THE DEVELOPMENT OF A HIGH VOLTAGE ELECTRO-PHOTOGRAPHIC DEVICE WHICH CAN BE USED IN THE RESEARCH OF KIRLIAN PHOTOGRAPHIC PHENOMENA

By

SYDWELL JOHN SAMUEL WILLIAMS

A DISSERTATION PRESENTED TO THE HIGHER DEGREES COMMITTEE
OF PENINSULA TECHNIKON IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF TECHNOLOGY ELECTRICAL ENGINEERING

PENINSULA TECHNIKON

2003

This thesis is dedicated to my family. Especially to my devoted wife, Gail, without her support and understanding these outputs would never have been realized.

ACKNOWLEDGMENTS

I would like to thank all my colleagues specifically, Wendal Koopman, Lindsey Wicomb, Marcel Berteler and Luis Lopes. Special thanks, to my supervisor Dr. Tariq Kahn and my Head of Department, Mr. Bennett Alexander for their encouragement and patience.

TABLE OF CONTENTS

page

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

KEY OF ABBREVIATIONS

CCD ..... Charged Coupled Device

GUI ..... Graphical User Interface

KET ..... Kirlian Experiment Template

LED ..... Light Emitting Diode

LOPT ..... Line Output Transformer

LSD ..... Light Sensitive Diode

PIN Diode ..... Positive-Intrinsic-Negative Diode

RH ..... Percentage Relative Humidity

RS232C ..... Standard Serial Interface Protocol

SFR ..... Special Function Register

SLR ..... Single Lens Reflective

TLE ..... Transparent Liquid Electrode

UML ..... Universal Modelling Language

USB ..... Universal Serial Bus

Abstract of Dissertation Presented to the Higher Degrees Committee
of Peninsula Technikon in Partial Fulfilment of the
Requirements for the Degree of Master of Technology

THE DEVELOPMENT OF A HIGH VOLTAGE ELECTRO-PHOTOGRAPHIC DEVICE WHICH CAN BE USED IN THE RESEARCH OF KIRLIAN PHOTOGRAPHIC PHENOMENA

By

SYDWELL JOHN SAMUEL WILLIAMS

March 2004

Supervisor: MTE Kahn
Faculty: Engineering
Department: Electrical Engineering

The scientific investigation of Kirlian photography has always been a contentious issue. Unreliable instrumentation and incomplete documentation characterized initial research efforts of most investigators. An instrument was designed and constructed which addressed these concerns. Using modern engineering techniques a portable instrument which can take near instant quality photographs was produced. Important design features include :

- Virtually Instant Pictures - The setup time can be very fast, a specimen can be ready in seconds
- Complete Digital Camera Control via software
- High voltage and frequency can also be controlled via the software
- Galvanic isolated high voltages which protects the user
- Automated monitoring and recording environment

- Offline/Internet Viewing – It is possible to review past experiments

The product presented in this dissertation makes use of standard microcontroller hardware and software design theory and practises as well as conventional personal computer PC programming techniques. Standard electrical theory was also applied throughout to deliver safe and effective device.

CHAPTER 1

INTRODUCTION

1.1 Brief History and Description of Kirlian Photography

Kirlian photography is a novel way of taking photographic images of animate and inanimate objects. This unique method of taking photographs produces a pattern of streamers rich in detail surrounding the object photographed. An example of this type of photograph can be seen in figure 1.1 [Johnson, 1975:121].

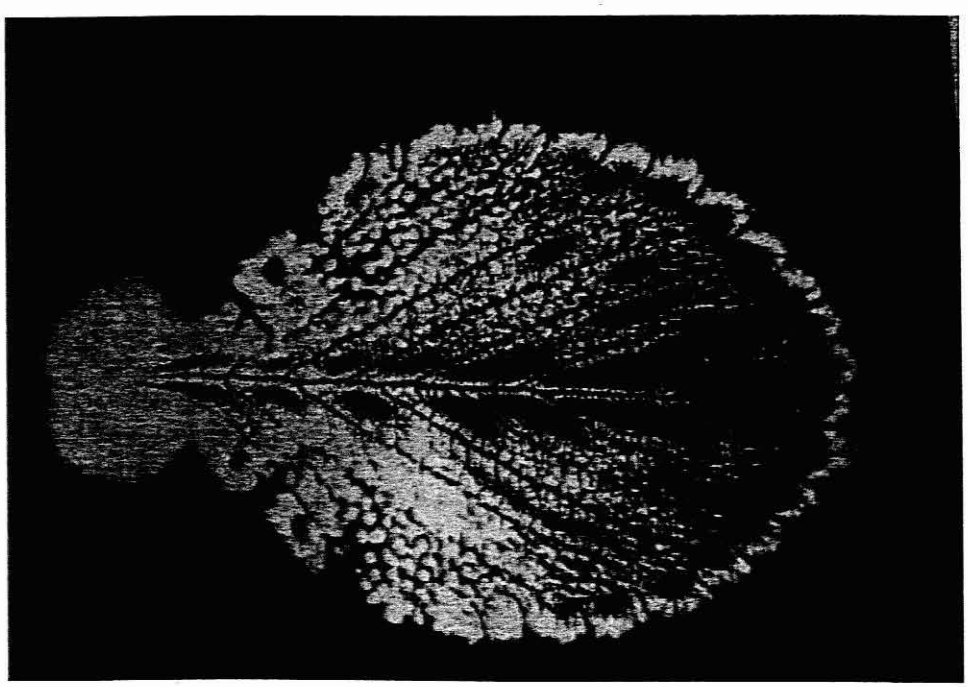


Figure 1-1: Typical Published Kirlian Photograph

This photograph is an example of a typical Kirlian photograph shown in literature. The object in this photograph is one of a strawberry leaf taken by Kendall

Johnson in 1996. The high voltage discharge created the pattern, which was captured by means of an unexposed photographic film.

The discovery of Kirlian photography was made by a husband and wife team, S. D. and V Kirlian, in 1939 in the Ukraine. It was however only brought to the attention of western investigators in the early 1960's [Kirlian, 1963]. Since then, the investigation into this phenomenon has been characterized by rash claims and counter claims as to the significance of these pictures. However, it is easy to explain in mundane terms the mechanics involved in taking Kirlian photographs.

Although steeped in controversy due to its paranormal connections, nothing out of the ordinary is required to explain what happens when a Kirlian photograph is taken. In order to take a Kirlian photograph the following method is generally used, which results in a picture of an electrical discharge between two objects. One of the objects is an electrode consists of flat piece of metal to which a high voltage source is applied. The other object is the subject of the photograph. The subject is usually electrically grounded. The two objects or electrodes are normally separated by an insulating material. The insulating material in a standard Kirlian photography is the actual photographic film used to capture the image. As can be deuced from the above description, electrical discharge occurs between an earthed object and a pulsed high voltage source [Tiller, 1973:78]. Depending on the potential difference and other factors a corona discharge, sparks or arc discharges can then be photographed. In the original discovery the photographic film served as the dielectric material.

There is however still a major controversy in scientific and paranormal quarters as to what the details represent. Some investigators insist that it is the so-called mythical aura, which has been photographed. While others maintain that it is just a discharge pattern of the electrical breakdown. Although this issue is central as

what kinds of practical benefits Kirlian photography will deliver, the answer to this question is not central to this investigation.

Notwithstanding the fact that there are disagreements as to what is being photographed, there have been successful demonstrations of the medical diagnostic ability of Kirlian photography. There are other diverse fields where Kirlian photography stands to deliver important contributions such as Psychology [Moss, 1981:26], Psychiatry and Parapsychology [Hossian, 1995:821], Metallurgy [Johnson, 1975:170], Medicine [Konikiewicz, 1979] & [Chouhan, 1989:Abstract] and more recently "electromagnetic fields associated with biological systems" [Russo et al, 2001]. Although Kirlian photography has been around for almost four decades, these exciting channels of investigation have yet to be thoroughly researched.

There are several of the reasons offered for the slow progress of developing this technology. Firstly, reliable instrumentation was not always used as it takes careful engineering to produce stable high voltage source. Secondly, there are many environmental factors, which if not taken into account during the experimentation will produce invalid results. Thirdly, because of the above reasons it was and still is difficult, sometimes impossible to reproduce other investigators results. Lastly, there were so many unsubstantiated claims made, whether they are true or not, that this impeded the scientific investigation into the use of this technology.

1.2 Problem statement and motivation

In order for significant progress to be made in the utilisation of the Kirlian Photographic Technique, repeatable and reliable instrumentation is required. The literature is full of investigations, which at the outset attempted at produce Kirlian effects often with out regard to the standardisation of the photographic process as

alluded to by researchers [Dakin 1974] and others. This dissertation describes the design and prototyping of a stable instrument, which will automatically document and record of most relevant information while taking a Kirlian Picture.

The techniques for producing Kirlian Photographs are quiet well defined. There are areas of controversy in the literature however, and these relate to the reliability or repeatability of the instrumentation especially the high voltage source [Konikiewicz, 1979]. There are other issues such as the interpretation of the photographs but this will be the subject of further study. The focus of this project is to document the design of a stable reliable instrumentation, which can be used to take Kirlian pictures. By documenting the design, it is hoped that a number of outcomes will be achieved. Firstly researchers of Kirlian photography phenomena will be able to use the prototype device to investigate reported Kirlian effects with out the serious concern of the scientific validity of there results. Secondly, other investigators could use these results to build similar or improved devices, as detailed documentation of Kirlian Photographic instrumentation is not readily available. With these outcomes in mind, the design of this Kirlian instrumentation was tackled.

1.3 Approach Used in the Study

The main object of the dissertation is documentation of the production of an instrument, which can be used in the research of the Kirlian photographic phenomena. Ideally, a safe, scientific, self-contained instrument with the ability to take instant, high quality Kirlian photographs should be delivered.

This investigation made use of the following Engineering techniques: High Voltage Engineering, Digital Photographic Techniques, Analogue Data Collection,

Microprocessors and Object Orientated software design. Standard design issues were addressed like the ease of use, availability and cost of components. More than this however thorough use of modern technological advancements was incorporated. Bearing in mind that most Kirlian instrumentation was designed before 1980 there were ample opportunity to improve on the traditional Kirlian photographic designs.

1.4 Outline of this Dissertation

The selection and measurement techniques employed to monitor the environment conditions are described in Chapter 2. This is a relatively straightforward approach to address to some extent the concerns of critics that environment conditions are important when evaluating this phenomenon.

In Chapter 3 the actual modalities of taking Kirlian photographs are discussed. A non-traditional method based on the transparent electrode technique is described as well as the digital camera techniques that are finally employed in this device.

The key area of performance issues relating to the generation of high voltages is discussed in Chapter 4.

A microcontroller system was designed to co-ordinate data transfer between Kirlian device and the PC. This system is described in Chapter 5. Using a microcontroller facilitates the scalability of the instrumentation, as it will be easier to add new functions and features later as required.

Incorporating the processing and presentation power of a modern PC with the interfacing of a microcontroller resulted in a design that seamlessly adds most of functionality expected in modern instrumentation. Using the PC program, described in

chapter 6, gives the researcher a much more control over the process of taking an experiment and thus easier for him or her to document and study the results.

Integrating the various systems to produce a working device is described in Chapter 7. Details for the power supplies and other constructional are also documented here.

Chapter 8, details the performance of the device. A walk through of an actual experiment and its eventual Kirlian pictures and data is described. Implications for human subjects are also discussed.

Finally, the inferences that can be drawn from of this dissertation discussed and concluded in Chapter 9.

1.5 Delimitations

The study will limit its investigation to the reproduction of high voltage photography and will not in any way presuppose what the actual photographs represent.

There were certain practical limitations which had to be dealt with. These were limited to the following. Ideally, controlling instead of monitoring the environmental parameters would increase the effectiveness of the instrument. This would however add greatly to the cost and complexity of the instrumentation.

In the similar manner monitoring of additional environmental factors could impact on the repeatability of pictures or reveal other modulation factors. These other factors include but are not limited to:

1. The ionisation potential of the atmosphere.
2. The composition and quantity of gasses in the atmosphere.
3. Surface conductivity of the subject.

1.6 Publications

Letters emanating from this research was published in the accredited journals listed below:

1. *Electro Photographic Instrument for Use in Kirlian Photographic Phenomena Research*. Published in Quantum Test & Measurement [Williams & Khan, 2003].

2. *Implementation of a PC and Embedded Microcontroller Based Data Acquisition System for a Corona Visualization Device.*
   Submitted to ACTA Press for publication in the "International Journal of Computers and Applications".

A conference poster entitled *"A Monitoring System for Environmental Parameters impacting on Corona Discharge Visualization"* was presented at the Cape Biotech 2003 Conference

CHAPTER 2

SELECTION AND MEASUREMENT OF ENVIRONMENT PARAMETERS

2.1 Introduction

In this chapter, the issues surrounding the environmental parameters are discussed. This task required selecting, from numerous possibilities, environment conditions to monitor. This approach was taken to address concerns of critics that environment conditions are important when evaluating this phenomenon. It has been argued that no valid interpretation of the result can take place without the due diligence of the environment. [Pehek et-al, 1976].

2.2 Selection of Conditions to Monitor

Many parameters known and possibly unknown, affect the detail of a Kirlian photographic picture, for this reason careful attention to the environment is required. To ascertain which parameters will be of significance to this device two a lines of investigation was embark upon. Firstly, an extensive interview with a leading expert on Kirlian photography in South Africa was carried out. Secondly, the published literature on this subject was consulted.

2.3 Interview with a Kirlian Photographic E xpert

On the 3rd of August 1999, an interview with Ebrahim Hossian of Medunsa was carried out. As leading expert in the field, he gave the current research new direction [Hossian, 1999]. One of the fruitful lines of investigation suggested was the reduction of the environmental variables to be monitored. As a result of this interview and reviewing the literature, the following five parameters for measurements were selected:

1. **Ambient Temperature**:

   This can be considered a universal measurement parameter. Almost all physical phenomena are influenced directly or indirectly by temperature.

2. **Atmospheric Pressure**:

   The corona discharge are influenced by atmospheric pressure according to Paschen's Law hence its inclusion as an environmental parameter to be measured.

3. **Ambient Light Level:**

   Its imperative that any picture taken should arise due to the Kirlian effect, therefore this measurement is chosen to make certain there is no contaminating light.

4. **Humidity:**

   Electrical Phenomena in general and corona discharges specifically are affected by the amount of moisture in the air.

5. **Pressure of the object:**

   The pressure of a specimen on the dielectric medium in most cases directly affects the surface area exposed to the high voltage system and thus the resultant picture.

Other possible candidates for measurement were excluded because either because it has minimal impact on the picture or because they were too complex to implement in practical terms. The measurements excluded are:

1. **Ionization potential:**

   The amount of charged particles in the air could most like effect the resultant picture. It was however not considered as it was to complex to obtain a significant real-time measurement of this parameter.

2. **Angle of object with respect to the dielectric medium:**

   This parameter definitely affects the Kirlian photograph. It was not considered as it only pertains to finger tips photographs and was considered superfluous at this stage of this design.

3. **Surface conductivity of the object:**

   The surface conductivity of certain subjects such as people will most probably be effect resultant photograph. Again this was too complex to measurement to perform with this "proof of concept" device.

4. **Qualitative and Quantitative composition of atmospheric gasses:**

   The distinctive blue colour which is characteristic of most Kirlian photographs is the result ionization of molecules and atoms of the atmosphere and then the subsequent recombination of the ions and electrons [Dakin, 1975] therefore composition of the surrounding

gasses directly affects the photographs. Needless to say that these measurements were not incorporated in the device presented.

2.4 Measurement Circuitry Implementation

A design feature of the instrument allows the straightforward inclusion of up to four additional environment parameters, provided the sensor and supporting circuitry supplies a linear 0 to 5 Volts in response to the measurand. It is perfectly conceivable that biometric data could also be included amongst these additional measurements.

2.4.1 Temperature Measurement Circuitry

The temperature circuitry employs a standard LM35 integrated sensor. The circuit diagram is shown in Figure 2-1 below. This sensor uses the effect that the difference in voltage between two transistors operating at different current densities are for the most directly related to the temperature of the transistors in Kelvin (National Semiconductor, 1986). The LM35 develop these characteristic to provide a stable, accurate and reliable temperature detector which output is directly related to the temperature in degrees Celsius.

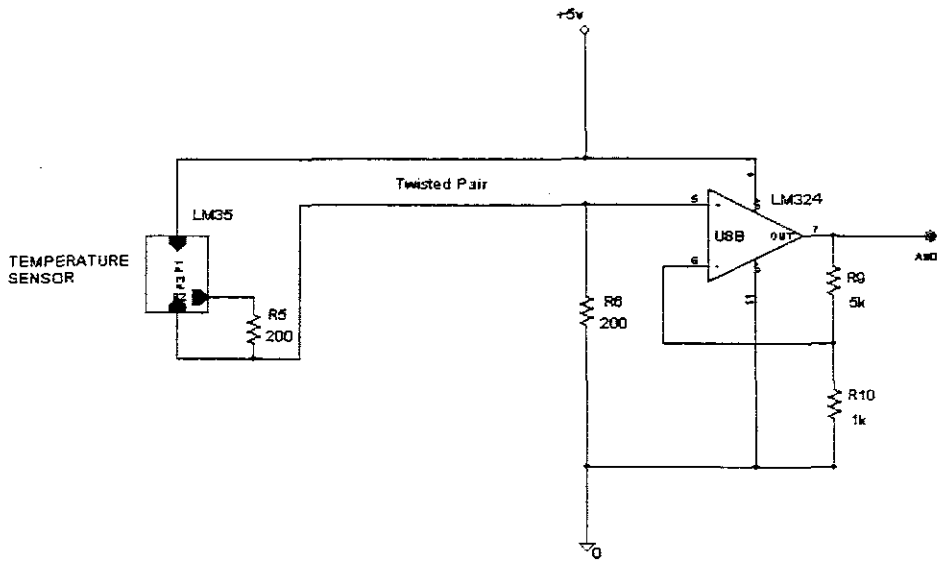


Figure 2-1: Temperature Sensor Circuitry

The circuit conforms to the two wire remote sensor configuration published in the National Semiconductor Application Note (National Semiconductor, 1986). In this configuration a single pair of twisted wires delivers the power to the sensor in addition to providing the signal to the operational amplifier. The operational amplifier is in non-inverting mode and operates from a single supply voltage of +5V. The output of sensor in this configuration is 10mV per degree centigrade with 0°C providing 0V. The circuit operates with a voltage gain of 6 calculated as follows:

$$VoltageGain = \frac{R_F + R_{in}}{R_{in}} = \frac{R_9 + R_{10}}{R_{10}} = 6 \qquad (2.1)$$

2.4.2 Humidity Measurement Circuitry

This measurement circuit consists of a Siemens capacitive humidity sensor and supporting circuitry. The sensor consists of thin-film polymer membrane and has a near linear response to humidity from 10% to 90% RH (relative humidity). See the relevant data sheet (Philips: 1997).

Three integrated circuits are employed to perform this measurement. The first stage consists of LM7555 is in astable mode. The sensor forms part of the main frequency generation ladder. This timer outputs a frequency inverse ly proportional to the ambient humidity, according to the following equation.

$$f = \frac{1.44}{(R_1 + 2R_2)C_h} \qquad (2.2)$$

where f is the frequency out
C_h is capacitance of the sensor
both R1 and R2 is 330KO

Extrapolating the minimum and maximum capacitance values from curves in the data sheet gives us values of 110pf and 150pf respectively. Using these values in equation 2.2 this gives us a minimum frequency 9696 Hz of and a maximum frequency of 13222 Hz respectively. The output of this timer was then taken straight to that of the frequency to voltage converter.

$$V_{out} = f_{in} \times 2.09 \times \frac{R_L}{R_S} \times (R_t C_t) \qquad (2.3)$$

where Vout is output voltage of this stage

fin is the input frequency

R_L is R10 it value is 100k

R_S is R6 it value is 15K

R_t is R7 it value is 6.8K

C_t is C3 it value is 10nF

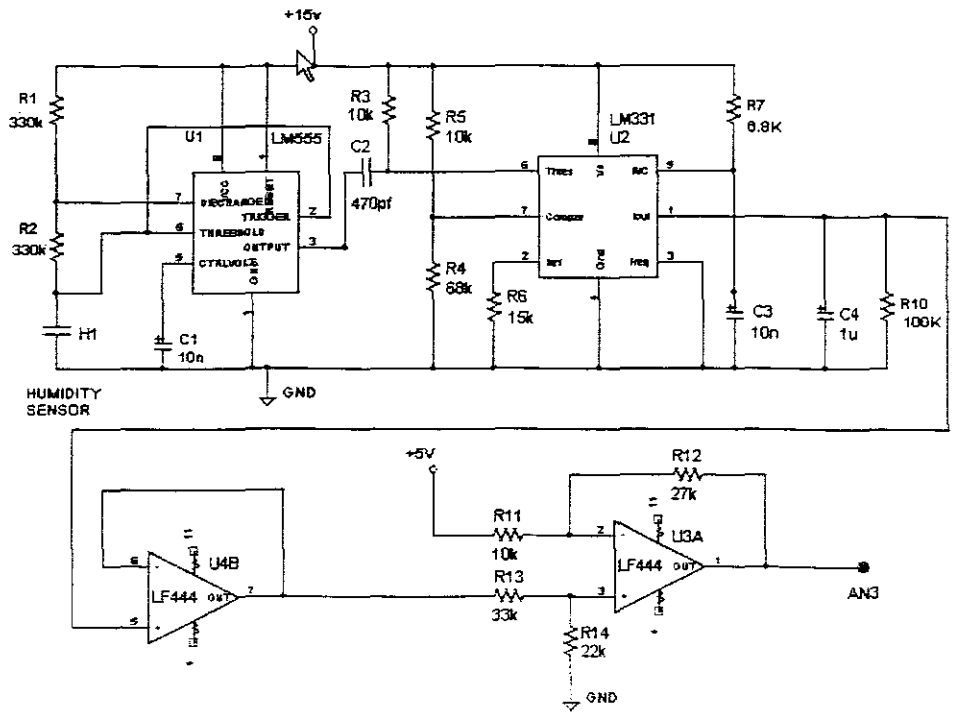


Figure 2-2: Humidity Measurement Circuitry

The output of this stage varies from 12.5V to 9.2V according to the above formula. A voltage follower buffers the next stage which is a difference amplifier. Selecting the correct values for the difference amplifier was done as follows:

Gain required 1.5; calculated from 5V/(12.5V - 9.2V).

Offset adjust required 13.8V; the minimum value of 9.2V needs to be subtract when amplified by 1.5.

$$V_{out} = \left(\frac{R11 + R12}{R13 + R14}\right)\frac{R14}{R11}V_{in} - \frac{R12}{R11}V_{ref} \qquad (2.4)$$

where Vout is output of the difference amplifier

Vin is the input from the voltage-follower stage

Vref is a reference voltage coming straight from the 5 volt supply line

R11 va lue is 10k

R12 value is 27k

R13 value is 33k

R14 value is 22k

Applying the fixed values to equation 2.4 results in the following:

$$V_{out} = 1.48 \times V_{in} - 13.5V \qquad (2.5)$$

This difference amplifier conditions the signal to be in the range of 0.12V to 5V, according to the formula above [National Semiconductor, 1980]. This output was then delivered to the analogue **AN3** input.

2.4.3 Light Measurement Circuitry

There are a number of measurement criteria when it comes to the formal process of measuring light levels. It is sufficient of this investigation that the measurand required is that of **illumenance** . The SI unit of illumenance is the **lux** which is expressed in units of lumens/m^2. These units refer to the response of the human eye; however the equivalent detector units would be that of **mW/cm^2**. The CIE standard of luminosity, often called the standard eyeball relates the two units [Squillante & Shah 1999].

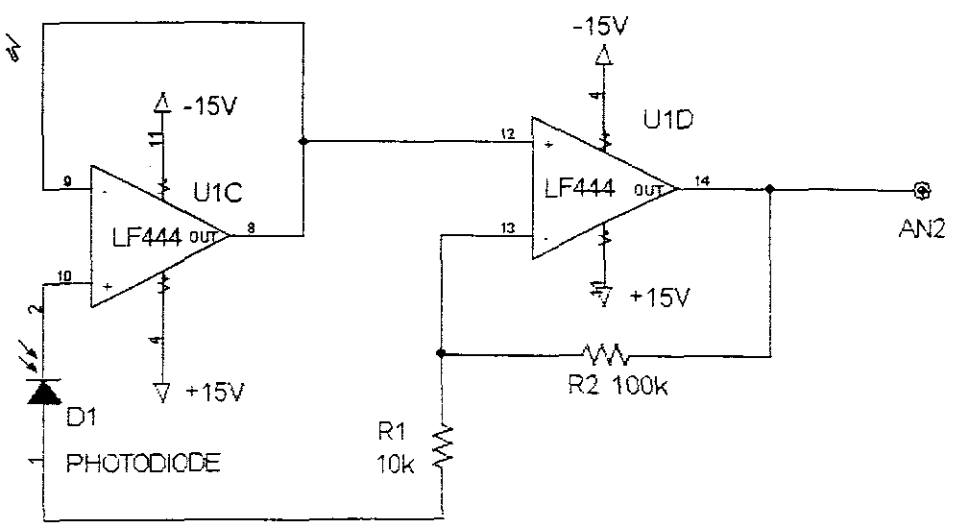


Figure 2-3: Light Measurement Circuit

The light intensity could vary drastically, this is because the instrument could under some conditions can take pictures even if the lid was open therefore it would be

ideal to display the light intensity values over a large scale. Very opportunely, the eye and photodiode detectors in an open circuit voltage mode respond logarithmically to light intensity, therefore the measurements are calculated and are displayed in this form. Straightforward circuit produces the signal we require see Figure 2-3. The first stage voltage follower input op-amp which ensured the diode is not loaded and thus open circuit voltage mode in ensured. According to the datasheet of BPW21 PIN photodiode of Infineon a light level of 8 Lux produces an open circuit voltage of 250 mV at the low-end of illumination scale whereas a light level of 2000 Lux produces a voltage of 450 mV at the high-end. The next stage is simply a non-inverted provides a gain of 11. This output was then delivered to the A/D input **AN2**.

Note that 8 Lux is minimum detectable light level of the BPW21 due to "dark current" swamping the signal at lower levels. A better PIN diode, such as the BPX63, would improve the detection limit, however dark-currents is a problem intrinsic to photodiodes technologies and is not easy or possible to reduce it completely.

2.4.4 Barometric Pressure Circuitry

The SI unit of measurement of pressure is the **pascal (Pa)** whereas the barometric pressures (P B) or ambient air pressure is commonly measured in **millibars**. There is linear relationship of 1 millibar to 100Pa between the two [Heeley, 1996]. Barometric pressure can change from about 940mb to 1060mb due to weather conditions alone.

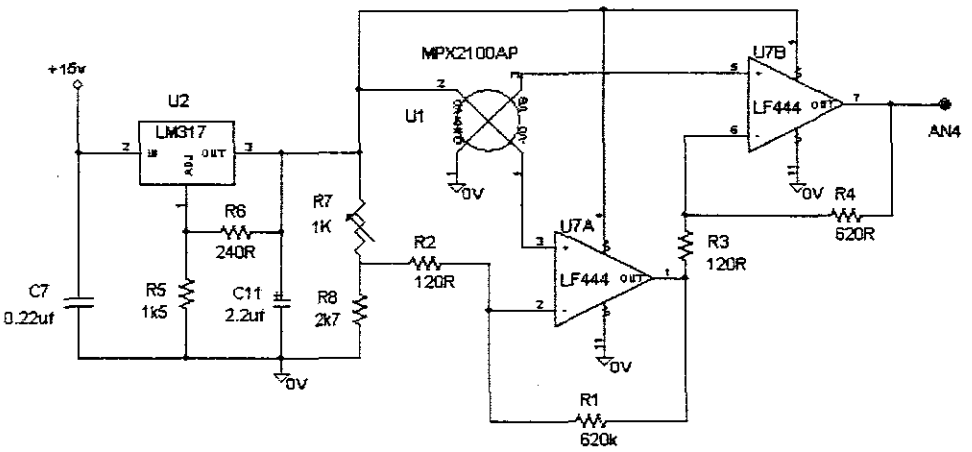


Figure 2-4: Barometric Pressure Circuitry

The two significant areas where barometric pressure measurements are required are in the meteorological and avionics industries. As the weather and altitude each significantly impacts on barometer readings this makes their implementation in any these domains reasonably complicated. For our purposes however we are only interested in the absolute air pressure therefore considerably simplifying this measurement. The circuitry employed is based partially on the Motorola Application note AN1326 [Winkler & Baum, 1997].

This measurement was accomplished by using the Motorola MPX2100 silicon piezoresistive pressure sensor. These devices are manufactured in difference configurations. The version used was the MPX2100AP, which features temperature compensation, absolute measurement, and a single measurement port. The maximum change in pressure due to reasonable altitude change and weather conditions is 24 kPa this corresponds to winters day at level (Cape Town) to a summer's day in Johannesburg at 1200km altitude. This will produce a voltage swing of 24% of full-scale swing (40 mV) or 9.6 mV. In order to have full-scale deflection at the A to D

input (5 V swing). The gain should be in the order of 5/0.009.6 or 520. The elements of this sensor are arranged in bridge formation. Using a classic instrumentation amplifier design a gain of 520 can be achieved with R1=R3 = 120 O and R2=R4=620KO. Note calibration of this device has not taken place thus far.

2.5 Software Monitoring Aids

The initial device specification, calls of a monitoring system which would aid the operator of the device when performing Kirlian photographic experiments. The computer program was designed in such a manner as to provide an "Experimenters View" that is; all the information and details pertinent when taking these photographs are provided to the operator. Figure 2-5 below shows the absolute maximum and minimum values of the alarm limits, thus all the measured variables are within limits and are shown in green. Whereas Figure 2-6 depicts the practical implementation of alarm limits with two measured variables causing alarm conditions. The first alarm condition that of temperature, which is measured at 19 degree Celsius whereas the limit is 10 to 16 degrees, thus the emphasis on 19 in the panel. The second alarm condition shows humidity at 40% RH whereas the limit for this experiment is 45% to 60%, thus the emphasis on 40 in the panel. An additional GUI feature is that the colour of alarm indicators gives a visual clue as to whether the specific environment variable is above or below the alarm limits. Note the sensor system to measure object pressure was incomplete and therefore the acronym NIY (Not Implemented Yet) was used.

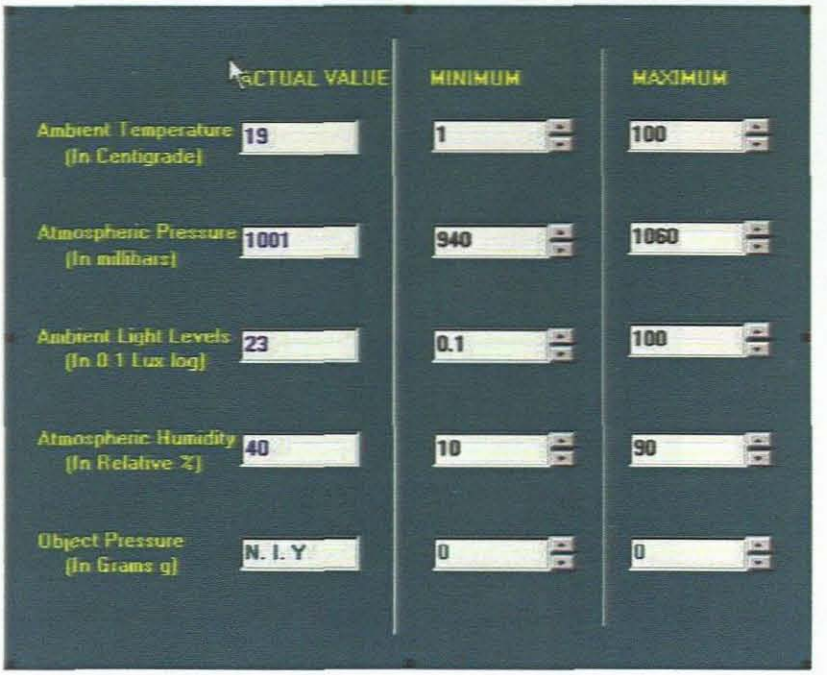


Figure 2-5: Environmental System Showing Absolute Limits

In addition to having all the data available two other invaluable resources is made available namely:

1. The operator can store an "Experiment Profile". This is all the applicable information required when taking a Picture and the ability to recall that data to reproduce that particular experiment.
2. The operator can also set high and low alarm levels of any of the measured variables. This feature if correctly utilized can further ensure that reliable experimentation takes place.

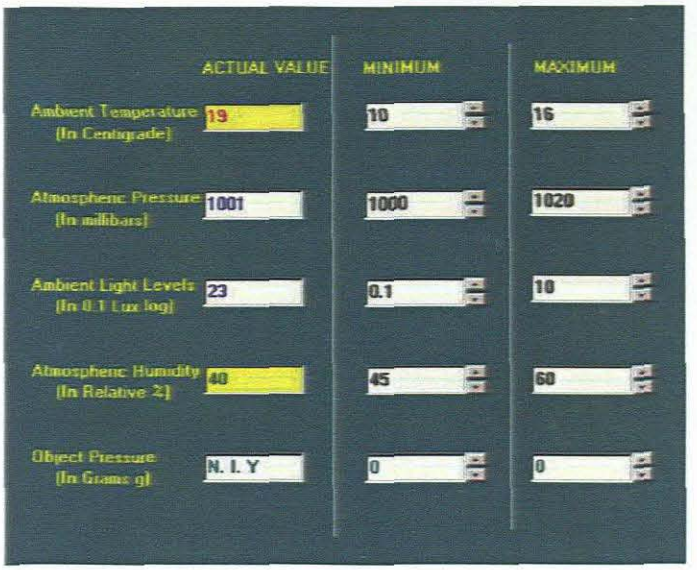


Figure 2-6: Environment Variables Showing Practical Limits with Alarm Conditions

The initial settings of these alarm settings do not prevent an operator from taking a picture but simply warns him that the tolerance of the particular measurements is in question, and is most likely out of bonds of his experiment design. As a method of guaranteeing the consistence of the experiment profiles, these limits are stored separately with each experiment's data.

2.6 Conclusion

The measurement of the environmental parameters although not inhibiting to the operation of the device it is critical in terms of providing scientifically valid results. Careful attention to the details of the measurements and its associated systems has to occur if any significant conclusions are to be drawn from the results. As shown above it is reasonable to assume that as other significant measurement parameters are discovered they could readily be added to this preliminary Kirlian device or to subsequent designs.

CHAPTER 3

TRANSPARENT LIQUID ELECTRODE TECHNIQUE WITH CCD IMAGING

3.1 Introduction

The original method of taking Kirlian Photographs involves using photographic film as the dielectric material between the subject and the high voltage source. Excellent quality pictures can be obtained using this method as there is no lens or camera involved in the process. There is however many limitations when using this direct contact method, these include but not limited to the following.

3.1.1 Darkroom Film Processing

The photographic film must be handled and processed in darkness during the entire process of taking a Kirlian picture [Iovine, 1993]. That means either a darkroom for black & white film or near total darkness for colour film is required. There are other techniques which do not suffer this limitation but they introduce other concerns these include the quality of pictures and other handling constraints. These techniques include keeping the unexposed film in a light tight envelope and then using this

envelope as the dielectric medium yet another technique uses fax or thermal ink paper.

3.1.2 Film Size

The developed film size is limited is particular photographic development system available. A normal automated laboratory processing of colour film is limited to 5 by 4 cm prints.

3.1.3 Film Flatness

Thirdly, special care must be made to ensure the film remains flat when taking the photographs. If this is not assured, the pictures will possibly show colour artefacts. [Tiller:1975]

3.1.4 White Balance

Lastly, if positive prints are used special care is required to insure that the white balance is set correctly, if not set correctly the colour purity of the photographs will be compromised.

3.2 The Transparent Electrode Defined

An alternate method of taking Kirlian Photographs involves the use of a so-called transparent electrode. This method was most likely first proposed by Leonard Konikiewicz in the seventies [Konikiewicz, 1977]. This is simply a glass plate that is electrical conductive on one side. The operation of electrode involves connecting the conductive side to the high voltage source. The object to be photographed is placed

on the non-conductive side. When the high voltage supply is switched on a discharge occurs between object through the glass to the high voltage source. This discharge is then photographed through glass plate, which allows the use of ordinary camera to take a Kirlian picture.

There are a few of things to be aware of before applying this so called transparent electrode technique.

3.2.1 Film sensitive

Firstly, because of the low levels of light emitted, the camera or the film needs to be sufficiently sensitive.

3.2.2 Macro lens

Secondly, to reduce dispersion of light, the camera needs to be as close to the electrode as possible therefore some sort of macro lens capability is required. (Photographic professionals regard a macro lens as a standard way of taking close-up photographs).

3.2.3 Manual or long exposure times

Thirdly, if the photographic film or CCD is not sensitive enough, longer exposure time will be required. Normally a manual shutter mechanism is required to achieve extended exposure times this is typically referred to as the bulb setting in photographic circles. As states previously, a dramatic increase in the exposure time would decrease the sharpness of the image [Iovine, 1993].

3.2.3 Protection of conductive layer

If a conductive glass sheet is used, care must be taken to prevent localized hot spots destroying the conductive layer.

When all these issues above are adequately addressed, as it was the case in this investigation, this technique suffers none of the debilitating issues described in the introduction of this chapter.

3.3 Construction of the Transparent Liquid Electrode

Sourcing of a conductive transparent glass sheet proved difficult, as there are no known local suppliers of this product. Homemade versions of this product do prove to be some what unreliable as the coating tends to be easily destroyed if there is a strong local discharge. Due to the unavailability of a conductive transparent glass a Transparent Liquid Electrode (TLE) was designed and constructed. See Figure 3-1 below for details of this construction. This electrode consisted of an opaque plastic PVC frame with a 3mm glass sheet glued to one side and a 2mm glass sheet on the other side. This mechanism resulted in a thin hollow sandwich be ing formed. A small tapped hole was drilled in one side to allow the sandwich to be filled with electrolyte. The electrolyte used was common table salt (sodium chloride) dissolved into distilled water. In the first electrode, a concentration of five grams per litre was used. It was found that the concentration of the solution is not critical. The tapped holed was then sealed with a matching stainless steel screw which was long enough to make contact with the electrolyte to provide a path to the high voltage supply. Different thickness of

glass sheeting was used, to allowing the research to observe in any difference in pictures using them alternately as the dielectric medium.

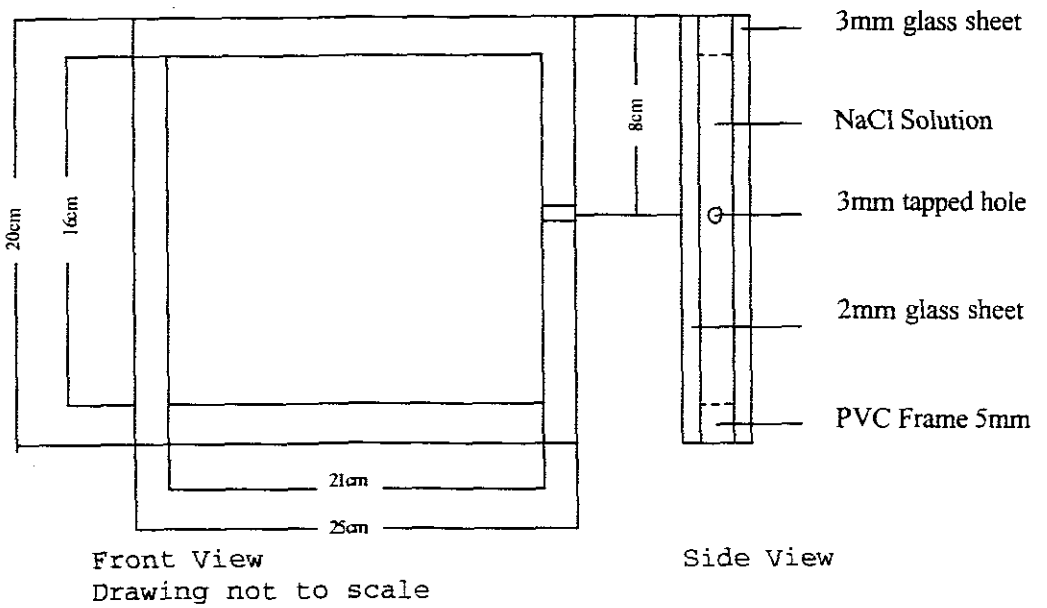


Figure 3-1: Transparent Liquid Electrode Construction

This electrode has added benefits of been practical immune to hot spots as well as relatively inexpensive to construct.

3.4 Initial Kirlian Photographic Experiments

A proof of concept experiment was needed in order to prevent a situation where a lot of work is put into the proposed instrument only to find out later that there is something flawed in the photographic system or the original assumptions. It was decided to construct a test rig that will evaluate the basics of the photographic subsystem. This test rig consisted of the following components:

- A Transparent Liquid Electrode (TLE)

- A laboratory insulation tester with an adjustable high-voltage supply of 5000V to 30 000V at fixed frequency of 50Hz
- A single lens reflex (SLR) camera, Nikon F50
- A freshly cut leaf which served as the subject
- And a wooden block sprayed matt black to support the subject and TLE

These components were arranged in a dark room of the Electrical Engineering department. The subject was placed the block of wood. The wood served as an insulator between the subject and the floor. The transparent electrode was placed on top of the subject. The camera mounted on the edge of the bench was focused on the subject.

The initial results were not promising as most of the pictures were "washed out" due to overexposure or black due to insufficient light from the subject. Once this was compensated for, by reducing the exposure time and increasing the speed of the film satisfactory results was obtained see Figure 3-2 for our first successful picture. From the experience obtained from these trail runs it was decided that our assumptions about taking Kirlian Photographic pictures was correct and it is with in our capability to complete the project as specified. It is important to note that the device designed and constructed in this research, routinely takes pictures of this quality and better with the environmental data automatically captured and with out the need of a darkroom.

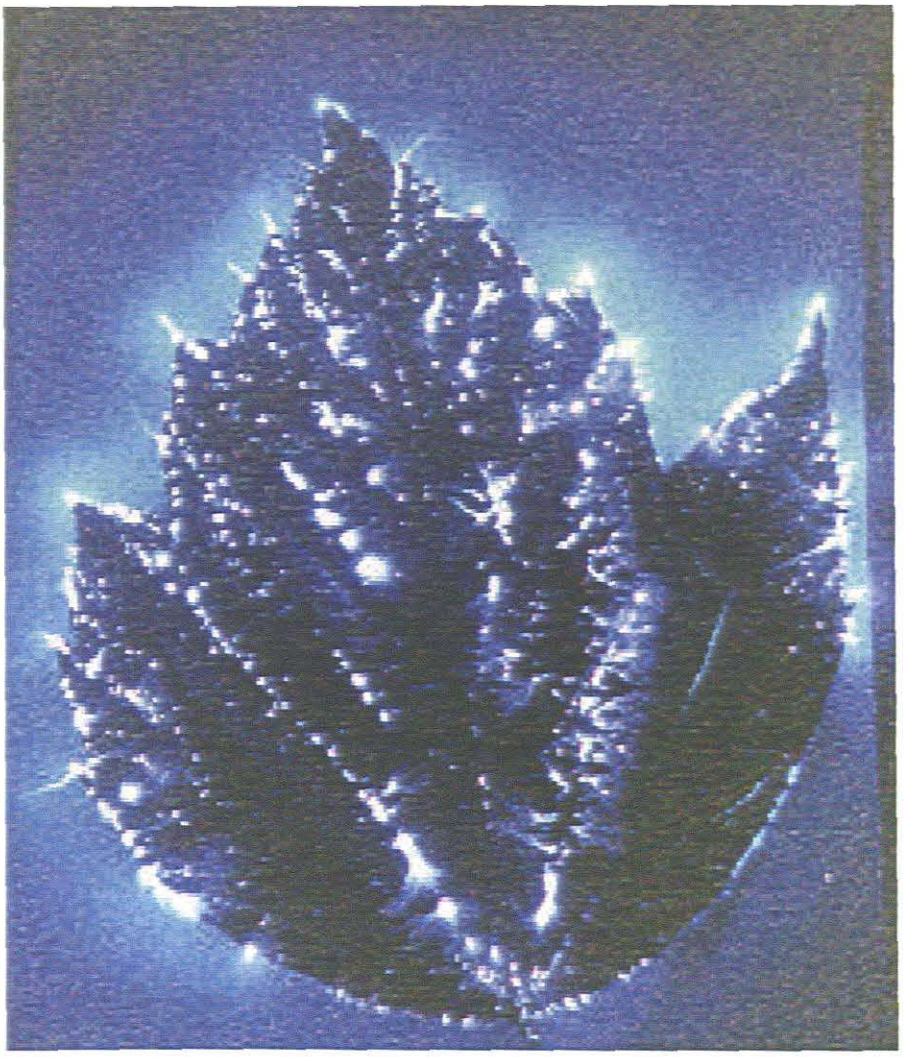


Figure 3-2: First Successful Picture with a Test Rig

3.5 Selecting Camera Technology

The TLE technique allows any conventional camera to take Kirlian photographs. Therefore a decision as to which camera technology would be suitable was required. Techniques exist which can take pictures without the use of film and its associated development processes. An obvious example of this is digital camera

technology. If digital camera technology could be used this would bring tremendous advantages over the traditional method of taking Kirlian photographs. A decade ago, this technology was not really an option for taking Kirlian Photographs as it was too expensive and the quality of the pictures was unacceptable. Today with the rapid increase in technology, it has become possible to consider digital cameras as a means of taking these pictures.

Some of the benefits of using digital camera include, no darkroom processing with its associated consumables and near instant rendering of the photographs. Another advantage of digital technology is that pictures can effortlessly be transferred to a computer. Using this technology for this project has meant that no scanning in of pictures is necessary as the Kirlian Experimenter software automatically captures them. This study is not the first to successfully use digital camera when taking Kirlian photographs, however published information surrounding this application is very scarce indeed.

There are limitations to the use of digital technology. Ordinary film cameras still provided better quality pictures than CCD or CMOS based digital cameras.

3.5.1 Evaluation of CCD Camera Performance

There are four main characteristic or features required by a Digital Camera or any other camera for that matter, to take Kirlian pictures using the transparent electrode technique. These features are listed below:

High Sensitivity The first requirement is sensitivity of the camera needs to be at least rated at ISO400 or else the exposure times required will produce ill-defined pictures.

Manual Shutter The ability to control the shutter speed of the camera externally. Mainly to achieve exposure times up to 15 seconds.

Short Focal Length The ability to focus at a short distance, 30cm or even shorter. This capability is usually document by camera manufactures as a camera that has a macro feature.

Large Aperture This allows the maximum amount light from the subject onto the CCD. This is more usually specified by the F-Stop number of the lens. The lower the F-Stop number the larger the aperture. An aperture setting in the region of 2.0 is suitable.

The relationship between the sensitivity, shutter speed and aperture settings for a particular light level is commonly referred to as the exposure setting of a particular camera setup. Ideally, for a given amount of available light from a subject there are myriad combinations for these three settings. For example, one could increase the aperture setting to compensate for decrease sensitivity of the film or CCD. Another scenario allows a more sensitive the photographic device to take picture with a slower shutter speed. A good quality camera operating under normal conditions would allow the aperture to decrease while increasing shutter speed. The net result of this is the exposure remains constant. However when taking Kirlian pictures most cameras will be operating at the very limits of their sensitivity and will not have much leeway in adjusting its exposure values. Careful consideration of the detection limits of the digital camera is required when attempting to take Kirlian photographs.

To observe whether suitable results can be obtained with CCD devices a video camera and CCD camera was used in similar trials. The first budget consumer CCD camera failed this trial as its sensitivity was only rated at an effective ISO100 and it shutter speed was only adjustable to a maximum of 33ms. The video camera was able to take recognizable Kirlian photographs because it hadthe relativity unique feature of a slow shutter speed setting of about one second and the sensor was based on CMOS technology. Unfortunately, the quality of pictures produced by the video camera was inadequate. As can be deduced the quality of pictures depends nearly exclusively on these exposure characteristics as well as the sensors resolution. Therefore it was important to have these trials as it showed which cameras or devices would be suitable to be included into the Kirlian device.

3.5.2 Digital Camera Selection

The process of selecting an appropriate digital camera was, in a sense, made easier by the numerous specific features required by this project. Firstly, budget constraints limited the choice to be made from the so called "prosumer" range of cameras. This range is known, industry wide, as products which falls, in marketing terms, between that of the professional and consumer ranges. Secondly, the photographic requirements which are listed above do severely limit the choice. Thirdly, electrical and mechanical compatibly was an issue as it was a requirement to control the camera from a PC. A short list of cameras which would fit the requirements was drawn-up. Key advantages and disadvantages is listed see Table 3-1 below.

Table 3-1: Digital Camera Short List

Digital Camera	**Advantage**	**Disadvantage**
Sony	Good all-round capabilities	Dubious PC control abilities.
Olympus C4040 Z	Excellent all-round capabilities	Limited Macro features
Fuji Film	Very good low Light capabilities	Aperture settings limited. Expensive.

The superior aperture settings as well as its high CCD pixel count (4.1 Mega Pixels) clinched the decision in favour of the Olympus camera. It is possible that the other candidate cameras could perform admirably but that is not relevant in this discussion. It is important to be cognizant of the fact that major manufactures of digital cameras release superior products approximately every 6 months.

3.6 Conclusion

The TLE technique together with the using modern digital imaging techniques introduces unique modalities for taking Kirlian photographic pictures. Moving this technique from the darkroom literally and figuratively introduces exciting new avenues of investigation.

CHAPTER 4

THE HIGH VOLTAGE SUPPLY SUB-SYSTEM

4.1 Introduction

The Kirlian photographic device should have a stable high-voltage supply for results to be scientifically significant. The initial design specified a high-voltage supply with the following features:

1. Adjustable voltage output from about 4 to 20kV
2. Adjustable pulse repetition rate of about 50Hz to100kHz
3. Able to source current of 5mA
4. The adjustable parameters should be able to be configured digitally.
5. Galvanic isolation of at least 2kV between the controlling microprocessor and high voltage generation circuitry

4.2 Topology Selection

The specifications listed in the introduction proved to require a more elaborate solution than was first expected. The requirements listed cannot be easily solved by conventional design theory because of two main reasons. Firstly, the relatively wide frequency range required. Secondly, the unknown loading factor of the subject introduces further complications. This is discussed in section 4.3. Nevertheless, progress was made in selection and initial construction of a solution.

Various designs topologies were considered these include voltage multiplication systems such as Cockcroft-Walton and simple Step-Up High Voltage transformers. It was made clear, from an earlier investigation, that the flyback converter topology would be most suited to the solution required. The flyback converter is a modified buck-boost converter with transformer-coupled output. The transformer acts the common inductor between the two main current loops in a typical buck-boost circuit. It would be more descriptive to call the transformer a "two winding inductor". Most often it is simply called the "flyback transformer".

Figure 4-1: High-Voltage Coils: The LOPT and the Autotransformer

Commercially, flyback transformers can be found wherever high-voltages are required. Two mature applications areas rely on this method almost exclusively when generating high-voltages. These applications are the driving circuitry for cathode ray tubes (CRT's) and for producing sparks in petrol combustion engines via an autotransformer. See Figure 4-1 for examples of these coils. The term "auto"-transformer can have dual connotations; firstly it is used in the "auto" industry and secondly, in the electrical sense it is an "autotransformer" since the primary and secondary winding has one common terminal. This text refers to its electrical meaning

unless noted otherwise. The term LOPT or Line Output Transformer is frequently used to describe the flyback transformer driving a CRT.

Designing and building a transformer from scratch was not considered as it would be beyond our level of skill and expertise as the high-voltage winding would consist of many accurately spaced turns of very fine wire. See Table 4-1 for a concise comparison of the two types of coils considered.

Table 4-1: Comparing LOPT with an Autotransformer

Parameters	**LOPT**	**Motor Vehicle Autotransformer**
Primary Voltages	50V to 100V	12V – Nominal 40V – Maximum
Frequency ranges	10kHz to500kHz	50Hz to 10kHz
Output Voltages	12kV for Monochrome to 30kV for Colour	15kV to about 30kV
Output Current	10mA	Up to 20mA
Advantages	• High Operating frequency provides a more pleasant experience • Well defined electrical specifications	Robust Easy to work with
Disadvantages	Sensitive to over voltages	• Latent capacitance can produce an unpleasant "shock" • Unknown constructional materials used

In engineering terms the superior coil would be the Line Output Transformer (LOPT) used on a colour television set, because of its well defined electrical

characteristics however there could be low frequency applications which would better suit the autotransformer [Schroeder, 2001] & [Sliwczynski & Krehlik, 1998]. Although the final version of the instrument was constructed with the LOPT, it is a relatively simple procedure to swap coils as the final drive stage uses the same electronic circuitry. When swapping coils however the high-voltage circuitry will obviously need recalibration.

4.3 Flyback Converter Theory

The basic principle of generating high voltages by the so-called "Flyback method" relies on switching a current in a large coil off. By magnetically coupling this primary coil to a high voltage secondary coil, substantial voltages can be generated.

It is important to note that in this configuration that there is no voltage per turn balance between the two windings that is;

$$Vo / No \neq Vi / Ni \qquad 4.1$$

The voltage generated at the primary is governed by this equation:

$$Vo = -L\delta I / \delta T \qquad 4.2$$

Where Vo is the voltage produced in Volts;
and L is the inductance of the coil in Henry's;
and dI is change in current measured in Amps;
and dT is the period over which the change occurred in Seconds;

The circuit can be effectively be modelled by substituting the flyback transformer with an equivalent circuit consisting of an ideal transformer in parallel with the magnetizing inductance as shown in Figure 4-2. Classic al theory would suggest that operating the flyback transformer in continuous mode would be a possible solution as in this mode the output voltage is directly proportional to the duty cycle and turn ratio. This mode would have the added advantage of low output impedance [Erickson, 1997:45]. Unfortunately, the current requirement on the secondary side, which dictates whether this continuous mode is in effect, would be impractical. Therefore, the non-linear, discontinuous mode of operation was chosen. This mode has the disadvantage of high output-impedance as the output is load dependant. An important feature of the discontinuous mode was however used to good effect namely that the output voltage is also a function of the input voltage [Erickson, 1997:36]. This fact was used to modulate the output voltage. See section 4.5.

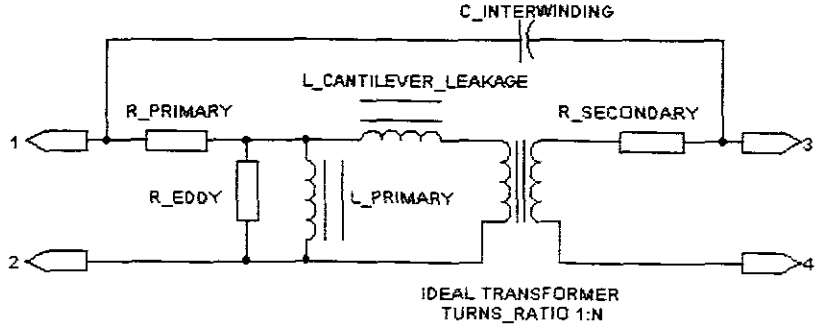


Figure 4-2: Cantilever Transformer Model

4.4 Practical High-Voltage Circuitry

This circuit description will trace high-voltage generation back through to the source. See the complete implementation of the high-voltage circuitry in Figure 4-3.

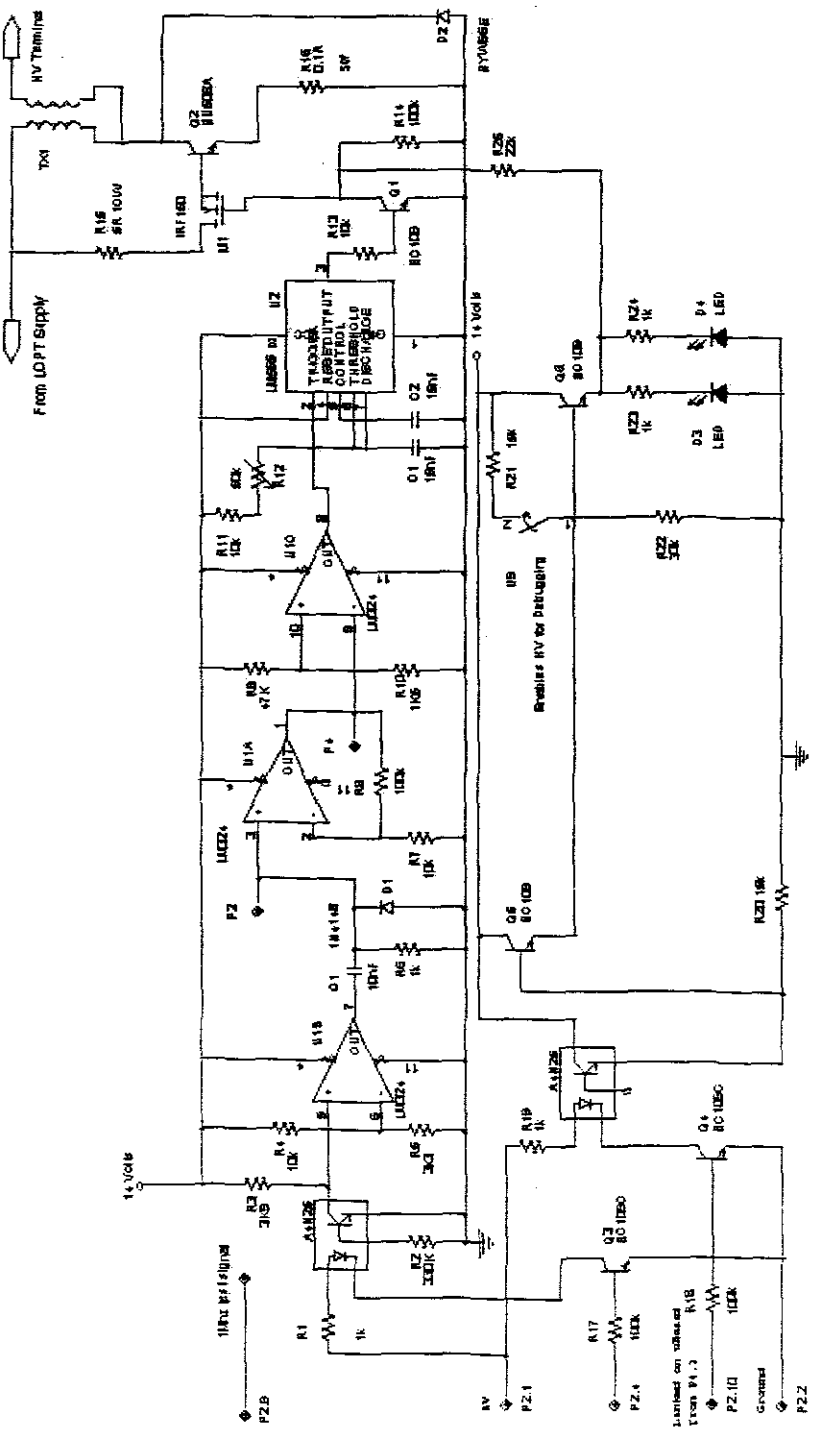


Figure 4-3: Flyback HV Implementation

A high-voltage fast-switching NPN power transistor Q2 was chosen as the switching device as supposed to a MOSFET because the flyback pulses regularly reach 1200 V peak, allowing for margins we needed a device which can withstand 1500V. There are few or no FETs available for that voltage range. The transistor used was the BU508A with maximum collector-emitter voltage is rated at 1500V.

A high-voltage avalanche diode D2 was used to protect the switching transistor. The diode used was the BYW96E rated at 1000V. A snubber circuit could also be used to perform this function it would require more analysis however and its operation would be frequency dependant. A power MOSFET IRF150 labelled M1 was used to drive power transistor Q2, since the current gain of Q2 is only 3 at required output current of 3 Amps. Using a M1 in this fashion means that it is easily driven from the frequency source which is the 555 Timer. A photograph of the practical implementation of this circuitry is shown in Figure 4-4.

The BC109 transistor Q1 performed a very important switching function; it either allowed or disallowed the frequency source, discussed above, to drive the MOSFET M1. Q1 could be turned on from two different sources the first being an onboard switch used during debugging or more importantly the exposure signal from the microprocessor. The exposure signal was isolated by means of an opto-isolator U4 aided by transistors Q5 and Q6.

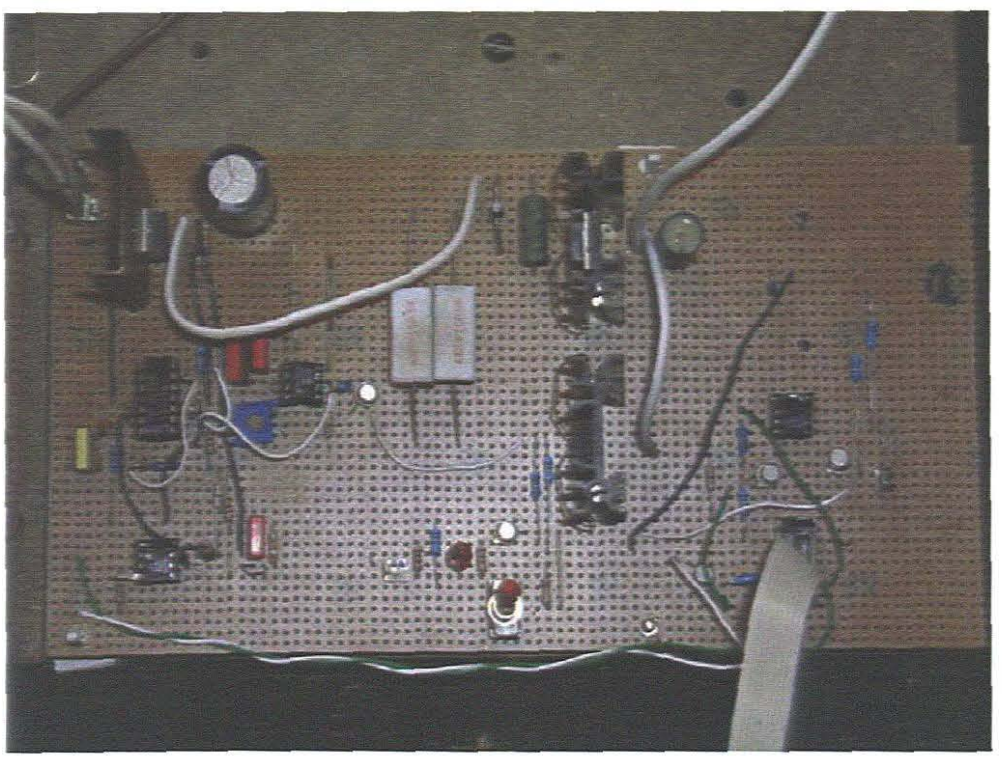


Figure 4-4: High-Voltage Supply Practical Implementation

The path of the frequency source consisted of an opto-isolator U3 as well as three op-amp stages U1A, U1B, and U1C. Briefly, U3 isolated the frequency generation from its microprocessor source. Op-amp U1B formed a comparator to ensure the correct level of signal is picked-up, as the output transistor was in common base mode to increase the frequency response. Capacitor C1, Resistor R6 and Diode D1 integrated this signal which ensured only the rising edge was amplified via U1A. A comparator stage formed with U1C to provide a clean negative going spike to trigger the monostable formed by the 555 timer U2. The monostable provides a convenient way of adjusting the duty cycle.

Modelling of the high voltage circuitry using the autotransformer was attempted using PSpice. The result from the model could not however be reconciled with real world measurements. The PSpice model considered required values which were unknown For example the effective air-gap length and as well as obscure materials parameter such as domain anisotropy were unobtainable [Tuinenga, 1988]. Other researchers had limited success with this simulation using their own modelling techniques [Schroeder, 2001].

Verification of the output voltage was achieved by using a custom designed voltage divider circuit as shown in Figure 4-5 below. No allowance was made for capacitive effects. A 999 mega ohm and 1 MO resistor was used series.

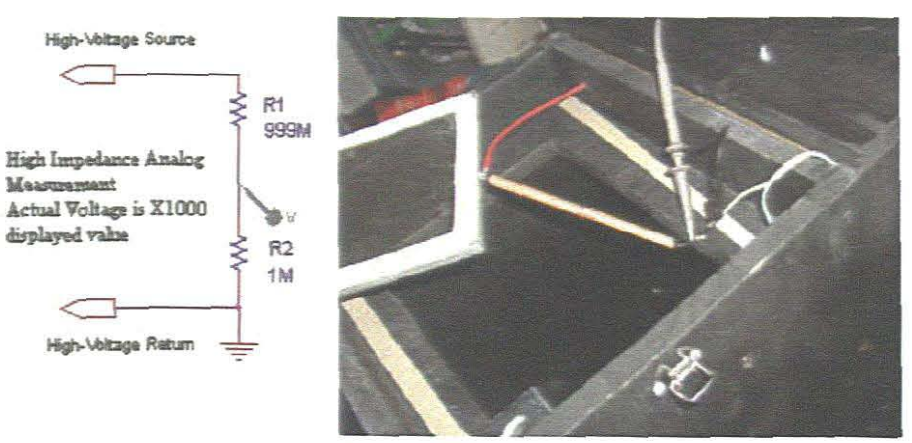


Figure 4-5: High-Voltage Measurement Implementation

4.5 Flyback Supply Circuitry

This circuitry supplied the positive terminal of the LOPT (traditionally noted as its B+ voltage) with a selectable voltage. The supply was based in main on the LM723C voltage regulator IC of National Semiconductor and a traditional unregulated supply consisting of a 150VA transformer TX1, a bridge diode pack D1 rated at 3A and smoothing capacitors C1 and C2. Refer to Figure 4-6. The negative supply rail was not used. The LM723 was configured to allow a series transistor Q1 to pass most of the current. The current was limited to 3A determined by R2.

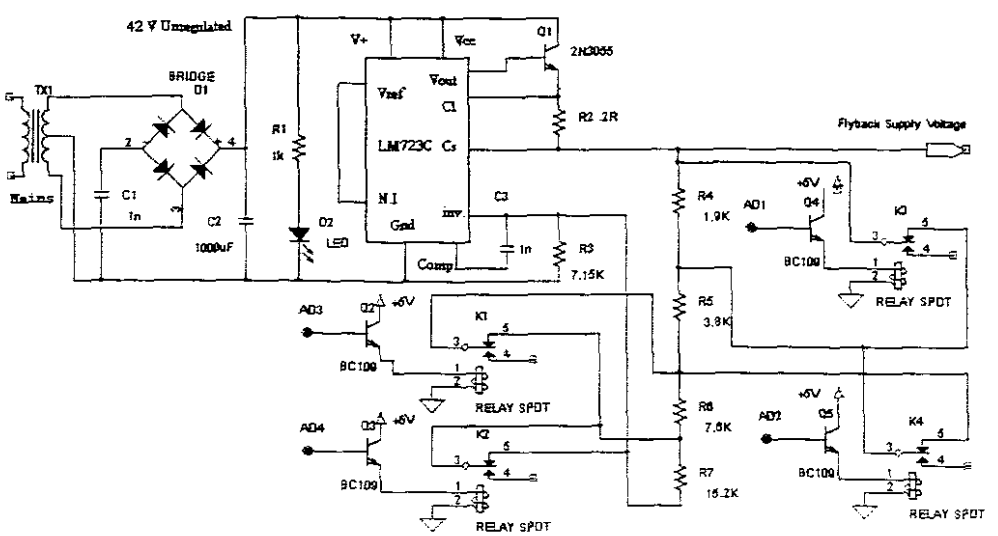


Figure 4-6: Circuit Diagram of the Flyback Supply

Voltage selection is achieved by switching four Printed Circuit Board (PCB) mounted rela ys. These relays individually select 4 resistors which in turn determine the voltage output of the regulator. Four digital outputs (AD1 to AD4) from the microprocessor are buffered via BC109C transistors Q1 to Q4 before driving the

relays. Using PCB relays in this manner result in the automatic galvanic isolation between the microcontroller and the high-voltage circuitry.

The voltage range of this supply is from 7V to 36V in 16 discrete steps. Calibration of this supply was dependant on the accuracy resistors in the ladder network. A mechanism to prevent this supply from sinking current from the high-voltage trigger circuit was not complete. That is why the practical minimum voltage was 10V.

4.6 High Voltage Isolation Mechanisms

The production of high voltage is almost always accompanied by various electromagnetic inferences. These include low to mid-frequency radio waves and induced voltages by means of capacitive or inductive coupling. In addition analogue measurement systems are also susceptible to ground loops [Hoover & Rempfer, 1988]. Even more significantly Personal Computer's switch-mode power supply and very fast switching (up towards 1 pico-second and faster) produces a significant number of electromagnetic noise. The PC itself is also susceptible to external noise sources.

Given the above compatibility issues, it would be prudent to isolate the various systems. Galvanic isolation was chosen to pre-emptively avoid these issues. The only other major source of noise to be concerned about would then be that of radio frequency interferences. Fortunately it is relative easy to electromagnetically separate the systems for the appropriate frequency range. In practical terms the galvanic isolation of 3750V was achieved between the high voltage circuitry and the

microprocessor by means of the 4N25 opto-isolators by Vishay Semiconductors and Günter PCB relays.

4.7 Conclusion

Careful design was required to produce stable high voltage supply. The proposed solution represented in this chapter can only be regarded as an initial attempt at this reasonably difficult engineering problem. This design requires attention to proper calibration.

A further enhancement would be to design a supply with a built-in feedback mechanism as this would negate the problems associated with high output impedances. The final product delivered an accurate output voltage of approximately 5% over the required scale on a fixed load resistance.

CHAPTER 5

HARDWARE AND SOFTWARE OF THE MICROPROCESSOR

5.1 Introduction

The Kirlian photographic device requires a system which would facilitate communication of device setting and environment data between a Personal Computer (PC) application and the device itself. Although it is conceivable to perform all the major instrument functions via a PC, this would mean serious constraints as to the specification and utilization of the PC. Typically a PC only version would require a dedicated unit and careful attention to real-time software concerns and more importantly a device driver may be required for each Operating System where this instrument would be employed. In a similar vain, discrete logic circuitry could be used to achieve similar results, this however would bring about unnecessary constraints and/or increased complexity. Therefore, it was decided to design the main functional controls around a microcontroller system, as this would achieve the design goals of a flexible instrument using a non-dedicated PC.

5.2 System Overview

A 32-bit Windows application was written to directly interface to a digital camera and, more importantly, for this discussion it was used as a "master" to control the instructions sent to the microcontroller "slave" unit. The theory and application of diverse

communication protocols has been the subject of numerous research efforts. This application, however calls for a relatively basic communication strategy as the data to be communicated can effortlessly be packaged into predetermined data types and sizes.

The decision to use to Siemens 80535 microcontroller was based on the fact that it provided a flexible and versatile platform to accomplish this and other development tasks. There is no doubt that, other microcontroller systems could perform these undertakings with similar results. [Semiconductor Group, 1991:14].

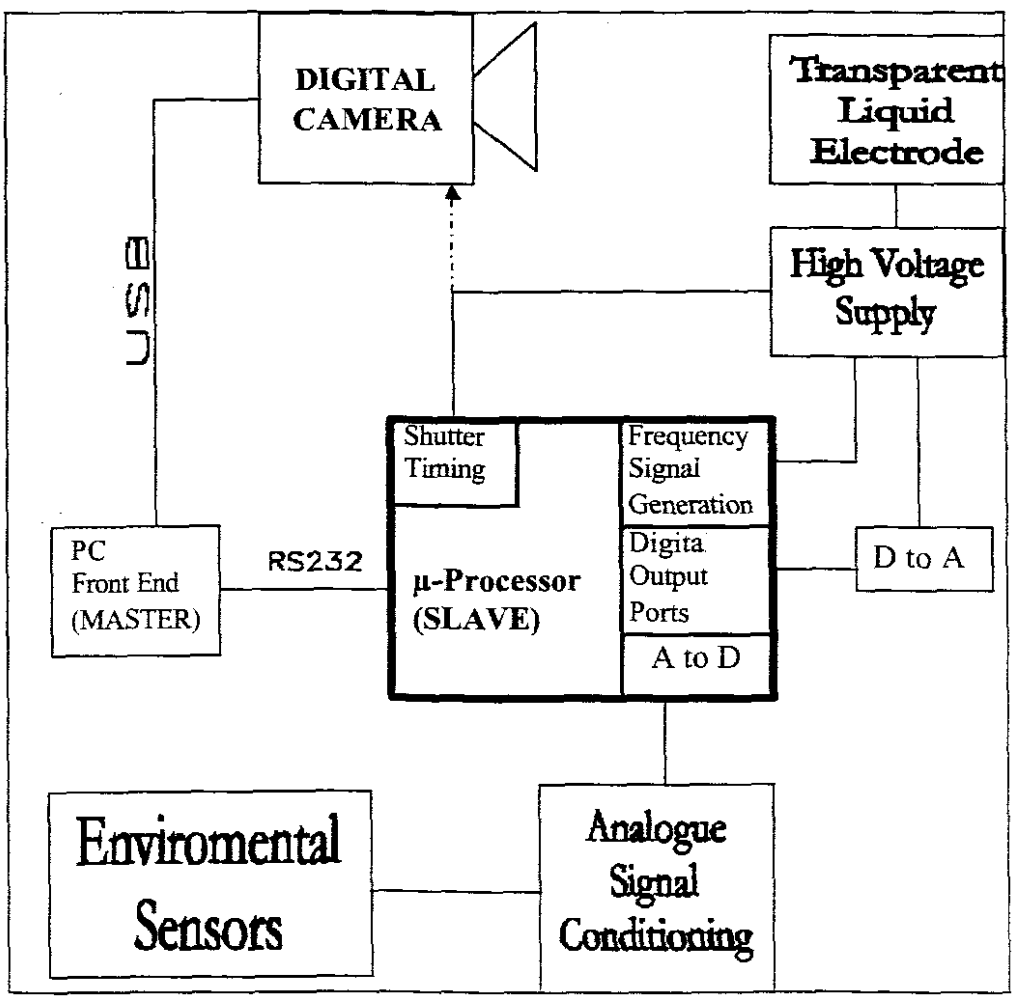


Figure 5-1: System Block Diagram

The 80535 is based on the widely used industry 8052 microcontroller. The key features of the 80C535 microcontroller as pertaining to this design is listed in the table below.

Table 5-1: Standard Microcontroller Features

Feature	**Details**
On-chip Program Memory	8 Kbytes (Disable for this project)
On-chip RAM	256 bytes (SFR's are located here)
Parallel I/O ports	Six 8-bit parallel ports
Serial ports	1 Full-duplex serial port (Not level compatible with a standard PC's serial port)
Timers	Three 16-bit timers
Analogue to digital converter	One 8 bit A/D converter with multiplexed inputs
Digital to analogue converter	None
Execution time	1μ instruction cycle at 12 MHz operation

Special function registers (SFR's) are used to control and configure all on chip peripheral components. The 80C535 microcontroller by its self provides most of the functionality required. Additional circuitry was required to provide:

1. Standard RS232C communication
2. Digital to Analogue functionality
3. The ability to store the microcontroller software on an EPROM

An example of a development board using 80C535 was adapted from [Ohsmann, 1997]. This adaptation extends the RS232 ability of the processor so that it was fully

RS232C compliant. It also provided a mechanism to execute a program stored on EPROM. These are two key features required by the Kirlian Device. The circuit was further developed as to include additional circuitry to provide Digital to Analogue capability.

This development board incorporates just 6 integrated circuits (IC's) and no other active components. A brief description of the IC devices follows

Table 5-2: Integrated Circuits Functional Description

IC – Part	**Description**	**Function**
80C535	Microcontroller	Provides the control and functionality
74HC573	Address Latch	Provide access to the 8 lower order address bits
74HC00	Address Decoder	Divides the memory into four 16 Kbytes section
MAX232	Serial line driver	Makes serial interface RS232 compatible
27C256	EPROM	Provides 32K program memory
62256-10	RAM	Provides 32K data memory

The development board requires various connections all of which is discussed. The circuitry requires 5V DC which is provided by the analogue power supply see Section 7.4. There is a reset switch which immediately restarts the program on the EPROM. Access to the microcontrollers' ports is via 10-way boxheaders. The boxheaders provided for the interconnection to the rest of the instrument as follows:

Table 5-3: Boxheader Description

Boxheader No.	**Description**
K4	Serial Interface
K5	Second Digital Port
K6	Analogue Input Port
K7	First Digital Input Port
K8	Frequency and Shutter outputs

Producing programming code for the microcontroller was accomplished by using a commercial C programming language cross-compiler, namely μVision (Rainbolt and Associates). The μVision implementation of C is based on the industry wide ANSI standard with specific language extensions for 80C535 and similar microcontrollers.

The compiled code in Intel Hex format was written to an EPROM. This EPROM was then inserted into the microcontroller system and became the only source of programming code, as the on-chip program memory was disabled.

5.3 Microcontroller Code Description

The chief function of the microcontroller software was to interpret the serial instructions transmitted by the software program on the Personal Computer. The software flow diagram is illustrated in figure 2 below. The microcontroller software operates in slave mode whereas PC program operates as the master. All the instruction to the microcontroller was sent and received via the RS232 port from the attached PC. The

main loop of the microcontroller software monitors the serial port then performs the required function and in some cases returns data back to the PC. Besides the main loop, two interrupts are also serviced

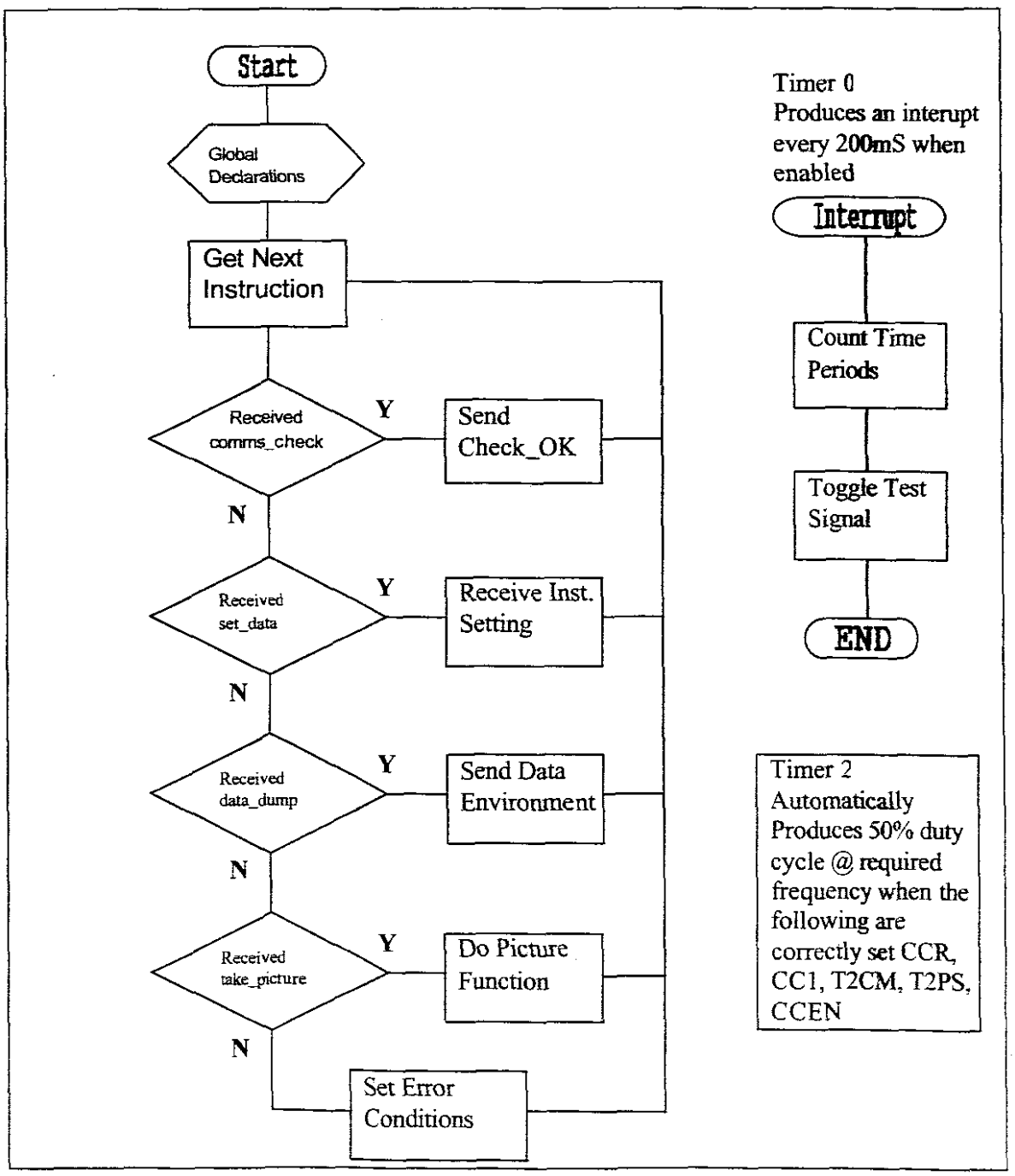


Figure 5-2: Main Program Loop and the Two Interrupt Routines

The main program loop follows typical C programming conventions. Global variables were placed in the main function and were kept to a minimum. The setting the Special Function Registers SFR of the microcontroller was key in setting the various functional blocks. A SFR specifically referrers to RAM based memory location on the microcontroller which accepts a byte as a control signal.

The system response time was non-critical as the values of the various device settings are communicated the moment contact was established and is only effectively used once the operator initiates the taking of a picture. Therefore a straightforward configured communications interface was used, namely that of the setting of 8 bits, 9600 baud, no Parity and 1 stop bit. To achieve those settings, the following SFR's were set. The SFR named SCON was set to 56_H which placed the UART into 8 bit receive mode. A standard baud rate was selected by placing $C0_H$ in the ADCON SFR. The SMOD bit was asserted in the PCON register which doubles the set baud rate frequency.

The interrupt system of the microcontroller was also configured by means of SFR's. Initially all interrupts were disabled by clearing the IEN0 and IEN1, while the global interrupt enable switch EAL was set. The 80c535 ports are easily configured via software as input or output ports. All that was required to make an output port was to write to the port in question; as such P4 and P5.

In addition to initialising SFR's and global variables the main function of the also receives a byte from the serial port and a switch statement then selects the appropriate alternate methods. The main loop would repeat it self loop until the microcontroller is reset or until the power is interrupted.

5.3.1 Exposure Time Signal Generation

The PC program allows the operator to specify the length of time the high voltage is enabled; this is also the required exposure time for the camera. This signal was generated by comparing the number of times Timer 0 was interrupted with the requested time period.

When an exposure output event was required the exposure digital output port was asserted high and while Timer 0 were enabled. Timer 0 generated an interrupt every 200ms when enabled. The code which serviced this interrupt routine counts how many times this interrupt was generated. This counted variable was compared to the count requested. When the values were the same, timer 0 was disabled and the exposure port was turned off. To achieve this functionality Timer 0 was placed in 8-bit auto reload counter mode also called mode 2. This was achieved by setting TMOD special function register to 22_H. The reload registers TH0 and TL0 were both set to $9C_H$. The exposure time signal was made available through second digital output point on Port 4. A test signal was available at line 8 of Port 4. This signal should be exactly 5 KHz and is available so long as timer 0 is enabled.

As stated earlier, the exposure signal could also be used to provide a shutter control for the camera. This method of activating the shutter of a camera could be cumbersome, as a mechanical system is required to depress the shutter via relays and solenoids. (Nonetheless commercial generic shutter release mechanisms are available). A better method would be to use the software to control exposure time of the camera. Software shutter control system was successfully implemented in the final prototype instrument.

5.3.2 Pulse Repetition Signal Generation

As with the other device parameters, the frequency of the device is adjustable by the operator. Timer 2 in conjunction with software mechanisms facilitates the pulse repetition signal or frequency generation. Two 8-Bit values are sent by PC software to microcontroller representing the period of the pulse repetition rate by this formula.

$$P = (freq\_h \times 256) + freq\_l \quad (5.1)$$

Where P is the pulse repetition rate in microseconds;
and freq_h is the high byte received;
and freq_l is low byte received;

Timer 2 consisted of a 16-Bit register (TH2 and TH1) it would increment this register every 1 microsecond (because a crystal 12MHz was used and the prescaler was disabled). When the timer overflows it would be reloaded from the CRC register. If the reloaded value was $FFFE_H$ the timer would overflow every microsecond. If the reloaded value was $FFFD_H$ the time would be reloaded every 2 microseconds. Thus, the reloaded value must be the exact number of microsecond away from $FFFF_H$. Hence we use

$$CRC = FFFF_H - P \quad (5.2)$$

Where CRC is the reloaded value of Timer 2

A unique feature of Timer 2 is the provision of compare and capture register. In order to implement a 50% duty cycle the CC1 register was loaded with a value half way between CRC and $FFFF_H$ it can thus be expressed by the following formula.

$$CC1 = FFFF_H - (P/2) \quad (5.3)$$

Where CC1 was the compare and capture value of Timer 2

The special function registers was used to place Timer 2 in a timer mode by setting T2CON.0 also called T2I0 to 1 while T2CON.1 also called T2I1 was cleared, with prescaler disabled by clearing T2CON.7 also called T2PS. Reload mode 0 was selected by clearing T2CON.2 also called T2CM. With automatic reloading taking place when Timer 2 overflowed. The compare function of register CC1 was enabled by setting the CCEN register to AA_H. This automatically configured line 1of Port 1 to generate 50% duty cycle square waves at the rate set by the PC software.

5.3.3 Communications Check

The purpose of this code was to establish whether communication between the PC and the microcontroller was successful. This was achieved by PC software writing the "comms_check" byte to the communications port. If all systems are operational, the microcontroller would respond by sending the byte represented by "check_ok" back to the PC. A typically scenario for its use, would be when the PC program needs to be certain that the microcontroller is ready for the next instruction, the microcontroller. Any other value, no response, or delayed response would indicate an error condition. The handling of error conditions is discussed in section 3.9.

5.3.4 Data Dump Routine

This code was used to transfer the values received by the analogue collection routine to the PC. The relevant data was stored in the appropriate global variables and then written to the serial port with a certain heading and trailing value. The PC program initiates this transfer by sending "data_dump" byte to the microcontroller and then expected received data to be in the correct format or else an error was flagged.

Table 5-4: Data Dump Packet Construction

Header	**Temperature**	**Pressure**	**Light**	**Humidity**	**Footer**
31_H	0 to FF_H	0 to FF_H	0 to FF_H	0 to FF_H	39_H

5.3.5 Set Values Routine

This code was used to obtain device setting from the PC. The packet received should be in the format shown in Table 3. An error condition was raised if the received packet was not in the prescribed format

Table 5-5: Set Values Packet Construction

Header	**Voltage**	**Frequency-Low**	**Frequency-High**	**Shutter**	**Footer**
67_H	0 to FF_H	0 to FF_H	0 to FF_H	0 to FF_H	57_H

5.3.6 Take Picture Routine

This crucial piece of code initiated the taking of a Kirlian picture. This sequence of events would be initiated when the PC sends a requested in the form of the byte represented by "take_picture". In the preceding communicating sequences the Set Values and Data Dump routines would have been executed. It therefore can be reasonably

assumed that PC software has all the required environment measurements and that the appropriate instrument settings are in effect. Consequently when the request for a photograph is made by the PC, all the microcontroller has to do is enable the Timer 0 interrupt and disable it again when the required number of interrupts has occurred. This occurs in the first interrupt routine itself.

5.3.7 Analogue Data Collection Method

The method to collect analogue data accepts an argument for the analogue port to be measured. It then returns the average of six A/D Analogue to Digital conversions for this particular port. Each port is internally multiplexed. Measurements are achieved by setting and reading the appropriate SFR's namely ADM, ADCON, DAPR, and ADDAT.

5.4 Error Handling Mechanisms

If an error condition was detected, it is responsibility of "master" device, in this case, the PC to display the error conditions to the user and halt all other device communication processes. The appropriate user response, in most scenarios, would be too reset the microcontroller. This reset simply places the communicating devices in a known state.

5.5 Additional Hardware Provisions

The high voltage system requires one Digital to Analogue channel; neither the microcontroller nor the development board has a Digital to Analogue capability. This

shortcoming was addressed by using 4 Digital Output with appropriate the external analogue circuitry to produce a D to A channel that had 16 discrete levels. See Chapter 4 for details.

5.6 Conclusion

The use of a microcontroller provided an elegant solution to controlling the main instrument functions of the Kirlian photographic device. The ability to easily add more data channels is a great boon to this instrument design. With integration of the processing power of a modern PC with the technology, advantages of a microcontroller and digital camera a powerful instrument can be constructed. The duplication of other significant research efforts could be accomplished relatively easily.

CHAPTER 6

DESIGN OF THE COMPUTER PROGRAM

6.1 Introduction

No matter what the capabilities of any instrumentation, its use and ultimate acceptance may depend on the operator to effectively interface with its controls. Reliable Kirlian experimentation can be tedious on the operator as there are many aspects of the environment and of the instrument to consider [Pehek et al, 1976]. Most critics of the Kirlian research agree that reliability and stability are key performance areas when investigating this phenomenon. The software designed and developed in this dissertation addresses these key issues by having an experiment centric view of the way pictures are taken. Restated, performing of scientific experiments are facilitated by the software program produced.

6.2 Programming Requirements

A non-trivial application program called "The Kirlian Experimenter" was developed to act as interface between the user and the instrument. The program is written for the 32-bit Microsoft Windows operating systems more specifically Windows NT, Windows 2000 or Windows XP, although it should work on the other Microsoft operating system products as well. The programming language used was C++ and the development environment was Borland C++ Builder version 5. The code consists of about 3400 lines spanning 6 forms. The body of the main form is listed in Appendix B.

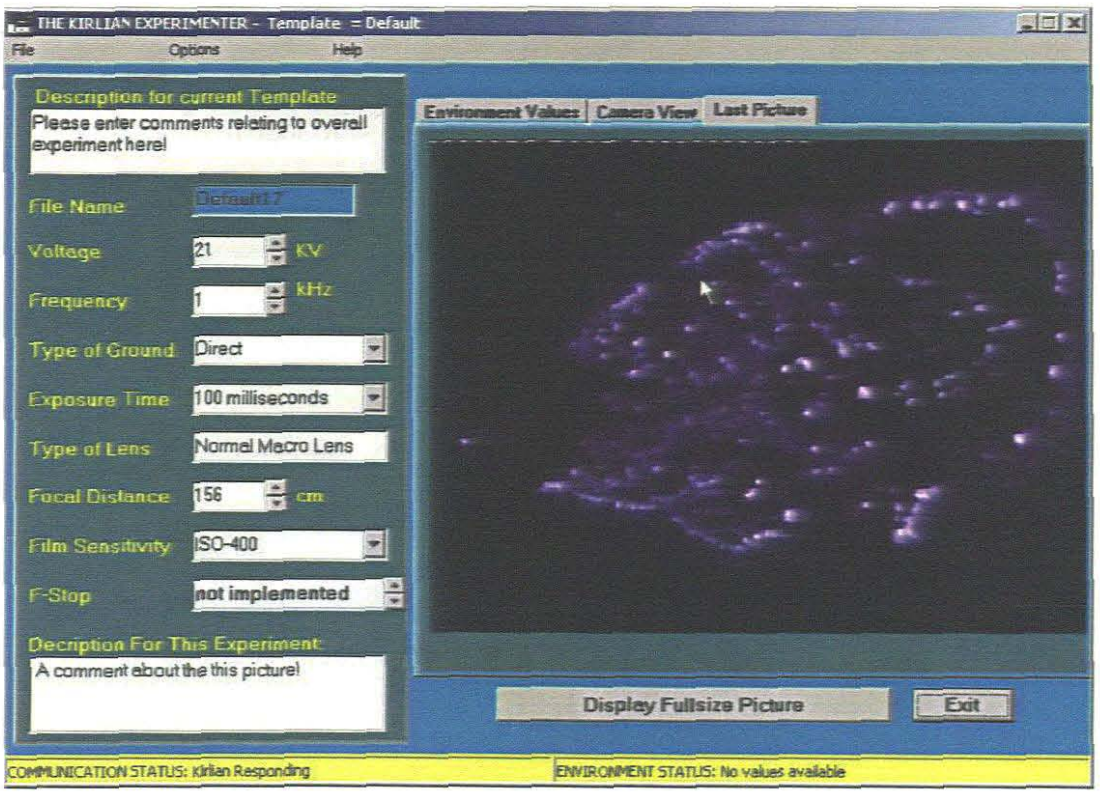


Figure 6-1: Main Windows for "The Kirlian Experimenter" Program

The primary task of the program is to facilitate the conduction of scientific experimentation into the phenomena of Kirlian photography. The software is purposely designed to act as interface between the operator/experimenter and the electronic and photographic hardware. Without this software, it will be almost impossible to test the capabilities of this hardware. The critical functions of the program are listed below:

1. To provide a means to set the voltage and frequency parameters of the High-voltage supply.
2. To collect analogue values of from the environmental subsystem.
3. To control the sequencing of events to take a Kirlian picture.

6.3 Overview of the Program Flow

The program was designed with standard Windows features to reduce the amount of learning required. The program starts with a splash picture showing a typical Kirlian picture. Thereafter an initial screen displays all the data of the first default experimental picture. Although the program interface appears daunting at first, as the main experiment features are navigable from this initial page, the interface is intuitive enough to allow rapid familiarization of the instrument and its software.

The typical steps required when taking a Kirlian picture, involve setting up the physical object of the experiment, and then using the software to take the picture. Consult section 8.2 for a detailed explanation as to the correct way to setup the subject. The software via the status bar will alert the operator whether communication with the Kirlian device is successful or not. If communication has been established the operator should select an appropriate Kirlian Experiment Template (KET) to use and modify device settings if required. The operator has then the option of either modifying the KET comment and/or comment about the picture about to be taken. The operator then selects the environment tab sheet. This tab sheet is mainly for information purposes and it is from this point that the "Take Picture" button is active. The operator has only to press "Take Picture" for the Kirlian picture of the subject to be taken. The software program then performs all necessary communication with the device hardware so that a picture is taken and downloaded to the PC. This picture will appear bounded in the "Last Picture" tab sheet. GUI buttons will be activated which allows actual or full-size picture to be viewed as required by the operator.

6.4 Offline Viewing

Offline viewing allows the operator or anyone else that has a PC to view the Kirlian pictures and its associated data. See Figure 6-2 for an example of offline viewing form. This functionality is accessed by selecting "View Previous Experiments" in the "Options" menu. There are three tab sheets in this form, one for a **Data View**, another for a **Picture View** and lastly a **Select Directory** tab sheet. All the pictures and data taken with a particular KET will be stored in its own directory and will automatically be listed the. This **Select Directory** tab sheet could have easily been called Select KET instead.

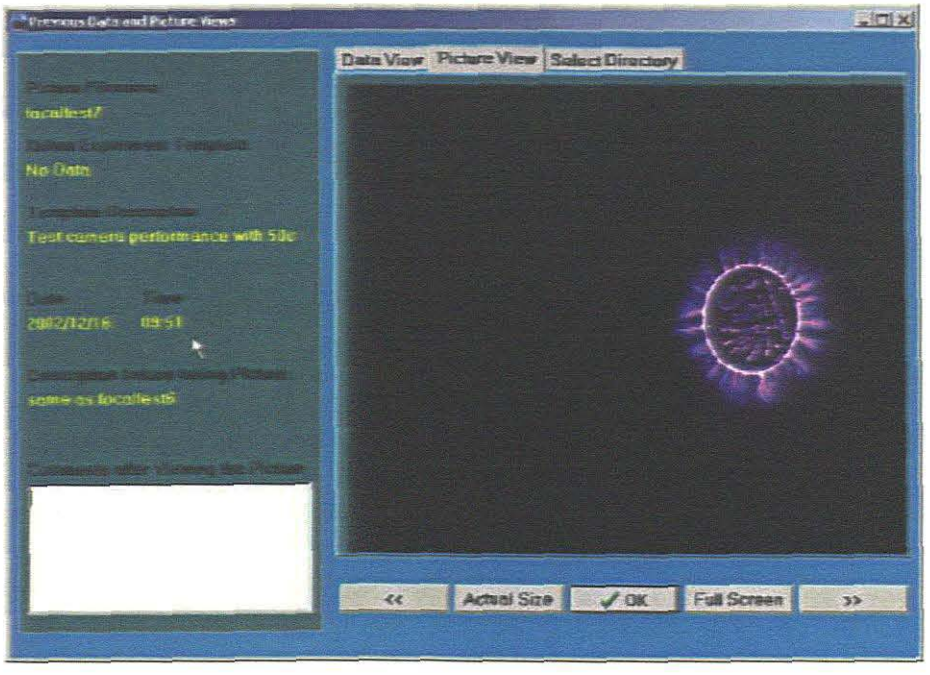


Figure 6-2: Example of Offline Viewing

6.5 Additional Operator Controls

Other options available to the operator include but are not limited to:

1. Changing the communication port (COM1 thru COM4)
2. Enable the High Voltage supply without a camera communication, Useful for debugging
3. Access Online Help
4. Access the About Box
5. Real Time camera view although Not Implemented Yet (NIY) at the time of this publication
6. Load the default KET
7. Save a new KET

There is no option to adjust the date and time via the programme itself following standard software application programming standards. It is therefore imperative that the date and time be adjusted correctly on the PC, as the program automatically obtains this information from the operating system.

6.6 Program Design

In order to achieve the required functionality and to cater for future enhancements a formal program design was attempted. Formal program design for typical GUI (Graphical User Interface) type programs are primary based on Object Orientated

principals and more specifically the Universal Modelling Language (UML) standard. The program under discussion requires real-time execution. This has a serious implication as true UML modelling would require further investigation. Therefore a sequentially based design approach was used. When analysing GUI programs in a sequential fashion it is prudent to use multiple flow diagrams to represent the various "threads" of execution. The flow charts used for this design are documented in Figure 6-4 through Figure 6-12

From an Object Orientated view, the base class for this program is the Kirlian Experiment itself with its associated data and methods. Thus, a class KDataClass1 was created to perform this requirement.

Table 6-1: Data and Data-Types Stored in KDataClass1

Data Encapsulated in the KDataClass1	**Type of Data**
Date and Time of the experiment	AnsiString
Comments about the Experiment	AnsiString
Comments about the Picture	AnsiString
Base File Name for the data and picture	int
Applied Voltage	int
Frequency of the supply	int
Type of Ground	int
Exposure Time	int
Type of Lens	AnsiString
Focal Distance	int
F-Stop or Aperture Setting	int
Measured Ambient Temperature and the lower and upper thresh holds	int
Measured Atmospheric Pressure and the lower and upper thresh holds	int
Measured Ambient Light Level and the lower and upper thresh holds	int
Measured Humidity and the lower and upper thresh holds	int
Measured Object Pressure and the lower and upper thresh holds	int

Following standard Object Orientated design approach, each datum has its own accessor and mutator methods. The code required for this functionality is listed in Appendix B.

Standard programming practises was applied throughout this application. Although using a GUI development package such as Borland Builder meant that standard Windows programming issues were taken care of, there were a numerous of thorny programming problems to overcome. The first of these problems was setting-up reliable communication via the serial port. Secondly, the real-time nature of the program meant that synchronizing the communication of events required a communication strategy implementing packet data transfer see Chapter 5 for a more complete discussion on this topic. Lastly, communicating with the digital camera via the USB proved extremely taxing programming issue, at first a partial solution was found through spawning an external program, which interfaced with the camera [Menchenin, 2003]. Ultimately, an elegant solution was achieved by using the Olympus SDK [Olympus 2003].

6.7 Program Versions

Anecdotal observation suggests that most complex software projects are never truly complete as there are always enhancements and bug fixes. This project is no exception. The current version of the "The Kirlian Experimenter" is 1.27. This information and direct access to latest software release is available via the About Box. There are no known serious bugs to report on this version.

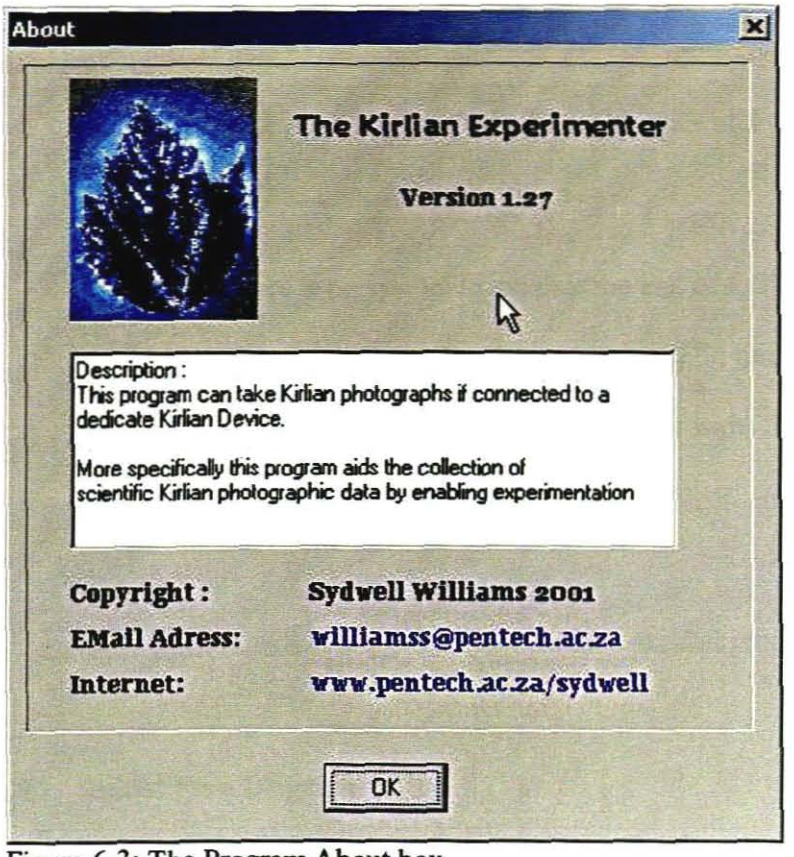


Figure 6-3: The Program About box

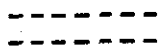
6.8 Flowcharts of the Main Form

This section lists the flow diagrams of the main form. These diagrams follow conventional flowcharting techniques. Most of the code of the Main Form is in methods. These methods are number in the flow chart and the code itself. See Appendix B. Main this form is called **K2001Main_f.**

The following symbols are used expanded the traditional flow charts information.

This symbol is used to indicate the start of a GUI event/thread

There is no symbol to indicate the end of a GUI event/thread. This was done to keep the picture sizes with in limits. At the end of a thread or an event the program simply waits for the next GUI event or Poll routine timeout.



This symbol is used to indicate a parallel process

6.8.1 Initial Flow Diagram

Methods 1.1 to 1.3 create, activate and initialize the main form. See Figure 6-4 the flow chart. Many initial GUI values are set in the original form designer of the IDE, although some setting which material affect the initial state of the displayed program are placed in the method 1.4 **HouseKeeping**. The default experiment template is also loaded from the file "default.ket". The page control method 1.5 **Main_pagectlChange** determines whether the last picture or the current environmental values are displayed. Currently the third page tab is not operational this should display the current camera view.

6.8.2 Picture Taking Sequencing Details

Method 1.7 (**TakePicture_butClick**) handles the code required to take a picture. See Figure 6-5. The actual code to be executed depends on which page is displayed. If the

page control displays the last picture then pressing the Take Picture button will display true size version of the last picture taken. On the other hand if the environment variables page is displayed then further preparation for taking a picture continues. The camera communication is checked and if it is ok, the camera settings are activated. A global variable (**PictureRequested**) is set so that when the poll routine knows a picture was requested see section 6.8.3. The method then waits until the picture is taken and then the download of the picture is commenced. The picture is then displayed in the main page control window.

Method 1.8 **(getKDataFilename)** is called to automatically rename the next picture to be taken. This code then takes the KET and the next filename in sequence from 001 to 999. Method 1.9 **(DisplayKInfo)** is also called redisplay this data.

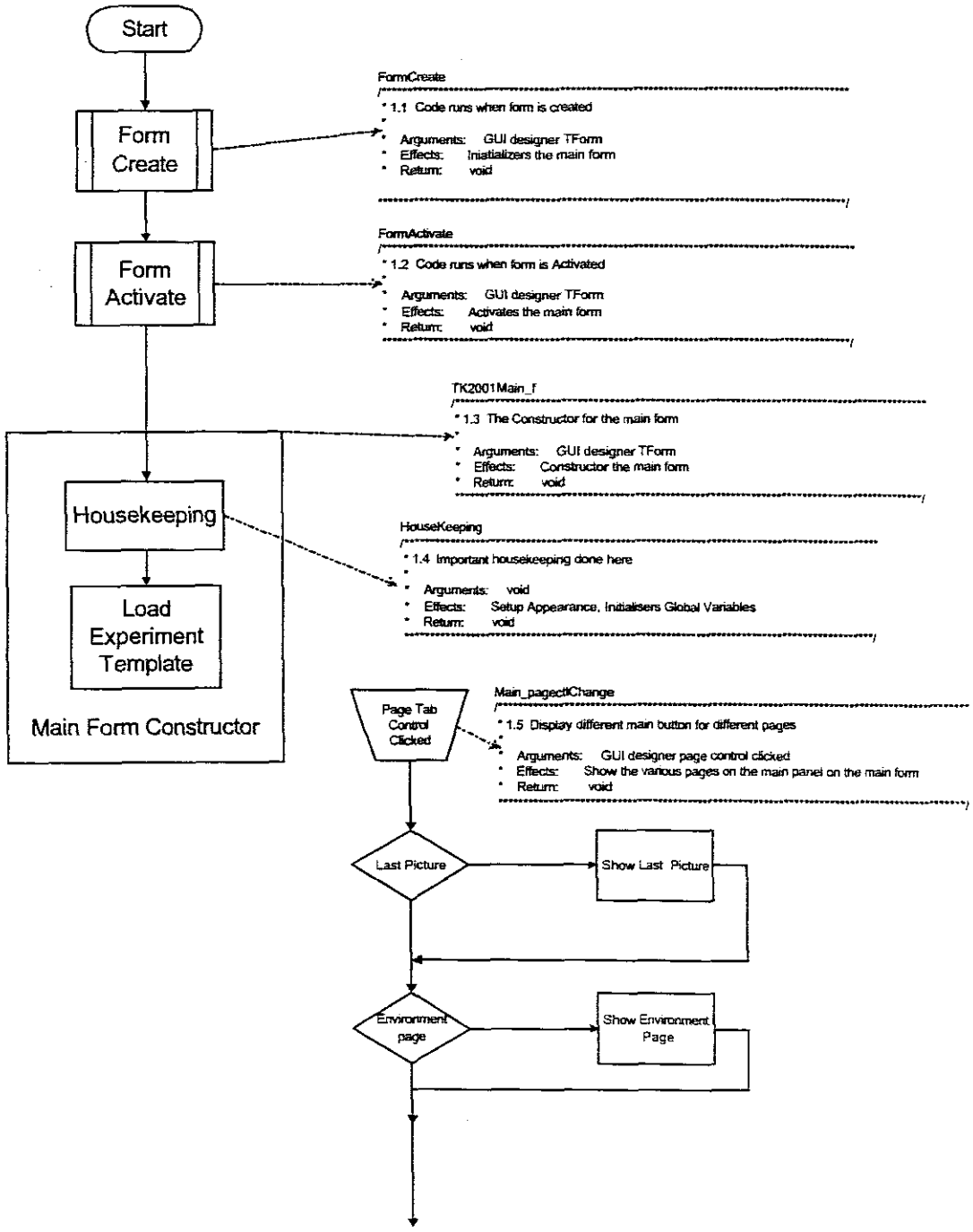


Figure 6-4: Initial Flow Diagram

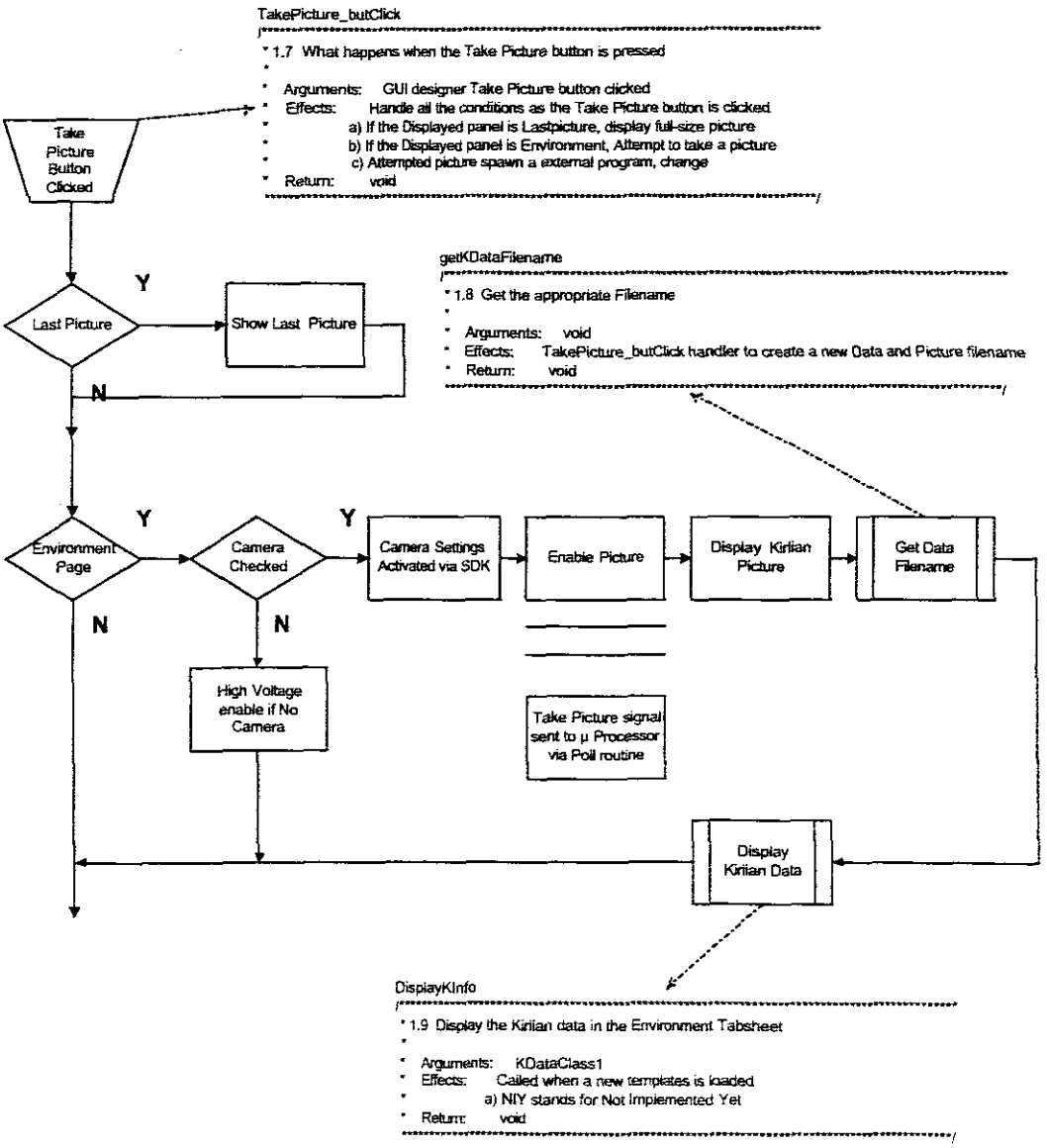


Figure 6-5: Picture Taking Sequencing

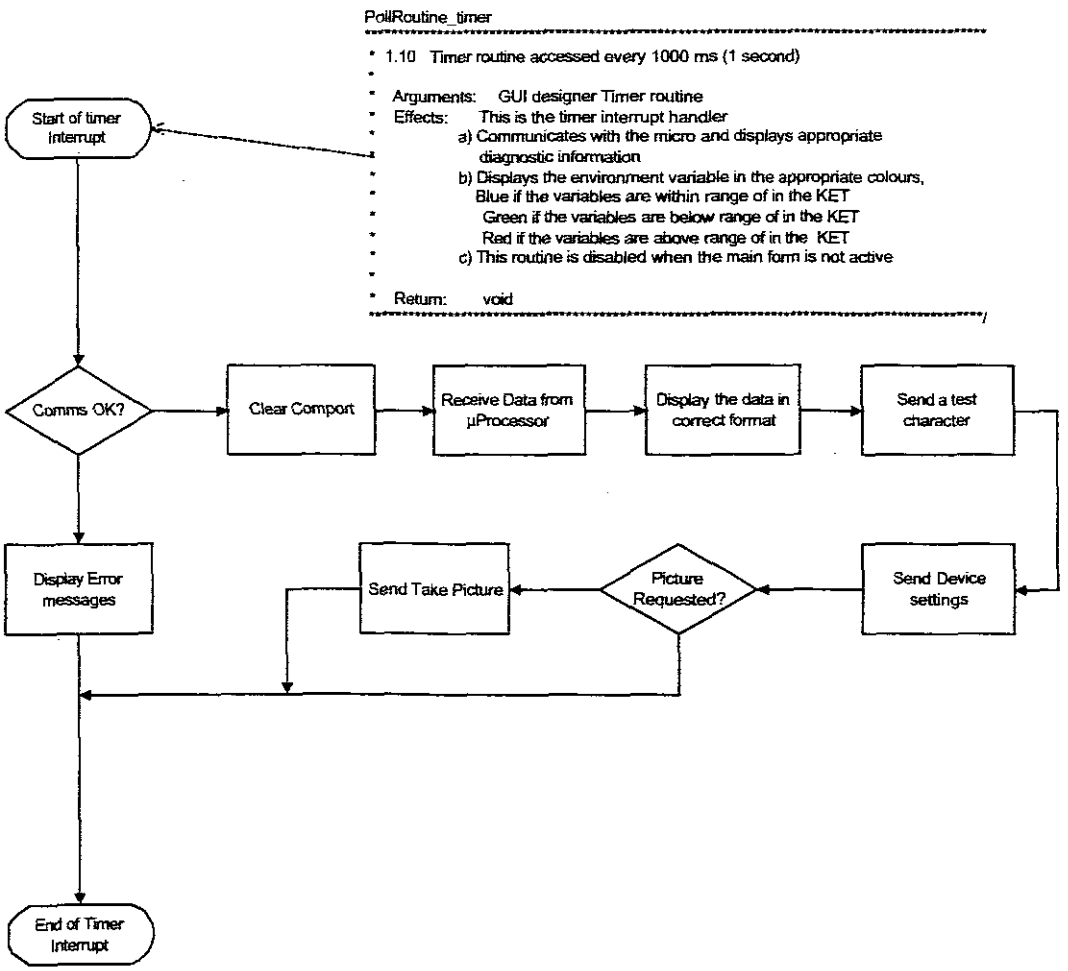


Figure 6-6: Poll Routine Flow Diagram

6.8.3 Poll Routine Details

The Polling Routine 1.10 checks RS232 comport and depending on whether there is an error or not, the environment variables are displayed or an error condition. See Figure 6-6. This routine is responsible for displaying the alarm condition as well as sending the take picture byte to the microcontroller if the global variable (**PictureRequested**) is set.

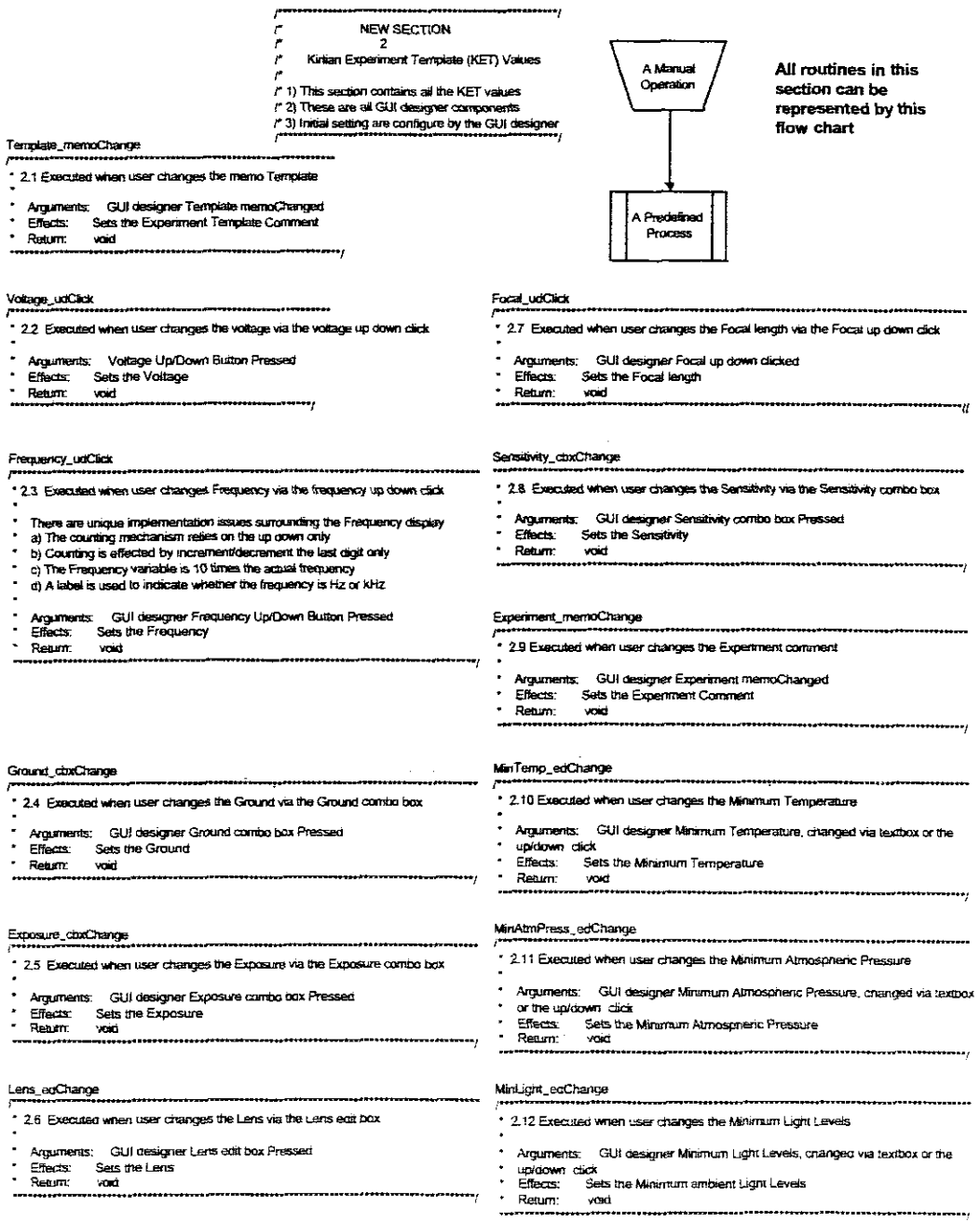


Figure 6-7: Kirlian Experiment Template (KET) Implementation

6.8.4 Kirlian Experiment Template (KET) Implementation

These routines are relatively straightforward all KET values whether they are text or integer are simply edit as the experimenter request it. All these items are listed in Table 6-1. Also see Figure 6-7 and Figure 6-8 for the relevant flow charts.

6.8.5 User Dialogue Boxes Details

Standard Windows user-interfacing techniques were employed, two examples are the KET can be loaded or saved under the file menu item and right clicking the status bar allows access to the communications port setting. The Figure 6-9 shows relevant method names and numbers used.

6.8.6 Serial Port Details

These routines allow the user to set the primary communication, which interfaces to the RS232 port. This allows any of the standard serial ports to be used namely COM1 thru COM4. Method 3.10 **(SetCommPort)** first initializes the port settings such as the Actual Comm. port to be used, Baud rate, Parity and Stop bits if any. Appropriate errors are flagged giving a visual indication to the user.

Method 3.9 (GetCommPort) simply returns the active Comm. setting while the Serial Port menu option, Method 3.6 **(SerialPort2Click)** displays a radio button form, which allows the user to select a new port.

```
/*****************************************************/
/*                NEW SECTION                        */
/*                  2  Continued                     */
/*****************************************************/

MinHumid_edChange
/*****************************************************************************
* 2.13 Executed when user changes the Minimum Humidity
*
* Arguments:    GUI designer Minimum Humidity, changed via textbox or the
* up/down click
* Effects:      Sets the Minimum Humidity
* Return:       void
*****************************************************************************/

MinObjPress_edChange
/*****************************************************************************
* 2.14 Executed when user changes the Minimum Object Pressure
*
* Arguments:    GUI designer Minimum Object Pressure, changed via textbox or
* the up/down click
* Effects:      Sets the Minimum Object Pressure
* Return:       void
*****************************************************************************/

MaxTemp_edChange
/*****************************************************************************
* 2.15 Executed when user changes the Maximum Temperature
*
* Arguments:    GUI designer Maximum Temperature, changed via textbox or the
* up/down click
* Effects:      Sets the Maximum Temperature
* Return:       void
*****************************************************************************/

MaxAtmPress_edChange
/*****************************************************************************
* 2.16 Executed when user changes the Maximum Atmospheric Pressure
*
* Arguments:    GUI designer Maximum Atmospheric Pressure, changed via textbox
* or the up/down click
* Effects:      Sets the Maximum Atmospheric Pressure
* Return:       void
*****************************************************************************/

MaxLight_edChange
/*****************************************************************************
* 2.17 Executed when user changes the Maximum Light Levels
*
* Arguments:    GUI designer Maximum Light Levels, changed via textbox or the
* up/down click
* Effects:      Sets the Maximum Light Levels
* Return:       void
*****************************************************************************/

MaxHumid_edChange
*****************************************************************************
* 2.18 Executed when user changes the Maximum Humidity
*
* Arguments:    GUI designer Maximum Humidity, changed via textbox or the
* up/down click
* Effects:      Sets the Maximum Humidity
* Return:       void
*****************************************************************************/

MaxObjPress_edChange
/*****************************************************************************
* 2.19 Executed when user changes the Maximum Object Pressure
*
* Arguments:    GUI designer Maximum Object Pressure, changed via textbox or the
* up/down click
* Effects:      Sets the Maximum Object Pressure
* Return:       void
*****************************************************************************/
```

Figure 6-8: KET Implementation Continued

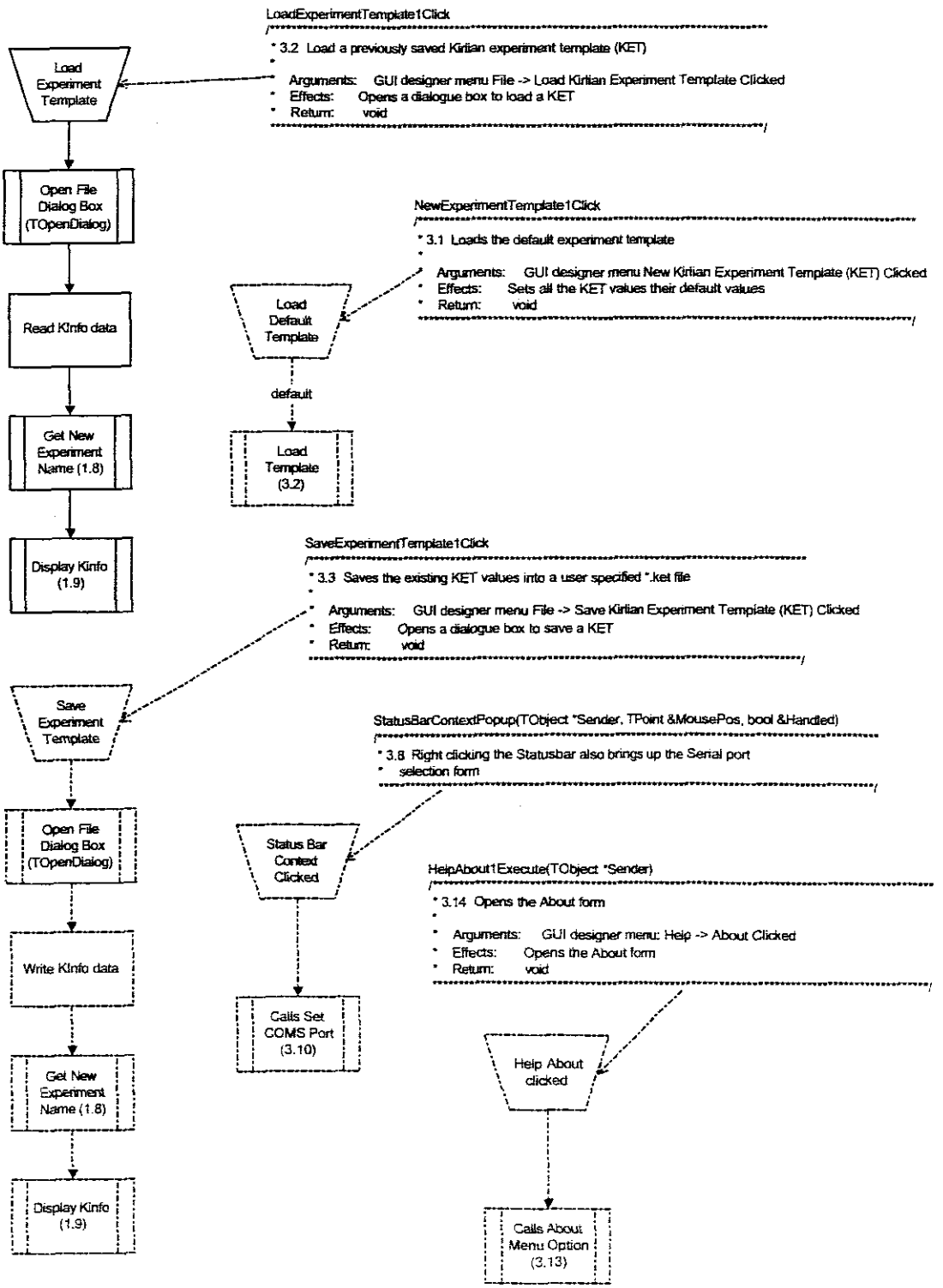


Figure 6-9: User Dialog Flow Charts

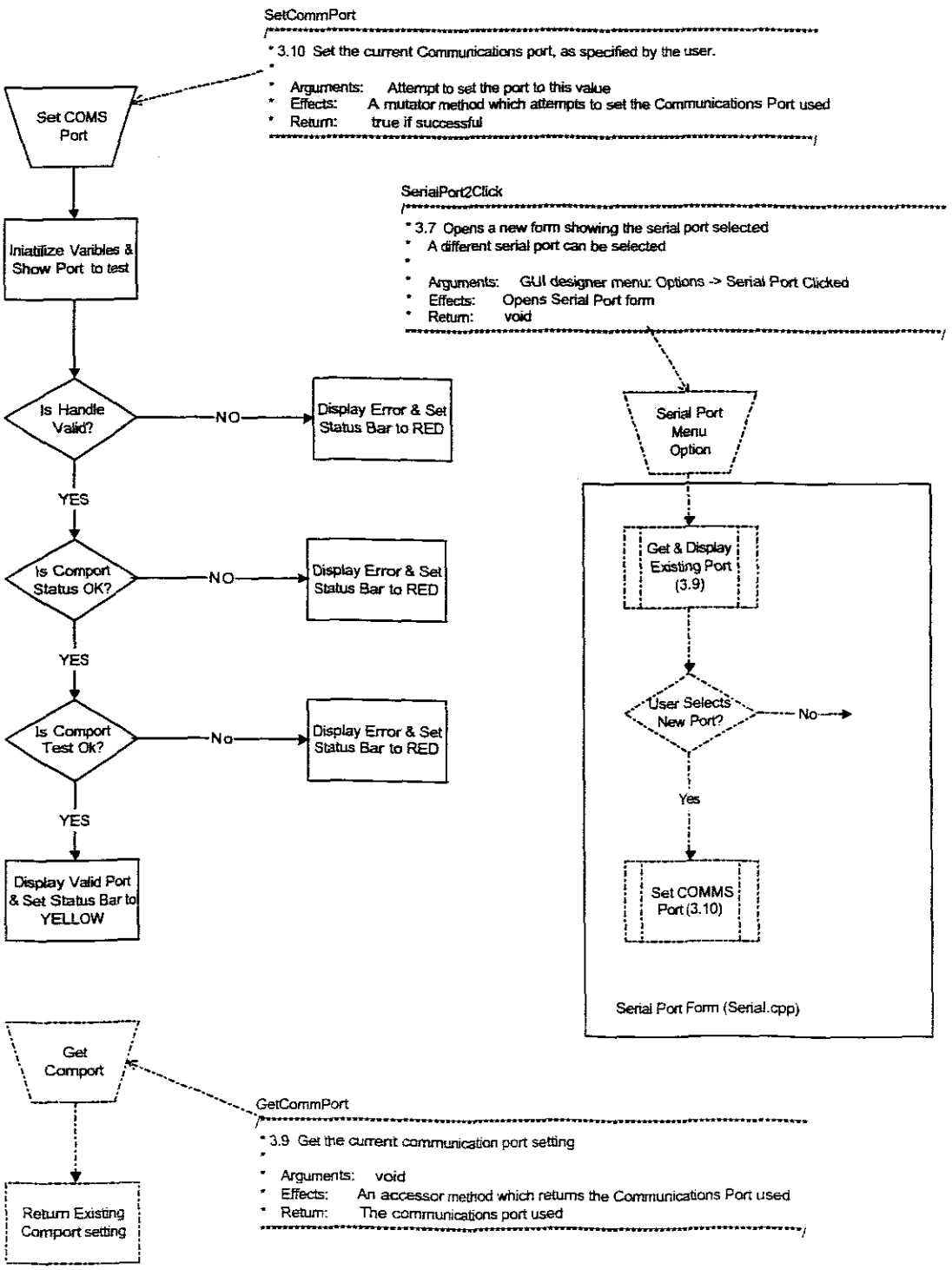


Figure 6-10: Serial Port Flow Diagram

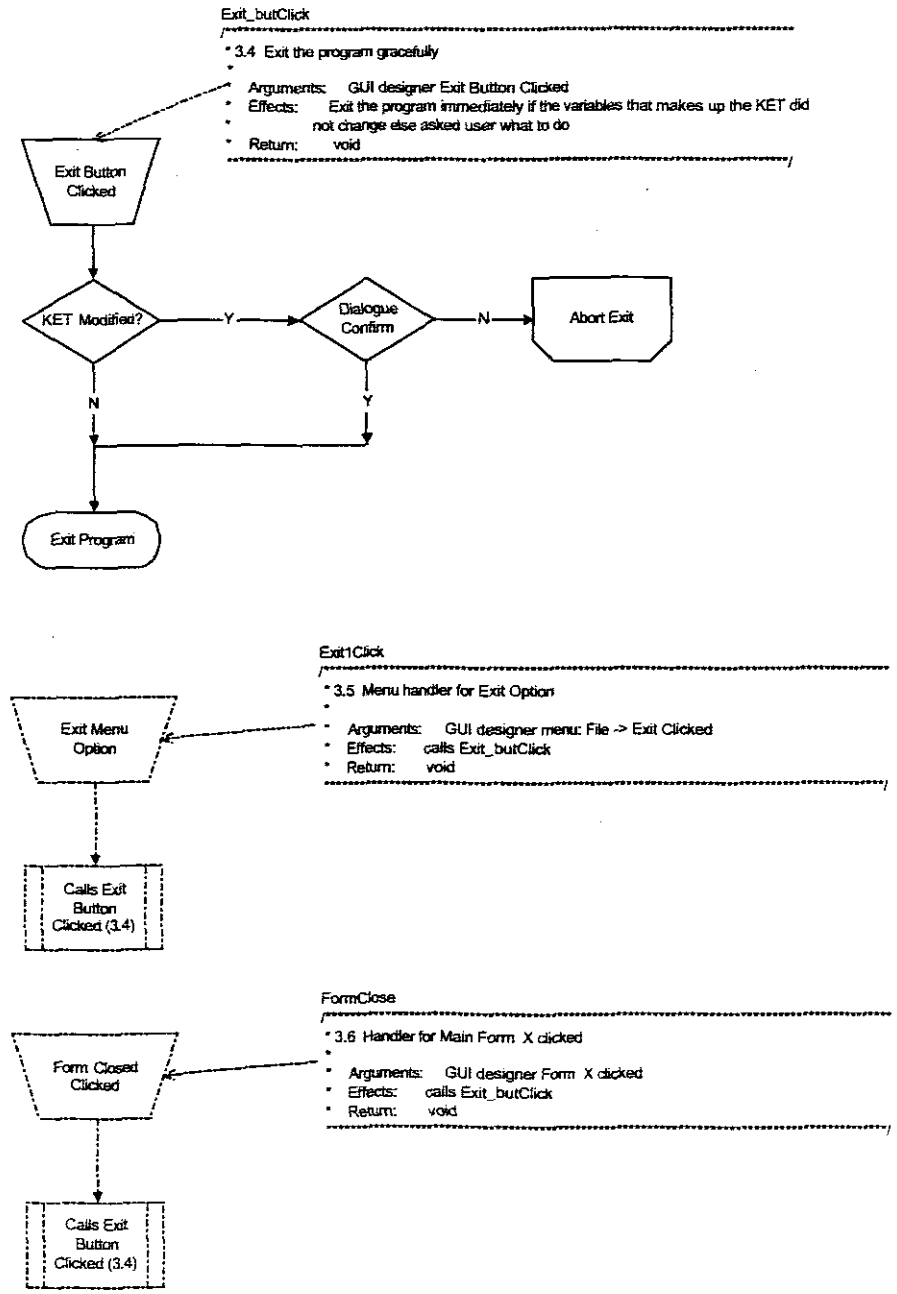


Figure 6-11: Exit Routines Flow Diagrams

6.8.7 Exit Routines Details

If the user modifies a loaded KET the user has an opportunity to cancel the exit command and then save the KET. The exit strategy employed is shown in Figure 6-11.

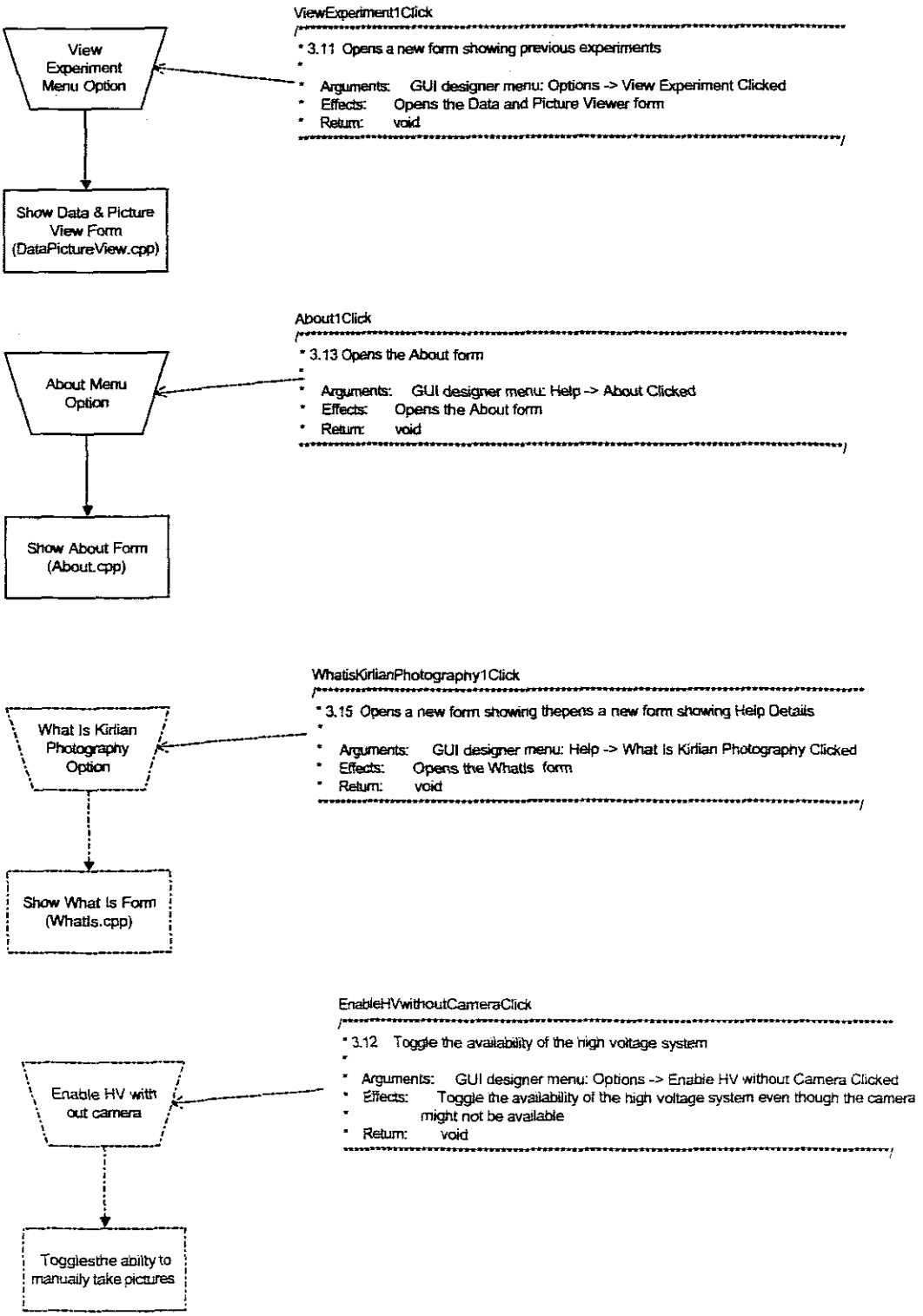


Figure 6-12: Menu Options Flow Diagrams

6.8.8 Menu Options Details

Other miscellaneous methods include the View Experiment option, method 3.11 **(ViewExperimentClick)** which opens a new form allowing the user to view previous experiments see the section 6.4 on Offline Viewing. The about box is activated through method 3.13 **(About1Click)** which opens a new form. Online help should be activated through the method 3.15 **(WhatisKirlianPhotography1Click)** this method is not implemented in this current release of the software version 1.27.

Method 3.12 **(EnableHVwithoutCameraClick)** is used during debugging as the name suggests it will enable the High Voltage output even if there is no digital camera connected. This is useful when testing other camera systems. These flow charts are listed in Figure 6-12.

6.9 Conclusion

The software engaged with this device, has been developed from a reasonably elaborate coding task. The final success of this project will depend on whether this code and its experimenter centric view of the instrumentation can be relied upon to produce repeated successful Kirlian photographs.

CHAPTER 7

INTEGRATING THE COMPONENTS

7.1 Introduction

The different engineering systems were integrated in a cohesive unit. In order to be an effective instrument a number of design issues were addressed. The completed instrument is shown below in Figure 7-1. General component interconnection is discussed in section 7.2. The mechanical construction is detailed in section 7.3. The power supplies and their construction are revealed in section 7.4.

Figure 7-1: Completed Kirlian Instrument Housing

7.2 Component Interconnection

A brief technical description of the system as a whole is outlined in this section. A block diagram of the instrument shows the various sub-sections that need integration (see Figure 7-2). Starting at the bottom of the diagram and working our way up in the bottom left we have the environmental sensors. The sensor signals are conditioned by supporting circuitry to provide outputs in the range 0 to 5 Volts (section 2.3). These signals are then feed in to the Digital to Analogue port of the microprocessor (section 5.1).

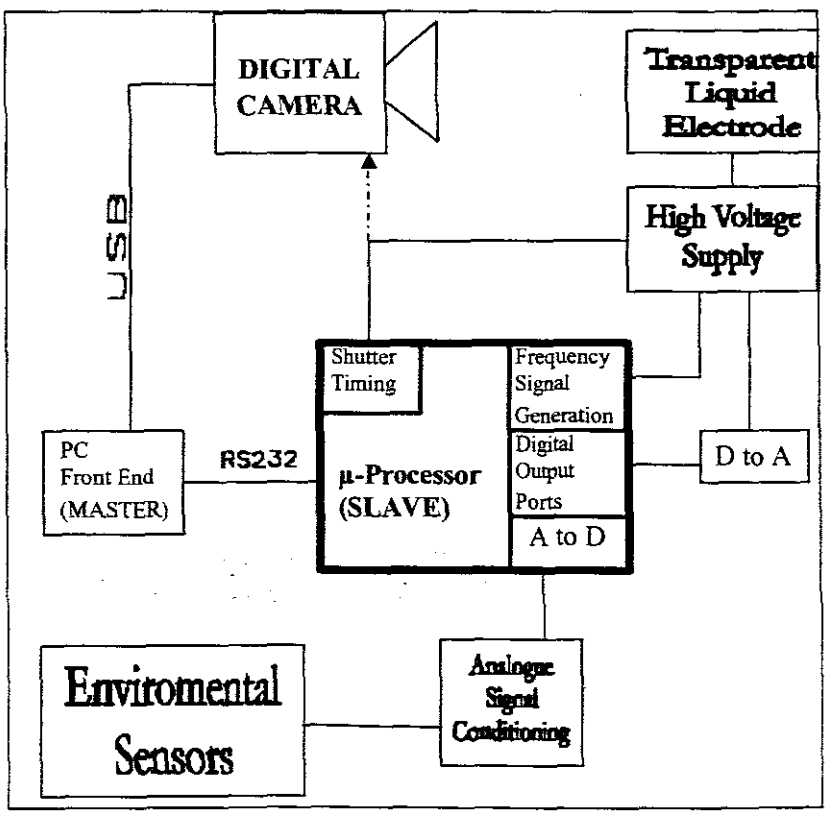


Figure 7-2: System Block Diagram

The microcontroller generates a square waves in response to the operator requested frequency value. This requested value and others comes from the PC program via the RS232 port. The square waves are ultimately used to generate the production of the high voltage source. This high voltage source requires galvanic isolation which provided by the opto-isolators (see section 4.6). The duty cycle of this square wave is adjustable and this in turn modifies the amplitude of the high voltage output.

This duty cycle input originates from the microprocessors digital output port which switches in the appropriate resistor according to required duty cycle and thus ultimately the high voltage output. In addition to communicating with the microcontroller the PC is required to communicate directly to the digital camera. This is achieved via a USB port and cable (see section 6.8.2). When using the Olympus Camedia C-4040 Zoom camera (see section 3.5.2). The software program developed provides full control of most camera settings.

7.3 Mechanical Construction

An important design feature of this Kirlian Device is that it is self-contained. All the optical mechanical and electrical systems are housed light tight box constructed out of wood. This has great benefits for the investigator, as there is no need to work in a darkroom. It has the additional feature of a light tight sleeve so that Kirlian pictures of human hands and/or fingers can easily been taken. The box was made out a standard laminated wood shelving available in hardware stores. Wood was chosen as the primary construction medium as it is relatively inexpensive and easy to work with. Most of the electronic circuitry was constructed on a popular prototyping strip board.

Interconnections of the boards were made with hook-up wire and standard keyed plugs and sockets. Figure 7-3 shows an internal view of the device with most of the electronic circuitry exposed.

Figure 7-3: Internal Picture of the Kirlian Device

The design features include a separate area for electronics, interchangeable camera mountings and easy accessible compartments of the TLE and camera. All accessible compartments are closed by means of canopy clips. The entire box is painted matt black to aid the expulsion external light sources.

Different cameras can be used in the box these include standard SLR cameras, digital cameras or even a video camera, although the latest "Kirlian Experimenter software" only supports the Camedia range of cameras by Olympus.

The Front panel of the Device contains a minimal number of switches and connectors. These are shown in the Figure 7.4.

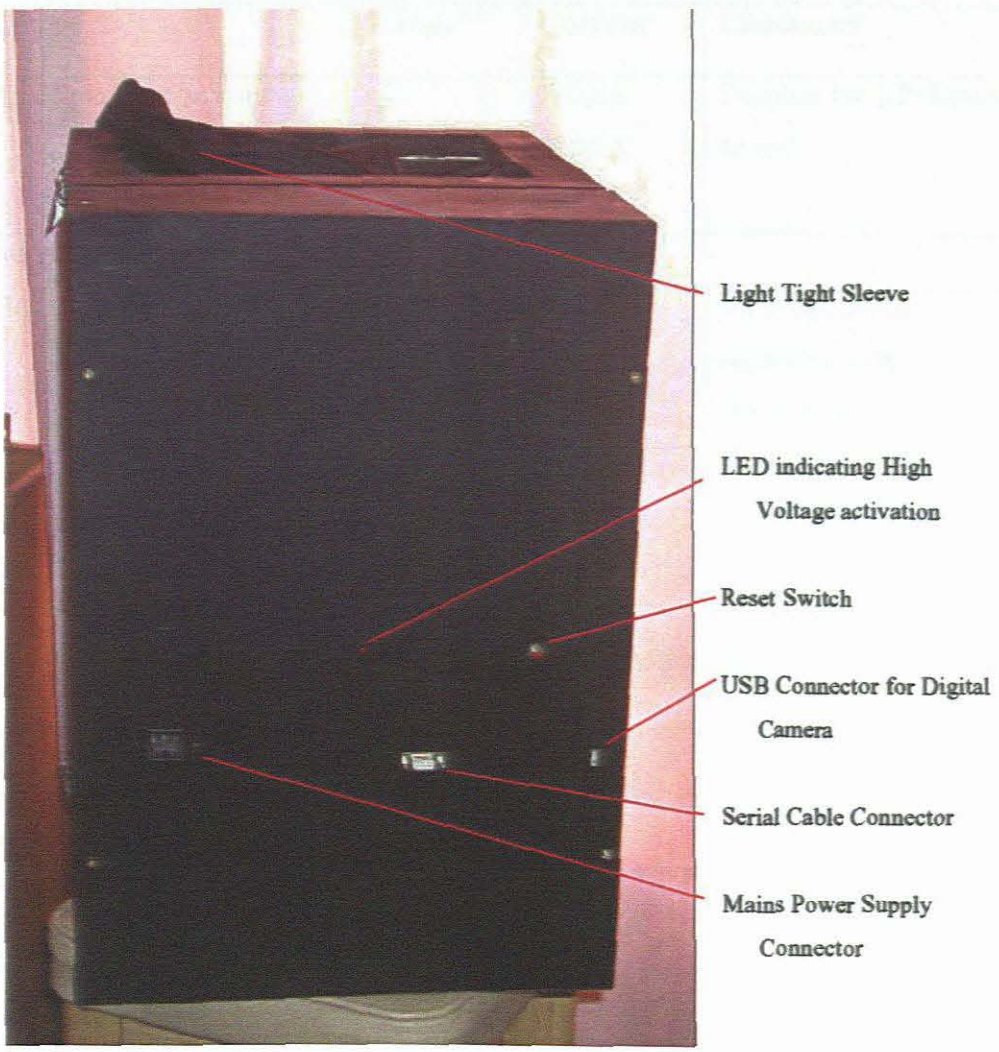


Figure 7-4: Front Panel Connections

7.4 Power Supplies

Standard linear power supply techniques were used to provide DC power to the various circuits. The table below details the voltages and currents provided for:

Table 7-1: Power Supplies used by the Device

Circuit	Voltage	Current	Comments
Analogue Data Acquisition	+5V +12V -12V	700mA 500mA 500mA	Supplies the µProcessor as well
Microcontroller Board	+5V	300mA	
High Voltage circuitry	+14V +20V	500mA 3A	Microprocessor regulated Supply (Section 4.5)
Digital camera	4 x 1.2V	1800mAh	Nickel Metal Hydride Batteries (NiMH)

Note although it would be a trivial task of providing DC power to the digital camera, it was decided not to do so for two main reasons. Firstly, the power supply requirements vary greatly between different digital cameras manufactures. Secondly, the warrantee of new digital camera does not permit the use of third-party power supplies.

7.5 Data Acquisition Power Supply

Basic supply generation techniques were used in designing data acquisition power supply. A dual complementary full-wave bridge topology was employed (See Figure 7.4). The RMS current rating of the transformer using this topology is 1.8 × DC current

requirements. The VA rating required by the mains transformer is equal to the sum of the three output power requirements.

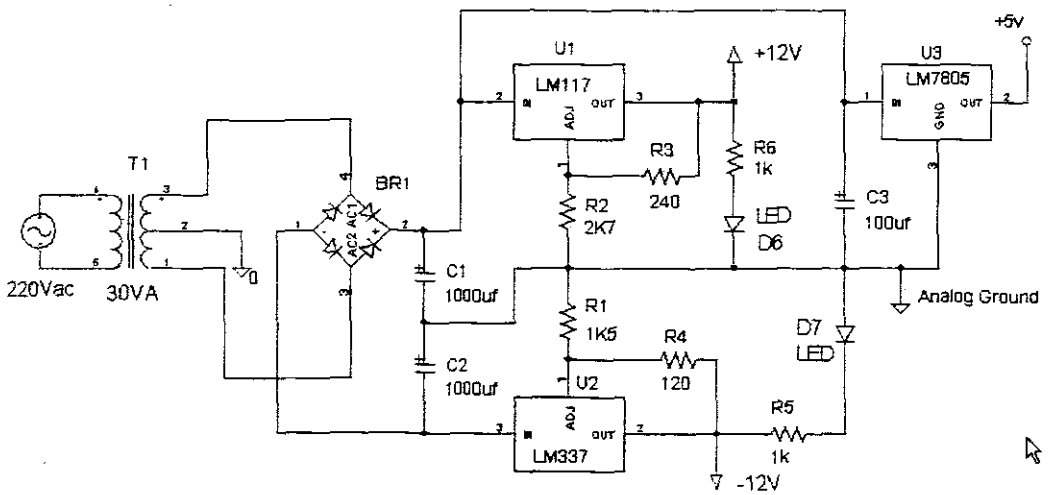


Figure 7-5: Data Acquisition and Microprocessor Power Supply

$$VA_T = [(+15V)Power + (-15V)Power + (5V)Power] \times 1.8 \qquad (7.1)$$

where: VA_T is the total power ratting for the transformer

$$VA_T = [6VA + 6VA + 3.5VA] \times 1.8 \qquad (7.2)$$

$$VA_T = 27.9VA \qquad (7.3)$$

A centre-tapped 30VA transformer (T1) with secondary winding of 17 volts, 0 volts, and 17 volts was used. A bridge rectifier (BR1) rated at 1A supplied the three 1000μF filter capacitors (C1 & C2 & C3). Empirical formula (7.4) relating peak-to-peak

ripple voltages Vp-p to capacitor C size and load current IL is given by the following formula [Polen, 1981].

$$C = \frac{I_L}{V_{p-p}} \times 7.2 \times 10^{-3} \qquad (7.4)$$

With this arrangement the maximum ripple voltage for the positive regulators at maximum load IL = 1.2A is 1.8V. The minimum transformer secondary voltage is given by this formula.

$$V_{ac} = \frac{\left(V_{out} + V_{reg} + V_{rect} + V_{ripple}\right)}{0.92} \times \frac{1}{\sqrt{2}} = 15.4\text{V} \qquad (7.5)$$

where: 0.92 = rectifier efficiency (typical)
Vac = Transformer secondary Output
Vout = Required output voltage = 15V
Vreg = required voltage differential of the regulator = 3V
Vrect = Voltage differential across the diode = 2 × 0.6V = 1.2V
Vripple = Ripple voltage = 1.8V

Note this equation does not cater for low line AC conditions. A transformer with a secondary voltage of 17AC was used. The positive & negative 15V regulators are variable output voltage regulators. The outputs are governed by the following equations. For positive voltages:

$$V_{out} = 1.25V\left(1 + \frac{R_3}{R_2}\right) \qquad (7.6)$$

where: R2 must be 240
R3 is selected to be 2K7

Resulting output = Vout = 15.3V

$$-V_{out} = 1.25V\left(1 + \frac{R_1}{R_4}\right) \qquad (7.7)$$

where: R4 must be 120O

R1 is selected to be 1K3

Resulting output = Vout = -14.8V

A fixed 5-volt IC regulator was used to generate the 5 volts required by the microprocessor. LED's (D6 & D7) was used as indicators on the power supply board.

Standard electronic and mechanical solutions were employed to integrate the various systems to a convenient instrument.

CHAPTER 8

PERFORMANCE

8.1 Introduction

As with any real world application of technology the proof that an instrument or system is effective is largely determined by it is operational efficiency. This chapter will demonstrate the effectiveness of this instrument as a potential research tool. Important aspects of the performance of this device are detailed in this chapter. Basic operation is detailed section 8.2. Image quality is discussed in section 8.3. A real world experiment is performed in section 8.4. Safety and other aspects relating to human subjects are discussed in section 8.5.

8.2 Experiment Walkthrough

A typical experiment would start by selecting an appropriate subject and possibly the experiment variables to modify. For this example, we selected a coin.

1. Place the coin the TLE and visually align it with the camera lens.
2. Connect the grounding electrode to the coin.
3. Close the lid of the Kirlian experimenter box.
4. Start the application if it has not already been started.

5. The experimenter then has the option of loading a previous KET or creating a new one.
    a. To load a new Kirlian Experiment Template (KET) click on **file** and then **load** and select the KET required.
    b. To create a KET by modifying the all the experiment conditions as per experiment design and then **save** these settings as a new KET.
6. Make sure the environment page is displayed by clicking on the environment tab.
7. Comment on the picture that is about to be taking a the text box provided
8. If all the setting are satisfactory simply press the **Take Picture** button

After a few seconds, depending on the exposure time selected, the Kirlian photograph will be displayed. This photograph together with its measured and selected data will automatically be saved in a folder bearing the name of the KET. Automatic naming is employed so that the operator does not accidentally overwrite an existing experiment's data.

To continue the experiment, modify a setting as per experiment design, then optionally add a comment, select the environment page and then press the Take Picture button. A new photograph is produced.

Standard interface buttons are provided to view the completed pictures in a **Full screen** or **Actual** size Windows. These buttons are available in the Offline viewing mode as well.

8.3 Image Quality

The software uses place the camera in JPEG mode although if software image processing is required a propriety raw format is available. The JPEG mode was chosen because the compression used reduces the file size considerably without noticeable image quality degradation. The small file sizes means downloads from the camera to the PC is relatively short. The 4.1 mega pixel, C4040Zoom camera from Olympus provides excellent detailed pictures. There is however, room for improvement and a better quality camera can be used. The most stringent requirement for a different camera would be that it is controllable via software, preferably through an USB interface.

The actual image size is 2272 by 1704 pixels. This is considerably more than most desktop PC settings thus, a means for viewing either the full screen image or the actual size image was introduced. Standard setting of 72 DPI is used for both horizontal and vertical resolutions.

8.4 Real World Experimentation

The device performs admirably as the following sequence of seven photographs with its associated data show. This was first of a series of sample experiments produced with most of the functional elements device in working order. There are a number of points of interest. Firstly, the time span of these pictures was a little over 20 minutes. The first was taken at 21:19 and the last at 21:40, in 21 minutes six successful photographs were taken all with immediate

feedback. The photographs show an increase of corona discharge as the frequency is incrementally decreased. This increase of corona is most like due to the frequency of the device approaching the resonant frequency of the leaf and TLE combination.

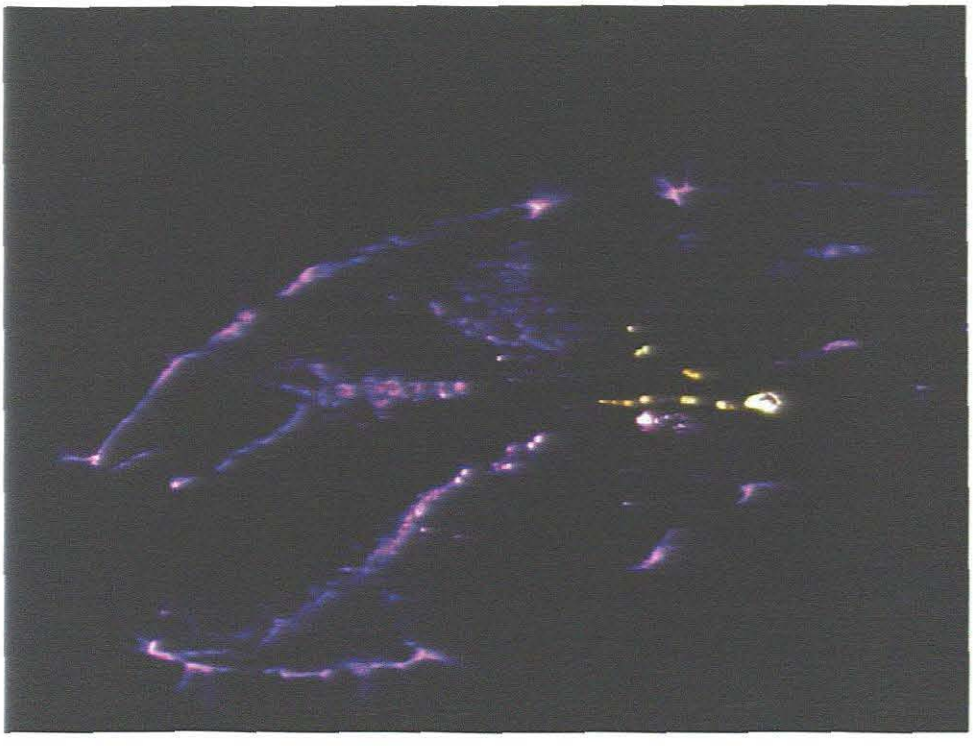


Figure 8-1: Kirlian photograph of experiment "TestHighVoltages6"

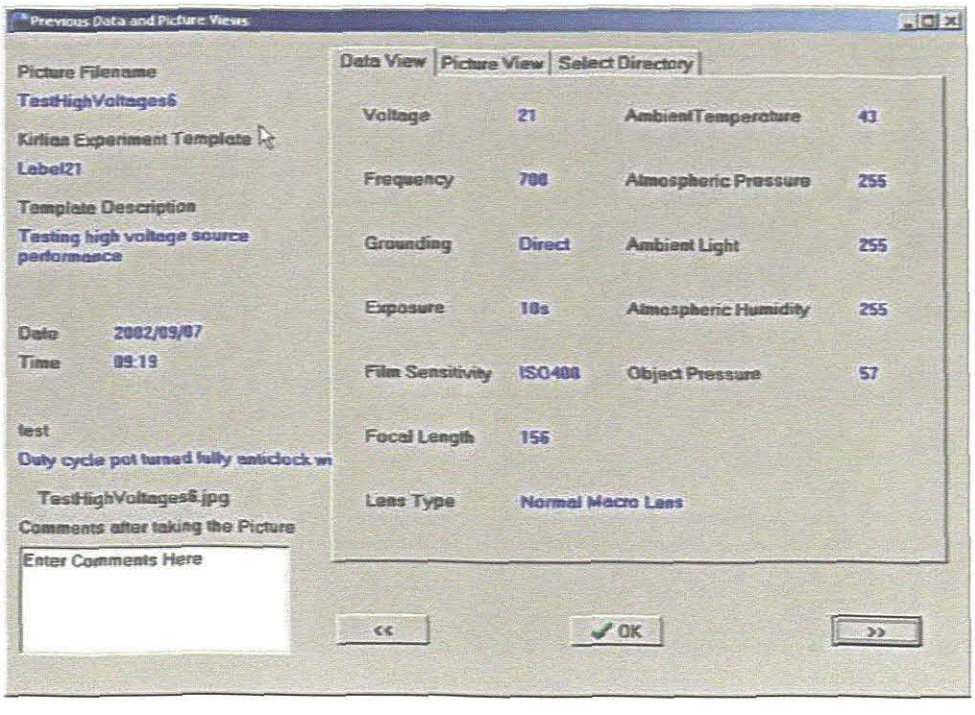


Figure 8-2: Data collected for experiment "TestHighVoltages6"

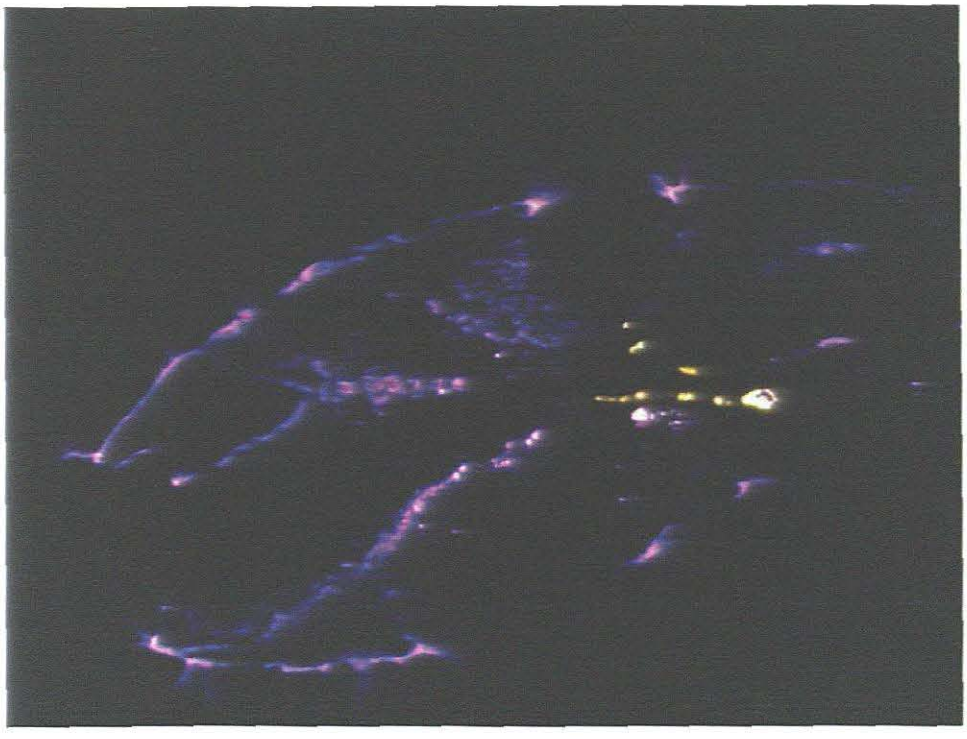


Figure 8-3: Kirlian photograph of experiment "TestHighVoltages7"

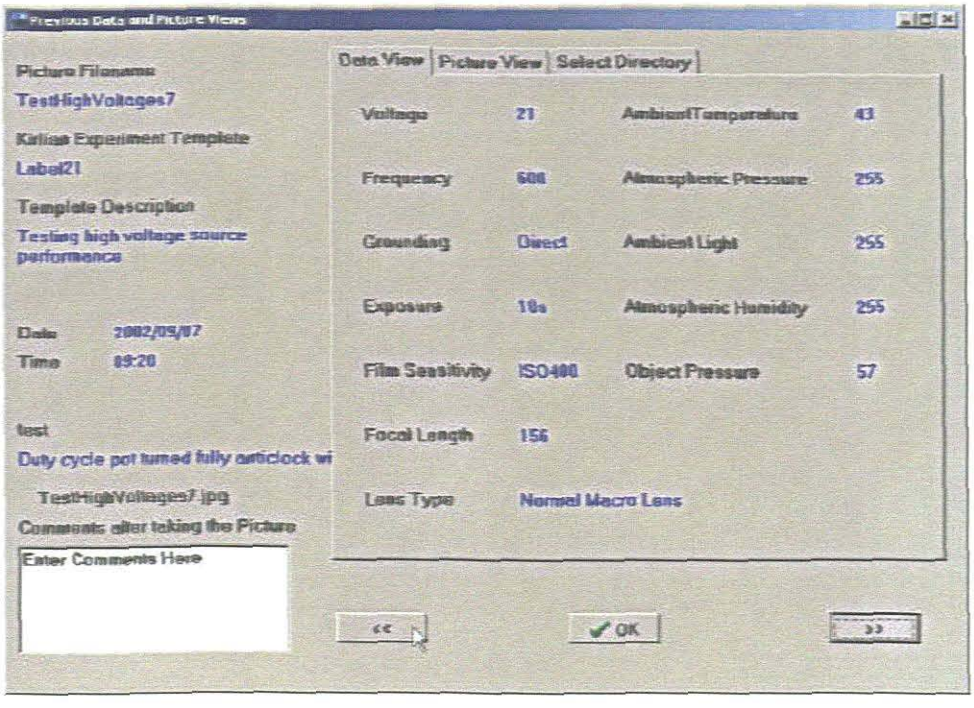


Figure 8-4: Data collected for experiment for "TestHighVoltages7"

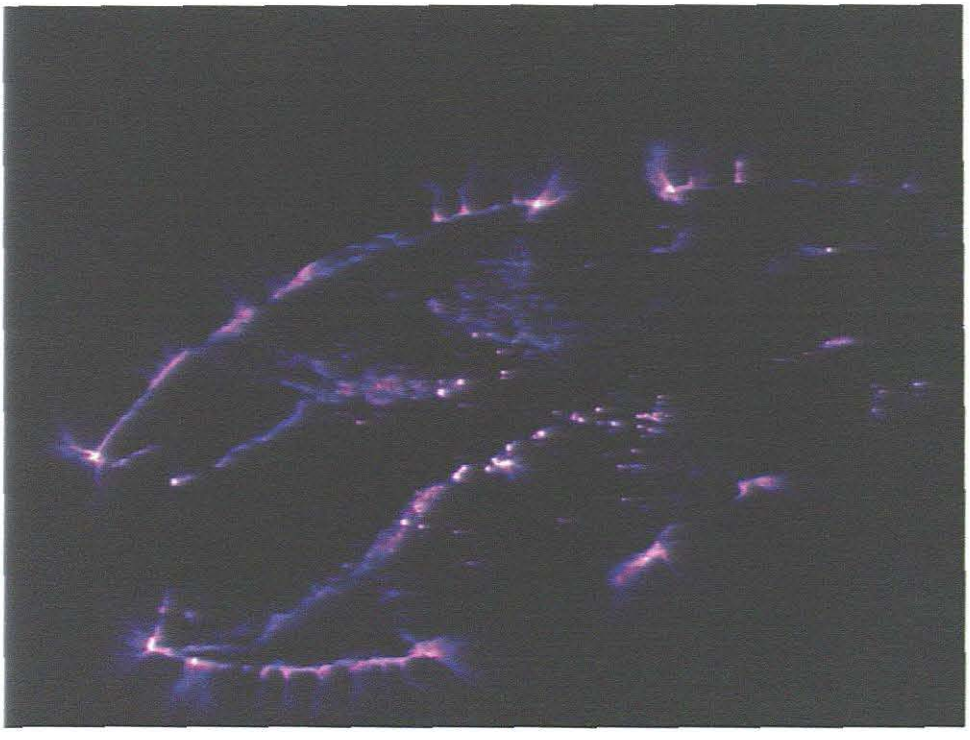


Figure 8-5: Kirlian photograph of experiment "TestHighVoltages8"

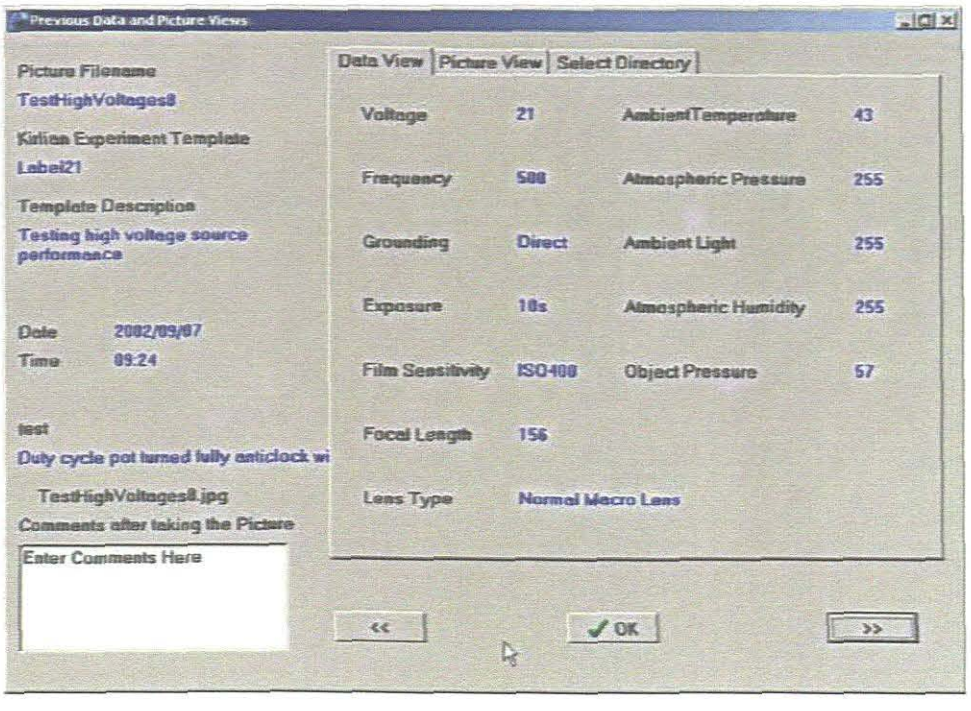


Figure 8-6: Data collected for experiment "TestHighVoltages8"

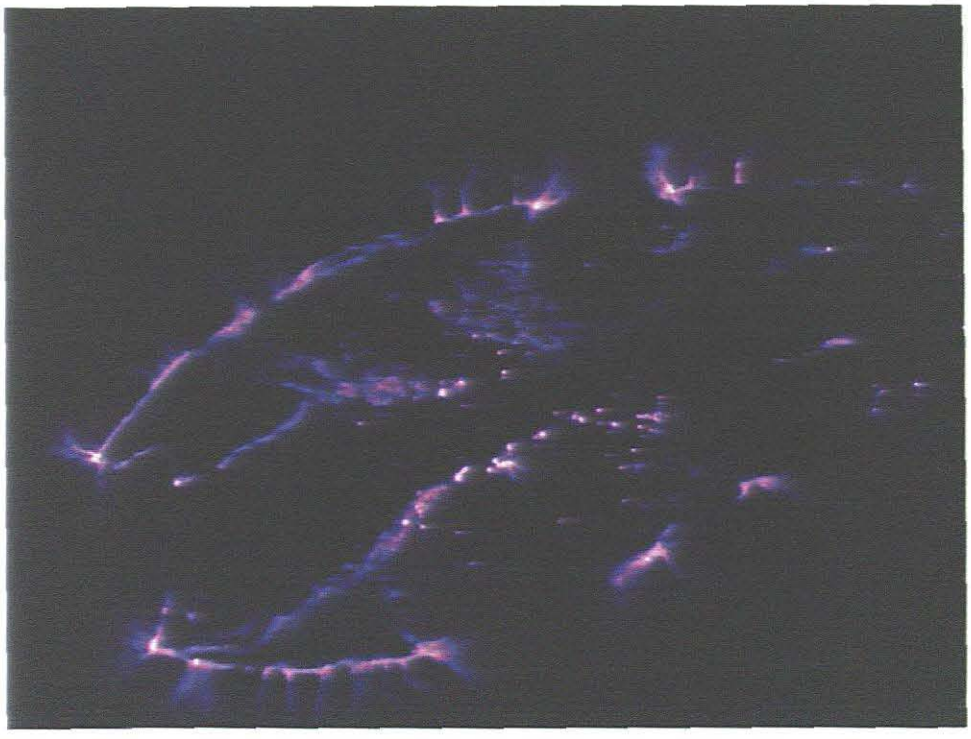


Figure 8-7: Kirlian photograph of experiment "TestHighVoltages9"

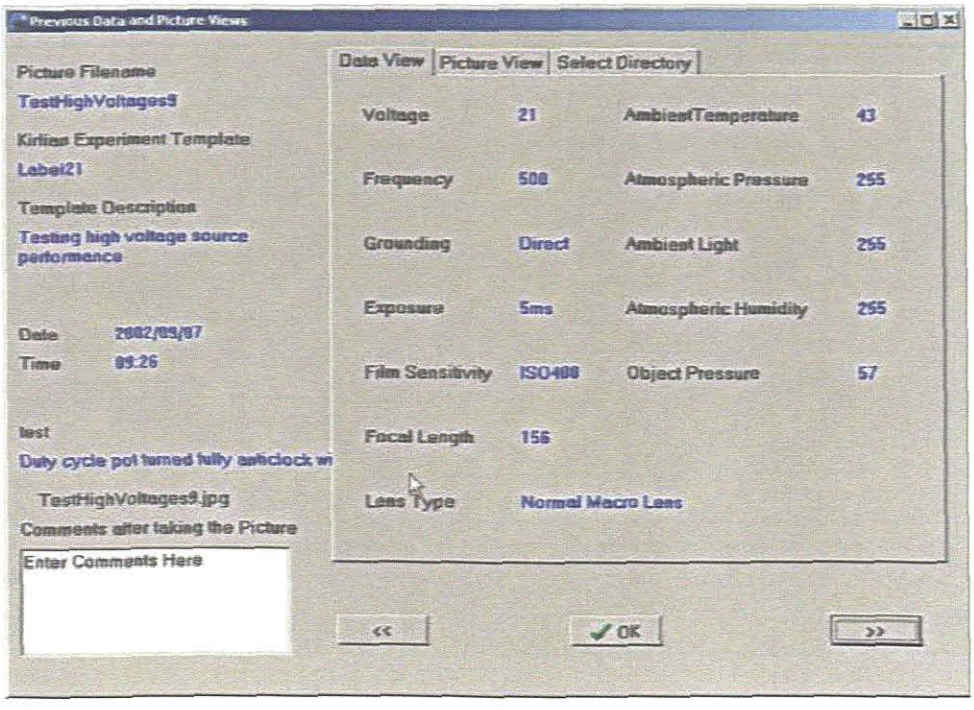


Figure 8-8: Data collected for experiment "TestHighVoltages9

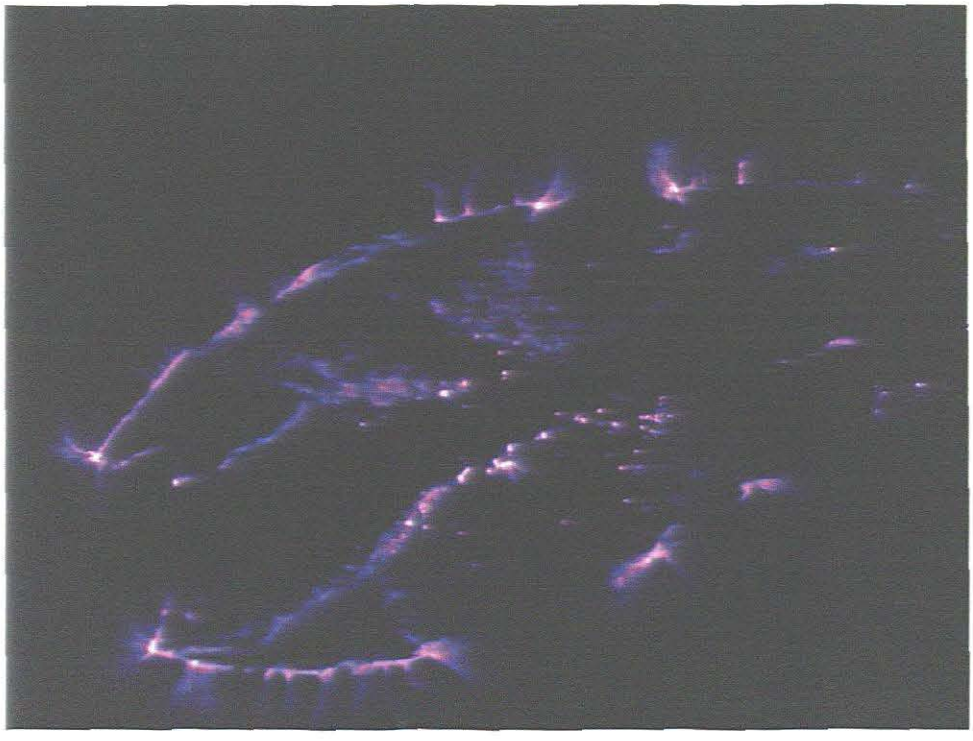


Figure 8-9: Kirlian photograph of experiment "TestHighVoltages12"

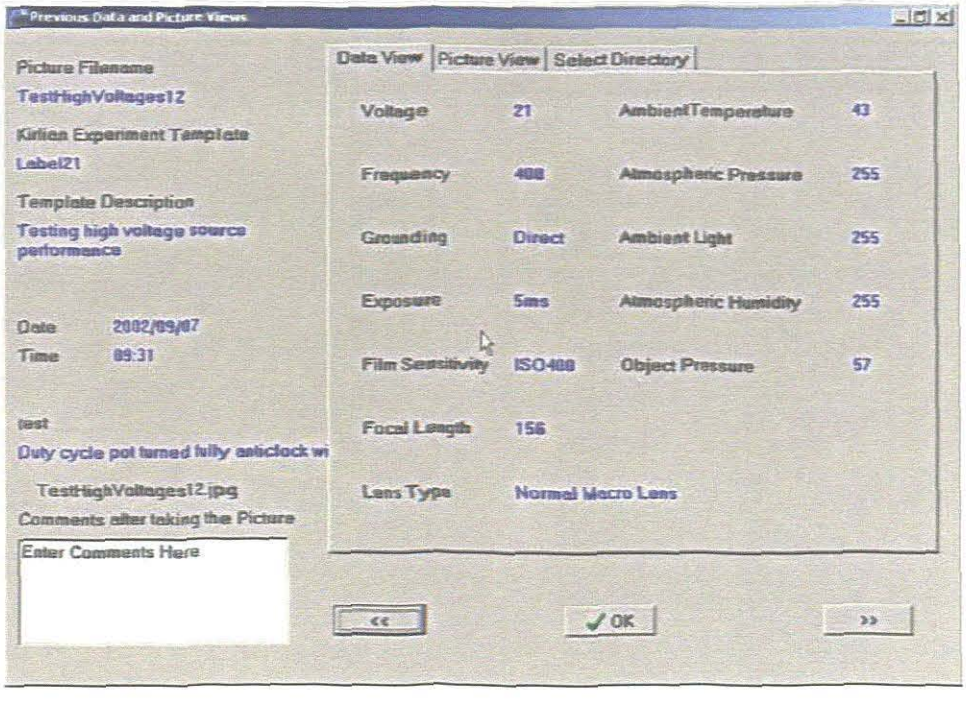


Figure 8-10: Data collected for experiment "TestHighVoltages12"

Figure 8-11: Kirlian photograph of experiment "TestHighVoltages13"

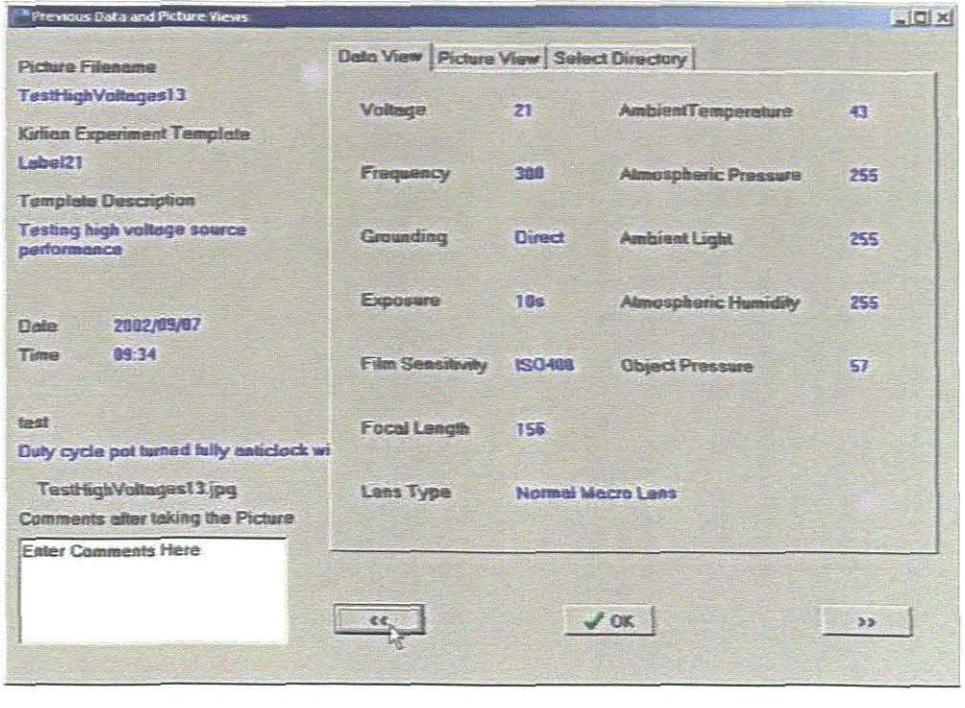


Figure 8-12: Data collected for experiment "TestHighVoltages13"

Figure 8-13: Kirlian photograph of experiment "TestHighVoltages14"

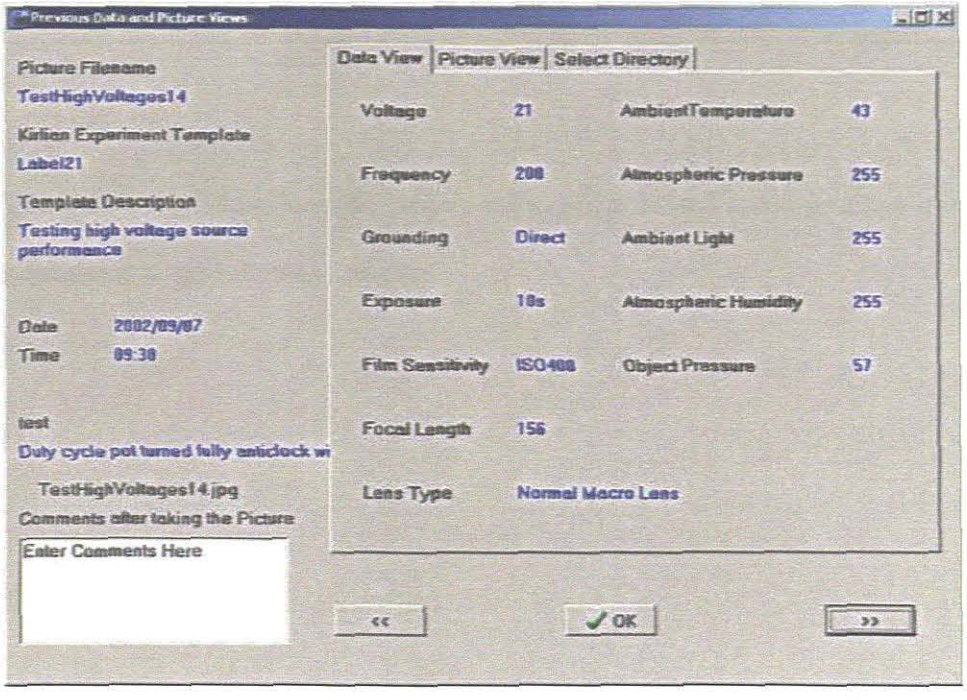


Figure 8-14: Data collected for experiment "TestHighVoltages14"

8.5 Safety Precautions

The instrument was designed with inherent safety aspects. These included the full Galvanic isolation of the high voltage generation circuitry. This provided flashover isolation of 1500V from the low voltage side. The maximum current was limited on secondary to 5mA this is twenty times less than what could be considered a dangerous.

Caution typified the initial human experiments as time progressed however confidence grew in the ability to produces instant quality Kirlian pictures and in the safety aspects of the device. One aspect requires further attention is the high voltage lead is some what exposed and requires the operator to make sure human subjects do not come within 10cm of this terminal.

8.6 Conclusions

Conclusions' resulting from this chapter refers to the overall instrument and it is therefore discussed in the following chapter.

CHAPTER 9

CONCLUSIONS AND RECOMMENDATIONS

9.1 Introduction

This dissertation is based on the design and practical construction of a prototype instrument, which can be used in research of the Kirlian photographic phenomena. The focus has been on using modern design techniques to provide reliable scientifically valid research results.

An integrated design was chosen to provide an empirical view when undertaking research. The device consists of a number of engineering systems that require synthesis. Each one of these tasks by them selves was not particularly difficult. Nevertheless, as far as is known, this is the first time a Kirlian photographic instrument with these features is envisaged and assembled.

9.2 Problems Solved in the Dissertation

The problem that this dissertation was required to solve was design and construction of a Kirlian photographic device was can be used in furthering research on this matter. This problem was divided into many sub-problems, which is described briefly in the following sections.

9.2.1 Environment Parameters

This task required selecting, from numerous possibilities, environment conditions to monitor. A personal interview with a leading expert was conducted as well as a literature review was done. The five, measurands were selected namely; ambient temperature, humidity, barometric pressure, background light levels as well as the object pressure. All of these measurements were implemented except that of object pressure which not because of time constraints. Appropriate sensors were selected and the required electronic circuitry was designed and constructed. A method of implementing alarm levels via software was also conceived.

9.2.2 Portable Photographic System

This task required the design and implementation of a portable photographic system that would allow taking of pictures without the aid of a darkroom. A Transparent Liquid Electrode (TLE) was crafted which enabled a digital camera to take Kirlian pictures. Near instant Kirlian pictures were produced, facilitated by the digital camera selected.

9.2.3 Microcontroller System

This task required the design, implementation and programming of microcontroller system that facilitates the communication of instructions from the personal computer (PC) software to the rest of the instrument. This system, based on an existing design, it collects all of the analogue and digital data from the instrument as well as setting the parameters of the high voltage system.

9.2.4 High Voltage System

This task required the design and implementation of a high voltage supply that can be controlled from a microcontroller. Galvanic isolation was provided between the high voltage supply and the microcontroller. The microcontroller provides the signals required to configure the frequency and voltage.

9.2.5 Computer Program

This task required the design and programming of a computer program which provides the operator with an experimenter centric view of the research. A profile of experiment conditions can be stored and recalled by the operator. It also provides an offline mechanism so that other parties via the internet or other software distribution methods could view the completed experiments. A completed experiment contains the Kirlian picture as well as the conditions of the experiment.

9.2.6 Integrating the Engineering Systems

This task required the integration of the above engineering systems into a synergistic design that makes practical sense. A portable light-tight wooden box was designed and constructed to contain these systems except the software program of course which is executed on an external personal computer (PC). This design also implements an easily removable cover with a light tight sleeve which allows a picture to be taken of a human hand.

9.2.7 Performance of the Instrument

The final task required the testing of the device. All engineering systems were implemented as discussed. The working prototype is in the main completed. It is currently housed in the Information Technology (Academic) department at the Peninsula Technikon. It progress can be tracked by visiting the http://www.pentech.ac.za/sydwell website.

The instrument proved reliable and robust, through three months of testing and demonstration. Excellent quality pictures are produced. The true potential of the device is at the moment undiscovered and will require further investigation.

9.3 Recommendations for Further Study

It can be said with reasonable confidence that this device opens up many research avenues. There are countless researchers, in this field, who strived that their sometimes controversial results could be duplicated [Moss, 1981]. This instrument or others using these design techniques could provide the validation required. There are obviously still contentious areas and finding significant result might be elusive even with this device. Techniques such as fractal, matrix and probability analysis might ultimately be needed to validate the efficacy of the instrument as done by other researchers [Korotkov, 2001], The design approach outlined in this research is an honest attempt to reinvigorate discussion, debate and more importantly experimentation surrounding the Kirlian photographic phenomena.

APPENDIX A
LISTING OF MICROCONTROLLER CODE

```
/*
Kirlian Project 2001
Author: Sydwell Williams
Date: 07 June 2001
      Version: D06
      Revision Date: 05 July 2001


NB!!!!!!!!!!!!!!!! This code must be Linker @ location at 0h


Implementing correct total Time Periods
0           10    milliseconds
1           20    milliseconds
2           50    milliseconds
3           100   milliseconds
4           200   milliseconds
5           500   milliseconds
6           1     second
7           2     seconds
8           5     seconds
9           10    seconds
10          20    seconds
11          50    seconds
*/


#pragma DEBUG OBJECTEXTEND CODE
#include <reg515.h>       /* special function register declarations */
#include <ctype.h>
#pragma REGISTERBANK (0)
#define     TRUE 1
#define     FALSE 0
#define     check_ok          'r'
#define     comms_check       'c'
#define     dump_data         'd'
#define     get_set_values    'g'
#define     testing           't'
#define     take_picture      'p'

extern char getkey ();
extern char putchar (char);

int DigitalOP (char pin, bit value);
sfr16 CRC   = 0xCA;  // 16bit compare reload and capture sfr, timer2
sfr16 CC1   = 0xC2;  // The first Capture and Compare sfr, timer2
```

```
sbit        shutter_dop       = P4^1;
sbit        waveType_dop      = P4^2;
sbit        ground1_dop       = P4^3;
sbit        ground2_dop       = P4^4;
sbit        safety_din        = P4^5;
sbit        DtoA_dop          = P4^6;
sbit        freq_dop          = P4^7;

char temperature, pressure, light, humidity, testc, time;
unsigned long total_time_periods, time_periods;
bit debug = FALSE;                    // Debug information during run
time


#pragma REGISTERBANK (0)
void timer0 (void) interrupt 1 using 0
{
time_periods++;
freq_dop=!freq_dop;      // Produces a fixed frequency of 5Khz
}


#pragma REGISTERBANK (0)

char GoodData (char port)
{
int i,  sum=0;
ADM=0;              // Single conversion
      ADCON&=0xF8;       // Clear old port settings
      ADCON|=port;       // Select port
      for (i=0; i<6; i++)
      {
            DAPR=0x00;         // Start conversion
            while (BSY);       // Do nothing  till end of conversion
            sum+=ADDAT;
}
return (sum/6);
}

int loop (void)
{
      ground2_dop=1;
      //    while (TRUE) putchar ('L');
}

int GetAllData (void)
{
      if (debug) putchar ('D');
      temperature = GoodData (0);
      pressure    = GoodData (1);
      light       = GoodData (2);
      humidity    = GoodData (3);
      return (TRUE);
}

int CommsError (void) //char *message)
```

```
{
      putchar ('E');
      ground1_dop=1;
      //printf ("\n\n Comms err \n\t%s", message);
      return (FALSE);
}

int SetValues (void)
{
      char voltage, ground, shape;
      unsigned int frequency;
      int freq_l, freq_h;      //Shouldn't be char as well
      int first=0;

      //    ground1_dop=0;
      safety_din=1;
      DtoA_dop=0;
      if (debug) putchar ('g');
      voltage = getkey ();          // Voltage output to be generated
      in KVolts
      P5=voltage;             // Remember the Low Nibble selects
      Voltages
      freq_l = getkey ();           // Frequency Low byte rem counter
      reload
      freq_h = getkey ();     // Freq high byte
      time= getkey ();        // The number of 100us time periods
      needed
      switch (time)
      {
            case 0x09:  total_time_periods = 100; break;
            case 0x0A:  total_time_periods = 200; break;
            case 0x0C:  total_time_periods = 500; break;
            case 0x11:  total_time_periods = 1000; break;
            case 0x12:  total_time_periods = 2000; break;
            case 0x14:  total_time_periods = 5000; break;
            case 0x21:  total_time_periods = 10000; break;  //!
            case 0x22:  total_time_periods = 20000; break;  //"
            case 0x24:  total_time_periods = 50000; break;  //$
            case 0x41:  total_time_periods = 100000; break; //A
            case 0x42:  total_time_periods = 200000; break; //B
            case 0x44:  total_time_periods = 500000; break; //D
            default: total_time_periods = 70000;
      }

      ground=getkey ();             // Get the type of grounding
      shape=getkey ();
      //humidity=voltage;           // Testing
      frequency = (freq_h*0x100) + freq_l;
      CRC = 0x10000 - frequency;
      CC1 = 0x10000 - (frequency/2);
      //P5=humidity;
      return (TRUE);
}

int TakeSnap (void)     //     a picture will be taken
{
```

```
      ET0 = 1;          // Enable Timer 0 interupt
      TR0 = 1;          // start timer 0
      time_periods=0;
      if (debug) putchar ('^');
      //    testc=getkey();
      shutter_dop =1;               // Switch shutter output on
      while (time_periods < (total_time_periods))
            //total_time_periods)
      {
            if (debug) putchar ('W');
            // Do nothing remember
            //interupt timer routines will be serviced
      }
      TR0 = 0;                      // stop timer 0
      if (debug) putchar (')');
      shutter_dop = FALSE;          // Switch shutter output off
      putchar ('P');                // Send Acknowledgement to PC
}

int Test (void)
{
      putchar ('t');
      CCH3=0x60;
      waveType_dop=!waveType_dop;
      ground2_dop=0;
      ground1_dop=0;
      shutter_dop=0;
      safety_din=0;
      DtoA_dop=0;
      return (TRUE);
}


/**
main program starts here
*/

void main (void)
{

      char received_instruction = 0;
      int  inst= 0;              // Number of instructions
      char  temp_lowbyte=0,
      press_lowbyte=0,
      humid_lowbyte=0,
      light_lowbyte=0;

      /* The following three commands set the Comms as follows
      Baud rate = 9600
      Data Bits = 8
      stop Bits = 1
      Parity    = none
      */

      SCON  = 0x56;     // SCON: mode 1, 8-bit UART, enable rcvr
      changed from 0x54
```

```
PCON  = 0x80;      // SMOD bit of PCON set to double frequency
ADCON = 0xC0;      // Standard Baud Rate used, BD Set
// and CLK output set ON
IEN0  = 0;
IEN1  = 0;         // Disable All interrupts
EAL   = 1;         // Enable all interrupts Global switch
P4    = 0x00;      // Clears Port 4
P5    = 0x0A;      // Port 5 to switch DA on, outputting 14.12v
TH0   = 0x9C;            // Timer 0 reload 100 counts = 100uS
TL0   = 0x9C;            // remember counts from 9C to FF or 100
counts interupt

TMOD  = 0x22;      // timer0 and 1 in mode 2
ground2_dop = 1;   // Indication that micro is working

//    printf ("\n \n All well that is, sydwell   \n");
putchar ('D');
putchar ('0');
putchar ('6');
putchar (' ');

// Setup Timer2 for Frequency Generation
// Frequency Out from CC1 - level changes at counter overflow and
at CCL1 & CCH1 = counter TL2 & TH2
// Signal Available at P1.1 also known as K8
T2CON=0x11;              // 1uS autoreload timer compare mode 0
// timer function selected T2CON.0=T2I0=1 and T2CON.1=T2I1=0
// reload mode selected T2CON.3=T2R0=0 and T2CON.4=T2R1=1
// compare mode 0 T2CON.2=T2CM=0
// prescaler not used T2CON.7=T2PS=0
CCEN=0xAA;// Compare enabled on all compare/capture registers CC
// testing timer2
CCH1 = 0xFF;
CCL1 = 0xFE;
CRC = 0xFFFD;
CCH2=0xE0;
CCH3=0xA0;

GetAllData ();
while (received_instruction != 'Q')
{
      received_instruction = getkey ();
      if (debug)
      {
            putchar ('R');
            putchar (received_instruction);
      }
      switch (received_instruction)
      {
            case comms_check: putchar (check_ok); break;
            case dump_data: putchar ('1'); putchar (temperature);
            putchar (pressure); putchar (light); putchar
            (humidity);
            putchar ('9'); DtoA_dop=1; safety_din=0; break;
            case get_set_values: SetValues (); break;
            case '1': loop (); break;
```

```
                case take_picture: TakeSnap (); break;
                case testing : Test (); break;
                default : CommsError (); //"Invalid instruction");
            }
            GetAllData ();
}
    // printf ("Kirlian 80C535 Halted By esc Please Reset");
    while (TRUE)
    {
    // Endless loop
    }
    //putchar ('Q');
}

/*          END OF MICROCONTROLLER CODE     */
```

APPENDIX B
LISTING OF SOFTWARE CODE FOR MAIN-FORM

```
/*************************************************************************
 *    Project:    Kirlian Experimenter
 *    Author:     Sydwell Williams
 *    Date:       20/3/2001
 *    Version:    1.27
 *    Revision:   05/05/2003
 *
 *  Currently: When setting up camera, use cam2com to set initial
shutter speed
 *  First Publicly Demonstrable Version
 *
 *    .1  Corrected Previous Displayed Picture
 *    .2  Store Experiment Results in Folders Only
 *    .3  Working Exception Handling Demostratored
 *    .4  Fixing File location issues
 *    .5  Include true fullscreen views - Document aspect ratio issue
 *    .6  Corrected Displayed Picture
 *
 * 1.21.0
 *  Major rewrite Olympus SDK used to control camera
 *    optical Zoom issues  Ok
 * 1.22
 *    Changed Automatic Naming to 00N
 *    Fixed Preview bug
 *    Still to check And Error return codes
 *    What about a fixed powersupply ?
 * 1.23
 *    Major error condition displayed - OK
 *    Changed Status Panels - OK
 *    Still to change Last Picture to work under restarted conditions
 *
 *1.24
 *    Click Camera for Camera Status Enabled
 *
 *1.25
 *    Fixed problem with rereading directory listing
 *
 *1.26
 *    Last Picture screen show last Picture now and bugs fixed
 *
 *1.27
 *    Drive box for Kirlian Data View window
 *    Error detection if PublicK.jpg not found
```

```
 *
*****************************************************************************
/
#include <vcl.h>
#pragma hdrstop
#include "K2001Main.h"
#pragma link "RYELib_OCX"
#pragma link "RYELib_OCX"
#pragma resource "*.dfm"

TK2001Main_f *K2001Main_f;
//---------------------------------------------------------------------------
                    /*****************************************/
                    /*            NEW SECTION                */
                    /*                 1                     */
                    /*            Main Routines              */
                    /*****************************************/

void __fastcall TK2001Main_f::FormCreate(TObject *Sender)
/*****************************************************************************
 * 1.1  Code runs when form is created
 *
 *    Arguments:     GUI designer TForm
 *    Effects:       Iniatializers the main form
 *    Return:        void
*****************************************************************************/
{
   StartUp = true;                //    True during the Startup Process
                                  //    When Debugging leave as false
 //  K2001Main_f->Visible = false;
}
//

void __fastcall TK2001Main_f::FormActivate(TObject *Sender)
/*****************************************************************************
 * 1.2  Code runs when form is Activated
 *
 *    Arguments:     GUI designer TForm
 *    Effects:       Activates the main form
 *    Return:        void

*****************************************************************************
/
{
   if (StartUp)
   {
      Splash_f->WindowState = wsNormal;
      Splash_f->ShowModal();
      K2001Main_f->Visible = true;
      StartUp = false;
   }
}
//

__fastcall TK2001Main_f::TK2001Main_f(TComponent *Owner)     :
TForm(Owner)
```

```
/*************************************************************************
 * 1.3  The Constructor for the main form
 *
 *    Arguments:     GUI designer TForm
 *    Effects:       Constructor the main form
 *    Return:        void
*************************************************************************
/
{
   HouseKeeping();
   LoadExperimentTemplate("default");
}
//

void __fastcall TK2001Main_f::HouseKeeping(void)
/*************************************************************************
 * 1.4  Important housekeeping done here
 *
 *    Arguments:     void
 *    Effects:       Setup Appearance, Initialisers Global Variables
 *    Return:        void
*************************************************************************
/
{
   K2001Main_f->Height=570;
   K2001Main_f->Width=800;
   K2001Main_f->Top=1;
   K2001Main_f->Left=1;
   isTimerRoutineRunning = false;
   Main_pagectl->ActivePage=LastPicture_tabsht;              // Set the
active Page
   TakePicture_but->Font->Style = TFontStyles()<< fsBold;;
   TakePicture_but->Caption="Actual Size Picture";
   K2001Main_f->Color=clAqua;
   Environment_tabsht->Brush->Color=clTeal;
   RTCamera_tabsht->Brush->Color=clTeal;
   LastPicture_tabsht->Brush->Color=clTeal;
   baseTitle = " THE KIRLIAN EXPERIMENTER -";

///   StatusBar->Color=clYellow;
///   StatusBar->Font->Color=clTeal;
///   StatusBar->Font->Style=TFontStyles()<< fsBold;
///   StatusBar->Font->Size=10;

/* Set Initial Gobal Variables */
   CommsTestBAD = true;
   SetCommPort(COMM1);
   K2001Main_f->Menu = MainMenu1;
   EnableHVwithoutCamera->Checked = true;
   PictureRequested = false;
   FileNameKET    = "Default";
   ProgramCurrentDir = GetCurrentDir();
   cameraNo = 0;         // Will create a problem if there is more than
one USB camera connected
   if (FileExists("fresh.jpg")) DeleteFile("fresh.jpg"); //Makesure
downloaded file doesn't exist
```

```
   // This mess up the Script path consider cmdline  ParamStr(0)
   BaseDataDir     = ProgramCurrentDir + "//Default";
   pollCount       = 0;
   checkRok        = 'r';
   commsCheck      = 'c';
   dataDump        = 'd';
   Get_set_values = 'g';
   Testing         = 't';
   takePicture     = 'p';

/* Load initial picture and configure its size */
   PreviousPic_jp = new TJPEGImage();  // Is two TJPEGImage 's really
need?
   DataView_jp = new TJPEGImage();
   try
   {
      PreviousPic_jp->LoadFromFile("PublicK.jpg");
      DataView_jp->Assign(PreviousPic_jp);
      Image2->Picture->Assign(DataView_jp);
      Image2->Stretch = true;
   }
   catch (const Exception &e)
   {
      Application->MessageBox(" No Previous Picture \"PublicK.jpg\" ",
NULL, MB_OK);
   }

 /*  if (CameraCheck())
   {
      CameraPanel->Color=clLime;
      CameraPanel->Caption="CAMERA STATUS: Connected";
   } else
   {
      CameraPanel->Color=clRed;
      CameraPanel->Caption="CAMERA STATUS: Disconnected";
   } */
}   //endmethod HouseKeeping
//


void __fastcall TK2001Main_f::Main_pagectlChange(TObject *Sender)
/*****************************************************************************
 * 1.5  Display different main button for different pages
 *
 *    Arguments:      GUI designer page control clicked
 *    Effects:        Show the various pages on the main panel on the
main form
 *    Return:         void
*****************************************************************************
/
{
   //TPageControl
   if (Main_pagectl->ActivePage == Environment_tabsht)
   {
       TakePicture_but->Font->Style = TFontStyles()<< fsBold;
       TakePicture_but->Caption="Take a Picture";
```

```
        TakePicture_but->Enabled = true;
        FullScreen_but->Visible = false;
    }
    if (Main_pagectl->ActivePage == RTCamera_tabsht)
    {
       TakePicture_but->Font->Style = TFontStyles()<< fsBold;;
       TakePicture_but->Caption = "Actual Size Picture";
       TakePicture_but->Enabled = false;
       FullScreen_but->Enabled = false;
       FullScreen_but->Visible = true;
    }
    if (Main_pagectl->ActivePage == LastPicture_tabsht)
    {
       TakePicture_but->Font->Style = TFontStyles()<< fsBold;;
       TakePicture_but->Caption = "Actual Size Picture";
       TakePicture_but->Enabled = true;
       FullScreen_but->Enabled = true;
       FullScreen_but->Visible = true;
    }
}
//

void __fastcall TK2001Main_f::NoCameraView_butClick(TObject *Sender)
/*****************************************************************************
 * 1.6  The camera view page
 *
 *    Arguments:      GUI designer Camera view panel button clicked
 *    Effects:        Display Environment Panel
 *    Return:         void
*****************************************************************************
/
{
   Main_pagectl->ActivePage=Environment_tabsht;
   TakePicture_but->Font->Style = TFontStyles()<< fsBold;
   TakePicture_but->Caption="Take a Picture";
}
//

void __fastcall TK2001Main_f::TakePicture_butClick(TObject *Sender)
/*****************************************************************************
 * 1.7  What happens when the Take Picture button is pressed
 *
 *    Arguments:      GUI designer Take Picture button clicked
 *    Effects:        Handle all the conditions as the Take Picture
button is clicked
 *                    a) If the Displayed panel is Lastpicture, display
full-size picture
 *                    b) If the Displayed panel is Environment, Attempt
to take a picture
 *                    c) Attempted picture spawn a external program,
change
 *    Return:         void
*****************************************************************************
/
{
    while (isTimerRoutineRunning)
```

```
      Label1->Caption = "timer";/* Do Nothing this label is not visible
*/
    K2001Main_f->Repaint();
    PollRoutine->Enabled = false;       //Stop the timer routine
   TJPEGImage *jp = new TJPEGImage();
   if (Main_pagectl->ActivePage == LastPicture_tabsht)
   {
      FullScreen_f->Image1->Height=FullScreen_f->Image1->Picture-
>Height;
      FullScreen_f->Image1->Width=FullScreen_f->Image1->Picture->Width;
     /* FullScreen_f->Image1->Height=Screen->Height;
      FullScreen_f->Image1->Width=Screen->Width;*/
      FullScreen_f->Image1->Stretch = false;
      FullScreen_f->Height = Screen->Height;
      FullScreen_f->Width = Screen->Width;
      FullScreen_f->BorderStyle = bsSizeable;
      FullScreen_f->ShowModal();
      PollRoutine->Enabled=true;
   }
   if (Main_pagectl->ActivePage == Environment_tabsht)
   {
      if (CameraCheck())
      {
         // Just take the picture NB!! camera must be correctly setup
         int err;
         SetCurrentDir (ProgramCurrentDir);
         CameraPanel->Visible = true;
         CameraPanel->Color=clLime;
         CameraPanel->Caption="CAMERA STATUS: Connected";
         PictureInProgress_b->Visible = true;
         PictureInProgress_b->Caption = "High Voltage now enabled";
         PictureInProgress_b->BringToFront();
         K2001Main_f->Repaint();
         long numPicCanTake = Rye1->propRemainCount[0];
         if (numPicCanTake < 1)
         {
            Application->MessageBox("Not enough space on camera card to
take picture", NULL, MB_OKCANCEL) != IDOK;
         }
         Rye1->propFlash[0] = RYE_FLASH_OFF;
         Rye1->propFocusMode[0] = RYE_FOCUS_MANUAL;
         Rye1->propWhiteBalance[0] = RYE_WB_DAYLIGHT;
         Rye1->propElectricZoom[0] = RYE_EZ_OFF;
         Rye1->propExpoBias[0] = 0;
         if (ZoomWide->Checked) Rye1->propOpticalZoom[0] = 1;
         if (ZoomNormal->Checked) Rye1->propOpticalZoom[0] = 100;
         if (ZoomTelephoto->Checked) Rye1->propOpticalZoom[0] = 225;
         Rye1->SetExposurePropVB(0, RYE_EXPOSURE_MANUAL,
kinfo.getFStopValue(), 1000 * kinfo.getExposureMillseconds());
         K2001Main_f->Rye1->propIsoSpeed[0] = kinfo.getSensitivity() +
1;
         K2001Main_f->Rye1->propFocusPosition[0] = kinfo.getFocal() -
15;
          //Make sure Program Directory is the current directory
          PictureRequested = true;       // Sets flag to enable
Highvoltage source
```

```
        Sleep(200);
        PollRoutine->Enabled = true;
                // Camera takes picture Now
        PollRoutine_timer(this);       // Force the taking of timer
routine
        PictureInProgress_b->Caption = "NOW"; K2001Main_f->Repaint();
        while (isTimerRoutineRunning){ Label1->Caption = "timer";/* Do
Nothing */};
        PollRoutine->Enabled=false;    // Disable the Pollroutine until
a picture is taken

        Rye1->Capture(0);
        PictureInProgress_b->Caption = "Please wait while transfering
Image";
        K2001Main_f->Repaint();
        { //This block of code downloads the image from the camera
           long noOfPics = Rye1->propPicCount[0];
           Rye1->propCurrentPicture[0] = noOfPics;
           const long nPicSize = Rye1->propPicSize[0];
           Variant V(OPENARRAY(int,(0,nPicSize-1)),varInteger);
           TVariant PicTV = new TVariant(V);
           TMemoryStream* picStream = new TMemoryStream();
           Rye1->GetPicture(0,nPicSize,PicTV.intVal );
           int *picBuffer;
           if ((picBuffer = (int *) malloc(nPicSize)) == NULL)
           {
              Application->MessageBox("Not enough memory in RAM for
JPEG image", NULL, MB_OKCANCEL) != IDOK;
              exit(1);  /* terminate program if out of memory */
           }
           picBuffer =(int *) VarArrayLock(PicTV);
           picStream->Position = 0;
           picStream->Write(picBuffer, nPicSize);
           picStream->Position = 0;
           picStream->SaveToFile("fresh.jpg");
           picStream->Free();
           Rye1->EraseLast(0);                  // Erase the picture in
the camera
           Rye1->propPowerSaveHost[0] = 30;     //Check this
           Rye1->DisConnect(0);
        }
// Create a JPEG object assign it to a Image2 to display it!
        while (!FileExists("fresh.jpg")){/* Do Nothing */};
        jp->LoadFromFile("fresh.jpg");
        Image2->Picture->Assign( jp );
        Image2->Stretch = true;
        kinfo.setDateTime();
        FullScreen_f->Image1->Picture->Assign( jp );
// Save data in file
      //TFileStream *KData2Save;
        int iFileHandle;
        iFileHandle = FileCreate (BaseDataDir + "\\" +
kinfo.getPictureFile() + ".KPD");
        FileWrite (iFileHandle, &kinfo, sizeof(kinfo));
        FileClose(iFileHandle);
```

```
         jp->SaveToFile(BaseDataDir + "\\" + kinfo.getPictureFile() +
".jpg");
        // PreviousPic_jp->Assign(jp);   this was commmented out in
version 1.27
         getKDataFilename (); // get the new filename
         DisplayKInfo(kinfo); // Diplay the new info
         if (FileExists("PublicK.jpg")) DeleteFile ("PublicK.jpg");
         if (FileExists("fresh.jpg")) RenameFile("fresh.jpg",
"PublicK.jpg");

         PictureInProgress_b->Visible = false;
         Main_pagectl->ActivePage = LastPicture_tabsht;
         Main_pagectlChange( Sender );
         // jp->Close();              */
      }  // endif CameraCheck
      else
      {
         CameraPanel->Visible = true;
         CameraPanel->Color = clRed;
         CameraPanel->Caption = "CAMERA STATUS: Disconnected";
         if (EnableHVwithoutCamera->Checked && !CommsTestBAD)
         {
            PictureRequested = true;
            CameraPanel->Visible = true;
            CameraPanel->Color=clYellow;
            CameraPanel->Caption="CAMERA STATUS: Disconnected but HV
enabled";
            ///StatusBar->Color=clOlive;
            ///StatusBar->Panels->Items[0]->Text="COMMUNICATION: High
Voltage Enabled ";
         }
      }  // endelse CameraCheck
      PollRoutine->Enabled=true;
   }  // endif Main_pagectl->ActivePage == Environment_tabsht
}  // endmethod TakePicture_butClick
//

void __fastcall TK2001Main_f::getKDataFilename (void)
/*****************************************************************
 * 1.8  Get the appropriate Filename
 *
 *    Arguments:      void
 *    Effects:        TakePicture_butClick handler to create a new Data
and Picture filename
 *    Return:         void
*****************************************************************
/
{
   AnsiString nam3, Wstr = FileNameKET;      // a working string
   int pos = Wstr.LastDelimiter("\\");
   Wstr.Delete( 1, pos );
   pos = Wstr.LastDelimiter( "." );
   Wstr.Delete( pos, Wstr.Length() );  // Make sure we have the raw
filename
   for (int n = 1; n < 1000; n++)
   {
```

```
      if (n > 99) nam3 = Wstr + n;
      else
         if (n > 9) nam3 = Wstr + "0" + n;
         else nam3 = Wstr + "00" + n;
      if ( !FileExists(BaseDataDir + "\\" + nam3 + ".KPD") ) break;
     }
    kinfo.setPictureFile (nam3);
}  //method getKDataFilename
//

void __fastcall TK2001Main_f::DisplayKInfo(KDataClass1 DInfo)
/*************************************************************************
 * 1.9  Display the Kirlian data in the Environment Tabsheet
 *
 *    Arguments:     KDataClass1
 *    Effects:       Called when a new templates is loaded
 *                    a) NIY stands for Not Implemented Yet
 *    Return:        void
*************************************************************************
/
{
   this->Caption = baseTitle + "  Template  = " + FileNameKET;
   FileName_ed->Text = DInfo.getPictureFile();
   Template_memo->Lines->SetText(
DInfo.getExperimentTemplateComment().c_str() );
   Experiment_memo->Lines->SetText( DInfo.getPictureComment().c_str()
);
   Voltage_ed->Text = DInfo.getVoltage();
   Voltage_ud->Position = DInfo.getVoltage();
   if ( 1000 >  DInfo.getFrequency())
   {
      Frequency_ed->Text = DInfo.getFrequency();
      Label22->Caption = "Hz";
   } else
   {
      Frequency_ed->Text = DInfo.getFrequency()/1000;
      Label22->Caption = "kHz";
   }
   Frequency_ud->Position = DInfo.getFrequency()/10;
   Ground_cbx->ItemIndex = DInfo.getGround();
   Exposure_cbx->ItemIndex = DInfo.getExposure();
   Lens_ed->Text = DInfo.getLens();
   Focal_ed->Text = DInfo.getFocal();
   Focal_ud->Position = DInfo.getFocal();
   Sensitivity_cbx->ItemIndex = DInfo.getSensitivity();
   FStop_cbx->ItemIndex = DInfo.getFStop();
   MinTemp_ed->Text = DInfo.getMinTemp();
   MinTemp_ud->Position = DInfo.getMinTemp();
//NIY    MinAtmPress_ed->Text = DInfo.getMinPress();
//NIY    MinAtmPress_ud->Position = DInfo.getMinPress();
   MinLight_ed->Text = DInfo.getMinLight();
   MinLight_ud->Position = DInfo.getMinLight();
   MinHumid_ed->Text = DInfo.getMinHumid();
   MinHumid_ud->Position = DInfo.getMinHumid();
//NIY    MinObjPress_ed->Text = DInfo.getMinOpress();
//NIY    MinObjPress_ud->Position = DInfo.getMinOpress();
```

```
   MaxTemp_ed->Text = DInfo.getMaxTemp();
   MaxTemp_ud->Position = DInfo.getMaxTemp();
//NIY     MaxAtmPress_ed->Text = DInfo.getMaxPress();
//NIY     MaxAtmPress_ud->Position = DInfo.getMaxPress();
   MaxLight_ed->Text = DInfo.getMaxLight();
   MaxLight_ud->Position = DInfo.getMaxLight();
   MaxHumid_ed->Text = DInfo.getMaxHumid();
   MaxHumid_ud->Position = DInfo.getMaxHumid();
//NIY     MaxObjPress_ed->Text = DInfo.getMaxOpress();
//NIY     MaxObjPress_ud->Position = DInfo.getMaxOpress();
} // endMethod DisplayKInfo
//

void __fastcall TK2001Main_f::PollRoutine_timer(TObject *Sender)
/*****************************************************************************
 *  1.10   Timer routine accessed every 1000 ms (1 second)
 *
 *    Arguments:     GUI designer Timer routine
 *    Effects:       This is the timer interrupt handler
 *                   a) Communicates with the micro and displays
appropriate
 *                       diagnostic information
 *                   b) Displays the environment variable in the
appropriate colours,
 *                       Blue if the variables are within range of in
the KET
 *                       Green if the variables are below range of in
the KET
 *                       Red if the variables are above range of in the
KET
 *                   c) This routine is disabled when the main form is
not active
 *
 *    Return:        void
*****************************************************************************
/
 {
   char         *receivedBuffer = new char[20], exposureChar;
   AnsiString   sendGetsetBuffer;
   DWORD        nBytesToWrite =1,     nBytesToRead=6;
   DWORD        nBytesWritten=3,        nBytesRead=2;

   int freq_period; //in Micro seconds
   char freq_highbyte, freq_lowbyte, exp_lb, exp_hb;
   char rxChar, voltage_byte;
   int errno;
   int totalExp;

   isTimerRoutineRunning = true;
   K2001Main_f->Repaint();
   receivedBuffer="623456";
   if (pollCount == 0) KETmodified = false;  // Template is not
modified, can only be placed here
   pollCount++;                              // General variable, not
absolutely necessary
```

```
   if (pollCount == MAXLONG) pollCount = 1;  // Prevent loop counting
throe 0
   if (CommsTestBAD)
   {
/*    Display Error Defaults */
     ActTemp_ed->Font->Color=clRed;
//NIY      ActAtmPress_ed->Font->Color=clRed;
      ActHumid_ed->Font->Color=clRed;
      ActLight_ed->Font->Color=clRed;
//NIY       ActObjPress_ed->Font->Color=clRed;
      ActTemp_ed->Text = "Not Available";
//NIY       ActAtmPress_ed->Text = "Not Available";
      ActHumid_ed->Text = "Not Available";
      ActLight_ed->Text = "Not Available";
//NIY       ActObjPress_ed->Text = "Not Available";
      SetCommPort (ActiveCommPort);
   }else  // endif CommsTestBAD
   {
//Purge all outstanding comms commands and buffers rem timeouts is
effective
      errno = PurgeComm (hCom, PURGE_TXABORT);
      errno = PurgeComm (hCom, PURGE_RXABORT);
      errno = PurgeComm (hCom, PURGE_TXCLEAR);
      errno = PurgeComm (hCom, PURGE_RXCLEAR);
      if (errno == 0) { ShowMessage("Error in PurgeComm");
CommsTestBAD=True; }
// Write to Comm Port to receive data dump
      errno=WriteFile (hCom, &dataDump, 1, &nBytesWritten, NULL);
      if (errno == 0) { ShowMessage("Error in WriteFile");
CommsTestBAD=True; }
// Read the data dump Comm Port
      errno=ReadFile (hCom, receivedBuffer, nBytesToRead, &nBytesRead,
NULL);
      if (errno == 0) { ShowMessage("Error in ReadFile");
CommsTestBAD=True; }
      Label1->Caption="receiving";
      if (receivedBuffer[0]!= '1')
      {
         CommsPanel->Color=clRed;
         CommsPanel->Caption="COMMUNICATION: Cannot read2 state";
         CommsTestBAD=True;
      }
      kinfo.setActTemp( 0.32*posit (receivedBuffer[1]) - 7 );
      kinfo.setActPress( posit (receivedBuffer[2]) );
      kinfo.setActHumid( posit (receivedBuffer[3]) );
      kinfo.setActLight( posit (receivedBuffer[4]) );
      kinfo.setActOpress( posit (receivedBuffer[5]) );
      receivedBuffer[6]= posit (receivedBuffer[6]);   // Should always
be '9'

/*     Display Colours    */
      ActTemp_ed->Text=kinfo.getActTemp();
      if ((kinfo.getActTemp() >=
kinfo.getMinTemp())&&(kinfo.getActTemp() <= kinfo.getMaxTemp()))
ActTemp_ed->Font->Color=clBlue;
```

```
      if (kinfo.getActTemp() < kinfo.getMinTemp()) ActTemp_ed->Font-
>Color=clGreen;
      if (kinfo.getActTemp() > kinfo.getMaxTemp())    ActTemp_ed->Font-
>Color=clRed;
/*NIY       ActAtmPress_ed->Text=kinfo.getActPress();
      if ((kinfo.getActPress() >=
kinfo.getMinPress())&&(kinfo.getActPress() <= kinfo.getMaxPress()))
ActAtmPress_ed->Font->Color=clBlue;
      if (kinfo.getActPress() < kinfo.getMinPress()) ActAtmPress_ed-
>Font->Color=clGreen;
      if (kinfo.getActPress() > kinfo.getMaxPress()) ActAtmPress_ed-
>Font->Color=clRed;
*/
      ActHumid_ed->Text=kinfo.getActHumid();
      if ((kinfo.getActHumid() >=
kinfo.getMinHumid())&&(kinfo.getActHumid() <=
kinfo.getMaxHumid()))ActHumid_ed->Font->Color=clBlue;
      if (kinfo.getActHumid() < kinfo.getMinHumid()) ActHumid_ed->Font-
>Color=clGreen;
      if (kinfo.getActHumid() > kinfo.getMaxHumid()) ActHumid_ed->Font-
>Color=clRed;
      ActLight_ed->Text=kinfo.getActLight();
      if ((kinfo.getActLight() >=
kinfo.getMinLight())&&(kinfo.getActLight() <= kinfo.getMaxLight()))
ActLight_ed->Font->Color=clBlue;
      if (kinfo.getActLight() < kinfo.getMinLight()) ActLight_ed->Font-
>Color=clGreen;
      if (kinfo.getActLight() > kinfo.getMaxLight()) ActLight_ed->Font-
>Color=clRed;

// Send Device parameter to Micro via serial port
      errno = TransmitCommChar (hCom, Testing);
      if (errno == 0) { ShowMessage("Error in TransmitCommChar
Testing"); CommsTestBAD=True; }
      errno=ReadFile (hCom, &rxChar, 1, &nBytesRead, NULL);
      errno=ReadFile (hCom, &rxChar, 1, &nBytesRead, NULL);
      if (errno == 0) { ShowMessage("Error in ReadFile");
CommsTestBAD=True; }
      if ('t' != rxChar) { CommsTestBAD=True;
ShowMessage("!!!STOPPED!!!!"); }
      Label1->Caption=rxChar;
// Send Get_set_values or 'g' to to put kirlian device into Get setable
data mode
      errno = TransmitCommChar (hCom, Get_set_values);
      if (errno == 0) { ShowMessage("Error in TransmitCommChar
Get_set_values"); CommsTestBAD=True; }

// Calculate correct voltage data bits to send
      voltage_byte = kinfo.getVoltage();
      voltage_byte = voltage_byte*8;  //the 5 highest bytes sets the
voltage
// Calculate correct frequency data bits to send
      // debugging only remove the following line
      // if ( 0 != kinfo.getFrequency())
      freq_period = (1000000/kinfo.getFrequency());      //microseconds
      freq_highbyte=freq_period/0x100;
```

```
      freq_lowbyte=freq_period - freq_highbyte*0x100;
      if (freq_lowbyte < 0) freq_highbyte++; // for added precision
//Calculate correct exposure bits to send
      switch (int (kinfo.getExposure()))
      {
         case 0: exposureChar=0x09; break;
         case 1: exposureChar=0x0A; break;
         case 2: exposureChar=0x0C; break;
         case 3: exposureChar=0x11; break;
         case 4: exposureChar=0x12; break;
         case 5: exposureChar=0x14; break;
         case 6: exposureChar=0x21; break;
         case 7: exposureChar=0x22; break;
         case 8: exposureChar=0x24; break;
         case 9: exposureChar=0x41; break;
         case 10: exposureChar=0x42; break;
         case 11: exposureChar=0x44; break;
         default:  ShowMessage("Incorrect Exposure value received " +
exposureChar);
      }
      sendGetsetBuffer.sprintf ("%c%c%c%c%c%c", voltage_byte,
freq_lowbyte, freq_highbyte, exposureChar, 'R', 'u');
      errno = WriteFile (hCom, sendGetsetBuffer.c_str(), 6,
&nBytesWritten, NULL);
      if (errno == 0) { ShowMessage("Error in TransmitCommChar ");
CommsTestBAD=True; }
      if (PictureRequested)
      {
        // PollRoutine->Enabled=false;
        // Send takePicture or 'p' to take a kirlian picture
        errno = TransmitCommChar (hCom, takePicture);
        Label1->Caption="takePicture";
        if (errno == 0) ShowMessage("Error in TransmitCommChar
takePicture HV enabled");
         PictureRequested = false;
      }
      if (errno == 0) { ShowMessage("Error in ReadFile");
CommsTestBAD=True; }
     // if ('t' != rxChar) { CommsTestBAD=True;
ShowMessage("!!!STOPPED!!!!"); }
   }  // endelse CommsTestBAD
   isTimerRoutineRunning = false;
   K2001Main_f->Repaint();
}  // end pollroutine
//



/*********************************************************/
/*                      NEW SECTION                      */
/*                           2                           */
/*       Kirlian Experiment Template (KET) Values        */
/*                                                       */
/* 1) This section contains all the KET values           */
/* 2) These are all GUI designer components              */
```

```
/* 3) Initial setting are configure by the GUI designer  */
/*************************************************************/

void __fastcall TK2001Main_f::Template_memoChange(TObject *Sender)
/*****************************************************************************
 *  2.1 Executed when user changes the memo Template
 *
 *    Arguments:     GUI designer Template memoChanged
 *    Effects:       Sets the Experiment Template Comment
 *    Return:        void
*****************************************************************************
/
{
   KETmodified=true;
   //kinfo.setExperimentTemplateComment( Template_memo->Lines-
>Strings[0] );
   kinfo.setExperimentTemplateComment( Template_memo->Text );
}
//

void __fastcall TK2001Main_f::Voltage_udClick(TObject *Sender,
TUDBtnType Button)
/*****************************************************************************
 *  2.2  Executed when user changes the voltage via the voltage up down
click
 *
 *    Arguments:     Voltage Up/Down Button Pressed
 *    Effects:       Sets the Voltage
 *    Return:        void
*****************************************************************************
/
{
   KETmodified=true;
   kinfo.setVoltage( Voltage_ud->Position );
}
//

void __fastcall TK2001Main_f::Frequency_udClick(TObject *Sender,
TUDBtnType Button)
/*****************************************************************************
 * 2.3  Executed when user changes Frequency via the frequency up down
click
 *
 *    There are unique implementation issues surrounding the Frequency
display
 *    a) The counting mechanism relies on the up down only
 *    b) Counting is effected by increment/decrement the last digit
only
 *    c) The Frequency variable is 10 times the actual frequency
 *    d) A label is used to indicate whether the frequency is Hz or kHz
 *
 *    Arguments:     GUI designer Frequency Up/Down Button Pressed
 *    Effects:       Sets the Frequency
 *    Return:        void
*****************************************************************************/
{
```

```
   Frequency_ud->Increment = 1;
   if (Frequency_ud->Position > 9) Frequency_ud->Increment = 10;
   if (Frequency_ud->Position > 99)  Frequency_ud->Increment = 100;
   if (Frequency_ud->Position > 999)  Frequency_ud->Increment = 1000;
   if (Frequency_ud->Position > 9999)  Frequency_ud->Increment = 10000;
/* Exceptions to the above rules */
//  Comctrls::btPrev used instead of btPrev because of Ambiguity
   switch ( OldFreqValue )
   {
   case 10:     if (Button == Comctrls::btPrev)
                {
                   Frequency_ud->Position  = 9;
                   Frequency_ud->Increment = 1;
                } else Frequency_ud->Position  = 20;
                break;
   case 100:    if (Button == Comctrls::btPrev)
                {
                   Frequency_ud->Position  = 90;
                   Frequency_ud->Increment = 10;
                } else  Frequency_ud->Position = 200;
                break;
   case 1000:   if (Button == Comctrls::btPrev)
                {
                   Frequency_ud->Position  = 900;
                   Frequency_ud->Increment = 100;
                } else Frequency_ud->Position = 2000;
                break;
   case 10000:   if (Button == Comctrls::btPrev)
                {
                   Frequency_ud->Position  = 9000;
                   Frequency_ud->Increment = 1000;
                } else Frequency_ud->Position = 20000;
                break;
   default:    break;
   } //endswitch
   kinfo.setFrequency(Frequency_ud->Position*10);
   OldFreqValue = Frequency_ud->Position;
   KETmodified  = true;
   if (kinfo.getFrequency() > 999)
   {
      Label22->Caption = "kHz";
      Frequency_ed->Text = kinfo.getFrequency()/1000;
   } else
   {
      Label22->Caption = "Hz";
      Frequency_ed->Text = kinfo.getFrequency();
   }
} // endmethod Frequency_udClick
//

void __fastcall TK2001Main_f::Ground_cbxChange(TObject *Sender)
/*****************************************************************
 *  2.4  Executed when user changes the Ground via the Ground combo box
 *
 *    Arguments:      GUI designer Ground combo box Pressed
 *    Effects:        Sets the Ground
```

```
 *    Return:        void
*******************************************************************/
{
   KETmodified=true;
   kinfo.setGround( Ground_cbx->ItemIndex );
}

void __fastcall TK2001Main_f::Exposure_cbxChange(TObject *Sender)
/*******************************************************************
 *  2.5  Executed when user changes the Exposure via the Exposure combo
box
 *
 *    Arguments:     GUI designer Exposure combo box Pressed
 *    Effects:       Sets the Exposure
 *    Return:        void
*******************************************************************/
{
   KETmodified=true;
   kinfo.setExposure( Exposure_cbx->ItemIndex );
}

void __fastcall TK2001Main_f::Lens_edChange(TObject *Sender)
/*******************************************************************
 *  2.6  Executed when user changes the Lens via the Lens edit box
 *
 *    Arguments:     GUI designer Lens edit box Pressed
 *    Effects:       Sets the Lens
 *    Return:        void
*******************************************************************/
{
   KETmodified=true;
   kinfo.setLens( Lens_ed->Text );
}
//

void __fastcall TK2001Main_f::Focal_udClick(TObject *Sender, TUDBtnType
Button)
/*******************************************************************
 *  2.7  Executed when user changes the Focal length via the Focal up
down click
 *
 *    Arguments:     GUI designer Focal up down clicked
 *    Effects:       Sets the Focal length
 *    Return:        void
*******************************************************************/
{
   kinfo.setFocal( Focal_ud->Position );
}


void __fastcall TK2001Main_f::Sensitivity_cbxChange(TObject *Sender)
/*******************************************************************
 *  2.8  Executed when user changes the Sensitivity via the Sensitivity
combo box
 *
 *    Arguments:     GUI designer Sensitivity combo box Pressed
```

```
 *    Effects:       Sets the Sensitivity
 *    Return:        void
*****************************************************************/
{
   KETmodified=true;
   kinfo.setSensitivity (Sensitivity_cbx->ItemIndex);
}

void __fastcall TK2001Main_f::Experiment_memoChange(TObject *Sender)
/*****************************************************************
 *  2.9 Executed when user changes the Experiment comment
 *
 *    Arguments:     GUI designer Experiment memoChanged
 *    Effects:       Sets the Experiment Comment
 *    Return:        void
*****************************************************************/
{
   KETmodified=true;
   kinfo.setPictureComment( Experiment_memo->Text );
}

void __fastcall TK2001Main_f::MinTemp_edChange(TObject *Sender)
/*****************************************************************
 *  2.10 Executed when user changes the Minimum Temperature
 *
 *    Arguments:     GUI designer Minimum Temperature, changed via
textbox or the up/down  click
 *    Effects:       Sets the Minimum Temperature
 *    Return:        void
*****************************************************************/
{
   KETmodified=true;
   kinfo.setMinTemp( MinTemp_ed->Text.ToIntDef(0) );
}

void __fastcall TK2001Main_f::MinAtmPress_edChange(TObject *Sender)
{
/*****************************************************************
 *  2.11 Executed when user changes the Minimum Atmospheric Pressure
 *
 *    Arguments:     GUI designer Minimum Atmospheric Pressure, changed
via textbox or the up/down  click
 *    Effects:       Sets the Minimum Atmospheric Pressure
 *    Return:        void
*****************************************************************/
   KETmodified=true;
   kinfo.setMinPress( MinAtmPress_ed->Text.ToIntDef(0) );
}

void __fastcall TK2001Main_f::MinLight_edChange(TObject *Sender)
/*****************************************************************
 *  2.12 Executed when user changes the Minimum Light Levels
 *
 *    Arguments:     GUI designer Minimum Light Levels, changed via
textbox or the up/down  click
 *    Effects:       Sets the Minimum ambient Light Levels
```

```
 *    Return:        void
 ************************************************************************/
{
   KETmodified=true;
   kinfo.setMinLight( MinLight_ed->Text.ToIntDef(0) );
}
//

void __fastcall TK2001Main_f::MinHumid_edChange(TObject *Sender)
/************************************************************************
 *  2.13 Executed when user changes the Minimum Humidity
 *
 *    Arguments:     GUI designer Minimum Humidity, changed via textbox
or the up/down  click
 *    Effects:       Sets the Minimum Humidity
 *    Return:        void
************************************************************************/
{
   KETmodified=true;
   kinfo.setMinHumid( MinHumid_ed->Text.ToIntDef(0) );
}


void __fastcall TK2001Main_f::MinObjPress_edChange(TObject *Sender)
/************************************************************************
 *  2.14 Executed when user changes the Minimum Object Pressure
 *
 *    Arguments:     GUI designer Minimum Object Pressure, changed via
textbox or the up/down  click
 *    Effects:       Sets the Minimum Object Pressure
 *    Return:        void
************************************************************************/
{
KETmodified=true;
   kinfo.setMinOpress( MinObjPress_ed->Text.ToIntDef(0) );
}


void __fastcall TK2001Main_f::MaxTemp_edChange(TObject *Sender)
/************************************************************************
 *  2.15 Executed when user changes the Maximum Temperature
 *
 *    Arguments:     GUI designer Maximum Temperature, changed via
textbox or the up/down  click
 *    Effects:       Sets the Maximum Temperature
 *    Return:        void
************************************************************************/
{
   KETmodified=true;
   kinfo.setMaxTemp( MaxTemp_ed->Text.ToIntDef(0) );
}

void __fastcall TK2001Main_f::MaxAtmPress_edChange(TObject *Sender)
/************************************************************************
 *  2.16 Executed when user changes the Maximum Atmospheric Pressure
 *
```

```
 *    Arguments:     GUI designer Maximum Atmospheric Pressure, changed
via textbox or the up/down  click
 *    Effects:       Sets the Maximum Atmospheric Pressure
 *    Return:        void
*****************************************************************/
{
   KETmodified=true;
   kinfo.setMaxPress( MaxAtmPress_ed->Text.ToIntDef(0) );
}

void __fastcall TK2001Main_f::MaxLight_edChange(TObject *Sender)
/*****************************************************************
 *  2.17 Executed when user changes the Maximum Light Levels
 *
 *    Arguments:     GUI designer Maximum Light Levels, changed via
textbox or the up/down  click
 *    Effects:       Sets the Maximum Light Levels
 *    Return:        void
*****************************************************************/
{
   KETmodified=true;
   kinfo.setMaxLight( MaxLight_ed->Text.ToIntDef(0) );
}

void __fastcall TK2001Main_f::MaxHumid_edChange(TObject *Sender)
/*****************************************************************
 *  2.18 Executed when user changes the Maximum Humidity
 *
 *    Arguments:     GUI designer Maximum Humidity, changed via textbox
or the up/down  click
 *    Effects:       Sets the Maximum Humidity
 *    Return:        void
*****************************************************************/
{
   KETmodified=true;
   kinfo.setMaxHumid( MaxHumid_ed->Text.ToIntDef(0) );
}

void __fastcall TK2001Main_f::MaxObjPress_edChange(TObject *Sender)
/*****************************************************************
 *  2.19 Executed when user changes the Maximum Object Pressure
 *
 *    Arguments:     GUI designer Maximum Object Pressure, changed via
textbox or the up/down  click
 *    Effects:       Sets the Maximum Object Pressure
 *    Return:        void
*****************************************************************/
{
   KETmodified=true;
   kinfo.setMaxOpress( MaxObjPress_ed->Text.ToIntDef(0) );
}


                    /*****************************************/
                    /*              NEW SECTION              */
                    /*                   3                   */
```

```
                    /*            Menu Options                  */
                    /*****************************************/

void __fastcall TK2001Main_f::NewExperimentTemplate1Click(TObject
*Sender)
/*************************************************************************
 * 3.1  Loads the default experiment template
 *
 *    Arguments:      GUI designer menu New Kirlian Experiment Template
(KET) Clicked
 *    Effects:        Sets all the KET values their default values
 *    Return:         void
*************************************************************************/
{
   LoadExperimentTemplate("default");
}

void __fastcall TK2001Main_f::LoadExperimentTemplate1Click(TObject
*Sender)
{
/*************************************************************************
 * 3.2  Load a previously saved Kirlian experiment template (KET)
 *
 *    Arguments:     GUI designer menu File -> Load Kirlian Experiment
Template Clicked
 *    Effects:        Opens a dialogue box to load a KET
 *    Return:         void
*************************************************************************/
      if (OpenDialog->Execute())
      {
      FileNameKET = OpenDialog->FileName;
            OpenDialog->Options.Contains(ofReadOnly);
      KDataOnFile = new TFileStream (OpenDialog->FileName,
fmOpenReadWrite|fmShareCompat);
      KDataOnFile->Read(&kinfo, sizeof(kinfo));
      KETmodified=false;
      delete(KDataOnFile);
      BaseDataDir = FileNameKET.Delete(FileNameKET.LastDelimiter(".") ,
4);
      getKDataFilename (); // get the new filename NOT experiment
Template name
      DisplayKInfo(kinfo); // Display the new info
      }
}

void __fastcall TK2001Main_f::SaveExperimentTemplate1Click(TObject
*Sender)
{
/*************************************************************************
 * 3.3  Saves the existing KET values into a user specified *.ket file
 *
 *    Arguments:      GUI designer menu File -> Save Kirlian Experiment
Template (KET) Clicked
 *    Effects:        Opens a dialogue box to save a KET
 *    Return:         void
*************************************************************************/
```

```
   String str;
      TVarRec vrs[1];

      if (SaveDialog->Execute())
      {
            if (FileExists(SaveDialog->FileName))
            {
                  str = FmtLoadStr(sOverwrite, OPENARRAY(TVarRec,
(SaveDialog->FileName)));
                  if (MessageDlg(str, mtConfirmation, TMsgDlgButtons()
<< mbYes << mbNo <<
                        mbCancel, 0) != IDYES)
                        return;
       }
       KDataOnFile = new TFileStream (SaveDialog->FileName,
fmCreate|fmShareCompat);
       KDataOnFile->WriteBuffer(&kinfo, sizeof(kinfo));
      // Label26->Caption = Format (" %d save ",
ARRAYOFCONST(((int)sizeof(kinfo))));
       FileNameKET = SaveDialog->FileName;
       KETmodified=false;
       delete(KDataOnFile);
       // Create Data directory
       BaseDataDir = FileNameKET.Delete(FileNameKET.LastDelimiter(".")
, 4);
       if (!CreateDir (BaseDataDir))
       {
         throw Exception(FileNameKET);
       }

       getKDataFilename (); // get the new filename NOT experiment
Template name
       DisplayKInfo(kinfo); // Diplay the new info
      }
}

int __fastcall TK2001Main_f::LoadExperimentTemplate(AnsiString
ATemplate)
/*************************************************************************
 * 3.4  Load the default experiment template
 *
 *    Arguments:     GUI designer menu File -> New Kirlian Experiment
Template (KET) Clicked
 *    Effects:       Sets all the KET values their default values
 *    Return:        void
*************************************************************************/
{
   if (ATemplate == "default")
   {
      kinfo.setDateTime();
      kinfo.setPictureFile("default.bmp");
   //   Label27->Caption=kinfo.getPictureFile();
      kinfo.setExperimentTemplateComment( "Please enter comments
relating to overall experiment here!" );
      kinfo.setPictureComment(" A comment about the this picture!");
      kinfo.setVoltage( 21 );
```

```
      kinfo.setFrequency( 1000 );
      OldFreqValue = kinfo.getFrequency( )/10; // No not change this
divison
      kinfo.setGround( K_DIRECT );
      kinfo.setExposure( 2 );
      kinfo.setLens( "Normal Macro Lens" );
      kinfo.setSensitivity( 1 );
      kinfo.setFocal( 23 );
      kinfo.setFStop( 3 );
      kinfo.setMaxTemp( 30 );
      kinfo.setMaxPress( 105 );
      kinfo.setMaxHumid( 40 );
      kinfo.setMaxLight( 50 );
      kinfo.setMaxOpress( 30 );
      kinfo.setMinTemp( 5 );
      kinfo.setMinPress( 45 );
      kinfo.setMinHumid( 0 );
      kinfo.setMinLight( 3 );
      kinfo.setMinOpress( 30 );
   } // endif ATemplate == "default"
   KETmodified=false;
   getKDataFilename();
   DisplayKInfo(kinfo);
   return((int)true);
}  //method LoadExperimentTemplate


void __fastcall TK2001Main_f::Exit_butClick(TObject *Sender)
/*************************************************************************
 * 3.4  Exit the program gracefully
 *
 *    Arguments:     GUI designer Exit Button Clicked
 *    Effects:       Exit the program immediately if the variables that
makes up the KET did
 *                   not change else asked user what to do
 *    Return:        void
 *************************************************************************/
{
   if (KETmodified)
   {
      if (MessageDlg("Experiment Template is not saved, Still Exit",
mtConfirmation,
                        TMsgDlgButtons() << mbCancel << mbYes , 0) ==
mrYes)
      {
         // Yes Pressed Terminate program
         Application->Terminate();
      } else
      {
         // Cancelled pressed, return to Program
         return;
      }
   } else Application->Terminate();
} // endmethod Exit_butClick
//
```

```
void __fastcall TK2001Main_f::Exit1Click(TObject *Sender)
/*************************************************************************
 * 3.5  Menu handler for Exit Option
 *
 *    Arguments:      GUI designer menu: File -> Exit Clicked
 *    Effects:        calls Exit_butClick
 *    Return:         void
 *************************************************************************/
{
   Exit_butClick(Sender);
}
//

void __fastcall TK2001Main_f::FormClose(TObject *Sender, TCloseAction
&Action)
/*************************************************************************
 * 3.6  Handler for Main Form  X clicked
 *
 *    Arguments:      GUI designer Form  X clicked
 *    Effects:        calls Exit_butClick
 *    Return:         void
 *************************************************************************/
{
   Exit_butClick(Sender);
}


void __fastcall TK2001Main_f::SerialPort2Click(TObject *Sender)
/*************************************************************************
 * 3.7  Opens a new form showing the serial port selected
 *    A different serial port can be selected
 *
 *    Arguments:      GUI designer menu: Options -> Serial Port Clicked
 *    Effects:        Opens Serial Port form
 *    Return:         void
 *************************************************************************/
{
   PollRoutine->Enabled=false;
   Serial_f->ShowModal();
   PollRoutine->Enabled=true;
}

void __fastcall TK2001Main_f::StatusBarContextPopup(TObject *Sender,
TPoint &MousePos, bool &Handled)
/*************************************************************************
 * 3.8  Right clicking the Statusbar also brings up the Serial port
 *    selection form
 *************************************************************************/
{
    SerialPort2Click(Sender);
}

commport __fastcall TK2001Main_f::GetCommPort(void)
/*************************************************************************
 * 3.9  Get the current communication port setting
 *
```

```
 *    Arguments:      void
 *    Effects:        An accessor method which returns the
Communications Port used
 *    Return:         The communications port used
 ********************************************************************/
{
   return(ActiveCommPort);
}

bool __fastcall TK2001Main_f::SetCommPort(commport thisPort)
/********************************************************************
 * 3.10  Set the current Communications port, as specified by the user.
 *
 *    Arguments:      Attempt to set the port to this value
 *    Effects:        A mutator method which attempts to set the
Communications Port used
 *    Return:         true if successful
 ********************************************************************/
{
   commport CommPort=thisPort;
   DCB dcb;
   DWORD dwError;
   bool fSuccess;
   AnsiString ComStr="COM5";
   CommsTestBAD=True;      /* SetCommPort main objective */
   ActiveCommPort=thisPort;
   CommsPanel->Color=clRed;
   CommsPanel->Caption="COMMUNICATION STATUS: Still to be tested";
   switch (CommPort)
   {
      case 0:     ComStr="COM1"; break;
      case 1:     ComStr="COM2"; break;
      case 2:     ComStr="COM3"; break;
      case 3:     ComStr="COM4"; break;
   }
   if (hCom != INVALID_HANDLE_VALUE) CloseHandle(hCom);
   hCom = CreateFile (
                        ComStr.c_str(),
                        GENERIC_READ | GENERIC_WRITE,
                        0,    /* sharemode: comm devices must be opened
w/exclusive-access */
                        NULL, /* no security attrs */
                        OPEN_EXISTING, /* comm devices must use this */
                        0,    /* not overlapped I/O */
                        NULL  /* hTemplate NULL for comm devices */
                      );
   if (hCom == INVALID_HANDLE_VALUE)
   {
      dwError = GetLastError();
CommsPanel->Color=clRed;
      CommsPanel->Caption=Format("COMMUNICATION: Port not Available
      ERROR %d", ARRAYOFCONST(((int)dwError)));        return(false);
   }
   /* Omit the call to SetupComm to use the default queue sizes.  * Get
the current configuration.  */
   fSuccess = GetCommState(hCom, &dcb);
```

```
   if (!fSuccess)
   {        /* Handle the error. */
CommsPanel->Color=clRed;
      CommsPanel->Caption="COMMUNICATION: Cannot read1 Port state";
return (false);
   } /* Fill the DCB: baud=9600, 8 data bits, no parity, 1 stop bit. */
   dcb.BaudRate = 9600;       //9600
   dcb.ByteSize = 8;          //5
   dcb.Parity = NOPARITY;           //NOPARITY
   dcb.StopBits = ONESTOPBIT;       //ONESTOPBIT
   fSuccess = SetCommState(hCom, &dcb);
   if (!fSuccess)
   {    /* Handle the error. */
CommsPanel->Color=clRed;
      CommsPanel->Caption="COMMUNICATION: Cannot read3 state";
      return ((int)false);
   }
/* Timeouts implemented set too 50ms provinsionally*/
   COMMTIMEOUTS lpCommTimeouts;
   GetCommTimeouts (hCom, &lpCommTimeouts);
   lpCommTimeouts.ReadIntervalTimeout=1;
   lpCommTimeouts.ReadTotalTimeoutMultiplier=1;
   lpCommTimeouts.ReadTotalTimeoutConstant=5;
   lpCommTimeouts.WriteTotalTimeoutMultiplier=1;
   lpCommTimeouts.WriteTotalTimeoutConstant=1;
   SetCommTimeouts (hCom, &lpCommTimeouts);
   //ClearCommError
   char received;
   DWORD nBytesToRead=1;
   DWORD nBytesRead;
   //WriteFile (hCom, to_send, nBytesToWrite, &nBytesWritten, NULL);
   errno = TransmitCommChar (hCom, commsCheck);
   if (errno == 0) ShowMessage("Error in TransmitCommChar 1");
   errno = ReadFile (hCom, &received, nBytesToRead, &nBytesRead, NULL);
   if (errno == 0) ShowMessage("Error in ReadFile 1");
   CommsTestBAD = (received != 'r');
   if (CommsTestBAD)
   {
CommsPanel->Color=clRed;
      CommsPanel->Caption="COMMUNICATION: Kirlian Device not
      Responding";
   } else
   {
CommsPanel->Color=clLime;
CommsPanel->Caption="COMMUNICATION: Kirlian Responding";
   }
   return(true);
} // endmethod SetCommPort

void __fastcall TK2001Main_f::ViewExperiment1Click(TObject *Sender)
/*****************************************************************
 * 3.11  Opens a new form showing previous experiments
 *
 *    Arguments:  GUI designer menu: Options -> View Experiment Clicked
 *    Effects:    Opens the Data and Picture Viewer form
 *    Return:     void
```

```
 ***********************************************************************/
 {
   PollRoutine->Enabled=false;
   SetCurrentDir( BaseDataDir );
   DataPictureView_f->ShowModal();
   SetCurrentDir( ProgramCurrentDir );
   PollRoutine->Enabled=true;
 }

void __fastcall TK2001Main_f::EnableHVwithoutCameraClick(TObject
*Sender)
/***********************************************************************
 * 3.12    Toggle the availability of the high voltage system
 *
 *    Arguments:     GUI designer menu: Options -> Enable HV without
Camera Clicked
 *    Effects:       Toggle the availability of the high voltage system
even though the camera
 *                   might not be available
 *    Return:        void
 ***********************************************************************/
{
   EnableHVwithoutCamera->Checked = !(EnableHVwithoutCamera->Checked);
}

void __fastcall TK2001Main_f::About1Click(TObject *Sender)
/***********************************************************************
 * 3.13 Opens the About form
 *
 *    Arguments:     GUI designer menu: Help -> About Clicked
 *    Effects:       Opens the About form
 *    Return:        void
 ***********************************************************************/
{
   PollRoutine->Enabled=false;
   AboutBox->ShowModal();
   PollRoutine->Enabled=true;
}

void __fastcall TK2001Main_f::HelpAbout1Execute(TObject *Sender)
/***********************************************************************
 * 3.14  Opens the About form
 *
 *    Arguments:     GUI designer menu: Help -> About Clicked
 *    Effects:       Opens the About form
 *    Return:        void
 ***********************************************************************/
{
   About1Click(Sender);
}

void __fastcall TK2001Main_f::WhatisKirlianPhotography1Click(TObject
*Sender)
/***********************************************************************
 * 3.15  Opens a new form showing thepens a new form showing Help
Details
```

```
 *
 *    Arguments:      GUI designer menu: Help -> What Is Kirlian
Photography Clicked
 *    Effects:        Opens the WhatIs  form
 *    Return:         void
 ************************************************************************/
{
   PollRoutine->Enabled=false;
   WhatIs_f->ShowModal();
   PollRoutine->Enabled=true;
}
```

LIST OF REFERENCES

BOXLER, C., & PAULSON, M.; 1977: *Kirlian photography – a new tool in biological research.* Journal of the Biological Photographic Association. 45(2) pp. 51 – 60.

CHOUHAN, P. S.; June 1989: *Bioelectrographic images in normal subjects and patients with cervical cancer.* A thesis submitted to the International Institute of Integral Human Sciences, Montreal, Quebec Canada, in fulfilment of the requirements for the degree of Doctor of Philosophy. Abstract from the Kirlian Institute.

DAKIN, H. S.; 1975: High voltage photography, Barrington, New Jersey: Edmund Scientific Company.

ELLISON, A. J.; 1982: *Kirlian photography.* In I. Guinness (Ed.), Psychical Research Wellingborough, England: Aquarian Press. pp. 188 – 192.

ERICKSON, R.; 1997: Fundamentals of Power Electronics. New York: Chapman and Hall, May 1997.

GADSBY, J. G.; : *Kirlian photography diagnosis – a recent study,* Complementary Therapies in Medicine

GADSBY, J. G.; 1991: *Kirlian photography: a critical analysis*, Complementary Therapies in Medicine, 5(1), pp. 23 – 28.

HEELEY, D.; 1996: *Understanding Pressure and Pressure Measurement,* Motorola Semiconductor Application Note, AN1573.

HOOVER, G., & REMPFER, W.; May 1988: *Electrically Isolating Data Acquisition Systems*, Design Notes number 10, Linear Technology Corporation.

HOSSIAN, E.; 1995: *Kirlian Photography – more than meets the eye? : Bioenergy update,* In SA family practice magazine. 16:12 pp. 812 – 821

HOSSIAN, E.; 1999: Personal interview of Hossian, E by Williams, S. J. S. at MEDUNSA Pretoria, South Africa.

IOVINE, J.; 1993: Kirlian photography: A Hands-on Guide. Boston, Massachusetts: TAB Books.

JOHNSON, K.; 1975: The Living Aura: Radiation Field Photography and the Kirlian Effect. New York: Hawthorn Books.

KIRLIAN, S. D., & KIRLIAN, V. Kh.; 1973: *Photographic and visual observation by means of high frequency currents,* Proceedings of the First Conference on Psychotronic Research Prague.

KONIKIEWICZ, L.; 1971: *Kirlian photography in theory and clinical application,* Journal of the Biological Photographic Association. Volume 45, Number 3,

KONIKIEWICZ, L.; 1979: Introduction to electrography. Harrisburg, Pennsylvania: Leonard's Associates

KRIPPNER, S.; 1974: *International developments in biological energy,* In S. Krippner & D. Rubin (Eds.). The Energies of Consciousness. New York: Gordon and Breach. (pp. 1 - 37)

KRIPPNER, S.; 1978: *Preface,* In M. Davis & E. Lane. Rainbows of Life. New York: Harper & Row. (pp. 13 - 15)

KRIPPNER, S., & RUBIN, D. (Eds.); 1974: Galaxies of Life. New York: Gordon & Breach, 1973. Second edition title: The Kirlian Aura. Papers from the First Western Hemisphere Conference on Kirlian Photography, Acupuncture, and the Human Aura. Garden City, New York: Doubleday.

KROPUENSKE, G.; 1993: *Understanding Horizontal Stages of Multi-Frequency CRT Video Displays,* Sencore News. September 1993. (pp. 19 – 23)

KOROTKOV, K.; 2001: *Advances in Diagnosis and Monitoring of the human quantum informational field with GDV technique,* ParaDigm 2001: Consciousness and Paranormal Phenomena. http://www.rajatiede.org/paradocs/tutkimuksia/paradigma2001.html

MENCHENIN, S.; 2003: *Cam2Com allows to control Olympus digital cameras from a computer and to take pictures directly onto the hard drive.* http://sabsik.com/Cam2Com

MOSS, T., & JOHNSON, K.; 1974: *Bioplasma or corona Discharge?* Galaxies of Life. New York: Gordon & Breach. (pp. 29 – 51)

MOSS, T., JOHNSON, K., & GRAY, J.; 1975: *Now you see it, now you don't,* In S. Krippner & D. Rubin (Eds.). The Energies of Consciousness. New York: Gordon and Breach. (pp. 41 - 74)

MOSS, T.; 1981: The body electric – a personal journey into the mysteries of parapsychology and Kirlian photography. St Albans, Granada.

NATIONAL SEMICONDUCTOR; OCTOBER 1986: *Application Note 460 LM34/LM35* Precision Monolithic Temperature Sensors

OHSMANN; 1997: *The 80c535 Single Board Computer.* Elektor Electronics. June 1997

OLYMPUS; 2003: Camedia Software Developer's Kit. Olympus SDK, Melville, NY

PEHEK, J. C., KYLER, H. J., & FAUST, D. L.; 1976: Image modulation in corona discharge photography. Science: 194(4262) (pp. 263 – 270)

PHILIPS ELECTRONICS; 1997: *Moisture control with Philips' humidity sensor.* Philips Electronics N.V.

PHUNG, B. T., BLACKBURN, T. R, HSU, J. & LEE, H. P.; 2001: *Study of partial discharges using Kirlian images.* Australasian Universities Power Engineering Conference (AUPEC) Melbourne, Australia 29th September to 2nd October 2002

POLEN, E.; 1981: *Power Supply Design.* Voltage Regulator Handbook. National Semiconductor Corporation. (pp. 8-3 to 8-6)

RUSSO M., CHOUDHRI F., WHITWORTH G., WEINBERG A., BICKEL W., & MEHMET C.; 2001: *Quantitative Analysis of Reproducible Changes in High-Voltage Electrophotography.* The Journal of Alternative & Complementary Medicine Volume 7 Number 6 December 2001

SCHROEDER, C.; 2001: *PSpice Non-Linear Magnetics Model,* Beyond Designs, http://www.beyond-designs.com/pspice_inductive.htm.

SEMICONDUCTOR GROUP; 1991: *Siemens Microcomputer Components. SAB 80515/ SAB 80C515 8-Bit Single-Chip Microcontroller Family: User's Manual.* Published by Siemens AG, Bereich Halbleiter, Marketing-Kommunikation, Balanstrabe 73, D-8000 Munchen 80. Edition 5.93

SLIWCZYNSKI, L., & KREHLIK, P.; 1998: *25-kV generator tests insulation,* EDN Magazine: MA, (pp. 122 - 123) 7th MAY 1998

SQUILLANTE, M. R., & SHAH, K. S.; 1999: *Photojunction Sensors,* In Webster J. G.'s ed. Measurement, Instrumentation, and Sensors Handbook. CRC Press LLC Corporate Blvd., N.W

TILLER, W. A.; 1975: *The light source in high-voltage photography,* In Krippner, S. & Rubin, D. (Eds.). The Energies of Consciousness. New York: Gordon and Breach. (pp. 100 - 156)

TUINENGA, P. W.; 1988: Spice – A guide to circuit simulation & Analysis using PSpice. New Jersey: Prentice Hall.

WALTON, R. S.; 1977: *Science or nonsense? Don't discount the value of Kirlian photography,* Biomedical Communications, 5(5), (pp. 16 – 28)

WINKLER, C., & BAUM, J.; 1997: *Barometric Pressure Measurement Using Semiconductor Pressure Sensors,* Motorola Semiconductor Application Note, AN1326

WLLIAMS, S. J. S., & KHAN, M. T. E.; 2003: *Electro Photographic Instrument for use in Kirlian Photographic Phenomena Research,* Quantum Test & Measurement, (pp. 19 – 21)

BIOGRAPHICAL SKETCH

Sydwell Williams started his career at a semi-government research organization (Council for Mineral Technology or Mintek). His evolvement up to his departure, was sole electronic technologist in charge of designing and commissioning a range of analytical instrument. He then moved to Cape Town and started a career as an network technician and in was transferred to the Multimedia Department in the capacity as a lecturer. Sydwell has self funded his research into Kirlian photography which he seriously embarked upon in 1997.