;          //}
```

```
    while(--timer && !detect);        //} Wait five seconds or until next trigger received

    LED1=1;                 //}
    LED2=1;                 //}
    LED3=1;                 //} all LEDs off before sleep instruction
    LED4=1;                 //}
    LED5=1;                 //}
    LED6=1;                 //}
    LED7=1;                 //}
    LED8=1;
  }
}
```

Second Prototype Software Listing

```
//*****************************************************************************************************
// FileName:                main_rtc.c
// Dependencies:            p33FJ256GP710.h
// Processor:               dsPIC33F
// Compiler:                MPLAB® C30 v2.01 or higher
// C30 Optimization Level: -O1
//*****************************************************************************************************

#include "p33FJ256GP710.h"
#include "lcd.h"
#include "common.h"

_FGS(GWRP_OFF & GCP_OFF);
_FOSCSEL(FNOSC_PRIPLL);
_FOSC(FCKSM_CSDCMD & OSCIOFNC_OFF & POSCMD_XT);
_FWDT(FWDTEN_OFF);

void Update_LCD( void );
const char mytext[] =  "Fault Meter v1.0";
const char mytext1[] = "Press S3 to cont";
const char time_msg[] = "Time 00: 00: 00 ";
const char adc_msg1[] =" RP5 = 0.00 Vdc  ";

int main ( void )
{
/* Initialize some general use variables */
hours, minutes, seconds = 0;
rtc_lcd_update = 0;

//The settings below set up the oscillator and PLL for 16 MIPS as
//follows:
//            Crystal Frequency  * (DIVISOR+2)
// Fcy =     ---------------------------------
//             PLLPOST * (PRESCLR+2) * 4
PLLFBD = 0x00A0;
CLKDIV = 0x0048;

/* set LEDs (D3-D10/RA0-RA7) drive state low */
LATA = 0xFF00;
/* set LED pins (D3-D10/RA0-RA7) as outputs */
TRISA = 0xFF00;

/* Initialize LCD Display */
Init_LCD();

/* Welcome message */
home_clr();
puts_lcd( (char*) &mytext[0], sizeof(mytext) -1 );
line_2();
puts_lcd( (char*) &mytext1[0], sizeof(mytext1) -1 );

while ( PORTDbits.RD6 );                    /* wait here until switch S3 is pressed */
```

```
/* Initialize Timer 1 for 32KHz real-time clock operation */
Init_Timer1();

/* Initial LCD message for TOD */
home_clr();
puts_lcd( (char*) &time_msg[0], sizeof(time_msg) -1 );
line_2();
puts_lcd( (char*) &adc_msg1[0], sizeof(adc_msg1) -1 );

/* Initialize ADC  */
Init_ADC();

/* Initialize DMA Channel 0 */
Init_DMAC0();

/* Infinite Loop */
while ( 1 )
{
if ( rtc_lcd_update ) {      /* check if time to update LCD with TOD data */
            hexdec( hours );
            Update_LCD();
            rtc_lcd_update = 0;
}

if ( adc_lcd_update ) {    /* check if time to update LCD with ADC data */
            line_2();
            advolt( temp1 );
            cursor_right();
            cursor_right();
            cursor_right();
            cursor_right();
            cursor_right();
            cursor_right();
            cursor_right();
             lcd_data( adones );
             cursor_right();
            lcd_data( adtens );
            lcd_data( adhunds );
            adc_lcd_update = 0;
}

if ( !PORTDbits.RD6 ) {
            AD1CON1bits.ADON = 1;          //start ADC
}
}
}


/*---------------------------------------------------------------------
  Function Name: Update_LCD
  Description:   Update LCD for real-time clock data
  Inputs:        None
  Returns:       None
-----------------------------------------------------------------------*/
void Update_LCD( void )
{
  /* position LCD cursor at column, row */
```

```
  home_it();
  cursor_right();
  cursor_right();
  cursor_right();
  cursor_right();
  cursor_right();
  lcd_data(tens + 0x30);
  lcd_data(ones + 0x30);

  hexdec( minutes );
  /* position LCD cursor at column, row */
  cursor_right();
  cursor_right();
  lcd_data(tens + 0x30);
  lcd_data(ones + 0x30);

  hexdec( seconds );
  /* position LCD cursor at column, row */
  cursor_right();
  cursor_right();
  lcd_data(tens + 0x30);
  lcd_data(ones + 0x30);
}


/************************************************************************
* FileName:        traps.c
* Dependencies:    p33FJ256GP710.h
* Processor:       dsPIC33F
* Compiler:        MPLAB® C30 v2.01 or higher
************************************************************************/

#include "p33FJ256GP710.h"

void __attribute__((__interrupt__)) _OscillatorFail(void);
void __attribute__((__interrupt__)) _AddressError(void);
void __attribute__((__interrupt__)) _StackError(void);
void __attribute__((__interrupt__)) _MathError(void);
void __attribute__((__interrupt__)) _DMACError(void);
void __attribute__((__interrupt__)) _AltOscillatorFail(void);
void __attribute__((__interrupt__)) _AltAddressError(void);
void __attribute__((__interrupt__)) _AltStackError(void);
void __attribute__((__interrupt__)) _AltMathError(void);
void __attribute__((__interrupt__)) _AltDMACError(void);


void __attribute__((__interrupt__)) _OscillatorFail(void)
{
    INTCON1bits.OSCFAIL = 0;        //Clear the trap flag
    while (1);
}

void __attribute__((__interrupt__)) _AddressError(void)
{
    INTCON1bits.ADDRERR = 0;        //Clear the trap flag
    while (1);
}
```

```
void __attribute__((__interrupt__)) _StackError(void)
{
	INTCON1bits.STKERR = 0;		//Clear the trap flag
	while (1);
}

void __attribute__((__interrupt__)) _MathError(void)
{
	INTCON1bits.MATHERR = 0;	//Clear the trap flag
	while (1);
}


void __attribute__((__interrupt__)) _DMACError(void)
{
			/* reset status bits, real app should check which ones */
			DMACS0 = 0;

	INTCON1bits.DMACERR = 0;	//Clear the trap flag
	while (1);
}


void __attribute__((__interrupt__)) _AltOscillatorFail(void)
{
	INTCON1bits.OSCFAIL = 0;
	while (1);
}

void __attribute__((__interrupt__)) _AltAddressError(void)
{
	INTCON1bits.ADDRERR = 0;
	while (1);
}

void __attribute__((__interrupt__)) _AltStackError(void)
{
	INTCON1bits.STKERR = 0;
	while (1);
}

void __attribute__((__interrupt__)) _AltMathError(void)
{
	INTCON1bits.MATHERR = 0;
	while (1);
}


void __attribute__((__interrupt__)) _AltDMACError(void)
{
	/* reset status bits, real app should check which ones */
	DMACS0 = 0;
	INTCON1bits.DMACERR = 0;	//Clear the trap flag
	while (1);
}
```

```
/************************************************************************
* FileName:        lcd.c
* Dependencies:    p33FJ256GP710.h
*                  lcd.h
*                  delay.h
* Processor:       dsPIC33F
* Compiler:        MPLAB® C30 v2.01 or higher
************************************************************************/

#include "p33FJ256GP710.h"
#include "lcd.h"
#include "delay.h"

// Control signal data pins
#define  RW  LATDbits.LATD5    // LCD R/W signal
#define  RS  LATBbits.LATB15   // LCD RS signal
#define  E   LATDbits.LATD4    // LCD E signal
//#define  E   LATFbits.LATF1    // LCD E signal

// Control signal pin direction
#define  RW_TRIS                 TRISDbits.TRISD5
#define  RS_TRIS                 TRISBbits.TRISB15
#define  E_TRIS                  TRISDbits.TRISD4
//#define  E_TRIS                  TRISFbits.TRISF1

// Data signals and pin direction
#define  DATA     LATE          // Port for LCD data
#define  DATAPORT  PORTE
#define  TRISDATA  TRISE         // I/O setup for data Port


/************************************************************************/
/*****     LCD SUBROUTINE  *****/

void Init_LCD( void )            // initialize LCD display
{
        // 15mS delay after Vdd reaches nnVdc before proceeding with LCD
        // initialization
        // not always required and is based on system Vdd rise rate
        Delay(Delay_15mS_Cnt);              // 15ms delay

        /* set initial states for the data and control pins */
        LATE &= 0xFF00;
        RW = 0;                  // R/W state set low
        RS = 0;                  // RS state set low
        E = 0;                   // E state set low

        /* set data and control pins to outputs */
        TRISE &= 0xFF00;
        RW_TRIS = 0;             // RW pin set as output
        RS_TRIS = 0;             // RS pin set as output
        E_TRIS = 0;              // E pin set as output
```

```
        /* 1st LCD initialization sequence */
        DATA &= 0xFF00;
        DATA |= 0x0038;
        E = 1;
        Nop();
        Nop();
        Nop();
        E = 0;                  // toggle E signal
        Delay(Delay_5mS_Cnt);       // 5ms delay

        /* 2nd LCD initialization sequence */
        DATA &= 0xFF00;
        DATA |= 0x0038;
        E = 1;
        Nop();
        Nop();
        Nop();
        E = 0;                  // toggle E signal
        Delay_Us( Delay200uS_count ); // 200uS delay

        /* 3rd LCD initialization sequence */
        DATA &= 0xFF00;
        DATA |= 0x0038;
        E = 1;
        Nop();
        Nop();
        Nop();
        E = 0;                  // toggle E signal
        Delay_Us( Delay200uS_count ); // 200uS delay

        lcd_cmd( 0x38 );            // function set
        lcd_cmd( 0x0C );            // Display on/off control, cursor blink off (0x0C)
        lcd_cmd( 0x06 );                        // entry mode set (0x06)
}


void lcd_cmd( char cmd )        // subroutiune for lcd commands
{
        TRISD &= 0xFF00;            // ensure RD0 - RD7 are outputs
        DATA &= 0xFF00;             // prepare RD0 - RD7
        DATA |= cmd;                // command byte to lcd
        RW = 0;                 // ensure RW is 0
        RS = 0;
        E = 1;                  // toggle E line
        Nop();
        Nop();
        Nop();
        E = 0;
        Delay(Delay_5mS_Cnt);       // 5ms delay
}


void lcd_data( char data )      // subroutine for lcd data
{
        TRISD &= 0xFF00;            // ensure RD0 - RD7 are outputs
        RW = 0;                                 // ensure RW is 0
```

```
        RS = 1;                 // assert register select to 1
        DATA &= 0xFF00;             // prepare RD0 - RD7
        DATA |= data;           // data byte to lcd
        E = 1;
        Nop();
        Nop();
        Nop();
        E = 0;                  // toggle E signal
        RS = 0;                 // negate register select to 0
        Delay_Us( Delay200uS_count ); // 200uS delay
        Delay_Us( Delay200uS_count ); // 200uS delay
}


void puts_lcd( unsigned char *data, unsigned char count )
{
        while ( count )
        {
                lcd_data( *data++ );
                count --;
        }
}


/************************************************************************
* FileName:       isr_timer1.c
* Dependencies:   p33FJ256GP710.h
* Processor:      dsPIC33F
* Compiler:       MPLAB® C30 v2.01 or higher
************************************************************************/

#include "p33FJ256GP710.h"

volatile unsigned char hours;
volatile unsigned char minutes;
volatile unsigned char seconds;
volatile unsigned char rtc_lcd_update;

void __builtin_btg(unsigned int *, unsigned int);


/*---------------------------------------------------------------------
  Function Name: _T1Interrupt
  Description:   Timer1 Interrupt Handler
  Inputs:        None
  Returns:       None
---------------------------------------------------------------------*/
void __attribute__((__interrupt__)) _T1Interrupt( void )
{

        if ( seconds < 59 )         // is cummulative seconds < 59?
        {
                seconds++;      // yes, so increment seconds
        }
        else                            // else seconds => 59
        {
                seconds = 0x00;         // reset seconds
```

```
                if ( minutes < 59 )      // is cummulative minutes < 59?
                {
                        minutes++;          // yes, so updates minutes
                }
                else                    // else minutes => 59
                {
                        minutes = 0x00;      // reset minutes
                        if ( hours < 23 )    // is cummulative hours < 23
                        {
                                hours ++;        // yes, so update hours
                        }
                        else
                        {
                                hours = 0x00;    // reset time
                        }
                }
        }

        /* set flag to update LCD */
        rtc_lcd_update = 1;

        /* Toggle LEDs at 1 Hz rate */
        __builtin_btg((unsigned int *)&LATA, 7);
        __builtin_btg((unsigned int *)&LATA, 6);

        /* reset Timer 1 interrupt flag */
        IFS0bits.T1IF = 0;
}


/*****************************************************************************
* FileName:        isr_DMAC0.c
* Dependencies:    p33FJ256GP710.h
* Processor:       dsPIC33F
* Compiler:        MPLAB® C30 v2.01 or higher
*****************************************************************************/

#include "p33FJ256GP710.h"

void __builtin_btg(unsigned int *, unsigned int);
volatile unsigned int temp1;
volatile unsigned char adc_lcd_update;
unsigned int readings[512];
unsigned int x;
unsigned int *pointer;


/*-----------------------------------------------------------------------
  Function Name: DMA0Interrupt
  Description:   DMA Channel0 Interrupt Handler
  Inputs:        None
  Returns:       None
-----------------------------------------------------------------------*/
void __attribute__((__interrupt__)) _DMA0Interrupt( void )
{
        AD1CON1bits.ADON = 0;
```

```
        __builtin_btg((unsigned int *)&LATA, 0);

        /* Save off the RP5 Potentiometer data */
        temp1 = ADC1BUF0;

        pointer=0x7800;
        for(x=0;x<512;x++) {
        readings[x]=*pointer;
        pointer++;
}

/* set flag to update LCD */
adc_lcd_update = 1;
IFS0bits.DMA0IF = 0;          // reset DMA interrupt flag
}


/*****************************************************************************
* FileName:        isr_ADC.c
* Dependencies:    p33FJ256GP710.h
* Processor:       dsPIC33F
* Compiler:        MPLAB® C30 v2.01 or higher
*****************************************************************************/

#include "p33FJ256GP710.h"
#include "common.h"

//volatile unsigned int temp1;
//volatile unsigned char adc_lcd_update;

/*-----------------------------------------------------------------------
  Function Name: ADCInterrupt
  Description:   ADC Interrupt Handler
  Inputs:        None
  Returns:       None
-----------------------------------------------------------------------*/
void __attribute__((__interrupt__)) _ADC1Interrupt( void )
{
        int count;

        /* Simple I am here indicator */
        if ( count++ == 2000 ) {
                __builtin_btg( (unsigned int *)&LATA, 3 );
                count = 0;
        }

        /* set flag to update LCD */
        adc_lcd_update = 1;

        /* reset ADC interrupt flag */
        IFS0bits.AD1IF = 0;
}
```

```
/*********************************************************************
* FileName:        init_timer1.c
* Dependencies:    p33FJ256GP710.h
* Processor:       dsPIC33F
* Compiler:        MPLAB® C30 v2.01 or higher
*********************************************************************/

#include "p33FJ256GP710.h"

/* Set SLEEP to 1 is dsPIC will enter sleep in main loop */
#define SLEEP  0


/*---------------------------------------------------------------------
  Function Name: Init_Timer1
  Description:   Initialize Timer1 for 1 second intervals
  Inputs:        None
  Returns:       None
---------------------------------------------------------------------*/
void Init_Timer1( void )
{
        /* declare temp variable for CPU IPL storing */
        int current_cpu_ipl;

        /* ensure Timer 1 is in reset state */
        T1CON = 0;

        /* reset Timer 1 interrupt flag */
        IFS0bits.T1IF = 0;

        /* set Timer1 interrupt priority level to 4 */
        IPC0bits.T1IP = 4;

        /* enable Timer 1 interrupt */
        IEC0bits.T1IE = 1;

        /* set Timer 1 period register */
        PR1 = 0x8000;

        /* select external timer clock */
        T1CONbits.TCS = 1;

        /* disable interrupts for unlock sequence below */
//      SET_AND_SAVE_CPU_IPL(current_cpu_ipl, 7);

        char a, b, c, *p;

        a = 2;
        b = 0x46;
        c = 0x57;
        p = (char *)&OSCCON;

        /* enable 32KHz Oscillator here
        low byte unlock sequence and enable LP Oscillator */
        asm volatile ("mov.b %1,[%0] \n"
        "mov.b %2,[%0] \n"
        "mov.b %3,[%0] \n" : /* no outputs */ : "r"(p), "r"(b), "r"(c),
```

```
                                        "r"(a));

        /* restore CPU IPL value after executing unlock sequence */
        RESTORE_CPU_IPL(current_cpu_ipl);

        /* enable Timer 1 and start the count */
        T1CONbits.TON = 1;


#if  SLEEP == 1
         /* If main loop will enter SLEEP mode then wait here
        until Oscillator starts and Timer begins to count */
        while ( TMR1 < 2 );
#endif
}


/************************************************************************
* FileName:        init_DMAC0.c
* Dependencies:    p33FJ256GP710.h
* Processor:       dsPIC33F
* Compiler:        MPLAB® C30 v2.01 or higher
************************************************************************/

#include "p33FJ256GP710.h"

/*-----------------------------------------------------------------------
  Function Name: Init_DMAC0
  Description:   Initialize DMA Channel 0
  Inputs:        None
  Returns:       None
-----------------------------------------------------------------------*/
void Init_DMAC0( void )
{
        /* DMA buffer 1 Start address for DPSRAM Address */
        DMA0STA = 0x0000;

        /* DMA buffer 2 Start address for DPSRAM Address */
        DMA0STB = 0x0200;

        /* initialize DMA Peripheral SFR Address */
        DMA0PAD = (int) &ADC1BUF0;

        /* Set transfer count to 512 words*/
        DMA0CNT = 0x01FF;

        /* reset status register */
        DMACS0 = 0x0000;

        /* DMA Channel Setup  */
   /* Word transfer, Read from peripheral/Write to DPSRAM
      Interrupt when block transfer complete
      Automatic DMA transfer initiation by DMA request
      Register indirect, post increment
      Continuous, ping-pong disabled */
        DMA0CON = 0x0000;
```

```
        /* ADC1 IRQ numer = 13 (D) */
        DMA0REQ = 0x000D;

        /* set DMA CH0 interrupt priority level to 4 */
        IPC1bits.DMA0IP = 4;
        /* reset DMA CH0 interrupt flag */
        IFS0bits.DMA0IF = 0;
        /* enable DMA CH0 interrupts */
        IEC0bits.DMA0IE = 1;

        /* enable DMA Channel 0 */
        DMA0CONbits.CHEN = 1;
}


/******************************************************************************
* FileName:        init_ADC.c
* Dependencies:    p33FJ256GP710.h
* Processor:       dsPIC33F
* Compiler:        MPLAB® C30 v2.01 or higher
******************************************************************************/

#include "p33FJ256GP710.h"

/*-----------------------------------------------------------------------
  Function Name: Init_ADC
  Description:   Initialize ADC module
  Inputs:        None
  Returns:       None
-----------------------------------------------------------------------*/
void Init_ADC( void )
{
        /* set port configuration here */
        AD1PCFGLbits.PCFG4 = 0;         // ensure AN4/RB4 is analog (Temp Sensor)
        AD1PCFGLbits.PCFG5 = 0;         // ensure AN5/RB5 is analog (Analog Pot)

        /* set channel scanning here, auto sampling and convert, with default read-
        format mode */
        AD1CON1 = 0x00E4;

        /* select 10-bit, 1 channel ADC operation */
        AD1CON1bits.AD12B = 0;

        /* enable DMA mode (ADC module sends all results to ADBUF0 and
        interrupts on each result */
        AD1CON1bits.ADDMABM = 1;

        /* No channel scan for CH0+, Use MUX A, SMPI = 1 per interrupt, Vref =
        AVdd/AVss */
        AD1CON2 = 0x0000;

        /* Set Samples and bit conversion time */
        AD1CON3 = 0x003F;

        /* Set number of DMA buffer locations */
        AD1CON4 = 0x0003;
```

```
        /* set channel scanning here for AN4 and AN5 */
        AD1CSSL = 0x003F;

        /* channel select AN5 */
        AD1CHS0 = 0x0005;

        /* reset ADC interrupt flag */
        IFS0bits.AD1IF = 0;

        /* enable ADC interrupts, disable this interrupt if the DMA is enabled */
        IEC0bits.AD1IE = 1;

        /* turn on ADC module */
        AD1CON1bits.ADON = 1;
}


/************************************************************************
* FileName:        hexdec.c
* Dependencies:    p33FJ256GP710.h
* Processor:       dsPIC33F
* Compiler:        MPLAB® C30 v2.01 or higher
************************************************************************/

#include "p33FJ256GP710.h"

volatile unsigned char hunds;
volatile unsigned char tens;
volatile unsigned char ones;

void hexdec( unsigned char count )
{
  hunds = 0;                                    //initialize hundred
  tens  = 0;                                    //initialize tens
  ones = 0;                                             //initialzise ones

  while ( count >= 10 ) {
    if ( count >= 200 )  {              //check two hundreds
      count -= 200;                             //subtract 200
      hunds = 0x02;                             //set for 2 hundred
    }

    if (count >= 100) {                         //check hundreds
      count -= 100;                             //subract 100
            hunds++;                                        //increment hundred register
    }

    if (count >= 10 ) {                         //check tens
      count -= 10;                      //subtract 10
      tens++;                                   //increment tens
    }
  }
  ones = count;                                 //remaining count equals ones
}
```

```
/************************************************************************
*                   Simple Delay Routines
************************************************************************
 * FileName:        delay.c
 * Dependencies:    delay.h
 * Processor:       dsPIC33F
 * Complier:        MPLAB C30 v2.01.00 or higher
 ************************************************************************/

#include "delay.h"

unsigned int temp_count;

void Delay( unsigned int delay_count )
{
        temp_count = delay_count +1;
        asm volatile("outer: dec _temp_count");
        asm volatile("cp0 _temp_count");
        asm volatile("bra z, done");
        asm volatile("do #3200, inner" );
        asm volatile("nop");
        asm volatile("inner: nop");
        asm volatile("bra outer");
        asm volatile("done:");
}


void Delay_Us( unsigned int delayUs_count )
{
        temp_count = delayUs_count +1;
        asm volatile("outer1: dec _temp_count");
        asm volatile("cp0 _temp_count");
        asm volatile("bra z, done1");
        asm volatile("do #1500, inner1" );
        asm volatile("nop");
        asm volatile("inner1: nop");
        asm volatile("bra outer1");
        asm volatile("done1:");
}


/************************************************************************
* FileName:        advolts.c
* Dependencies:    p33FJ256GP710.h
* Processor:       dsPIC33F
* Compiler:        MPLAB® C30 v2.01 or higher
************************************************************************/

volatile unsigned char adones;
volatile unsigned char adtens;
volatile unsigned char adhunds;
volatile unsigned char adthous;

/*----------------------------------------------------------------------
  Function Name: advolt
  Description:   Convert Raw ADC data to volts for LCD display
  Inputs:        Raw ADC data
```

```
  Returns:      None
-------------------------------------------------------------------------*/
void advolt( unsigned int adc_conv_data )
{
        adones = 0;                             // reset values
        adtens = 0;
        adhunds = 0;
        adthous = 0;

         while ( adc_conv_data > 0 )
         {
                if( adc_conv_data > 300 )       //test for 1 volt or greater
                {
                        adones++;               //increment 1 volt counter
                        adc_conv_data -= 300;   //subtract 1 volt
                }

        else if( adc_conv_data > 30 )           //test for 0.1 volt
        {
                if ( adtens < 9 )
                {
                        adtens++;               //increment tenths
                }
                else
                {
                        adones++;               //tenths has rolled over
                        adtens = 0;             //so increment ones and reset tenths
                }
                adc_conv_data -= 30;            //test for 0.01 volt
        }

        else if( adc_conv_data > 3 )
        {
                adhunds++;                      //increment hundreths
                adc_conv_data -= 3;
        }

        else if ( adc_conv_data <= 3 )
        {
                adthous++;
                adc_conv_data --;
        }
 }

adones += 0x30;
adtens += 0x30;
adhunds += 0x30;
adthous += 0x30;
}
```

Third Prototype Software Listing

```
/*****************************************************
Author  : Glen McGillan
Chip type           : ATmega48
Clock frequency     : 8.000000 MHz
*****************************************************/

#include <mega48.h>
#include <sleep.h>
#define ADC_VREF_TYPE 0x40


unsigned char n_edge;
unsigned char p_edge;
unsigned int delay;
unsigned long on_timer;
unsigned int battery;
unsigned int voltage;
unsigned int p_current;
unsigned int n_current;
unsigned char current;
unsigned char fault_direction;


unsigned char convert(unsigned char value, unsigned char DP)
{
  unsigned char temp;

  switch(value) {
    case 0:
      temp=0x11;
      break;
    case 1:
      temp=0x7D;
      break;
    case 2:
      temp=0x23;
      break;
    case 3:
      temp=0x29;
      break;
    case 4:
      temp=0x4D;
      break;
    case 5:
      temp=0x89;
      break;
    case 6:
      temp=0x81;
      break;
    case 7:
      temp=0x3D;
```

```
      break;
    case 8:
      temp=0x01;
      break;
    case 9:
      temp=0x09;
      break;
    default:
      temp=0xFF;
  }
  if(DP) {
    temp&=0xFE;
  }
  return temp;
}


void displayout(unsigned char shiftout)
{
  unsigned char i;
  for(i=0;i<8;i++) {
    PORTB.0=shiftout&0x80;
    shiftout=shiftout<<1;
    PORTB.2=1; //} clock pulse
    PORTB.2=0; //}
  }
}


void displayV(unsigned int value, unsigned char DP)
{
  unsigned char ones=0;
  unsigned char tens=0;
  unsigned char hundreds=0;

  while(value>=100) {
    hundreds++;
    value-=100;
  }
  while(value>=10) {
    tens++;
    value-=10;
  }
  ones=value;

  if(!hundreds) {
    hundreds=0xFF;
  }

  displayout( convert(ones,0) );
  displayout( convert(tens,DP) );
  displayout( convert(hundreds,0) );
}


unsigned int read_adc(unsigned char adc_input)
{
```

```
  ADMUX=adc_input|ADC_VREF_TYPE;
  ADCSRA|=0x40;                         // Start the AD conversion
  while ((ADCSRA & 0x10)==0);           // Wait for the AD conversion to complete
  ADCSRA|=0x10;
  return ADCW;
}


/*
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{
  TCNT1H=0;
  TCNT1L=0;
  TCCR1B&=0xF8;        // stop the timer (runs only once each time)
}
*/


void display(void)
{
  //store previous readings into memory table (circular buffer)
  PRR=0xFE;              // enable the ADC system
  voltage=read_adc(5);   // first reading to ensure that the ADC is fully running

  if(n_edge) {
    voltage=read_adc(5); // read negative voltage level (pin28)
  }
  else {
    voltage=read_adc(4); // read positive voltage level (pin27)
  }

  p_current=read_adc(3);          // read positive current level (pin26)
  n_current=read_adc(2);          // read negative current level (pin25)
  //}calculate which current value to use:
  //}   -ve current, current=0
  //}   +ve current, current=1

  //} calculate fault direction:
  if(p_edge && current) {//}   +ve V, +ve I = left
    fault_direction=0;
  }
  else if(n_edge && !current) {     //}   -ve V, -ve I = left
    fault_direction=0;
  }
  else if(p_edge && !current) {     //}   +ve V, -ve I = right
    fault_direction=1;
  }
  else {                        //}   -ve V, +ve I = right
    fault_direction=1;
  }

  battery=read_adc(1);
  if(battery<=100) {
//    low battery indication
  }

  displayV((unsigned char)voltage/4,1);
```

```
  // display information (LCD?)
  //   -direction (if current below 5A, then don't display direction)
  //   -voltage
  //   -current
  //   -flash to indicate new pulse received
}


// Pin change 0-7 interrupt service routine
interrupt [PCINT0] void pin_change_isr0(void)
{
  PORTC.0=1;              // power on
  on_timer=144928;
}


// External Interrupt 0 service routine (falling edge)
interrupt [EXT_INT0] void ext_int0_isr(void)
{
  if(!delay && !p_edge) {
    n_edge=1;
    PORTC.0=1;            // power on
    on_timer=144928;
  }
}


// External Interrupt 1 service routine (rising edge)
interrupt [EXT_INT1] void ext_int1_isr(void)
{
  if(!delay && !n_edge) {
    p_edge=1;
    PORTC.0=1;            // power on
    on_timer=144928;
  }
}


void main(void)
{
  #pragma optsize-
    CLKPR=0x80;                 //} Crystal Oscillator division factor: 8
    CLKPR=0x03;                 //}
    #ifdef _OPTIMIZE_SIZE_
      #pragma optsize+
    #endif

  PORTB=0xFA;//}
  DDRB=0x05;  //}
  PORTC=0x4E;//} port initialisation
  DDRC=0x01;  //}
  PORTD=0xF0;//}
  DDRD=0x03;  //}
  TCCR0A=0x00;                  //}
  TCCR0B=0x00;                  //} timer 0:
  TCNT0=0x00;                   //}  stopped
```

```
OCR0A=0x00;           //}
OCR0B=0x00;           //}
TCCR1A=0x00;          //}
TCCR1B=0x05;          //}
TCNT1H=0x00;          //} timer 1:
TCNT1L=0x00;          //}   977Hz clock
ICR1H=0x00;           //}   overflow interrupt
ICR1L=0x00;           //}
OCR1AH=0x00;          //}
OCR1AL=0x00;          //}
OCR1BH=0x00;          //}
OCR1BL=0x00;          //}
ASSR=0x00;            //}
TCCR2A=0x00;          //} timer 2:
TCCR2B=0x00;          //}   stopped
TCNT2=0x00;           //}
OCR2A=0x00;           //}
OCR2B=0x00;           //}
EICRA=0x0E;           //}
EIMSK=0x03;           //} external interrupts:
EIFR=0x03;            //}   INT0 - falling edge
PCICR=0x01;           //}   INT1 - rising edge
PCMSK0=0x02;          //}   PCINT1 - any change
PCIFR=0x01;           //}
TIMSK0=0x00;          //}
TIMSK1=0x00;          //} timer interrupts
TIMSK2=0x00;          //}
ACSR=0x80;            //} analogue comparator off
ADCSRB=0x00;          //}
DIDR0=0x3E;           //} ADC initialisation:
ADMUX=ADC_VREF_TYPE;  //}   125kHz, AVCC pin
ADCSRA=0x83;                //}   ADC1-5 active
PRR=0xFF;     // disable ADC system (power reduction)

sleep_enable();

#asm("sei")     // global interrupt enable

while (1)
{
  if(n_edge) {          //} negative edge detected:
    PORTD.0=1;          //}   negative LED on
    display();
    n_edge=0;           //}   clear flag
    delay=17647;        //}   set delay before new edge can be detected
  }

  if(p_edge) {          //} positive edge detected:
    PORTD.1=1;
    display();
    p_edge=0;
    delay=17647;
  }

  while(delay) {//} wait delay/blanking period before looking for additional edges
    delay--;      //}
  }
```

```
    PORTD.0=0;          //} green LEDs off
    PORTD.1=0;          //}

    while(on_timer && !n_edge && !p_edge) {
      on_timer--;
    }
    if(!on_timer) {
      PORTC.0=0;                    // secondary power off
      CLKPR=0x80;                   //} Crystal Oscillator division factor: 256 (32kHz)
      CLKPR=0x08;                   //}
      idle();
    }
    CLKPR=0x80;                     //} Crystal Oscillator division factor: 8 (1MHz)
    CLKPR=0x03;                     //}
  }
}
```

Fourth Prototype Software Listing

```
//*************************************************************************************************
#include <mega32.h>
#include <OLED.h>
#include <delay.h>
#include <stdlib.h>
#include <logo.h>

//*************************************************************************************************/
/ Main routine
//*************************************************************************************************
void main(void)
{
  unsigned char edge=0;
  unsigned int Vpos_reading,Vneg_reading,Ipos_reading,Ineg_reading;
  unsigned long timing;
  unsigned char idle=0;

  initialise();                         // initialise microcontroller

  OLED_power=1;
  delay_ms(100);

  OLED_initialise();                    // initialise OLED

  delay_ms(120);

  I_power=1;
  V_power=1;

  display_logo();

  while(1) {
    delay_ms(100);
    timing=500000;
    idle=0;
    while(!edge) {
      if(!Button1_pin || trigger_pos) {
        edge=positive;
      }
      else if(!Button2_pin || !trigger_neg) {
        edge=negative;
      }
      else {
        if(!idle) {
          if(timing) {
            timing--;
          }
          else {
            idle=1;
            display_logo();
            timing=500000;
          }
        }
```

```
  }
}
delay_ms(5);
Vpos_reading=read_adc(V_pos);
Vneg_reading=read_adc(V_neg);
Ipos_reading=read_adc(I_pos);
Ineg_reading=read_adc(I_neg);

if(Vpos_reading>999) {
  Vpos_reading=999;
}
if(Vneg_reading>999) {
  Vneg_reading=999;
}
if(Ipos_reading>999) {
  Ipos_reading=999;
}
if(Ineg_reading>999) {
  Ineg_reading=999;
}


OLED_write_cmd(0x8e);                  // clear window command
OLED_write_data(0);
OLED_write_data(0);
OLED_write_data(130);
OLED_write_data(130);

delay_ms(100);

OLED_write_cmd(0x92);                  // fill enable command
OLED_write_data(0x01);

delay_ms(10);

if(Ipos_reading>=50 || Ineg_reading>=50) {
  if(Ipos_reading>Ineg_reading) {
    if(edge==positive) {
      display_leftarrow(0,0);
    } else {
      display_rightarrow(0,0);
    }
  } else {
    if(edge==positive) {
      display_rightarrow(0,0);
    } else {
      display_leftarrow(0,0);
    }
  }
}


if(edge==positive) {
  display_value( (Vpos_reading/6.5)-1 ,42);
}
else {
  display_value( (Vneg_reading/5) ,42);
```

```
    }
    display_kV(112,42);


    if(Ipos_reading>Ineg_reading) {
      display_value(Ipos_reading,84);
    }
    else {
      display_value(Ineg_reading,84);
    }
    display_A(112,84);

    edge=0;
  }
}



//***********************************************************************************************
// Initialisation routine
//***********************************************************************************************
void initialise(void)
{
// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=Out Func4=Out Func3=Out Func2=In Func1=In Func0=In
// State7=T State6=T State5=0 State4=0 State3=0 State2=T State1=T State0=T
PORTA=0x00;
DDRA=0x38;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=Out Func3=Out Func2=Out Func1=In Func0=In
// State7=P State6=P State5=P State4=0 State3=0 State2=0 State1=P State0=P
PORTB=0xE3;
DDRB=0x1C;

// Port C initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTC=0x00;
DDRC=0xFF;

// Port D initialization
// Func7=Out Func6=In Func5=In Func4=Out Func3=In Func2=In Func1=Out Func0=Out
// State7=0 State6=P State5=P State4=0 State3=T State2=T State1=0 State0=0
PORTD=0x60;
DDRD=0x93;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 8000.000 kHz
// Mode: Phase correct PWM top=FFh
// OC0 output: Non-Inverted PWM
```

```
TCCR0=0x61;
TCNT0=0x00;
OCR0=0x7F;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC Clock frequency: 250.000 kHz
// ADC Voltage Reference: AVCC pin
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x85;
}
```

```
//*****************************************************************************************************
// OLED Data Write routine
//*****************************************************************************************************
void OLED_write_data(unsigned long Data)
{
  OLED_read_disable();                  //}
  OLED_data();                          //} IOSET=DC|RD;

  OLED_write_enable();                  //}
  OLED_active();                        //} IOCLR=WR|CS;

  OLED_data_bus=0;                      // IOCLR=0x3ffff;

  OLED_data_bus=Data;                   // IOSET=Data;

  OLED_write_disable();                 //}
  OLED_inactive();                      //} IOSET=WR|CS;
}


//*****************************************************************************************************
// OLED Command Write routine
//*****************************************************************************************************
void OLED_write_cmd(unsigned char Command)
{
  OLED_read_disable();                  // IOSET=RD;

  OLED_command();                       //}
  OLED_write_enable();                  //} IOCLR=WR|CS|DC;
  OLED_active();                        //}

  OLED_data_bus=0;                      // IOCLR=0xff;

  OLED_data_bus=Command;                //} IOSET=Command;

  OLED_write_disable();                 //} IOSET=WR|CS;
  OLED_inactive();                      //}
}


//*****************************************************************************************************
// OLED Initialisation routine
//*****************************************************************************************************
void OLED_initialise(void)
{
  OLED_reset();                         //}
  delay_ms(100);                        //} reset OLED
  OLED_non_reset();                     //}

  OLED_write_cmd(0xA0);                 //} 262K 8bit R->G->B
  OLED_write_data(0xB4);                //}


  OLED_write_cmd(0xA1);                 //} Display_Start_Line
  OLED_write_data(0x00);                //}
```

```
  OLED_write_cmd(0xA2);              //} Display_Offset
  OLED_write_data(0x80);             //}

  OLED_write_cmd(0xCA);              //} Mux_Ratio
  OLED_write_data(0x7F);             //}

  OLED_write_cmd(0xA6);              // Display_Mode

  OLED_write_cmd(0xAD);              //} Master_Conf
  OLED_write_data(0x8E);             //}

  OLED_write_cmd(0xC7);              //} Master_Contrast
  OLED_write_data(0x0f);             //}

  OLED_write_cmd(0xC1);              //}
  OLED_write_data(0xff);             //} Contrast_Current
  OLED_write_data(0xff);             //}
  OLED_write_data(0xff);             //}

  OLED_write_cmd(0xBE);              //} VCOMH
  OLED_write_data(0x3f);             //}

  OLED_write_cmd(0xB0);              //} Power_Saving
  OLED_write_data(0x05);             //}

  OLED_write_cmd(0xBB);              //}
  OLED_write_data(0x1c);             //} Precharge_Color
  OLED_write_data(0x1c);             //}
  OLED_write_data(0x1c);             //}

  OLED_write_cmd(0xB1);              //} Reset_Pre_charge
  OLED_write_data(0x11);             //}

  OLED_write_cmd(0xB3);              //} Osc_Freq
  OLED_write_data(0xf0);             //}

  OLED_write_cmd(0xAF);              // Display_ON
}


//***********************************************************************************************
// Read the AD conversion result
//***********************************************************************************************
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}
```

```
//*************************************************************************************************
void display_number(unsigned char top_left_x, unsigned char top_left_y)
{
  unsigned char x,y,z,k,j;

  for(y=0;y<36;y++) {
    OLED_write_cmd(0x15);                    // set to start of column
    OLED_write_data(top_left_x);
    OLED_write_cmd(0x75);                    // set to next row
    OLED_write_data((int)y+top_left_y);
    OLED_write_cmd(0x5c);                    // write directly to ram
    for(x=0;x<4;x++) {
      z=one[x+((int)y*4)];
      for(k=0;k<8;k++) {
        if( z & 0x80>>k ) {
          j=255;
        } else {
          j=0;
        }
        OLED_write_data(j);
        OLED_write_data(j);
        OLED_write_data(j);
      }
    }
  }
}

//*************************************************************************************************
void display_decimal(unsigned char top_left_x, unsigned char top_left_y)
{
  unsigned char x,y,z,k,j;

  for(y=0;y<36;y++) {
    OLED_write_cmd(0x15);                    // set to start of column
    OLED_write_data(top_left_x);
    OLED_write_cmd(0x75);                    // set to next row
    OLED_write_data((int)y+top_left_y);
    OLED_write_cmd(0x5c);                    // write directly to ram
    for(x=0;x<2;x++) {
      z=decimal_point[x+((int)y*2)];
      for(k=0;k<8;k++) {
        if( z & 0x80>>k ) {
          j=255;
        } else {
          j=0;
        }
        OLED_write_data(j);
        OLED_write_data(j);
        OLED_write_data(j);
      }
    }
  }
}
```

```
//*************************************************************************************************
void display_arrow(unsigned char top_left_x, unsigned char top_left_y)
{
  unsigned char x,y,z,k,j;

  for(y=0;y<36;y++) {
    OLED_write_cmd(0x15);                       // set to start of column
    OLED_write_data(top_left_x);
    OLED_write_cmd(0x75);                       // set to next row
    OLED_write_data((int)y+top_left_y);
    OLED_write_cmd(0x5c);                       // write directly to ram
    for(x=0;x<16;x++) {
      z=left_arrow[x+((int)y*16)];
      for(k=0;k<8;k++) {
        if( z & 0x80>>k ) {
          j=255;
        } else {
          j=0;
        }
        OLED_write_data(j);
        OLED_write_data(j);
        OLED_write_data(j);
      }
    }
  }
}


//*************************************************************************************************
void display_logo(void)
{
  unsigned char y;
  unsigned int x;

  OLED_write_cmd(0x8e);                 // clear window command
  OLED_write_data(0);
  OLED_write_data(0);
  OLED_write_data(130);
  OLED_write_data(130);
  delay_ms(100);
  OLED_write_cmd(0x92);                 // fill enable command
  OLED_write_data(0x01);
  delay_ms(10);

  for(y=0;y<49;y++) {
    OLED_write_cmd(0x15);               // set to start of column
    OLED_write_data(0);
    OLED_write_cmd(0x75);               // set to next row
    OLED_write_data((int)y+30);
    OLED_write_cmd(0x5c);               // write directly to ram
    for(x=0;x<384;x=x+3) {
      OLED_write_data(  logo[x+((int)y*384)]  );
      OLED_write_data(  logo[x+((int)y*384)+1]  );
      OLED_write_data(  logo[x+((int)y*384)+2]  );
    }
  }
}
```

```
//*********************************************************************************************
void display_value(unsigned int value, unsigned char y)
{
  unsigned char x100,x10,x1;

  x100=0;
  x10=0;
  x1=0;

  while(value>=100) {
    value-=100;
    x100++;
  }
  while(value>=10) {
    value-=10;
    x10++;
  }
  x1=value;

  if(x100) {
    display_number(x100,0,y);
  }
  display_number(x10,32,y);
  display_decimal(64,y);
  display_number(x1,80,y);
}


//*********************************************************************************************
flash unsigned char zero[144] =
{
   0x00 , 0x0F , 0xF8 , 0x00 , 0x00 , 0x7F , 0xFE , 0x00
 , 0x01 , 0xFF , 0xFF , 0x80 , 0x03 , 0xFF , 0xFF , 0xC0
 , 0x07 , 0xFF , 0xFF , 0xE0 , 0x0F , 0xFF , 0xFF , 0xE0
 , 0x0F , 0xFC , 0x3F , 0xF0 , 0x1F , 0xF8 , 0x1F , 0xF0
 , 0x1F , 0xF0 , 0x0F , 0xF8 , 0x1F , 0xF0 , 0x0F , 0xF8
 , 0x1F , 0xF0 , 0x0F , 0xF8 , 0x3F , 0xE0 , 0x07 , 0xF8
 , 0x3F , 0xE0 , 0x07 , 0xFC , 0x3F , 0xE0 , 0x07 , 0xFC
 , 0x3F , 0xE0 , 0x07 , 0xFC , 0x3F , 0xE0 , 0x07 , 0xFC
 , 0x3F , 0xE0 , 0x07 , 0xFC , 0x3F , 0xE0 , 0x07 , 0xFC
 , 0x3F , 0xE0 , 0x07 , 0xFC , 0x3F , 0xE0 , 0x07 , 0xFC
 , 0x3F , 0xE0 , 0x07 , 0xFC , 0x3F , 0xE0 , 0x07 , 0xFC
 , 0x3F , 0xE0 , 0x07 , 0xFC , 0x3F , 0xE0 , 0x07 , 0xFC
 , 0x1F , 0xE0 , 0x07 , 0xF8 , 0x1F , 0xF0 , 0x0F , 0xF8
 , 0x1F , 0xF0 , 0x0F , 0xF8 , 0x1F , 0xF0 , 0x0F , 0xF8
 , 0x0F , 0xF8 , 0x1F , 0xF0 , 0x0F , 0xFC , 0x3F , 0xF0
 , 0x07 , 0xFF , 0xFF , 0xF0 , 0x07 , 0xFF , 0xFF , 0xE0
 , 0x03 , 0xFF , 0xFF , 0xC0 , 0x01 , 0xFF , 0xFF , 0x80
 , 0x00 , 0x7F , 0xFE , 0x00 , 0x00 , 0x1F , 0xF8 , 0x00
};

//*********************************************************************************************
flash unsigned char one[144] =
{
   0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x3F , 0xC0
 , 0x00 , 0x00 , 0x7F , 0xC0 , 0x00 , 0x00 , 0xFF , 0xC0
 , 0x00 , 0x00 , 0xFF , 0xC0 , 0x00 , 0x01 , 0xFF , 0xC0
```

```
 , 0x00 , 0x07 , 0xFF , 0xC0 , 0x00 , 0x0F , 0xFF , 0xC0
 , 0x00 , 0x3F , 0xFF , 0xC0 , 0x01 , 0xFF , 0xFF , 0xC0
 , 0x03 , 0xFF , 0xFF , 0xC0 , 0x03 , 0xFF , 0xFF , 0xC0
 , 0x03 , 0xFF , 0xFF , 0xC0 , 0x03 , 0xFE , 0xFF , 0xC0
 , 0x03 , 0xFC , 0xFF , 0xC0 , 0x03 , 0xF0 , 0xFF , 0xC0
 , 0x03 , 0xC0 , 0xFF , 0xC0 , 0x03 , 0x00 , 0xFF , 0xC0
 , 0x00 , 0x00 , 0xFF , 0xC0 , 0x00 , 0x00 , 0xFF , 0xC0
 , 0x00 , 0x00 , 0xFF , 0xC0 , 0x00 , 0x00 , 0xFF , 0xC0
 , 0x00 , 0x00 , 0xFF , 0xC0 , 0x00 , 0x00 , 0xFF , 0xC0
 , 0x00 , 0x00 , 0xFF , 0xC0 , 0x00 , 0x00 , 0xFF , 0xC0
 , 0x00 , 0x00 , 0xFF , 0xC0 , 0x00 , 0x00 , 0xFF , 0xC0
 , 0x00 , 0x00 , 0xFF , 0xC0 , 0x00 , 0x00 , 0xFF , 0xC0
 , 0x00 , 0x00 , 0xFF , 0xC0 , 0x00 , 0x00 , 0xFF , 0xC0
 , 0x00 , 0x00 , 0xFF , 0xC0 , 0x00 , 0x00 , 0xFF , 0xC0
 , 0x00 , 0x00 , 0xFF , 0xC0 , 0x00 , 0x00 , 0xFF , 0xC0
};

//*********************************************************************************************
flash unsigned char two[144] =
{
   0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x1F , 0xFC , 0x00
 , 0x00 , 0x7F , 0xFF , 0x80 , 0x01 , 0xFF , 0xFF , 0xE0
 , 0x03 , 0xFF , 0xFF , 0xF0 , 0x07 , 0xFF , 0xFF , 0xF0
 , 0x07 , 0xFF , 0xFF , 0xF8 , 0x0F , 0xFE , 0x1F , 0xF8
 , 0x0F , 0xF8 , 0x0F , 0xFC , 0x1F , 0xF8 , 0x07 , 0xFC
 , 0x1F , 0xF8 , 0x07 , 0xFC , 0x1F , 0xF0 , 0x07 , 0xFC
 , 0x01 , 0xF0 , 0x07 , 0xFC , 0x00 , 0x00 , 0x0F , 0xFC
 , 0x00 , 0x00 , 0x1F , 0xF8 , 0x00 , 0x00 , 0x3F , 0xF8
 , 0x00 , 0x00 , 0x7F , 0xF0 , 0x00 , 0x00 , 0xFF , 0xF0
 , 0x00 , 0x01 , 0xFF , 0xE0 , 0x00 , 0x07 , 0xFF , 0xC0
 , 0x00 , 0x0F , 0xFF , 0x80 , 0x00 , 0x1F , 0xFF , 0x00
 , 0x00 , 0x3F , 0xFC , 0x00 , 0x00 , 0x7F , 0xF8 , 0x00
 , 0x00 , 0xFF , 0xF0 , 0x00 , 0x01 , 0xFF , 0xC0 , 0x00
 , 0x03 , 0xFF , 0x80 , 0x00 , 0x07 , 0xFF , 0x00 , 0x00
 , 0x07 , 0xFF , 0xFF , 0xFC , 0x0F , 0xFF , 0xFF , 0xFC
 , 0x1F , 0xFF , 0xFF , 0xFC , 0x1F , 0xFF , 0xFF , 0xFC
 , 0x1F , 0xFF , 0xFF , 0xFC , 0x3F , 0xFF , 0xFF , 0xFC
 , 0x3F , 0xFF , 0xFF , 0xFC , 0x3F , 0xFF , 0xFF , 0xFC
};

//*********************************************************************************************
flash unsigned char three[144] =
{
   0x00 , 0x1F , 0xF8 , 0x00 , 0x00 , 0x7F , 0xFF , 0x00
 , 0x01 , 0xFF , 0xFF , 0x80 , 0x03 , 0xFF , 0xFF , 0xC0
 , 0x07 , 0xFF , 0xFF , 0xE0 , 0x0F , 0xFF , 0xFF , 0xE0
 , 0x0F , 0xFC , 0x3F , 0xF0 , 0x0F , 0xF8 , 0x1F , 0xF0
 , 0x1F , 0xF8 , 0x1F , 0xF0 , 0x0F , 0xF0 , 0x1F , 0xF0
 , 0x00 , 0x70 , 0x1F , 0xF0 , 0x00 , 0x00 , 0x1F , 0xF0
 , 0x00 , 0x00 , 0x3F , 0xE0 , 0x00 , 0x00 , 0x7F , 0xE0
 , 0x00 , 0x07 , 0xFF , 0xC0 , 0x00 , 0x07 , 0xFF , 0x80
 , 0x00 , 0x07 , 0xFF , 0x00 , 0x00 , 0x07 , 0xFF , 0xC0
 , 0x00 , 0x07 , 0xFF , 0xF0 , 0x00 , 0x07 , 0xFF , 0xF8
 , 0x00 , 0x00 , 0x1F , 0xF8 , 0x00 , 0x00 , 0x0F , 0xFC
 , 0x00 , 0x00 , 0x07 , 0xFC , 0x00 , 0x00 , 0x07 , 0xFC
 , 0x00 , 0x00 , 0x07 , 0xFC , 0x01 , 0xE0 , 0x07 , 0xFC
 , 0x3F , 0xE0 , 0x07 , 0xFC , 0x3F , 0xF0 , 0x0F , 0xFC
```

```
 , 0x1F , 0xF0 , 0x0F , 0xF8 , 0x1F , 0xFC , 0x3F , 0xF8
 , 0x0F , 0xFF , 0xFF , 0xF0 , 0x07 , 0xFF , 0xFF , 0xF0
 , 0x07 , 0xFF , 0xFF , 0xE0 , 0x01 , 0xFF , 0xFF , 0xC0
 , 0x00 , 0xFF , 0xFF , 0x00 , 0x00 , 0x1F , 0xF8 , 0x00
};

//*************************************************************************************************
flash unsigned char four[144] =
{
  0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x7F , 0xE0
 , 0x00 , 0x00 , 0xFF , 0xE0 , 0x00 , 0x01 , 0xFF , 0xE0
 , 0x00 , 0x03 , 0xFF , 0xE0 , 0x00 , 0x07 , 0xFF , 0xE0
 , 0x00 , 0x07 , 0xFF , 0xE0 , 0x00 , 0x0F , 0xFF , 0xE0
 , 0x00 , 0x1F , 0xFF , 0xE0 , 0x00 , 0x3F , 0xFF , 0xE0
 , 0x00 , 0x7F , 0xFF , 0xE0 , 0x00 , 0x7F , 0xFF , 0xE0
 , 0x00 , 0xFF , 0x7F , 0xE0 , 0x01 , 0xFF , 0x7F , 0xE0
 , 0x03 , 0xFE , 0x7F , 0xE0 , 0x07 , 0xFC , 0x7F , 0xE0
 , 0x0F , 0xF8 , 0x7F , 0xE0 , 0x0F , 0xF0 , 0x7F , 0xE0
 , 0x1F , 0xE0 , 0x7F , 0xE0 , 0x3F , 0xE0 , 0x7F , 0xE0
 , 0x7F , 0xC0 , 0x7F , 0xE0 , 0xFF , 0x80 , 0x7F , 0xE0
 , 0xFF , 0xFF , 0xFF , 0xFE , 0xFF , 0xFF , 0xFF , 0xFE
 , 0xFF , 0xFF , 0xFF , 0xFE , 0xFF , 0xFF , 0xFF , 0xFE
 , 0xFF , 0xFF , 0xFF , 0xFE , 0xFF , 0xFF , 0xFF , 0xFE
 , 0xFF , 0xFF , 0xFF , 0xFE , 0xFF , 0xFF , 0xFF , 0xFE
 , 0x00 , 0x00 , 0x7F , 0xE0 , 0x00 , 0x00 , 0x7F , 0xE0
 , 0x00 , 0x00 , 0x7F , 0xE0 , 0x00 , 0x00 , 0x7F , 0xE0
 , 0x00 , 0x00 , 0x7F , 0xE0 , 0x00 , 0x00 , 0x7F , 0xE0
};

//*************************************************************************************************
flash unsigned char five[144] =
{
  0x00 , 0x00 , 0x00 , 0x00 , 0x03 , 0xFF , 0xFF , 0xF0
 , 0x03 , 0xFF , 0xFF , 0xF0 , 0x03 , 0xFF , 0xFF , 0xF0
 , 0x07 , 0xFF , 0xFF , 0xF0 , 0x07 , 0xFF , 0xFF , 0xF0
 , 0x07 , 0xFF , 0xFF , 0xF0 , 0x07 , 0xFF , 0xFF , 0xF0
 , 0x07 , 0xFF , 0xFF , 0xF0 , 0x07 , 0xF8 , 0x00 , 0x00
 , 0x07 , 0xF8 , 0x00 , 0x00 , 0x0F , 0xF8 , 0x00 , 0x00
 , 0x0F , 0xF8 , 0x00 , 0x00 , 0x0F , 0xF9 , 0xFC , 0x00
 , 0x0F , 0xFF , 0xFF , 0x80 , 0x0F , 0xFF , 0xFF , 0xC0
 , 0x0F , 0xFF , 0xFF , 0xE0 , 0x1F , 0xFF , 0xFF , 0xF0
 , 0x1F , 0xFF , 0xFF , 0xF8 , 0x1F , 0xFC , 0x3F , 0xF8
 , 0x07 , 0xF0 , 0x0F , 0xFC , 0x00 , 0x00 , 0x0F , 0xFC
 , 0x00 , 0x00 , 0x07 , 0xFC , 0x00 , 0x00 , 0x07 , 0xFC
 , 0x00 , 0x00 , 0x07 , 0xFC , 0x00 , 0x00 , 0x07 , 0xFC
 , 0x03 , 0xE0 , 0x07 , 0xFC , 0x3F , 0xF0 , 0x0F , 0xF8
 , 0x1F , 0xF0 , 0x0F , 0xF8 , 0x1F , 0xFC , 0x3F , 0xF8
 , 0x0F , 0xFF , 0xFF , 0xF0 , 0x0F , 0xFF , 0xFF , 0xE0
 , 0x07 , 0xFF , 0xFF , 0xC0 , 0x03 , 0xFF , 0xFF , 0x80
 , 0x00 , 0xFF , 0xFF , 0x00 , 0x00 , 0x1F , 0xF8 , 0x00
};

//*************************************************************************************************
flash unsigned char six[144] =
{
  0x00 , 0x0F , 0xF8 , 0x00 , 0x00 , 0x3F , 0xFF , 0x00
 , 0x00 , 0xFF , 0xFF , 0x80 , 0x01 , 0xFF , 0xFF , 0xC0
```

```
 , 0x03 , 0xFF , 0xFF , 0xE0 , 0x07 , 0xFF , 0xFF , 0xF0
 , 0x0F , 0xFC , 0x3F , 0xF0 , 0x0F , 0xF8 , 0x1F , 0xF8
 , 0x1F , 0xF0 , 0x0F , 0xF8 , 0x1F , 0xF0 , 0x0E , 0x00
 , 0x1F , 0xF0 , 0x00 , 0x00 , 0x1F , 0xE0 , 0x00 , 0x00
 , 0x3F , 0xE1 , 0xFC , 0x00 , 0x3F , 0xE7 , 0xFF , 0x00
 , 0x3F , 0xEF , 0xFF , 0xC0 , 0x3F , 0xFF , 0xFF , 0xE0
 , 0x3F , 0xFF , 0xFF , 0xF0 , 0x3F , 0xFF , 0xFF , 0xF0
 , 0x3F , 0xFC , 0x3F , 0xF8 , 0x3F , 0xF8 , 0x0F , 0xF8
 , 0x3F , 0xF0 , 0x0F , 0xFC , 0x3F , 0xF0 , 0x07 , 0xFC
 , 0x3F , 0xE0 , 0x07 , 0xFC , 0x3F , 0xE0 , 0x07 , 0xFC
 , 0x1F , 0xE0 , 0x07 , 0xFC , 0x1F , 0xE0 , 0x07 , 0xFC
 , 0x1F , 0xF0 , 0x07 , 0xFC , 0x1F , 0xF0 , 0x0F , 0xFC
 , 0x0F , 0xF8 , 0x0F , 0xF8 , 0x0F , 0xFC , 0x3F , 0xF8
 , 0x07 , 0xFF , 0xFF , 0xF0 , 0x03 , 0xFF , 0xFF , 0xF0
 , 0x01 , 0xFF , 0xFF , 0xE0 , 0x00 , 0xFF , 0xFF , 0xC0
 , 0x00 , 0x7F , 0xFF , 0x00 , 0x00 , 0x0F , 0xF8 , 0x00
};

//*************************************************************************************************
flash unsigned char seven[144] =
{
   0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x3F , 0xFF , 0xFF , 0xFC , 0x3F , 0xFF , 0xFF , 0xFC
 , 0x3F , 0xFF , 0xFF , 0xFC , 0x3F , 0xFF , 0xFF , 0xFC
 , 0x3F , 0xFF , 0xFF , 0xFC , 0x3F , 0xFF , 0xFF , 0xFC
 , 0x3F , 0xFF , 0xFF , 0xFC , 0x3F , 0xFF , 0xFF , 0xF8
 , 0x00 , 0x00 , 0x3F , 0xF0 , 0x00 , 0x00 , 0x3F , 0xE0
 , 0x00 , 0x00 , 0x7F , 0xC0 , 0x00 , 0x00 , 0xFF , 0x80
 , 0x00 , 0x01 , 0xFF , 0x00 , 0x00 , 0x01 , 0xFF , 0x00
 , 0x00 , 0x03 , 0xFE , 0x00 , 0x00 , 0x03 , 0xFC , 0x00
 , 0x00 , 0x07 , 0xFC , 0x00 , 0x00 , 0x07 , 0xF8 , 0x00
 , 0x00 , 0x0F , 0xF8 , 0x00 , 0x00 , 0x0F , 0xF0 , 0x00
 , 0x00 , 0x1F , 0xF0 , 0x00 , 0x00 , 0x1F , 0xF0 , 0x00
 , 0x00 , 0x3F , 0xE0 , 0x00 , 0x00 , 0x3F , 0xE0 , 0x00
 , 0x00 , 0x3F , 0xE0 , 0x00 , 0x00 , 0x3F , 0xC0 , 0x00
 , 0x00 , 0x7F , 0xC0 , 0x00 , 0x00 , 0x7F , 0xC0 , 0x00
 , 0x00 , 0x7F , 0xC0 , 0x00 , 0x00 , 0x7F , 0xC0 , 0x00
 , 0x00 , 0x7F , 0x80 , 0x00 , 0x00 , 0xFF , 0x80 , 0x00
 , 0x00 , 0xFF , 0x80 , 0x00 , 0x00 , 0xFF , 0x80 , 0x00
};

//*************************************************************************************************
flash unsigned char eight[144] =
{
   0x00 , 0x1F , 0xF8 , 0x00 , 0x00 , 0xFF , 0xFF , 0x00
 , 0x01 , 0xFF , 0xFF , 0x80 , 0x07 , 0xFF , 0xFF , 0xE0
 , 0x07 , 0xFF , 0xFF , 0xE0 , 0x0F , 0xFF , 0xFF , 0xF0
 , 0x0F , 0xFC , 0x3F , 0xF0 , 0x1F , 0xF8 , 0x1F , 0xF8
 , 0x1F , 0xF0 , 0x0F , 0xF8 , 0x1F , 0xF0 , 0x0F , 0xF8
 , 0x1F , 0xF0 , 0x0F , 0xF8 , 0x1F , 0xF0 , 0x0F , 0xF8
 , 0x1F , 0xF8 , 0x1F , 0xF0 , 0x0F , 0xFC , 0x3F , 0xF0
 , 0x07 , 0xFF , 0xFF , 0xE0 , 0x07 , 0xFF , 0xFF , 0xE0
 , 0x01 , 0xFF , 0xFF , 0x80 , 0x03 , 0xFF , 0xFF , 0xC0
 , 0x07 , 0xFF , 0xFF , 0xE0 , 0x0F , 0xFF , 0xFF , 0xF0
 , 0x1F , 0xFC , 0x3F , 0xF8 , 0x1F , 0xF0 , 0x0F , 0xF8
 , 0x3F , 0xF0 , 0x0F , 0xFC , 0x3F , 0xE0 , 0x07 , 0xFC
 , 0x3F , 0xE0 , 0x07 , 0xFC , 0x3F , 0xE0 , 0x07 , 0xFC
```

```
 , 0x3F , 0xE0 , 0x07 , 0xFC , 0x3F , 0xF0 , 0x0F , 0xFC
 , 0x3F , 0xF0 , 0x0F , 0xFC , 0x1F , 0xFC , 0x3F , 0xF8
 , 0x1F , 0xFF , 0xFF , 0xF8 , 0x0F , 0xFF , 0xFF , 0xF0
 , 0x07 , 0xFF , 0xFF , 0xE0 , 0x03 , 0xFF , 0xFF , 0xC0
 , 0x01 , 0xFF , 0xFF , 0x00 , 0x00 , 0x3F , 0xF8 , 0x00
};

//*****************************************************************************************************
flash unsigned char nine[144] =
{
   0x00 , 0x1F , 0xF0 , 0x00 , 0x00 , 0xFF , 0xFE , 0x00
 , 0x03 , 0xFF , 0xFF , 0x00 , 0x07 , 0xFF , 0xFF , 0x80
 , 0x0F , 0xFF , 0xFF , 0xC0 , 0x0F , 0xFF , 0xFF , 0xE0
 , 0x1F , 0xFC , 0x3F , 0xF0 , 0x1F , 0xF0 , 0x1F , 0xF0
 , 0x3F , 0xF0 , 0x0F , 0xF8 , 0x3F , 0xE0 , 0x0F , 0xF8
 , 0x3F , 0xE0 , 0x07 , 0xF8 , 0x3F , 0xE0 , 0x07 , 0xF8
 , 0x3F , 0xE0 , 0x07 , 0xFC , 0x3F , 0xE0 , 0x07 , 0xFC
 , 0x3F , 0xE0 , 0x0F , 0xFC , 0x3F , 0xF0 , 0x0F , 0xFC
 , 0x1F , 0xF0 , 0x1F , 0xFC , 0x1F , 0xFC , 0x3F , 0xFC
 , 0x0F , 0xFF , 0xFF , 0xFC , 0x0F , 0xFF , 0xFF , 0xFC
 , 0x07 , 0xFF , 0xFF , 0xFC , 0x03 , 0xFF , 0xF7 , 0xFC
 , 0x00 , 0xFF , 0xE7 , 0xFC , 0x00 , 0x3F , 0x87 , 0xFC
 , 0x00 , 0x00 , 0x0F , 0xF8 , 0x00 , 0x00 , 0x0F , 0xF8
 , 0x00 , 0x70 , 0x0F , 0xF8 , 0x1F , 0xF0 , 0x0F , 0xF8
 , 0x1F , 0xF8 , 0x1F , 0xF0 , 0x0F , 0xFC , 0x3F , 0xF0
 , 0x0F , 0xFF , 0xFF , 0xE0 , 0x07 , 0xFF , 0xFF , 0xC0
 , 0x03 , 0xFF , 0xFF , 0x80 , 0x01 , 0xFF , 0xFF , 0x00
 , 0x00 , 0xFF , 0xFC , 0x00 , 0x00 , 0x1F , 0xF0 , 0x00
};

//*****************************************************************************************************
flash unsigned char decimal_point[72] =
{
   0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x1F , 0xF8 , 0x1F , 0xF8 , 0x1F , 0xF8 , 0x1F , 0xF8
 , 0x1F , 0xF8 , 0x1F , 0xF8 , 0x1F , 0xF8 , 0x1F , 0xF8
 , 0x1F , 0xF8 , 0x1F , 0xF8 , 0x00 , 0x00 , 0x00 , 0x00
};

//*****************************************************************************************************
flash unsigned char kV[72] =
{
   0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x1E , 0x00
 , 0x1E , 0x00 , 0x1E , 0x00 , 0x1E , 0x78 , 0x1E , 0xF0
 , 0x1F , 0xE0 , 0x1F , 0xC0 , 0x1F , 0xC0 , 0x1F , 0xE0
 , 0x1F , 0xF0 , 0x1E , 0xF0 , 0x1E , 0x78 , 0x1E , 0x7C
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x78 , 0x3C , 0x3C , 0x78 , 0x3C , 0x78
 , 0x3C , 0x78 , 0x1C , 0x70 , 0x1E , 0xF0 , 0x1E , 0xF0
 , 0x0E , 0xE0 , 0x0E , 0xE0 , 0x0F , 0xE0 , 0x07 , 0xC0
 , 0x07 , 0xC0 , 0x07 , 0xC0 , 0x00 , 0x00 , 0x00 , 0x00
};
```

```
//*********************************************************************************************
flash unsigned char A[72] =
{
  0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x03 , 0xE0 , 0x07 , 0xF0 , 0x07 , 0xF0
 , 0x07 , 0xF0 , 0x0F , 0x78 , 0x0F , 0x78 , 0x1F , 0x7C
 , 0x1E , 0x3C , 0x1F , 0xFC , 0x3F , 0xFE , 0x3F , 0xFE
 , 0x3C , 0x1E , 0x78 , 0x0F , 0x00 , 0x00 , 0x00 , 0x00
};

//*********************************************************************************************
flash unsigned char left_arrow[576] =
{
  0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x80 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x0F , 0x80 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0xFF , 0x80 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x0F
 , 0xFF , 0x80 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0xFF
 , 0xFF , 0x80 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x0F , 0xFF
 , 0xFF , 0x80 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0xFF , 0xFF
 , 0xFF , 0x80 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x0F , 0xFF , 0xFF
 , 0xFF , 0x80 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0xFF , 0xFF , 0xFF
 , 0xFF , 0x80 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x0F , 0xFF , 0xFF , 0xFF
 , 0xFF , 0x80 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0x80 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x0F , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xF8
 , 0x00 , 0x00 , 0x00 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xF8
 , 0x00 , 0x00 , 0x0F , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xF8
 , 0x00 , 0x00 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xF8
 , 0x00 , 0x0F , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xF8
 , 0x00 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xF8
 , 0x0F , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xF8
 , 0x07 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xF8
 , 0x00 , 0x7F , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xF8
```

```
 , 0x00 , 0x07 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xF8
 , 0x00 , 0x00 , 0x7F , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xF8
 , 0x00 , 0x00 , 0x07 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xF8
 , 0x00 , 0x00 , 0x00 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xF8
 , 0x00 , 0x00 , 0x00 , 0x0F , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xF8
 , 0x00 , 0x00 , 0x00 , 0x00 , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0x80 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x0F , 0xFF , 0xFF , 0xFF
 , 0xFF , 0x80 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0xFF , 0xFF , 0xFF
 , 0xFF , 0x80 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x0F , 0xFF , 0xFF
 , 0xFF , 0x80 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0xFF , 0xFF
 , 0xFF , 0x80 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x0F , 0xFF
 , 0xFF , 0x80 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0xFF
 , 0xFF , 0x80 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x0F
 , 0xFF , 0x80 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0xFF , 0x80 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x0F , 0x80 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x80 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
};

//***********************************************************************************************
flash unsigned char right_arrow[576] =
{
   0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x01 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x01 , 0xF0
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x01 , 0xFF
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x01 , 0xFF
 , 0xF0 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x01 , 0xFF
 , 0xFF , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x01 , 0xFF
 , 0xFF , 0xF0 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x01 , 0xFF
 , 0xFF , 0xFF , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x01 , 0xFF
 , 0xFF , 0xFF , 0xF0 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x01 , 0xFF
 , 0xFF , 0xFF , 0xFF , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x01 , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xF0 , 0x00 , 0x00 , 0x00 , 0x00
```

```
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x01 , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0x00 , 0x00 , 0x00 , 0x00
 , 0x1F , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0xF0 , 0x00 , 0x00 , 0x00
 , 0x1F , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0x00 , 0x00 , 0x00
 , 0x1F , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xE0 , 0x00 , 0x00
 , 0x1F , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFE , 0x00 , 0x00
 , 0x1F , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xE0 , 0x00
 , 0x1F , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFE , 0x00
 , 0x1F , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xE0
 , 0x1F , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xF0
 , 0x1F , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0x00
 , 0x1F , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xF0 , 0x00
 , 0x1F , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0x00 , 0x00
 , 0x1F , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xF0 , 0x00 , 0x00
 , 0x1F , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0x00 , 0x00 , 0x00
 , 0x1F , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0xF0 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x01 , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xFF , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x01 , 0xFF
 , 0xFF , 0xFF , 0xFF , 0xF0 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x01 , 0xFF
 , 0xFF , 0xFF , 0xFF , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x01 , 0xFF
 , 0xFF , 0xFF , 0xF0 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x01 , 0xFF
 , 0xFF , 0xFF , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x01 , 0xFF
 , 0xFF , 0xF0 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x01 , 0xFF
 , 0xFF , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x01 , 0xFF
 , 0xF0 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x01 , 0xFF
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x01 , 0xF0
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x01 , 0x00
 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00 , 0x00
};
```

Fifth Prototype Software Listing

This program is the same as the previous software listing, except for the functions listed below.

```
//*************************************************************************************************
// Fault Meter with OLED display
// version 0.04:
//*************************************************************************************************

#include <mega324.h>
#include <OLED.h>
#include <delay.h>
#include <stdlib.h>
#include <logo.h>

//*************************************************************************************************
// Main routine
//*************************************************************************************************
void main(void)
{
  unsigned char edge=0;
  unsigned int Vpos_reading;
  unsigned int Vneg_reading;
  unsigned int Ipos_reading;
  unsigned int Ineg_reading;
  unsigned long timing;
  unsigned char idle=0;

  initialise();                                 // initialise microcontroller

  OLED_power=1;                                 //}
  delay_ms(100);                                //} initialise OLED
  OLED_initialise();                            //}

  I_power=1;                                    //} enable peak detect circuits
  V_power=1;                                    //}

  delay_ms(120);                                //}
  display_logo();                               //} display splash screen
  idle=1;                                       //}

  while(1) {                                    // main program loop
    delay_ms(100);                              // brief delay between readings
    timing=500000;                              //}
    while(!edge) {                              //} wait in this loop until a positive or
                                                   negative edge is detected
      if(!Button1_pin || trigger_pos) {         //} if no edge within approx 5
                                                   seconds, then display logo
        edge=positive;                          //}
      }                                         //}
      else if(!Button2_pin || !trigger_neg) { //}
        edge=negative;                          //}
```

```
  }                                   //}
  else {                              //}
    if(!idle) {                       //}
      if(timing) {                    //}
        timing--;                     //}
      }                               //}
      else {                          //}
        idle=1;                       //}
        display_logo();               //}
        timing=500000;                //}
      }                               //}
    }                                 //}
  }                                   //}
}                                     //}
idle=0;                               //}

delay_ms(5);                          // allow peak detect circuits to stabilise
Vpos_reading=read_adc(V_pos);         //}
Vneg_reading=read_adc(V_neg);         //} take readings of peak detect circuits
Ipos_reading=read_adc(I_pos);         //}
Ineg_reading=read_adc(I_neg);         //}

OLED_write_cmd(0x8e);                 //}
OLED_write_data(0);                   //} OLED clear window command
OLED_write_data(0);                   //}
OLED_write_data(130);                 //}
OLED_write_data(130);                 //}

delay_ms(100);                        // delay for OLED

OLED_write_cmd(0x92);                 //} OLED fill enable command
OLED_write_data(0x01);                //}

delay_ms(10);                         // delay for OLED

if(Ipos_reading>Ineg_reading) {       //}
  if(edge==positive) {                //} determine direction of arrow to display:
    display_leftarrow(0,0);           //}   - positive current, positive voltage = left
  }                                   //}   - positive current, negative voltage = right
  else {                              //}   - negative current, positive voltage = right
    display_rightarrow(0,0);          //}   - negative current, negative voltage = left
  }                                   //}
}                                     //}
else {                                //}
  if(edge==positive) {                //}
    display_rightarrow(0,0);          //}
  }                                   //}
  else {                              //}
    display_leftarrow(0,0);           //}
  }                                   //}
}                                     //}

if(edge==positive) {                          //}
  display_value( (Vpos_reading/6.5)-1 ,42);   //} display voltage reading
}                                             //}
else {                                        //}
  display_value( (Vneg_reading/5) ,42);       //}
```

```
    }                                       //}
    display_kV(112,42);                     //}

    if(Ipos_reading>Ineg_reading) {         //}
      display_value(Ipos_reading,84);       //} display current reading
    }                                       //}
    else {                                  //}
      display_value(Ineg_reading,84);       //}
    }                                       //}
    display_A(112,84);                      //}

    edge=0;                                 // clear edge detection
  }
}


//***********************************************************************************************
// Initialisation routine
//***********************************************************************************************
void initialise(void)
{
  PORTA=0x00;                               //}
  DDRA=0x38;                                //}
  PORTB=0xE3;                               //} port definitions
  DDRB=0x1C;                                //}
  PORTC=0x00;                               //}
  DDRC=0xFF;                                //}
  PORTD=0x60;                               //}
  DDRD=0x93;                                //}

  TCCR0A=0xB1;                              //}
  TCCR0B=0x01;                              //} timer0:
  TCNT0=0x00;                               //}  *8MHz
  OCR0A=0x7E;                               //}  *Phase correct PWM
  OCR0B=0x81;                               //}

  TCCR1A=0x00;                              //}
  TCCR1B=0x00;                              //} timer1:
  TCNT1H=0x00;                              //}   *stopped
  TCNT1L=0x00;                              //}
  ICR1H=0x00;                               //}
  ICR1L=0x00;                               //}
  OCR1AH=0x00;                              //}
  OCR1AL=0x00;                              //}
  OCR1BH=0x00;                              //}
  OCR1BL=0x00;                              //}

  ASSR=0x00;                                //}
  TCCR2A=0x00;                              //} timer2:
  TCCR2B=0x00;                              //}   *stopped
  TCNT2=0x00;                               //}
  OCR2A=0x00;                               //}
  OCR2B=0x00;                               //}

  EICRA=0x00;                               //}
  EIMSK=0x00;                               //} external interrupts: disabled
```

```
  PCICR=0x00;                           //}

  TIMSK0=0x00;                          //}
  TIMSK1=0x00;                          //} timer interrupts: disabled
  TIMSK2=0x00;                          //}

  ACSR=0x80;                            //} analogue comparator: disabled
  ADCSRB=0x00;                          //}

  DIDR0=0xC7;                           //}
  ADMUX=ADC_VREF_TYPE & 0xff;           //} ADC initialisation:
  ADCSRA=0x85;                          //}   *250kHz, AVCC reference
}
```

Sixth Prototype Software Listing

This program is the same as the previous prototype software listing, except for the functions listed below.

```
//*************************************************************************************************
// Fault Meter with OLED display
// Version 1.00
//*************************************************************************************************

#include <mega644.h>
#include <OLED.h>
#include <delay.h>
#include <stdlib.h>
#include <logo.h>
#include <sleep.h>


//*************************************************************************************************
// Main routine
//*************************************************************************************************
void main(void)
{
  unsigned char edge=0;
  unsigned int Vpos_reading;
  unsigned int Vneg_reading;
  unsigned int Ipos_reading;
  unsigned int Ineg_reading;
  unsigned long timing;
  unsigned char idle=0;

  initialise();                                 // initialise microcontroller

  OLED_initialise();                            //

  I_power=on;                                   //} enable peak detect circuits
  V_power=on;                                   //}

  delay_ms(120);                                //}
  display_logo();                               //} display splash screen
  idle=1;                                       //}

  while(1) {                                    // main program loop
    delay_ms(100);                              // brief delay between readings
    timing=500000;                              //}
    while(!edge) {                              //} wait in this loop until a positive or
      if(!Button_pin || trigger_pos) {          //} negative edge is detected
        edge=positive;                          //}   *if no edge within approx
      }                                         //}     5 seconds, then display logo
      else if(!trigger_neg) {                   //}
        edge=negative;                          //}
      }                                         //}
```

```
    else {                                  //}
     if(!idle) {                            //}
      if(timing) {                          //}
       timing--;                            //}
      }                                     //}
      else {                                //}
       idle=1;                              //}
       display_logo();                      //}
       timing=500000;                       //}
      }                                     //}
     }                                      //}
    }                                       //}
   }                                        //}
   idle=0;                                  //}

   delay_ms(5);                             // allow peak detect circuits to
stabilise
   Vpos_reading=read_adc(V_pos);            //}
   Vneg_reading=read_adc(V_neg);            //} take readings of peak
   Ipos_reading=read_adc(I_pos);            //}  detect circuits
   Ineg_reading=read_adc(I_neg);            //}

   OLED_write_cmd(0x8e);                    //}
   OLED_write_data(0);                      //} OLED clear window command
   OLED_write_data(0);                      //}
   OLED_write_data(130);                    //}
   OLED_write_data(130);                    //}

   delay_ms(100);                           // delay for OLED

   OLED_write_cmd(0x92);                    //} OLED fill enable command
   OLED_write_data(0x01);                   //}

   delay_ms(10);                            // delay for OLED

   if(Ipos_reading>Ineg_reading) {          //}
    if(edge==positive) {                    //} determine direction of arrow to
display:
     display_leftarrow(0,0);                //}  - positive current, positive voltage
= left
    }                                       //}      - positive current, negative
voltage = right
    else {                                  //}      - negative current, positive
voltage = right
     display_rightarrow(0,0);               //}      - negative current, negative
voltage = left
    }                                       //}
   }                                        //}
   else {                                   //}
    if(edge==positive) {                    //}
     display_rightarrow(0,0);               //}
    }                                       //}
    else {                                  //}
     display_leftarrow(0,0);                //}
    }                                       //}
   }                                        //}
```

```
    if(edge==positive) {                          //}
      display_value( (Vpos_reading/6.5)-1 ,42);   //} display voltage reading
    }                                             //}
    else {                                        //}
      display_value( (Vneg_reading/5) ,42);       //}
    }                                             //}
    display_kV(112,42);                           //}

    if(Ipos_reading>Ineg_reading) {               //}
      display_value(Ipos_reading,84);             //} display current reading
    }                                             //}
    else {                                  //}
      display_value(Ineg_reading,84);             //}
    }                                             //}
    display_A(112,84);                            //}

    edge=0;                                       // clear edge detection


    I_power=off;                                  //disable current peak detector
    V_power=off;                                  //disable voltage peak detector
    OLED_high_voltage=off;                        //
    OLED_write_cmd(0xAE);                         //OLED sleep mode
    powerdown();
  }
}


//***********************************************************************************************
// Initialisation routine
//***********************************************************************************************
void initialise(void)
{
  PORTA=0x00;                                     //}
  DDRA=0x38;                                      //}
  PORTB=0xFD;                                     //} port definitions
  DDRB=0x02;                                      //}
  PORTC=0x00;                                     //}
  DDRC=0xFF;                                      //}
  PORTD=0x40;                                     //}
  DDRD=0xB3;                                      //}

  TCCR0A=0x00;                                    //}
  TCCR0B=0x00;                                    //} timer0:
  TCNT0=0x00;                                     //}  *stopped
  OCR0A=0x00;                                     //}
  OCR0B=0x00;                                     //}

  TCCR1A=0x00;                                    //}
  TCCR1B=0x00;                                    //} timer1:
  TCNT1H=0x00;                                    //}   *stopped
  TCNT1L=0x00;                                    //}
  ICR1H=0x00;                                     //}
  ICR1L=0x00;                                     //}
  OCR1AH=0x00;                                    //}
  OCR1AL=0x00;                                    //}
```

```
  OCR1BH=0x00;                          //}
  OCR1BL=0x00;                          //}

  ASSR=0x00;                            //}
  TCCR2A=0x00;                          //} timer2:
  TCCR2B=0x00;                          //}   *running
  TCNT2=0x00;                           //}
  OCR2A=0x00;                           //}
  OCR2B=0x00;                           //}

  EICRA=0x00;                           //}
  EIMSK=0x00;                           //} external interrupts: enabled
  PCICR=0x00;                           //}

  TIMSK0=0x00;                          //}
  TIMSK1=0x00;                          //} timer interrupts: T2 overflow
  TIMSK2=0x01;                          //}

  ACSR=0x80;                            //} analogue comparator: disabled
  ADCSRB=0x00;                          //}

  DIDR0=0xC7;                           //}
  ADMUX=ADC_VREF_TYPE & 0xff;           //} ADC initialisation:
  ADCSRA=0x85;                          //}   *250kHz, AVCC reference

  sleep_enable();
}


//*************************************************************************************************
// OLED Data Write routine
//*************************************************************************************************
void OLED_write_data(unsigned long Data)
{
  OLED_read_disable();                  //}
  OLED_data();                          //} IOSET=DC|RD;

  OLED_write_enable();                  //}
  OLED_active();                        //} IOCLR=WR|CS;

  OLED_data_bus=0;                      // IOCLR=0x3ffff;

  OLED_data_bus=Data;                   // IOSET=Data;

  OLED_write_disable();                 //}
  OLED_inactive();                      //} IOSET=WR|CS;
}


//*************************************************************************************************
// OLED Command Write routine
//*************************************************************************************************
void OLED_write_cmd(unsigned char Command)
{
  OLED_read_disable();                  // IOSET=RD;
```

```
  OLED_command();                 //}
  OLED_write_enable();            //} IOCLR=WR|CS|DC;
  OLED_active();                  //}

  OLED_data_bus=0;                // IOCLR=0xff;

  OLED_data_bus=Command;          //} IOSET=Command;

  OLED_write_disable();           //} IOSET=WR|CS;
  OLED_inactive();                //}
}


//*************************************************************************************************
// OLED Initialisation routine
//*************************************************************************************************
void OLED_initialise(void)
{
  OLED_logic_power=1;             //}
  delay_ms(100);                  //}

  OLED_reset();                   //}
  delay_ms(100);                  //} reset OLED
  OLED_non_reset();               //}

  OLED_write_cmd(0xA0);           //} 262K 8bit R->G->B
  OLED_write_data(0xB4);          //}


  OLED_write_cmd(0xA1);           //} Display_Start_Line
  OLED_write_data(0x00);          //}

  OLED_write_cmd(0xA2);           //} Display_Offset
  OLED_write_data(0x80);          //}

  OLED_write_cmd(0xCA);           //} Mux_Ratio
  OLED_write_data(0x7F);          //}

  OLED_write_cmd(0xA6);           // Display_Mode

  OLED_write_cmd(0xAD);           //} Master_Conf
  OLED_write_data(0x8E);          //}

  OLED_write_cmd(0xC7);           //} Master_Contrast
  OLED_write_data(0x0f);          //}

  OLED_write_cmd(0xC1);           //}
  OLED_write_data(0xff);          //} Contrast_Current
  OLED_write_data(0xff);          //}
  OLED_write_data(0xff);          //}

  OLED_write_cmd(0xBE);           //} VCOMH
  OLED_write_data(0x3f);          //}

  OLED_write_cmd(0xB0);           //} Power_Saving
```

```
  OLED_write_data(0x05);                //}

  OLED_write_cmd(0xBB);                 //}
  OLED_write_data(0x1c);                //} Precharge_Color
  OLED_write_data(0x1c);                //}
  OLED_write_data(0x1c);                //}

  OLED_write_cmd(0xB1);                 //} Reset_Pre_charge
  OLED_write_data(0x11);                //}

  OLED_write_cmd(0xB3);                 //} Osc_Freq
  OLED_write_data(0xf0);                //}

  OLED_write_cmd(0xAF);                 // Display_ON

  OLED_high_voltage=on;
}
```