Work Report for Rakon Limited

Employer Details:
Name of Company: Rakon Limited
Address: 01 Pacific Rise, Mt. Wellington, Auckland
Telephone: 5735554
Supervisor's Name: Mr. Andrew Daken
Department: Engineering

Employee Details:
Name: Thusitha Dananjaya De Silva Mabotuwana
Student ID: 9790416
Department: Electrical and Electronic / Computer Systems
Number of hours completed: Sub Professional - 400 hours

Table of Contents

List of figures

1.0 Introduction

1.1 Introduction to Rakon Limited

Founded in 1967, Rakon limited is a manufacturer of high quality crystal oscillators. Since then they have seen their market grow exponentially and Rakon has been nimble enough to adapt itself to supply the changing needs of the customers. Now Rakon has production facilities in Australia and Singapore to supply the rapidly expanding Asian market and employs over 400 people in New Zealand. Rakon has a wide range of products and customers and supplies about 65% of the world's high precision oscillators.

1.2 What is covered in this report

This report covers a detailed overview of the company, its people, products and facilities and a summary of the work I was involved in, followed by general comments about the work from a subjective point of view and conclusions. It contains confidential information of Rakon Limited and therefore should not be used for any purpose other than what it is intended for. Any part of this report should not be reproduced without the written consent of the author and Rakon Limited.

2.0 Rakon Limited

2.1 Company History

Founded in 1967 by New Zealander Warren Robinson, Rakon Limited is at the forefront of New Zealand's high-tech industry at present. It was soon obvious that Rakon was poised to supply the rapidly expanding market for crystal oscillators. The Company's growth has been tremendous over the last 30 years. In 1972, Rakon Singapore was established to supply the growing Asian markets.

Over the years the needs of the customers have changed. In the 1980s with the growth of the mobile telephone technology, the industry required high precision and high stability oscillators. Rakon approached this demand by developing Temperature Compensated Crystal Oscillators (TCXOs). The development work continued in Rakon through the 1980s and eventually Rakon was hailed as the market leaders in crystal oscillator technology with unsurpassed frequency and temperature stability and miniaturisation. Among Rakon's early high volume customers were NEC Australia in 1989 and Rockwell in 1981 while Motorola, Nortel Networks, Trimble, Rockwell, Raytheon, Tyco, IBM, Conexant, Metricom, Garmin, Nokia, Adaptive, Creative, Samsung, Qualcomm, Ericsson, Magellan, and Hewlett Packard are among the major customers at present. In 1994 Rakon moved into their first purpose built facility in Mt.Wellington with a promise to deliver higher standards and quantities. With a state of the art clean room, Rakon developed their RSX crystals in 1995 and in 1996 was awarded ISO9002 accreditation. Subsequently in 1997, Rakon achieved ISO9001 accreditation for their TCXOs and began production on a mass scale.

2.2 Company mission, philosophy and values

In general Rakon's focus is to maintain their position as the producer of the highest quality temperature compensated crystal oscillators. They are targeting the fast growing high-tech industries such as GPS and cellular phone industries as their customers. They have a strong commitment towards the environment, quality of products and the society. They also strive to have good relationships with their staff, customers and suppliers. The success of the company to date is also based on good planning and readiness to adapt to new markets. For a full copy of Rakon's mission statement refer to Appendix 1.

2.3 Production facilities and structure

Rakon Limited has its main production facility in Mt.Wellington, Auckland, which is where I worked (Figure 01). There were approximately 450 employers in mid February, but this is expected to increase to about 500 by the end of 2003. The Mt.Wellington base is comprised of the purpose built Rakon building and two adjacent building that have been leased. The reason to move into leased building is that the company has grown unexpectedly and was finding it hard to cope with. Rakon is making plans for another purpose built site with a new clean room and millions of dollars worth of equipment. Rakon's production is separated into different areas. What separates the areas are the nature of the products and the technology used in the manufacturing process. All the crystal processing and assembly is done in the main building where the clean room is used.

Figure 01: Rakon's Auckland Production Facility

The clean room (Figures 2,3) contains laser welders for encasing crystals within metal cans, pick and place machines to place components on PCBs, chemical etching machines to precisely control the thickness of the crystal flakes and a multitude of testing equipment.

Figure 02: Employees working in the high-tech clean room

Figure 03: Employees working in high-tech clean room

Rakon's main products are the TCXOs. These are manufactured in buildings two and three. All TCXOs other than the IC TCXOs are produced in the same process using the same machines. There is a Surface Mounting Department (SMD) which does all the PCB assembly, soldering and other related processes. They have many pick and place machines to apply solder paste and to place the components and reflow ovens to complete the soldering process.

The Product Testing Department is separate and tests the units for frequency and temperature stability. They have ovens that are cooled using liquid nitrogen or carbon dioxide to test the products across a large temperature range. The testing ovens (Figure 04) are all designed and built on site as is the case with most of the testing equipment used.

Figure 04: Crystal and oscillator testing ovens

The IC TCXO is a different type of product. It applies temperature compensation to the oscillators via a programmable chip capacitor. This is an extremely high volume product and Rakon is capable of producing about 500,000 of these extremely high precision products in a month. Since these products require very few, yet highly sophisticated machines, they are produced independently of other products. There is also a Custom Products Department which designs and builds customised products for customers in low volumes. Most of the assembly in the Custom Products Department is done manually due to low volume.

2.4 Management structure

Rakon's complete management structure is shown in Appendix 2. Basically it is a fairly flat structure where any engineer would only have one person to report to. Senior management has very well defined roles and clearly knows whom they have to manage. The factory operators report to their individual Team Leaders about day to day business activities and operations but may report to a senior manager for any other work related issues.

2.5 Employee relations, work conditions and company culture

Rakon has quite a relaxed environment to work in. All daytime employees work eight and a half hour shifts including a thirty-minute lunch break and two ten-minute coffee breaks which they may take at anytime they wish. The staff is provided with a lunch room with microwaves, refrigerators, a sink and clean cutlery and crockery. This common room is always well stocked with tea, coffee and sugar.

Production happens around the clock at Rakon. There are three shifts in the production department and assembly lines, namely 7 am – 3.30 pm, 3.30 pm – 12 mn and 12 mn – 7 am and employees may arrange for hours that suit them with the team leader's and/or manager's approval. The production lines have a majority of women employed and have a multicultural environment. There are two male and female toilets per building. Employees go through basic induction training before they begin working and may go through other in-house training programs if they show interest. This gives the employees the opportunity to up-skill and increase their value within the company. The starting rate for an operator is $11.60 per hour, but each employee is reviewed and given a pay raise accordingly every six months.

The engineers are on salary basis and have very flexible working hours, but normal hours are 8 am – 4.30 pm. There are very strong company cultures and subcultures. For example, whenever an employee's leaving the job or else gets promoted, a collection is made and a gift is given to that person who's leaving. There are sports events arranged between departments as well, which include sports activities such as rugby, soccer and cricket. Also when a shipment or some other target is achieved under a tight time schedule, the workers are given a small party to show the management's thankfulness for all the hard work they've done thus giving the employees real encouragement and appreciation. All in all the company has a very friendly environment where people help each other and I think this is one of the key issues to Rakon's success.

2.6 Marketing strategy and competitiveness

All Rakon products are made to meet orders from customers. They have customer representatives who deal with delivery schedules and work closely with production. When an order is placed with Rakon, the Customer Products Department designs and makes a small number of units and if the customer is satisfied the design gets approved and passed onto the Production Department for high volume production.

Rakon's success is partly because of their strategies. They undertake orders for oscillators with specifications that other producers around the world would not undertake. The reason is that Rakon is willing to rework units that failed the initial tests. For some of the tighter specification units, after the automated production, the pass rate from the final tests is as low as 30%. This means 70% would either have to be thrown away or reworked. Rakon chooses to rework them by manually removing the offending components and replacing them in an effort to make the units fit within the specifications. This means that the people who are reworking the units need to have a thorough understanding of what is required to change the characteristics of the units and therefore make them fit the specifications better. It is because of this added value that Rakon can produce units of very high standards. Although it increases the costs, because Rakon is the only company that would undertake such orders, customers are charged high prices for such Rakon products.

2.7 Products

Rakon produces a number of high precision products. In the subsections that follow, a few of the main products are briefly outlined.

2.7.1 Crystals

A major part of Rakon's production facilities are devoted to the manufacture of crystals with extremely accurate frequencies. The frequency of the crystal is highly dependent on the ambient temperature. The crystal may be modelled as impedance with a capacitor in parallel with a capacitor, resistor and an inductor connected in series. The production process of a crystal is shown in the flowchart below (Figure 05).

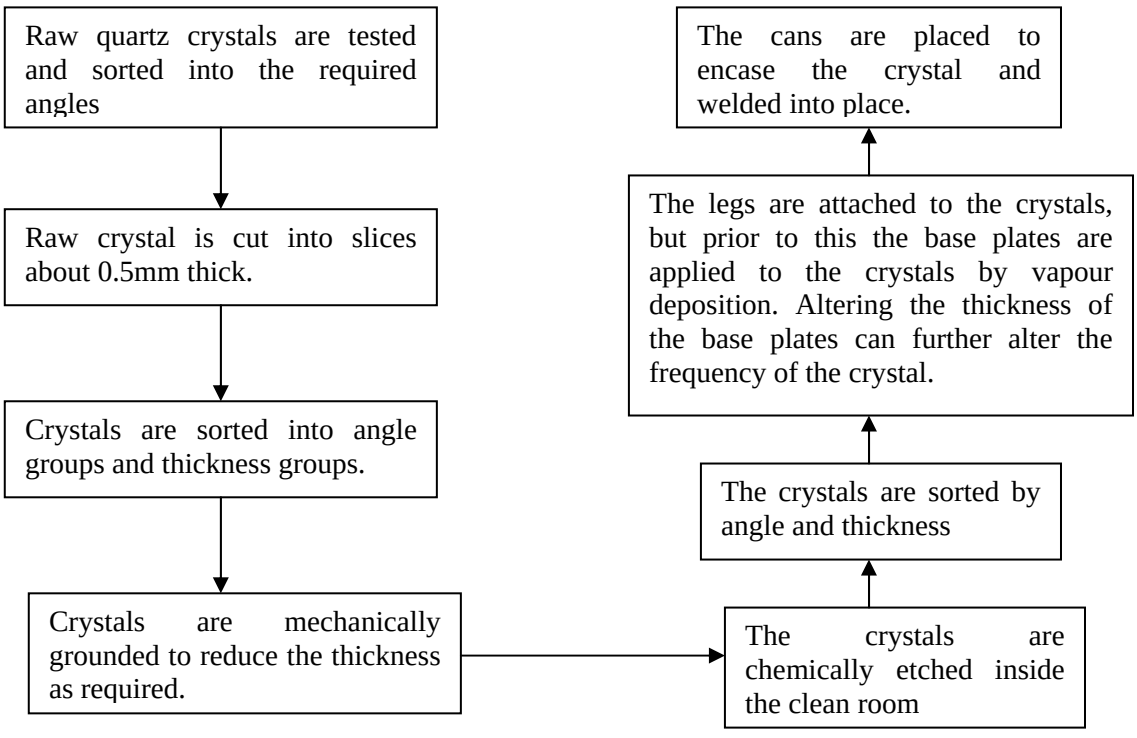


Figure 05: Production process of a crystal

Rakon uses their own crystals in the production of about 95% of the oscillators. The RSX crystals are the newest product Rakon produces and these oscillators are surface mountable and are extremely low profile. For a more detailed description of the process please visit the Rakon website (Reference 1).

2.7.2 Oscillators

Rakon uses their crystals to produce oscillators that produce a stable frequency sinusoidal output. The frequency of such oscillators is dependent on the ambient temperature and therefore has limited uses in industry. Exciting the piezo electric quartz crystals which have been cut to the correct thickness and angle produces the required oscillation. The crystal is surrounded by a network of capacitors and resistors which produces a stable frequency.

2.7.3 Temperature Compensated and Voltage Controlled Crystal Oscillators (TCXOs and VCTCXOs)

The temperature compensation is applied to the oscillator by making the capacitance of the circuit change with respect to temperature. A network of capacitors and thermistors does this. The voltage controlled TCXOs use a varactor diode to alter the capacitance by the voltage applied to it.

2.7.4 IC based TCXOs

This product is extremely high volume. The reason is that this product is tested and produced in the same automated process. The crystals are tested in the ovens for temperature characteristics and a programmable chip capacitor is used to apply the correct capacitance into the thermistors and capacitor network. Since the network is tailored to the individual characteristics of the crystals on the oscillator, the stability of these products is extremely high.

3.0 Work I was involved in

I was mainly involved in the Output Level, Harmonic, Phase Noise and Micro-Jump Testing and was under the mentorship of Andre Daken. A photograph of my work area is shown below.

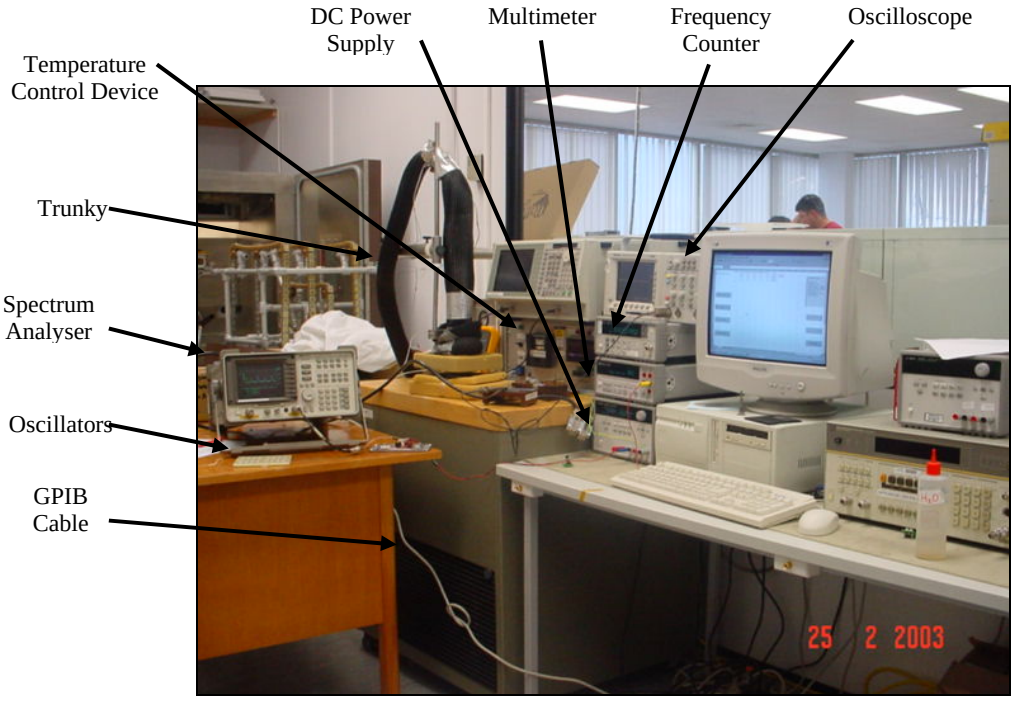
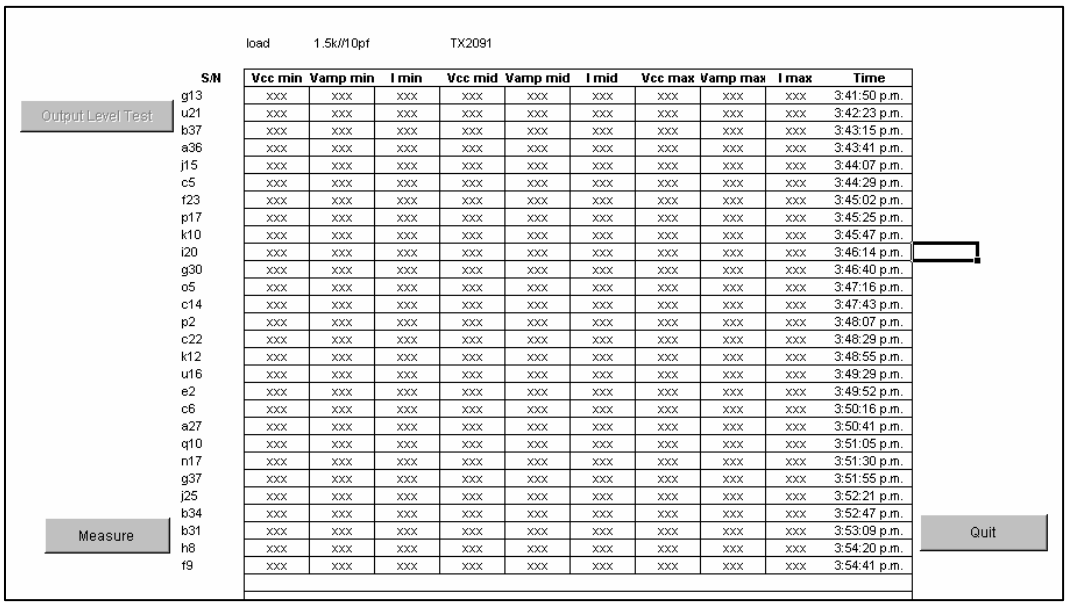


Figure 06: Area I was working in

3.1 Output Level Testing

The main idea of carrying out this test is to ensure that the oscillators have a minimum specified output voltage at the specified minimum supply voltage. The required output voltage, minimum and maximum supply voltages and other relevant parameters are given by the customer in the product specification details. The test is generally carried out at three different temperatures, namely -40°C, 25°C and 85°C but these values could vary if a customer has requested otherwise.

The measurements to be taken during the test were supplied voltage, current drawn and the peak-to-peak voltage. A DC power supply, digital multimeter and an oscilloscope were used for this purpose and a trunky was used to maintain the required temperature. Although very accurate temperature controlled ovens along with programmed instruments were used in the IT Department to do output level tests for largely produced units, only a basic Microsoft Excel Macro using a GPIB interface was being used for the test by the Engineering and Customer Products Departments due to fewer quantities. A screen shot of the macro that was being used is shown below (Figure 07). Please note that actual values cannot be shown due to confidential reasons.



S/N	Vcc min	Vamp min	I min	Vcc mid	Vamp mid	I mid	Vcc max	Vamp max	I max	Time
g13	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	3:41:50 p.m.
u21	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	3:42:23 p.m.
b37	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	3:43:15 p.m.
a36	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	3:43:41 p.m.
j15	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	3:44:07 p.m.
c5	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	3:44:29 p.m.
f23	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	3:45:02 p.m.
p17	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	3:45:25 p.m.
k10	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	3:45:47 p.m.
i20	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	3:46:14 p.m.
g30	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	3:46:40 p.m.
o5	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	3:47:16 p.m.
c14	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	3:47:43 p.m.
p2	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	3:48:07 p.m.
c22	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	3:48:29 p.m.
k12	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	3:48:55 p.m.
u16	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	3:49:29 p.m.
e2	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	3:49:52 p.m.
c6	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	3:50:16 p.m.
a27	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	3:50:41 p.m.
q10	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	3:51:05 p.m.
n17	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	3:51:30 p.m.
g37	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	3:51:55 p.m.
j25	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	3:52:21 p.m.
b34	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	3:52:47 p.m.
b31	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	3:53:09 p.m.
h8	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	3:54:20 p.m.
f9	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	3:54:41 p.m.

Figure 07: Screen shot of the macro used for Output Level Test

3.1.1 Drawbacks of the Output Level software used

- The program crashed if any of the instruments were missing or not properly connected.
- If an instrument was unplugged or turned off after starting the test, the program would crash.
- Although an average of values was considered using the minimum, mid and maximum voltages as set out in the specification sheets, these voltages could not be changed by a normal testing person with no programming knowledge in Visual Basic (VB). Therefore the tester always had to seek for a programmer's help whenever these values needed to be changed.
- There was no way of undoing measured values and changing the data input row. For example, if the terminals of the oscillator weren't connected properly to the test jig, the data sent through would be mere noise present and the reading needed to be taken again. The practice was to get data using the macro as usual and at the end of the batch, the row with invalid data deleted and other rows moved upwards. This was quite a drawback when considering even a hundred units.
- The file couldn't be closed after the test started and continue from the last stopped row since the macro would always start entering data from the very first row. If there were too many

units that couldn't be tested in one day, the practice was to do them in two separate files and then copy the data into one file at the end.

- The buttons of the macro were placed at a fixed position thus scrolling up to the very first page had to be done every time there were more units than Excel rows that would fit into one page.
- Other than these major drawbacks there were problems with large file size and text formatting since numbers as well as times and dates were also included in the Excel sheet.

As a solution to this I started developing a fully functional system using Visual Basic and the user interface of this new system is shown below in Figure 08.

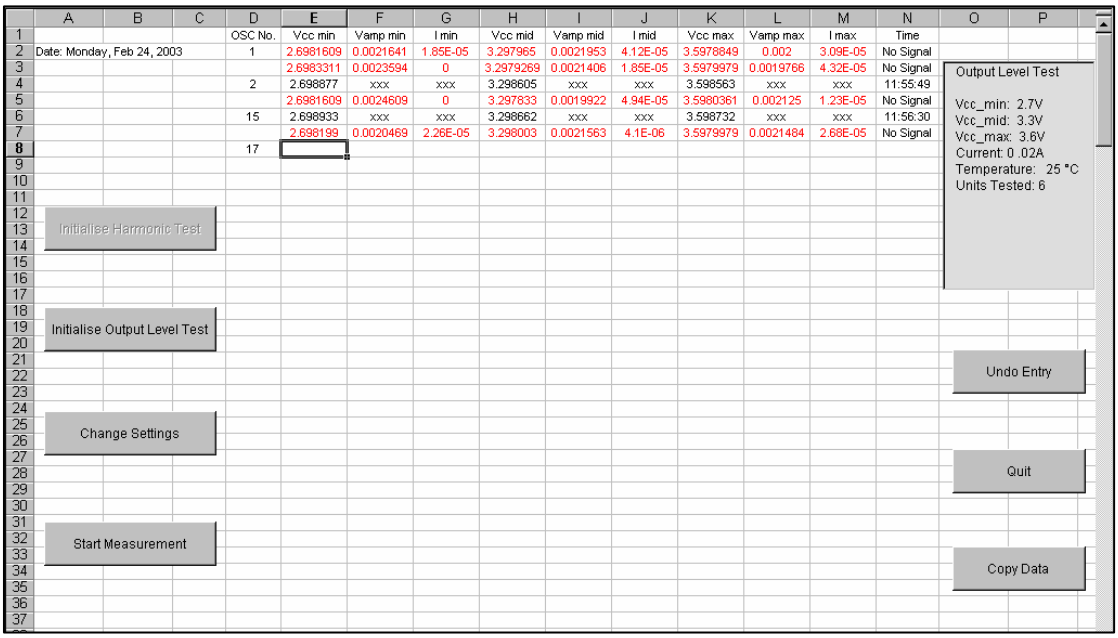


Figure 08: User interface of the new Output Level Test software

3.1.2 Features of the automated Output Level Test developed by me

- Error detection was used to detect if any of the instruments required were missing or not properly connected. The program wouldn't crash but give an appropriate error message instead, telling the user which instrument wasn't detected.
- The user is asked to enter the tested temperature when initialising the test and this temperature along with other test conditions are listed on the side of the Excel sheet using a VB ListBox. Therefore test conditions could easily be identified if the sheet is looked at, at a later date.
- If an instrument was unplugged or turned off after starting the test, an appropriate warning would be displayed. Once the instrument is connected it would resume measurements from where it last stopped.
- A separate button for changing values is provided allowing the user to change the minimum, mid and maximum voltages and the maximum allowable current. The ListBox is also updated immediately if the instruments were present and the values changed. A maximum of 10V and 1A were set as means of safety for oscillators in the event of a large number being accidentally entered. Also error detection was used so that any invalid numbers, such as text, negative voltages and currents and invalid temperatures cannot be entered.
- The ListBox also shows the number of units tested, making the unit count easier specially when testing a few hundred of units.
- A test can be continued even after closing the file, thus enabling to continue the test with the same file avoiding the hazard of copying data from separate files.
- An 'Undo' button is provided allowing the user to delete earlier values if necessary. The next measurement would continue from the next blank row.

- At times the tested oscillator could not have been properly turned on which results in very low values (mere noise due to the absence of a signal from the oscillator). This is detected by the program and such rows are highlighted with red showing the user that a proper signal was not present.
- Whenever the user clicked 'Initialise', all the previous values, except the oscillator numbers are deleted and the sheet is made ready for a new test.
- All the buttons were designed to move up or down depending on the row values are being entered to, so that the user doesn't have to scroll up or down to click on the buttons.

3.2 Harmonic Test

The second type of testing I was involved in was the Harmonic Test. Customers requiring oscillators for GPS need to know about the harmonics present in the oscillators since the presence of too many harmonics could result in poor GPS signal detection. Generally all the oscillators have five significant harmonics present although the range of frequency these harmonics are present could slightly differ on the type of oscillator.

The instruments used for this test were the DC power supply, spectrum analyser and the trunky. This test is not done on most of the units produced at Rakon unless specifically requested for by the customer. Hence there was no automated system for this test and all the units had to be tested and values entered into Excel manually. Five values were to be entered per each oscillator and harmonics tested at three temperatures (Figure 09) giving a total of 15 entries per oscillator. This meant 1500 manual entries even for 100 units which not surprisingly took at least a few days.

	A	B	C	D	E	F
1	Harmonic Content of Oscillator					
2						
3	tx2091 grp FD					
4						
5	Vcc	xxx Vdc			Calibrated	Yes
6	Output Load	xxx			300MHz Level	xxx
7	Temperature	25			300MHz Freq	xxx
8					VBW AVG	20
9	Date	01/17/03	Time	15:00		
10						
11	Osc S/N	Fundamental Level (dBm)	2nd Harmonic (dBm)	3rd Harmonic (dBm)	4th Harmonic (dBm)	5th Harmonic (dBm)
12	D1	xxx	xxx	xxx	xxx	xxx
13–22		Actual values cannot be given out due to confidential reasons				
23	F16	xxx	xxx	xxx	xxx	xxx
24						

Figure 09: Screen shot of results of harmonic test

I was given 5 palettes of oscillators in my first week and each palette contained 128 oscillators. I was to run the Output Level and Harmonic Tests on these units and give the results as soon as I could. Output Level Test was done in a reasonably short period of time despite the many drawbacks present in the system (as mentioned above). However the harmonics test took much longer since all the values had to be entered into Excel manually. The whole process was not only time consuming and tiring, but also the chances of incorrect values being accidentally entered were quite high.

I soon realised the necessity of automating the process and discussed the matter with my supervisor Andrew Daken after completing the task I was given. He agreed to what I said and encouraged me to find and develop a system that would automate the process. He also found the Hewlett Packard Spectrum Analyser Programmer's Guide for me so that I could read through it and automate the process.

After spending some time understanding how the spectrum analyser worked and how it could be programmed to get the harmonics with the press of a button, I was able to write a Microsoft Excel Macro to automate the process. Shown on the next page is the user interface of this system.

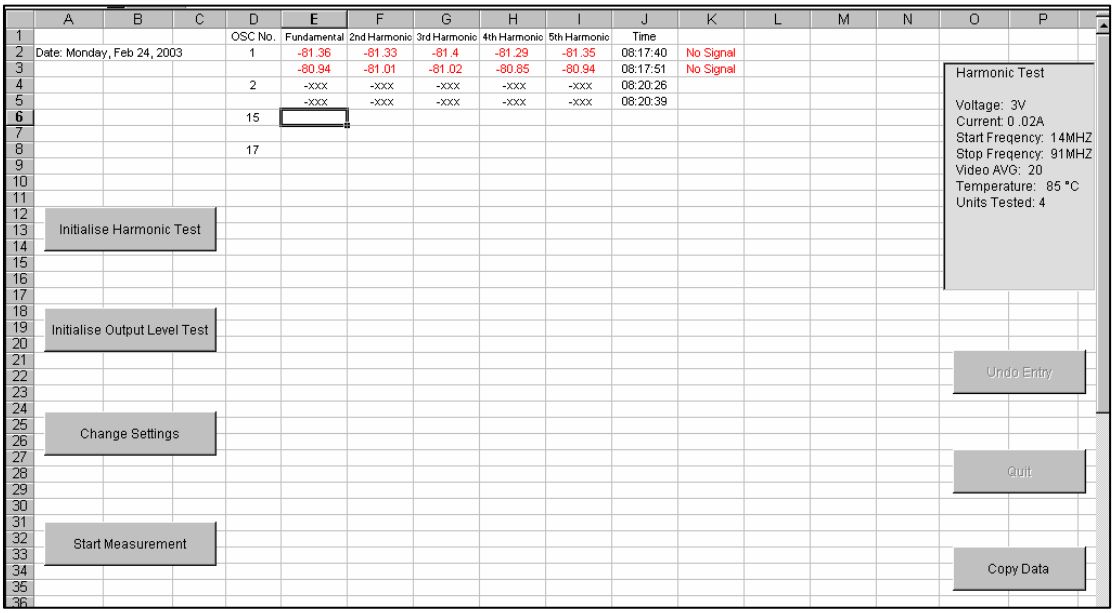


Figure 10: User interface of the Harmonic Test software

The features of the final design of this automated process are listed below.

3.2.1 Features of the automated Harmonic Test designed by me

- Error detection was used to detect if any of the instruments required were missing or not properly connected. The program wouldn't crash but give an appropriate error message instead, telling the user which instrument wasn't present.
- The user is asked to enter the tested temperature when initialising the test and this temperature along with other test conditions are listed on the side of the Excel sheet using a VB ListBox. Therefore test conditions could easily be identified if and when the sheet is looked at, at a later date.
- If an instrument was unplugged or turned off after starting the test, an appropriate warning would be displayed. Once the instrument was connected it would resume measurements from where it last stopped.
- A separate button for changing values was provided allowing the user to change the voltage, maximum allowable current, Start Frequency, Stop frequency and the Video Average to be used. The ListBox was also updated immediately if the instruments were present and the values changed. A maximum of 10V and 1A were set as means of safety for oscillators in the event of a large number being accidentally entered. Also error detection was used so that any invalid numbers, such as text, negative voltages, currents, video averages and/or frequencies and invalid temperatures cannot be entered.
- The ListBox also shows the number of units tested, making the unit count easier specially when testing a few hundred of units.
- A test could be repeated even after closing the file, thus enabling to continue the test with the same file avoiding the hazard of copying data from separate files.
- An 'Undo' button is provided allowing the user to delete earlier values if necessary. The next measurement would continue from the next blank row.
- At times the tested oscillator could not have been properly turned on which results in very low values. This is detected by the program and such rows are highlighted with red showing the user that a proper signal from the oscillator was not present.
- Whenever the user clicks 'Initialise', all the previous values, except the oscillator numbers are deleted and the sheet is made ready for a new test.

- All the buttons were designed to move up or down depending on the row values are being entered to, so that the user doesn't have to scroll up or down to click on the buttons.

Along with these features both the applications were designed to show the 'Tested Time' and also give a system beep after getting the required values. A 'Copy Data' button was also provided which copied all the test values onto a new Excel Sheet. The user could use this file to save the test values thus saving substantial amount of space. The file containing the macros was around 565Kb, whereas the new file containing only the values and headings would be around 40-50Kb depending on the number of units tested.

Although the Output Level and Harmonic Test applications were developed independent of each other, the final design consisted of a fully integrated system where the user could select between the two tests and initialise instruments, change settings and get values accordingly. The VB code of this final system can be found in Appendix 3.

3.3 Phase Noise Test

This test is done mainly on oscillators used for GPS. It measures noise present in the oscillators that could affect the GPS signals. Instruments used for this test are the DC power supply, frequency counter, dynamic signal analyser, thermal couple and the phase noise interference machine. My task was to test the units using the computerised system and hand in the results.

3.4 Micro-Jump Test

This test is also done mainly on oscillators used for GPS and Rakon is the only company in the world that offers this test. It detects if there are any spontaneous frequency jumps in the oscillators since these brief jumps are not detected by the Phase Noise Test. This test is done using an automated process designed using MATLAB and my task was to run the tests on the oscillators and give printouts of the results.

3.5 Other Involvements

Apart from being actively involved in the above-mentioned tests and software development, I was also involved with other general engineering tasks and writing Visual Basic based macros for making certain measurements. General engineering tasks include soldering components onto oscillators using SMT (Surface Mounted) components, solder paste and solder wick, making manual measurements using oscilloscope, multimeter, thermal couple, frequency counter and DC power supply and data manipulation using Microsoft Excel.

3.5.1 Macros written for testing

- Measuring the frequency at three different voltages was required and 80 measurements were to be taken. A Macro was written (Appendix 4) to program the Frequency Counter and the Power Supply. This was called the Vcc Stability Test.
- Measuring the frequency at various temperatures and plotting the graph, Frequency Vs. Temperature was required. A program to show temperature and frequencies was written for this purpose (Appendix 5)
- Measuring voltages and frequencies at a constant Vcc but varying Vco was required. Vco was to change from 0.1V to 4.5V with a 0.1V step size and frequencies read. This test was to be performed three times with appropriate time delays between each experiment. See Appendix 6 for the code written to achieve this task.

4.0 Conclusions

All in all I thoroughly enjoyed working at Rakon Limited. It provided me with an insight of how high-tech facilities operate while giving me the experience of working in the industry. I learnt about writing software for instruments using Visual Basic, which was totally a new experience for me. Learning how to operate electrical instruments such as spectrum analysers, frequency counters, signal analysers and phase noise interference machines and also dealing with day to day engineering issues were among the other important skills I've gained by working for Rakon Limited. I thank the company for giving me the opportunity to work there and wish them all the best in all its future endeavors.

References

Rakon. Retrieved February 14, 2003, from http://www.rakon.co.nz/

Appendix 1: Mission Statement

Our Vision

Rakon strives to be the technology-leading supplier of high performance and high quality quartz crystals, Temperature Compensated Crystal Oscillators (TCXOs) and frequency control products. We meet our customers' needs by:

- Producing products of exceptionally high quality
- Providing a high degree of added value
- Providing customized technological solutions
- Being flexible with product specification and delivery

Major applications currently targeted by Rakon include:

- Global Positioning System (GPS) applications
- Communications Systems – Cellular systems; satellite communications; data transmission
- Any other applications demanding precision, high-stability reference oscillators
- Any applications requiring customised TCXOs

Our Philosophy and Values

Rakon's philosophy and values to guide us in achieving our vision are:

The Environment
Rakon is committed to leaving a healthy, clean environment for future generations. Actively pursue conservation of energy and physical resources. Ensure all products and processes are developed and improved to be environmentally sustainable.

Staff
Recognise that our staff are vital to the success of the company. Employ and develop staff to have the skills needed to achieve our goals. Provide opportunities for them to achieve their full potential. Treat all staff fairly, with honesty and dignity. Maintain regular, open communication at all levels of the organisation.

Customers
Value our customers. Listen carefully and be responsive to their stated needs. Anticipate their future needs.

Quality
Instill a company culture and environment where all staff contribute to the continuous improvement of our products, processes and business practices.

Suppliers
Build long-term, trusting relationships with our suppliers and encourage them to adopt the same quality improvement philosophy that we have ourselves.

Planning
Make plans and decisions that are oriented towards the long-term success of the company, we aim to balance the needs of our staff, customers and suppliers with the need to generate profit.

Society
Rakon will be a responsible member of the community. We will be fair, ethical and professional in conducting all aspects of our business.

Appendix 2: Management structure

Appendix 3: Visual Basic Code written for Output Level and Harmonic Tests

```
Const start_row As Integer = 2
Const maxVolt As Integer = 10, maxCurr As Integer = 1, maxStop As Integer = 1000, maxAVG
As Integer = 100
Const defvolt As Single = 3#, defcurr As Single = 0.02, defVAVG As Integer = 20
Const defVcc_min As Single = 2.7, defVcc_mid As Single = 3.3, defVcc_max As Single = 3.6
Const defOutCurr As Single = 0.02

Const Vcc_min_col = 5, Vpp_min_col = 6, I_min_col = 7
Const Vcc_mid_col = 8, Vpp_mid_col = 9, I_mid_col = 10
Const Vcc_max_col = 11, Vpp_max_col = 12, I_max_col = 13

Dim curr_row As Integer
Dim counter As Integer, power As Integer, dvm As Integer, osc As Integer
Dim actual As Long
Dim VAVG As Integer, volt As Single, curr As Single
Dim Vcc_min As Single, Vcc_mid As Single, Vcc_max As Single, outCurr As Single
Dim startFrq As Integer, stopFrq As Integer
Dim callSpec As Integer, calliwriteS As Integer, calliwriteP As Integer, outOrHarm As
Integer
Dim isDvmSet As Integer, isOscSet As Integer, firstTime As Integer, firstTimeOut As
Integer

Private Sub cmdInit_Click()
  Call Initialise
End Sub

Private Sub cmdQUIT_Click()
 Dim returned As String

 If (outOrHarm = 1) Then
        If (calliwriteS = 0 Or calliwriteP = 0) Then
           returned = MsgBox("No instrument(s) found.", vbOKOnly + vbExclamation,
"Warning")
           returned = MsgBox("Please ensure all instruments are properly connected",
vbOKOnly + vbExclamation, "Warning")
        Else
           returned = MsgBox("Sure you want to quit the program?", vbYesNo +
vbExclamation, "Warning")
           If returned = 6 Then
              Call quitMsgs(1, power)
           End If
        End If
 Else
        If (calliwriteP = 0 Or isDvmSet = 0) Then
            returned = MsgBox("No instrument(s) found.", vbOKOnly + vbExclamation,
"Warning")
            returned = MsgBox("Please ensure all instruments are properly connected",
vbOKOnly + vbExclamation, "Warning")
        Else
           returned = MsgBox("Sure you want to quit the program?", vbYesNo +
vbExclamation, "Warning")
           If returned = 6 Then
              Call quitMsgs(2, power)
           End If
        End If
 End If
End Sub

Private Sub quitMsgs(outOrH As Integer, po As Integer)
    Dim co As Integer
    If (outOrH = 1) Then
        co = counter
        counter = 0
        calliwriteS = 0
    Else
        co = dvm
        dvm = 0
        isDvmSet = 0
    End If
    Call ilocal(co)
    Call iclose(co)
    Call iclose(po)
    Call siclcleanup
    cmdInit.Enabled = True
    output_level.Enabled = True
    cmdStart_meas.Enabled = False
    cmdQUIT.Enabled = False
    change.Enabled = False
    del.Enabled = False
    calliwriteP = 0
End Sub

Private Sub cmdStart_meas_Click()
   Dim count As Integer
   Dim count2 As Integer
   Dim curr_column As Integer
   Dim strres As String * 20

   outOrHarm = Cells(50, 33)
   If ((calliwriteS = 0 Or calliwriteP = 0) And (outOrHarm = 1)) Then
```

```
        firstTime = 1
    End If

    If (firstTime = 1 And outOrHarm = 1) Then
        counter = iopen("hpib7,18")
        Call itimeout(counter, 5000)
        power = iopen("hpib7,10")
        Call itimeout(power, 5000)
        calliwriteS = 1
        Call setSpec
        If (calliwriteS) Then
            calliwriteP = 1
            Call SetPower
        End If
        firstTime = 0

        If (calliwriteS And calliwriteP) Then
            For count = 1 To 20000
                For count2 = 1 To 1000
                Next count2
            Next count
            cmdQUIT.Enabled = True
            output_level.Enabled = False
            del.Enabled = True
            Call List1
        End If
    End If
    If (outOrHarm = 1 And calliwriteS = 1 And calliwriteP = 1) Then
        curr_column = 5
        count2 = iwrite(counter, "VAVG OFF;" + Chr$(10), 9, 1, 0&)
        If count2 = 0 Then
            Call iwrite(counter, "VAVG ON;" + Chr$(10), 8, 1, 0&)
            delay (VAVG / 5)
            Call iwrite(counter, "MKPK HI" + Chr$(10), 8, 1, 0&)
            For count1 = 1 To 5
                Call iwrite(counter, "MKA?" + Chr$(10), 5, 1, 0&)
                Call iread(counter, strres, 20, 0&, actual)
                Cells(curr_row, curr_column) = val(strres)
                'Cells(curr_row, curr_column + 6) = ""
                Call iwrite(counter, "MKPK NR" + Chr$(10), 8, 1, 0&)
                If (Cells(curr_row, curr_column) < -80) Then
                   Cells(curr_row, curr_column).Font.ColorIndex = 3
                Else
                   Cells(curr_row, curr_column).Font.ColorIndex = 0
                End If
                curr_column = curr_column + 1
            Next count1
            Cells(curr_row, curr_column).Font.ColorIndex = 0
            Cells(curr_row, curr_column) = Format$(Now, "hh:mm:ss")
            If (Cells(curr_row, curr_column - 2).Font.ColorIndex = 3) Then
               Cells(curr_row, curr_column + 1).Font.ColorIndex = 3
               Cells(curr_row, curr_column + 1) = "No Signal"
            End If
            curr_row = curr_row + 1
            Call iwrite(counter, "MKOFF;" + Chr$(10), 7, 1, 0&)
            count2 = iwrite(power, "SYST:BEEP" + Chr$(10), 10, 1, 0&)
            If count2 <> 0 Then
               calliwriteP = 0
            End If
            Cells(50, 31) = curr_row
            If (dis.ListCount = 9) Then
               dis.RemoveItem (8)
            End If
            dis.AddItem "Units Tested:" & Str$(curr_row - start_row)
            calliwriteS = 1
         Else
            count2 = MsgBox("Please ensure all instrument(s) are properly connected and
initialised", vbOKOnly + vbExclamation, "Warning")
            dis.Clear
         End If
    ElseIf (outOrHarm = 2) Then
        Call output_meas
    End If
    If (curr_row = 39) Or (curr_row = 77) Or (curr_row = 115) Or (curr_row = 153) Then
        cmdInit.Top = cmdInit.Top + 430
        cmdStart_meas.Top = cmdStart_meas.Top + 430
        cmdQUIT.Top = cmdQUIT.Top + 430
        del.Top = del.Top + 430
        change.Top = change.Top + 430
        output_level.Top = output_level.Top + 430
        copy_data.Top = copy_data.Top + 430
        ActiveWindow.LargeScroll Down:=1
    End If
End Sub

Private Sub Initialise()
    Dim temper As Integer
    Dim tempS As Integer
    Dim tempP As Integer
    Dim temp3 As String
```

```
   Dim temp4 As Integer

   temp3 = Cells(50, 32)
   temper = val(InputBox("Enter Temperature", "Value Entry Form", temp3, 4800, 3800))
   If (temper > -100 And temper < 400 And temper <> 0) Then
        tempP = 999
        counter = iopen("hpib7,18")
        Call itimeout(counter, 5000)
        tempS = iwrite(counter, "TIMEDSP ON;" + Chr$(10), 12, 1, 0&)
        If tempS = 0 Then
            Cells(50, 33) = 1
            outOrHarm = 1
            Cells(50, 32) = temper
            callSpec = 0
            VAVG = defVAVG
            Cells(51, 28) = VAVG
            startFrq = 14
            Cells(51, 29) = startFrq
            stopFrq = 91
            Cells(51, 30) = stopFrq
            calliwriteS = 1
            Call setSpec
         Else
            temper = MsgBox("Spectrum analyser not found", vbOKOnly + vbExclamation,
"Warning")
            output_level.Enabled = True
            dis.Clear
         End If
         If tempS = 0 Then
            power = iopen("hpib7,10")
            Call itimeout(power, 5000)
            tempP = iwrite(power, "*CLS" + Chr$(10), 5, 1, 0&)
         End If
         If tempP = 0 Then
            Cells(2, 1) = "Date: " & Format$(Now, "dddd, mmm d, yyyy")
            For temper = start_row To curr_row - 1
                For temp4 = 5 To 14
                    Cells(temper, temp4) = ""
                Next temp4
            Next temper
            firstTime = 0
            volt = defvolt
            Cells(50, 26) = volt
            curr = defcurr
            Cells(50, 27) = curr
            curr_row = start_row
            Call List1
            calliwriteP = 1
            Call SetPower
            output_level.Enabled = False
            cmdQUIT.Enabled = True
            cmdStart_meas.Enabled = True
            change.Enabled = True
            del.Enabled = True
            Cells(1, 4) = "OSC No."
            Cells(1, 5) = "Fundamental"
            Cells(1, 6) = "2nd Harmonic"
            Cells(1, 7) = "3rd Harmonic"
            Cells(1, 8) = "4th Harmonic"
            Cells(1, 9) = "5th Harmonic"
            Cells(1, 10) = "Time"
            Cells(1, 11) = ""
            Cells(1, 12) = ""
            Cells(1, 13) = ""
            Cells(1, 14) = ""
            Columns("J:J").Select
            Selection.NumberFormat = "h:mm:ss"
            Range("E1:I1").Select
            With Selection.Font
                .name = "Arial"
                .size = 7
            End With
            Cells(2, 5).Select
            ActiveWindow.ScrollRow = 1
         ElseIf tempP <> 999 Then
            tempP = MsgBox("DC power supply not found", vbOKOnly + vbExclamation,
"Warning")
         End If
         cmdInit.Top = 124.5
         cmdStart_meas.Top = 342.75
         cmdQUIT.Top = 292.5
         del.Top = 223.5
         change.Top = 266.25
         output_level.Top = 194.25
         copy_data.Top = 360
    Else
         temper = MsgBox("Initialise again and enter valid value for temperature to
start new test", vbOKOnly + vbExclamation, "Invalid Entry")
    End If
```

```
End Sub

Private Sub setSpec()

        Call getCounter("FA ", Str$(startFrq), "MHZ", 2)
        Call getCounter("FB ", Str$(stopFrq), "MHZ", 2)
        Call getCounter("VAVG ", Str$(VAVG), ";", 2)
        If (calliwriteS) Then
            Call iwrite(counter, "PKSORT 1;" + Chr$(10), 9, 1, 0&)
            Call iwrite(counter, "PKTBL 1;" + Chr$(10), 9, 1, 0&)
            Cells(50, 28) = Cells(51, 28)
            Cells(50, 29) = Cells(51, 29)
            Cells(50, 30) = Cells(51, 30)
        Else
            startFrq = Cells(50, 29)
            stopFrq = Cells(50, 30)
            VAVG = Cells(50, 28)
        End If
End Sub

Private Sub SetPower()
   Dim count2 As Integer

   count2 = iwrite(power, "*RST" + Chr$(10), 5, 1, 0&)
   If count2 <> 0 Then
        calliwriteP = 0
   End If
   If (calliwriteP) Then
        Call iwrite(power, "*CLS" + Chr$(10), 5, 1, 0&)
        Call iwrite(power, "OUTPUT ON" + Chr$(10), 10, 1, 0&)
        Call iwrite(power, "INST:NSEL 1" + Chr$(10), 12, 1, 0&)
        Call iwrite(power, "DISPLAY:MODE VI" + Chr$(10), 16, 1, 0&)
        Call getCounter("SOURCE:CURRENT:LEVEL ", Str$(curr), "", 1)
        Call getCounter("VOLT ", Str$(volt), "", 1)
    Else
        count2 = MsgBox("Error reading power supply", vbOKOnly + vbExclamation,
"Warning")
        calliwriteP = 0
    End If
End Sub

Private Sub getCounter(st1 As String, st2 As String, st3 As String, inst As Integer)
  Dim newsiclAddr As String
  Dim count2 As Integer
  If st3 <> "" Then
     newsiclAddr = st1 & st2 & st3
  Else
     newsiclAddr = st1 & st2
  End If
  If inst = 1 Then
     count2 = iwrite(power, newsiclAddr + Chr$(10), Len(newsiclAddr), 1, 0&)
     If (count2 <> 0) Then
        calliwriteP = 0
        count2 = MsgBox("Error reading power supply", vbOKOnly + vbExclamation,
"Warning")
     End If
  Else
     If (calliwriteS) Then
         count2 = iwrite(counter, newsiclAddr + Chr$(10), Len(newsiclAddr), 1, 0&)
         If count2 <> 0 Then
              calliwriteS = 0
         End If
     End If
  End If
End Sub

Private Sub change_Click()
    If (outOrHarm = 1) Then
        Call changeVal("Voltage", maxVolt, 1)
        Call changeVal("Current", maxCurr, 2)
        Call changeVal("Start Frequency", maxStop, 3)
        Call changeVal("Stop Frequency", maxStop, 4)
        Call changeVal("Video Average", maxAVG, 5)
    ElseIf (outOrHarm = 2) Then
        Call changeValOut("Vcc_Min", maxVolt, 1)
        If calliwriteP = 1 Then
            Call changeValOut("Vcc_Mid", maxVolt, 2)
            Call changeValOut("Vcc_Max", maxVolt, 3)
            Call changeValOut("Current", maxCurr, 4)
            volt = Vcc_min
            Call SetPower
        End If
    End If
End Sub

Private Sub List1()
    Dim temp As Integer
    temp = Cells(50, 33)

    If (temp = 1) Then
```

```
        If (dis.ListCount <> 0) Then
            Do While dis.ListCount <> 0
                dis.RemoveItem (dis.ListCount - 1)
            Loop
        End If
        dis.AddItem "Harmonic Test"
        dis.AddItem ""
        dis.AddItem "Voltage: " & Str$(volt) & "V" '1
        If curr < 1 Then
            dis.AddItem "Current: 0" & Str$(curr) & "A"    '2
        Else
            dis.AddItem "Current: " & Str$(curr) & "A"
        End If
        dis.AddItem "Start Freqency: " & Str$(startFrq) & "MHZ"    '3
        dis.AddItem "Stop Freqency: " & Str$(stopFrq) & "MHZ"  '4
        dis.AddItem "Video AVG: " & Str$(VAVG)   '5
        dis.AddItem "Temperature:  " & Str$(Cells(50, 32)) & " °C"
        If (dis.ListCount = 9) Then
            dis.RemoveItem (8)
        End If
        dis.AddItem "Units Tested:" & Str$(curr_row - start_row)
    Else
        Call listOut
    End If
End Sub

Private Sub changeVal(st1 As String, instMax As Integer, inst As Integer)
    Dim returned As String
    Dim newVolt As String

    returned = MsgBox("Enter new " & st1 & "?", vbYesNo + vbExclamation, "Change " &
st1)
    If returned = 6 Then
        newVolt = InputBox("Change " & st1, "Value Entry Form", "Enter New " & st1 & "
Here...", 500, 700)
        If limits(newVolt, instMax) Then
            If inst = 1 Then
                calliwriteP = 1
                volt = val(newVolt)
                Call SetPower
                If (calliwriteP) Then
                    Cells(50, 26) = volt
                    Call List1
                End If
            ElseIf inst = 2 Then
                calliwriteP = 1
                curr = val(newVolt)
                Call SetPower
                If (calliwriteP) Then
                    Cells(50, 27) = curr
                    Call List1
                End If
            ElseIf inst = 3 Then
                startFrq = val(newVolt)
                If (calliwriteS) Then
                    Cells(51, 29) = startFrq
                End If
                callSpec = 1
            ElseIf inst = 4 Then
                stopFrq = val(newVolt)
                If (calliwriteS) Then
                    Cells(51, 30) = stopFrq
                End If
                callSpec = 1
            ElseIf inst = 5 Then
                VAVG = val(newVolt)
                If (calliwriteS) Then
                   Cells(51, 28) = VAVG
                End If
                callSpec = 1
            End If
         Else
            returned = MsgBox(st1 & " Not Changed", vbOKOnly + vbExclamation, "Invalid
Entry or over maximum limit")
            Call List1
         End If
    End If
    If callSpec And inst = 5 Then
        callSpec = 0
        calliwriteS = 1
        Call setSpec
        Call List1
    End If
End Sub

Private Function limits(value As String, max As Integer) As Integer
    If val(value) > 0 And val(value) < max Then
        limits = 1
    Else
        limits = 0
```

```
    End If
End Function

Public Sub List2()
   outOrHarm = Cells(50, 33)
   cmdInit.Enabled = True
   cmdStart_meas.Enabled = True
   cmdQUIT.Enabled = False
   change.Enabled = True
   del.Enabled = False
   output_level.Enabled = True
   Columns("D:N").Select
   With Selection.Font
        .name = "Arial"
        .size = 8
   End With
   With Selection
        .HorizontalAlignment = xlCenter
   End With
   If (outOrHarm = 1) Then
        callSpec = 0
        volt = Cells(50, 26)
        curr = Cells(50, 27)
        VAVG = Cells(50, 28)
        startFrq = Cells(50, 29)
        stopFrq = Cells(50, 30)
        curr_row = Cells(50, 31)
        calliwriteS = 0
        calliwriteP = 0
        Call List1
        firstTime = 1
        Range("E1:I1").Select
        With Selection.Font
            .size = 7
        End With
    Else
        firstTimeOut = 0
        isDvmSet = 0
        calliwriteP = 0
        isOscSet = 0
        Call listOut
     End If
     Cells(Cells(50, 31), 5).Select
End Sub

Private Sub copy_data_Click()
    Call send_data(curr_row)
End Sub

Private Sub del_Click()
    Dim i As Integer

    If curr_row >= 3 Then
        For i = 5 To 14
          Cells(curr_row - 1, i) = ""
          Cells(curr_row - 1, i).Font.ColorIndex = 0
        Next i
        curr_row = curr_row - 1
        Cells(50, 31) = curr_row
        If (dis.ListCount = 8 And outOrHarm = 2) Then
            dis.RemoveItem (7)
        ElseIf (dis.ListCount = 9 And outOrHarm = 1) Then
            dis.RemoveItem (8)
        End If
        dis.AddItem "Units Tested:" & Str$(curr_row - start_row)
    ElseIf curr_row = 2 Then
        For i = 5 To 14
          Cells(curr_row, i) = ""
          Cells(curr_row, i).Font.ColorIndex = 0
        Next i
        Cells(50, 31) = curr_row
    End If
    If (curr_row = 38) Or (curr_row = 76) Or (curr_row = 114) Or (curr_row = 152) Then
         cmdInit.Top = cmdInit.Top - 430
         cmdStart_meas.Top = cmdStart_meas.Top - 430
         cmdQUIT.Top = cmdQUIT.Top - 430
         del.Top = del.Top - 430
         change.Top = change.Top - 430
         output_level.Top = output_level.Top - 430
         copy_data.Top = copy_data.Top - 430
         count_row = count_row - 37
         ActiveWindow.LargeScroll Down:=-1
    End If
End Sub

Private Sub output_level_Click()
    Dim temp3 As String

    temp3 = Cells(50, 32)
    temper = val(InputBox("Enter Temperature", "Value Entry Form", temp3, 4800, 3800))
```

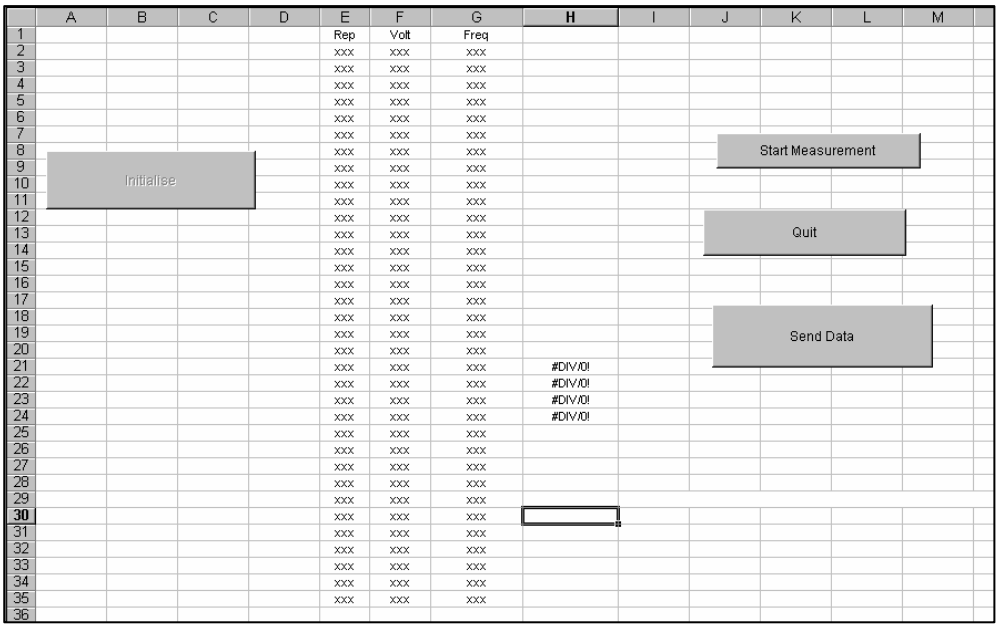
```
    If (temper > -100 And temper < 400 And temper <> 0) Then
        Cells(50, 33) = 2
        outOrHarm = 2
        Cells(50, 32) = temper
        dvm = iopen("hpib7,22")
        Call itimeout(dvm, 5000)
        isDvmSet = Setdvm(dvm)
        If (isDvmSet) Then
            power = iopen("hpib7,10")
            Call itimeout(power, 5000)
            Vcc_min = defVcc_min
            Cells(60, 28) = Vcc_min
            Vcc_mid = defVcc_mid
            Cells(60, 29) = Vcc_mid
            Vcc_max = defVcc_max
            Cells(60, 30) = Vcc_max
            curr = defOutCurr
            Cells(60, 31) = curr
            calliwriteP = 1
            volt = Vcc_min
            Call SetPower
            If (calliwriteP) Then
                curr_row = start_row
                Cells(50, 31) = curr_row
                osc = iopen("hpib7,6")
                Call itimeout(osc, 5000)
                isOscSet = Setosc(osc)
                If (isOscSet = 1) Then
                    cmdInit.Enabled = False
                    output_level.Enabled = True
                    cmdStart_meas.Enabled = True
                    cmdQUIT.Enabled = True
                    change.Enabled = True
                    del.Enabled = True
                    Cells(2, 1) = "Date: " & Format$(Now, "dddd, mmm d, yyyy")
                    Call listOut
                    cmdInit.Top = 124.5
                    cmdStart_meas.Top = 342.75
                    cmdQUIT.Top = 292.5
                    del.Top = 223.5
                    change.Top = 266.25
                    output_level.Top = 194.25
                    copy_data.Top = 360
                    Cells(1, 4) = "OSC No."
                    Cells(1, 5) = "Vcc min"
                    Cells(1, 6) = "Vamp min"
                    Cells(1, 7) = "I min"
                    Cells(1, 8) = "Vcc mid"
                    Cells(1, 9) = "Vamp mid"
                    Cells(1, 10) = "I mid"
                    Cells(1, 11) = "Vcc max"
                    Cells(1, 12) = "Vamp max"
                    Cells(1, 13) = "I max"
                    Cells(1, 14) = "Time"
                    Range("E2:N200").Select
                    Selection.Font.ColorIndex = 0
                    Selection.Cells = ""
                    Selection.NumberFormat = "General"
                    Columns("N:N").Select
                    Selection.NumberFormat = "h:mm:ss"
                    Range("E1:I1").Select
                    With Selection.Font
                        .name = "Arial"
                        .size = 8
                    End With
                    Cells(2, 5).Select
                    ActiveWindow.ScrollRow = 1
                End If
            Else
                dis.Clear
            End If
        Else
            calliwriteP = 0
            dis.Clear
        End If
    Else
        temper = MsgBox("Initialise again and enter valid value for temperature to start
new test", vbOKOnly + vbExclamation, "Invalid Entry")
    End If
End Sub

Private Sub listOut()
    Vcc_min = Cells(60, 28)
    Vcc_mid = Cells(60, 29)
    Vcc_max = Cells(60, 30)
    curr = Cells(60, 31)
    curr_row = Cells(50, 31)
    If (dis.ListCount <> 0) Then
            dis.Clear
    End If
```

```
    dis.AddItem "Output Level Test"
    dis.AddItem ""
    dis.AddItem "Vcc_min: " & Str$(Vcc_min) & "V"    '3
    dis.AddItem "Vcc_mid: " & Str$(Vcc_mid) & "V"  '4
    dis.AddItem "Vcc_max: " & Str$(Vcc_max) & "V"  '5
    If curr < 1 Then
        dis.AddItem "Current: 0" & Str$(curr) & "A"    '2
    Else
        dis.AddItem "Current: " & Str$(curr) & "A"
    End If
    dis.AddItem "Temperature:  " & Str$(Cells(50, 32)) & " °C"
    If (dis.ListCount = 8) Then
        dis.RemoveItem (7)
    End If
    dis.AddItem "Units Tested:" & Str$(curr_row - start_row)
End Sub

Private Sub output_meas()
    Dim count2 As Single
    Dim count3 As Single
    Dim count4 As Single
    Dim temp As Integer

    If (isDvmSet = 0 Or calliwriteP = 0 Or isOscSet = 0) Then
        firstTimeOut = 1
    End If
    If (firstTimeOut = 1) Then
        osc = iopen("hpib7,6")
        Call itimeout(osc, 5000)
        isOscSet = Setosc(osc)
        If (isOscSet <> 1) Then
            firstTimeOut = 0
        End If
    End If
    If (firstTimeOut = 1) Then
        dvm = iopen("hpib7,22")
        Call itimeout(dvm, 5000)
        isDvmSet = Setdvm(dvm)
        If (isDvmSet) Then
            power = iopen("hpib7,10")
            Call itimeout(power, 5000)
            temp = iwrite(power, "*CLS" + Chr$(10), 5, 1, 0&)
            If (temp = 0) Then
                calliwriteP = 1
                Vcc_min = Cells(60, 28)
                volt = Vcc_min
                curr = Cells(60, 31)
                Call SetPower
                cmdInit.Enabled = False
                cmdQUIT.Enabled = True
                del.Enabled = True
                delay (3)
            Else
                calliwriteP = 0
            End If
        End If
        firstTimeOut = 0
    End If
    If (calliwriteP And isDvmSet And isOscSet) Then
        count2 = 999
        count3 = 998
        count4 = 997
        If (calliwriteP <> 0) Then
            count2 = getPower("MEASure:VOLTage:DC?", power)
            count3 = getDVM("MEAS:CURR:DC?", dvm)
            count4 = getOSC("MEASU:MEAS1:VAL?", osc)
        End If
        If (count2 <> 999 And count3 <> 998 And count4 <> 997) Then
             'Font.Color.Index = 0
             Cells(curr_row, Vcc_min_col) = count2
             Cells(curr_row, I_min_col) = count3 * 1000
             Cells(curr_row, Vpp_min_col) = count4

             volt = Vcc_mid
             Call getCounter("VOLT ", Str$(volt), "", 1)
             delay (1)
             Cells(curr_row, Vcc_mid_col) = getPower("MEASure:VOLTage:DC?", power)
             Cells(curr_row, I_mid_col) = getDVM("MEAS:CURR:DC?", dvm) * 1000
             Cells(curr_row, Vpp_mid_col) = getOSC("MEASU:MEAS1:VAL?", osc)

             volt = Vcc_max
             Call getCounter("VOLT ", Str$(volt), "", 1)
             delay (1)
             Cells(curr_row, Vcc_max_col) = getPower("MEASure:VOLTage:DC?", power)
             Cells(curr_row, I_max_col) = getDVM("MEAS:CURR:DC?", dvm) * 1000
             Cells(curr_row, Vpp_max_col) = getOSC("MEASU:MEAS1:VAL?", osc)

             Cells(curr_row, 14) = Format$(Now, "hh:mm:ss")
             Call iwrite(power, "SYST:BEEP" + Chr$(10), 10, 1, 0&)
             volt = Vcc_min
```

```
            Call getCounter("VOLT ", Str$(volt), "", 1)

            curr_row = curr_row + 1
            If (dis.ListCount = 8) Then
               dis.RemoveItem (7)
            End If
            dis.AddItem "Units Tested:" & Str$(curr_row - start_row)
            Cells(50, 31) = curr_row
            If (Cells(curr_row - 1, I_max_col) < 0.0001) Then
               For temp = 5 To 13
                   Cells(curr_row - 1, temp).Font.ColorIndex = 3
               Next temp
               Cells(curr_row - 1, 14) = "No Signal"
            End If
        ElseIf count3 = 998 Then
             isDvmSet = 0
        End If
    Else
        If (isDvmSet = 0) Then
            count2 = MsgBox("Error reading multimeter", vbOKOnly + vbExclamation,
"Warning")
        ElseIf (isOscSet = 0) Then
            count2 = MsgBox("Error reading oscilloscope", vbOKOnly + vbExclamation,
"Warning")
        ElseIf (calliwriteP = 0) Then
            count2 = MsgBox("Error reading power supply", vbOKOnly + vbExclamation,
"Warning")
        End If
    End If
End Sub

Private Sub changeValOut(st1 As String, instMax As Integer, inst As Integer)
    Dim returned As String
    Dim newVolt As String

    returned = MsgBox("Enter new " & st1 & "?", vbYesNo + vbExclamation, "Change " &
st1)
    If returned = 6 Then
        newVolt = InputBox("Change " & st1, "Value Entry Form", "Enter New " & st1 & "
Here...", 500, 700)
        If limits(newVolt, instMax) Then
            If inst = 1 Then
                calliwriteP = 1
                volt = val(newVolt)
                Call SetPower
                If (calliwriteP) Then
                    Cells(60, 28) = volt
                    Call listOut
                End If
            ElseIf inst = 2 Then
                calliwriteP = 1
                volt = val(newVolt)
                Call SetPower
                If (calliwriteP) Then
                    Cells(60, 29) = volt
                    Call listOut
                End If
            ElseIf inst = 3 Then
                calliwriteP = 1
                volt = val(newVolt)
                Call SetPower
                If (calliwriteP) Then
                    Cells(60, 30) = volt
                    Call listOut
                End If
            ElseIf inst = 4 Then
                calliwriteP = 1
                curr = val(newVolt)
                Call SetPower
                If (calliwriteP) Then
                    Cells(60, 31) = curr
                    Call listOut
                End If
            End If
        Else
            returned = MsgBox(st1 & " Not Changed", vbOKOnly + vbExclamation, "Invalid
Entry or over maximum limit")
        End If
    End If
End Sub
```

Appendix 4: User interface and code written for Vcc Stability Test

User Interface



Code

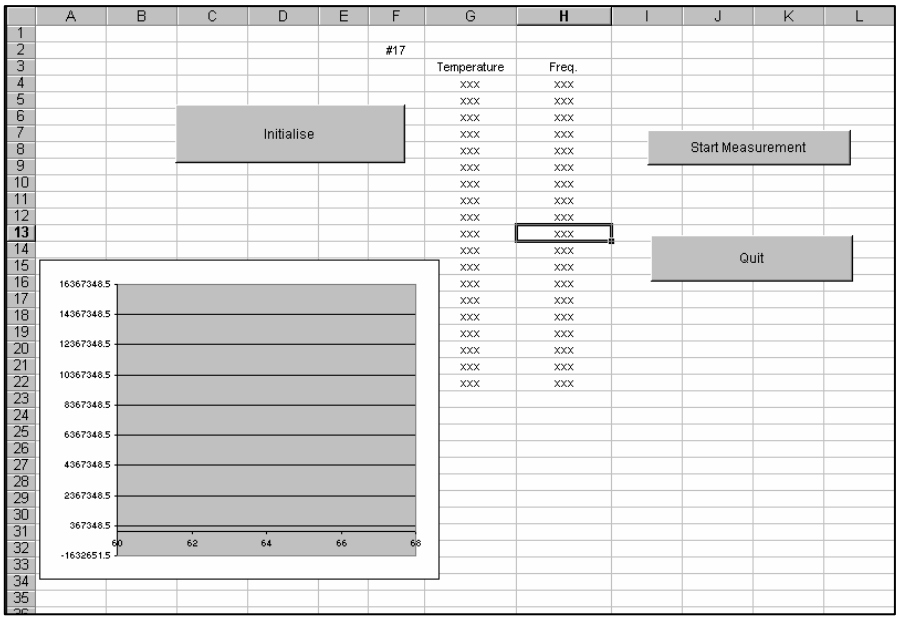
```
Const start_row As Integer = 2
Dim curr As Integer
Dim counter As Integer
Dim dvm As Integer
Dim power As Integer
Dim supply As Integer
Dim scope As Integer
Dim strres As String * 20
Dim actual As Long
Dim col As Integer

Private Sub cmdInit_Click()
  Call Initialise
  cmdInit.Enabled = False
  cmdStart_meas.Enabled = True
  cmdQUIT.Enabled = True
End Sub

Private Sub cmdQUIT_Click()
' Call iclose(counter)
 Call siclcleanup
 cmdInit.Enabled = True
 cmdStart_meas.Enabled = False
  cmdQUIT.Enabled = False
End Sub

Private Sub cmdStart_meas_Click()
dvm = 1
Worksheets("Sheet1").Range("E2:g100").value = ""
'1st
  For scope = 1 To 20
    Call iwrite(counter, "READ:FREQ?" + Chr$(10), 11, 1, 0&)
    Call iread(counter, strres, 20, 0&, actual)
    Cells(dvm, 7) = Val(strres)
    Cells(dvm, 5) = dvm
    Call iwrite(supply, "INST:NSEL 1" + Chr$(10), 12, 1, 0&)
    Call iwrite(supply, "MEASURE:VOLTAGE?" + Chr$(10), 17, 1, 0&)
    Call iread(supply, strres, 20, 0&, actual)
    Cells(dvm, 6) = Val(strres)
    dvm = dvm + 1
  Next scope
Call iwrite(supply, "INST:NSEL 1" + Chr$(10), 12, 1, 0&)
Call iwrite(supply, "VOLT 5" + Chr$(10), 7, 1, 0&)

For curr = 1 To 1000
 Worksheets("Sheet1").Range("H10").value = curr
```

```
Next curr
Worksheets("Sheet1").Range("H10").value = ""

'2nd
  For scope = 1 To 20
    Call iwrite(counter, "READ:FREQ?" + Chr$(10), 11, 1, 0&)
    Call iread(counter, strres, 20, 0&, actual)

    Cells(dvm, 7) = Val(strres)
    Cells(dvm, 5) = dvm

    Call iwrite(supply, "INST:NSEL 1" + Chr$(10), 12, 1, 0&)
    Call iwrite(supply, "MEASURE:VOLTAGE?" + Chr$(10), 17, 1, 0&)
    Call iread(supply, strres, 20, 0&, actual)

    Cells(dvm, 6) = Val(strres)

    dvm = dvm + 1
  Next scope

Call iwrite(supply, "INST:NSEL 1" + Chr$(10), 12, 1, 0&)
Call iwrite(supply, "VOLT 5.25" + Chr$(10), 10, 1, 0&)

For curr = 1 To 1000
 Worksheets("Sheet1").Range("H10").value = curr
Next curr
Worksheets("Sheet1").Range("H10").value = ""

'3rd

For scope = 1 To 20
    Call iwrite(counter, "READ:FREQ?" + Chr$(10), 11, 1, 0&)
    Call iread(counter, strres, 20, 0&, actual)

    Cells(dvm, 7) = Val(strres)
    Cells(dvm, 5) = dvm

    Call iwrite(supply, "INST:NSEL 1" + Chr$(10), 12, 1, 0&)
    Call iwrite(supply, "MEASURE:VOLTAGE?" + Chr$(10), 17, 1, 0&)
    Call iread(supply, strres, 20, 0&, actual)

    Cells(dvm, 6) = Val(strres)
    dvm = dvm + 1
  Next scope

Call iwrite(supply, "INST:NSEL 1" + Chr$(10), 12, 1, 0&)
Call iwrite(supply, "VOLT 4.75" + Chr$(10), 10, 1, 0&)

For curr = 1 To 1000
 Worksheets("Sheet1").Range("H10").value = curr
Next curr
Worksheets("Sheet1").Range("H10").value = ""

For scope = 1 To 20
    Call iwrite(counter, "READ:FREQ?" + Chr$(10), 11, 1, 0&)
    Call iread(counter, strres, 20, 0&, actual)

    Cells(dvm, 7) = Val(strres)
    Cells(dvm, 5) = dvm

    Call iwrite(supply, "INST:NSEL 1" + Chr$(10), 12, 1, 0&)
    Call iwrite(supply, "MEASURE:VOLTAGE?" + Chr$(10), 17, 1, 0&)
    Call iread(supply, strres, 20, 0&, actual)

    Cells(dvm, 6) = Val(strres)

    dvm = dvm + 1
  Next scope
End Sub

Private Sub Initialise()
   col = 12
   curr_row = start_row
   supply = iopen("hpib7,10")
   Call itimeout(supply, 5000)
   Call iwrite(supply, "*RST" + Chr$(10), 5, 1, 0&)

   Call iwrite(supply, "OUTPUT ON" + Chr$(10), 10, 1, 0&)
   Call iwrite(supply, "INST:NSEL 1" + Chr$(10), 12, 1, 0&)
   Call iwrite(supply, "DISPLAY:MODE VI" + Chr$(10), 15, 1, 0&)
```

```
   Call iwrite(supply, "SOURCE:CURRENT:LEVEL 0.100" + Chr$(10), 26, 1, 0&)
   Call iwrite(supply, "VOLT 4.75" + Chr$(10), 10, 1, 0&)

   Call iwrite(supply, "INST:NSEL 2" + Chr$(10), 12, 1, 0&)
   Call iwrite(supply, "DISPLAY:MODE VI" + Chr$(10), 15, 1, 0&)
   Call iwrite(supply, "SOURCE:CURRENT:LEVEL 0.100" + Chr$(10), 26, 1, 0&)
   Call iwrite(supply, "VOLT 5" + Chr$(10), 7, 1, 0&)

   counter = iopen("hpib7,3")
   Call itimeout(counter, 5000)
   Call iwrite(counter, "*RST" + Chr$(10), 5, 1, 0&)
   Call iwrite(counter, "INP:IMP 50" + Chr$(10), 11, 1, 0&)
End Sub

Private Sub CommandButton1_Click()
    Range("G1:G80").Select
    Selection.Copy
    Workbooks("Book1.xls").Activate
   ' Worksheets("Sheet1").Range("D4").Select
    Worksheets("Sheet1").Cells(4, col).Select
    Selection.PasteSpecial Paste:=xlValues, Operation:=xlNone, SkipBlanks:= _
        False, Transpose:=False
    col = col + 1
End Sub
```

Appendix 5: User interface and code written to get frequencies at different temperatures

User Interface

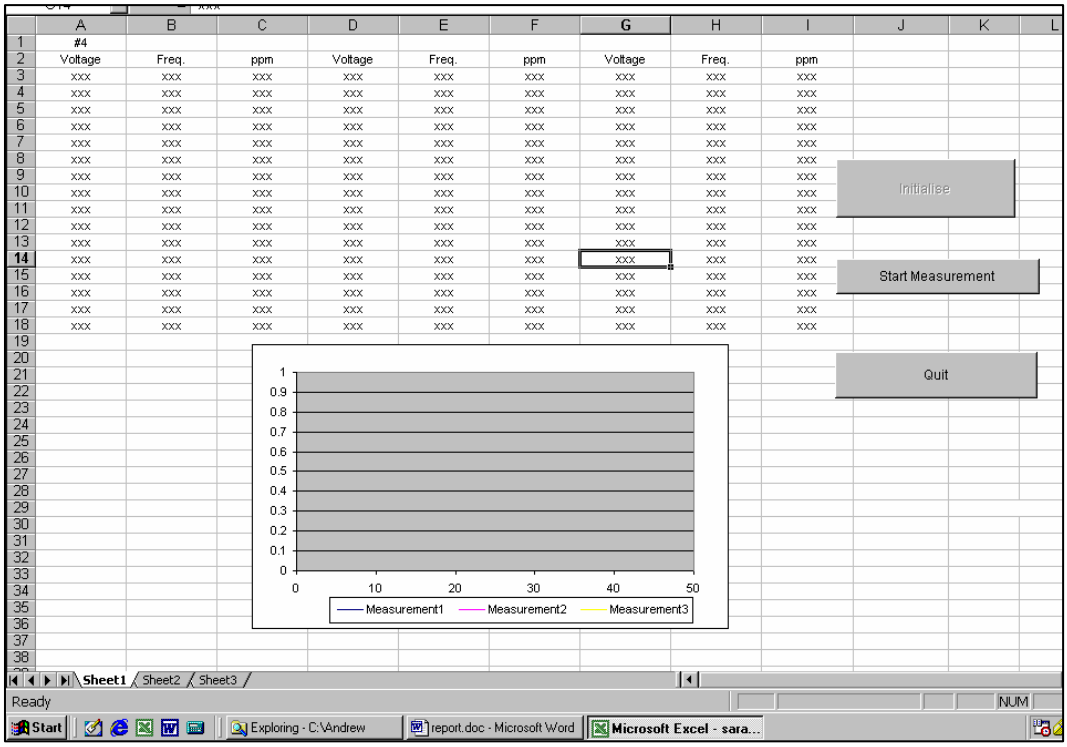


Code

```
Const start_row As Integer = 2
Dim curr As Integer
Dim counter As Integer
Dim dvm As Integer
Dim power As Integer
Dim supply As Integer
Dim scope As Integer
Dim strres As String * 20
Dim actual As Long
Dim col As Integer
Dim crow As Integer


Private Sub cmdInit_Click()
  Call Initialise
  cmdInit.Enabled = True
  cmdStart_meas.Enabled = True
  cmdQUIT.Enabled = True
End Sub

Private Sub cmdQUIT_Click()
 Call iclose(counter)
 Call siclcleanup
 cmdInit.Enabled = True
 cmdStart_meas.Enabled = False
  cmdQUIT.Enabled = False
End Sub

Private Sub cmdStart_meas_Click()
    Call iwrite(counter, "READ:FREQ?" + Chr$(10), 11, 1, 0&)
    Call iread(counter, strres, 20, 0&, actual)
    Cells(crow, 8) = Val(strres)
    crow = crow + 1
    Call ilocal(counter)
End Sub

Private Sub Initialise()
   col = 3
   curr_row = start_row
   supply = iopen("hpib7,10")
   Call itimeout(supply, 5000)
   Call iwrite(supply, "*RST" + Chr$(10), 5, 1, 0&)
   Call iwrite(supply, "OUTPUT ON" + Chr$(10), 10, 1, 0&)
   Call iwrite(supply, "INST:NSEL 1" + Chr$(10), 12, 1, 0&)
   Call iwrite(supply, "DISPLAY:MODE VI" + Chr$(10), 15, 1, 0&)
   Call iwrite(supply, "SOURCE:CURRENT:LEVEL 0.200" + Chr$(10), 26, 1, 0&)
   Call iwrite(supply, "VOLT 5" + Chr$(10), 7, 1, 0&)
   crow = 8
   counter = iopen("hpib7,3")
   Call itimeout(counter, 5000)
   Call iwrite(counter, "*RST" + Chr$(10), 5, 1, 0&)
End Sub
```

Appendix 6: User interface and code written to measure voltages and frequencies at a constant Vcc but varying VCO

User Interface



Code

```
Const start_row As Integer = 2
Dim curr As Integer
Dim counter As Integer
Dim dvm As Integer
Dim power As Integer
Dim supply As Integer
Dim scope As Integer
Dim strres As String * 20
Dim actual As Long
Dim col As Integer
Dim row As Integer

Private Sub cmdInit_Click()
  Call Initialise
  cmdInit.Enabled = False
  cmdStart_meas.Enabled = True
  cmdQUIT.Enabled = True
End Sub

Private Sub cmdQUIT_Click()
 Call iclose(counter)
 Call siclcleanup
 cmdInit.Enabled = True
 cmdStart_meas.Enabled = False
  cmdQUIT.Enabled = False
End Sub

Private Sub cmdStart_meas_Click()
    Dim i As Single
    Dim j As Integer
    Dim st As String
    Dim strr As String * 20

    For j = 1 To 5 Step 2
        row = 3
        Call iwrite(supply, "VOLT .1" + Chr$(10), 8, 1, 0&)
        For i = 0.2 To 4.5 Step 0.1
            delay (10)
            strres = ""
            Call iwrite(counter, "READ:FREQ?" + Chr$(10), 11, 1, 0&)
            Call iread(counter, strres, 20, 0&, actual)
            Cells(row, j + 1) = Val(strres)
```

```
            strr = ""
            Call iwrite(supply, "INST:NSEL 2" + Chr$(10), 12, 1, 0&)
            Call iwrite(supply, "MEASure:VOLTage:DC?" + Chr$(10), 20, 1, 0&)
            Call iread(supply, strr, 20, 0&, actual)
            Cells(row, j) = Val(strr)

            row = row + 1
            st = "VOLT" + Str$(i)
            'Cells(5, 5) = st
            Call iwrite(supply, st + Chr$(10), Len(st), 1, 0&)
        Next i
        delay (10)
        strres = ""
        Call iwrite(counter, "READ:FREQ?" + Chr$(10), 11, 1, 0&)
        Call iread(counter, strres, 20, 0&, actual)
        Cells(row, j + 1) = Val(strres)
        strr = ""
        Call iwrite(supply, "INST:NSEL 2" + Chr$(10), 12, 1, 0&)
        Call iwrite(supply, "MEASURE:VOLTAGE?" + Chr$(10), 17, 1, 0&)
        Call iread(supply, strr, 20, 0&, actual)
        Cells(row, j) = Val(strr)
    Next j
End Sub

Private Sub Initialise()
   row = 3
   supply = iopen("hpib7,10")
   Call itimeout(supply, 5000)
   Call iwrite(supply, "*RST" + Chr$(10), 5, 1, 0&)

   Call iwrite(supply, "OUTPUT ON" + Chr$(10), 10, 1, 0&)
   Call iwrite(supply, "INST:NSEL 1" + Chr$(10), 12, 1, 0&)
   Call iwrite(supply, "DISPLAY:MODE VI" + Chr$(10), 15, 1, 0&)
   Call iwrite(supply, "SOURCE:CURRENT:LEVEL 0.100" + Chr$(10), 26, 1, 0&)
   Call iwrite(supply, "VOLT 5" + Chr$(10), 7, 1, 0&)

   Call iwrite(supply, "INST:NSEL 2" + Chr$(10), 12, 1, 0&)
   Call iwrite(supply, "DISPLAY:MODE VI" + Chr$(10), 15, 1, 0&)
   Call iwrite(supply, "SOURCE:CURRENT:LEVEL 0.100" + Chr$(10), 26, 1, 0&)
   Call iwrite(supply, "VOLT .1" + Chr$(10), 8, 1, 0&)

   counter = iopen("hpib7,3")
   Call itimeout(counter, 5000)
   Call iwrite(counter, "*RST" + Chr$(10), 5, 1, 0&)
   Call iwrite(counter, ":FREQ:ARM:STAR:SOUR IMM" + Chr$(10), 24, 1, 0&)
   Call iwrite(counter, ":FREQ:ARM:STOP:SOUR TIM" + Chr$(10), 24, 1, 0&)
   Call iwrite(counter, ":FREQ:ARM:STOP:TIM 1" + Chr$(10), 21, 1, 0&)
   Call iwrite(counter, "INP:IMP 50" + Chr$(10), 11, 1, 0&)
End Sub

 Sub delay(delay_time As Single)
  Dim Finish As Single

  Finish = Timer + delay_time
  Do
  Loop Until Finish <= Timer
End Sub
```