ES4B8 – Individual Project (Part B) Report

Development of LabView® Virtual Instrument Graphical User Interface for Hewlett Packard HP4145A Semiconductor Parameter Analyser

Project Supervisor – Professor PA Mawby
ES4B8 – Individual Project Part B: Final Report
University of Warwick – School of Engineering
Student ID: 0320078 – Adam Foster

0 – Author's Assessment of the Project

Overall the project has been a success from both an engineering and personal point of view. From an engineering perspective, the Virtual Instrument (VI) produced should allow more efficient use of the HP4145A compared to the older VI already available from National Instruments (http://sine.ni.com/apps/utf8/niid_web_display.download_page?p_id_guid=E3B19B3E9376659CE034080020E74861). Although newer analysis machines have surpassed the HP4145A, they are still in use in some places (see section 6.1 – this VI has been downloaded 100 times in 2 months (as of 5th November 2007)) and so it is useful for these engineers and institutions.

Other engineers can still contribute to this project as the Virtual Instrument can be edited and extended very easily. Unlike other programming languages, LabView allows easy access to the source code itself (no decompilers are required). As identified in section 6, there are still some improvements that can be made in terms of performance and reliability (for certain measurement situations). Some more functionality can also be added, such as allowing the user to setup profiles that can be loaded into the VI to make measurement setup quicker and easier.

This project should be considered an achievement as it has yielded a working Virtual Instrument for use with the HP4145A. The VI has made significant improvements over the existing solution from National Instruments allowing complete control of the HP4145A from one interface instead of many smaller applications each controlling one specific function. The Virtual Instrument is now available to other engineers to use for free (as long as they have a LabView licence) and to develop further if they wish, prolonging the use of the HP4145A at these

institutions. From a personal standpoint, my knowledge of LabView has increased significantly, together with general software development skills

The main weakness of this project is the lack of testing outside the lab and away from the development computer. The VI has been made available on the National Instruments website (see section 6), and this was done to try and get feedback from 'real world' testing of the software, but so far no comments or suggestions have been contributed. Weaknesses in the software itself include no control for repeated measurements (i.e. how many measurements to user wants the device to make) and overall speed of the program when it is being executed (although this may be to do with the development computer – see section 6).

1.0 – Summary

The aim of this project was to produce a computer based Virtual Instrument (VI) allowing full control of the HP4145A semiconductor parameter analyser from a Windows® based computer, using LabView®. The VI created allows full control of the measurement process and for data to be obtained from the HP4145A for analysis. There was already a LabView program that did this, but it did not work correctly (data was not returned to the computer from the HP4145A) and did not offer the best user interface (see section 2.3) as it was designed for the later HP4145B model. This new VI allows control of the HP4145A's functions from one interface without the need to open lots of programs at once to control each individual step in the measurement process. The VI created for this project uses some of the basic functions used in this older version (such as sending commands along to the device and receiving messages back – see section 5), but the main program has been completed from scratch. Testing of the VI showed that it did function correctly and met the specification written at the start of the project. Thus far, only the development computer has been used to test the VI, as it was hard to find another computer with a GPIB card and the associated drivers required. For this reason the VI was published on the National Instruments website for free download, to allow feedback from 'real world' testing. Thus far, there have been no comments or suggestions fed back from users, even though there have been 100 downloads). Further work on this VI can still be carried out to improve overall speed of execution (the program is slow with large amounts of data) and to add some more functionality such as repeat measurement control (i.e. how many repeat measurements the user wants) and the ability to load profiles from the HP4145A's floppy drive (see section 6 for more detail).

Table of Contents

List of Figures and Diagrams

2.0 – Introduction

This report details what has been achieved in developing a LabView® based solution to control the Hewlett Packard HP4145A Semiconductor Parameter Analyser using a Microsoft™ Windows® based PC and National Instruments™ LabView® software. The control software is known as a virtual instrument (or VI) as it should be a virtual representation of the HP4145A. A virtual instrument did already exist from National Instruments™ (the creators of LabView®) but did not offer the best solution to controlling the device (see section 2.2) as it was mainly designed for the HP4145B model from Hewlett Packard. The VI created for this solution has made use of aspects of this available VI (e.g. some functions have been copied) as this allowed quicker development, but these needed to be integrated into an overall control solution (see section 4) allowing a user to operate one the device using one seamless Virtual Instrument. The HP4145A itself is used to analyse semiconductors by sending test signals to the different pins of the device. The output from the device is then fed back to the HP4145A and this data can be manipulated and displayed on the devices CRT (Cathode Ray Tube) screen (for more on the device itself see section 2.1 below).

Semiconductor analysis is important in the field of electronic engineering and particularly as a teaching aid for academic institutions. For example, when a transistor of any type is placed into a circuit, the outside network of resistors, capacitors and any other attached devices sets the levels of current and voltage into the transistor. These currents and voltages are important as these define the operating point where amplification of an applied signal will take place. The points are referred to as quiescent points of the circuit, and are important for circuits as output AC signals may be clipped if the quiescent voltages and currents are not selected wisely enough[5].

A line drawn onto the characteristic graph can represent all the possible quiescent points for a particular circuit configuration. These show the current and voltage levels for the output of the transistor, and allow the use of load-lines in order to asses the operating points of the devices. An output characteristic graph plots the current level at the collector (for a bipolar junction transistor – BJT) of the transistor against the applied voltage between the collector and emitter. The different collector current levels at specific base currents are then drawn onto the graph, to represent all the possible input currents that may be present at the base of the transistor. The q-point (the point defined by the quiescent voltage and current) can then reside along any point of this line, depending on the values of important components within the circuit, such as resistors or the level of the supply voltage. Load lines offer the ability to see what occurs when an AC signal is applied to the device, as this will cause the q-point to move along the load line in time with the applied signal. If q-point is too close to either the x or y-axis, then this can clip the output signal, as the current or voltage cannot exceed past the axis to draw the graph. This type of analysis is used in determining the effectiveness of a network when it is being used to amplify AC signals applied to the device.

Although the HP4145A isn't usually used to analyse an entire circuit, it is very important to test that the transistor's input and output characteristics are those expected. The HP4145A is used to simulate the voltages and currents that the circuit components would apply to each pin if the semiconductor were used in an actual circuit. The HP4145A can then produce the characteristic graph (an output characteristic graph for a BJT is shown in figure 2.0).

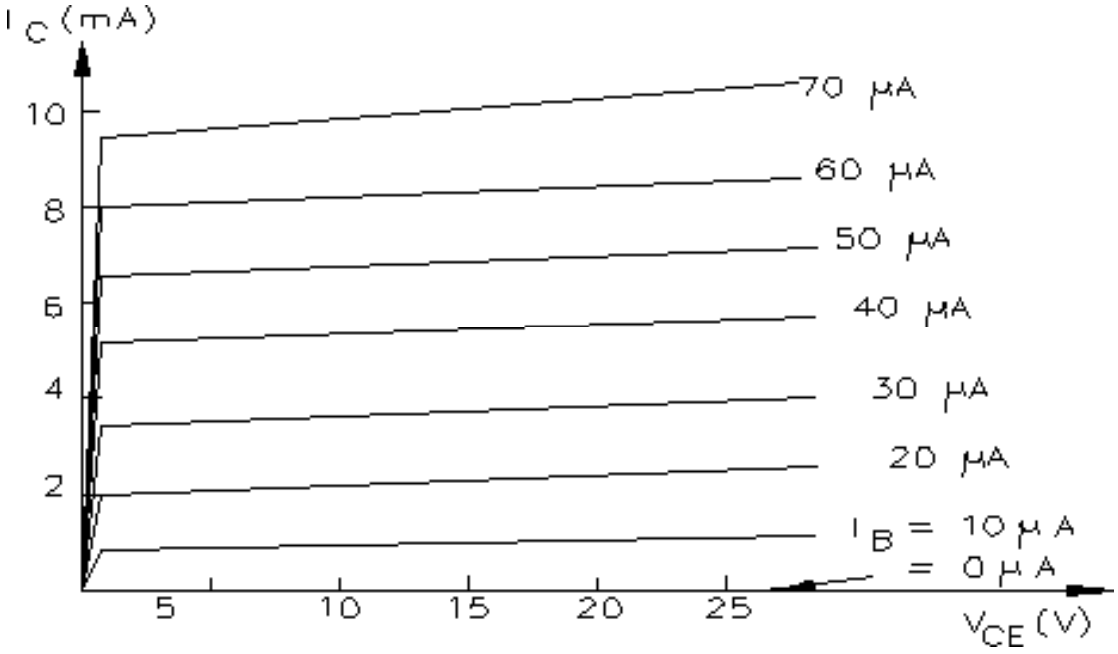

Figure 2.0 – *A typical BJT output characteristic graph showing the voltage between the collector and emitter (V_{CE}) plotted against the collector current I_C for different base currents (I_b)*[6]

Another important consideration is the frequency response of the device, as this will let the designer know what frequencies the transistor can operate within while maintaining a constant gain. The HP4145A allows a user to test all these aspects of a semiconductor device, and so the Virtual Instrument should also do this (see specification).

2.1 – Introduction to the HP4145A

The semiconductor analyser itself consists of a measurement box (which houses the semiconductors for testing) that is connected to the actual measurement device (HP4145A) using 3 cables (but can have up to 8 channels). These three cables carry signals to measure the device connected to the measurement box and can be configured using the HP4145A to transmit a voltage or current or to conduct a sweep. The 3 cables are connected to the different pins of the semiconductor (e.g. for a BJT (Bipolar Junction Transistor) one may be connected to the base terminal, one to the

collector and one to the emitter). The HP4145A itself consists of a small Cathode Ray Tube (CRT) screen, a Floppy Disk (B) drive along with text and number entry buttons as well as measurement control buttons and switches. Figure 2.1 shows a picture of the HP4145A used for this project.

To conduct a measurement the user primarily uses the 'NEXT' and 'PREVIOUS' page control keys to cycle through the different 'pages' that appear on the screen and using the soft-keys around the edge of the display to select the options that appear. In conducting a measurement the user will go through the pages in the sequence shown in figure 2.2 cycling along using the 'NEXT' and 'PREVIOUS' keys.

Figure 2.1 – *HP4145A Semiconductor Parameter Analyser*

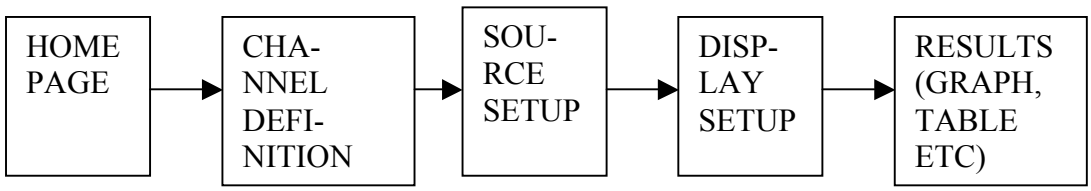

Figure 2.2: *Measurement setup process* [adapted from source 3]

The Home page allows the user to skip to any of the other pages instantly using the soft-keys on the side of the screen to select different options. The 'Channel Definition' page allows the user to control what signals are sent along the three cables connected to the measurement box. These can be set up in any way the user wants but there are also pre-defined setups for different devices including a diode, Bipolar Junction Transistors (BJT) and Field Effect Transistors (FET) that can be loaded from the floppy disk. The Source setup page allows the user to define which of the signals they want to be constant and which ones are to be the variable parameters. This page also allows the user to set the ranges and accuracy of sweeps conducted as well as set the compliance of each variable and how many measurement steps will be used.

Display setup allows the user to select which of the variables are shown in the results as well as how the results are displayed and the choices include graph, list, matrix or schmoo. The final page triggers the measurement process and then displays the results to the user.

The HP4145A also allows the user to control it remotely using code commands sent along a GPIB (General Purpose Interface Bus – may also be known as HP-IB (Hewlett-Packard Interface Bus[1])) from a computer. This interface also allows for the connection of printers and plotters to the HP4145A. The commands that can be used are detailed in the manual for the machine, and allow all aspects of the measurement process described above to be controlled remotely using ASCII commands.

2.2 – Introduction to LabView©

To allow control of the HP4145A device, LabView was chosen as an appropriate technology to use. This was because it specialises in allowing easy control of measurement devices within a laboratory environment where users may not have

adequate knowledge of more complicated high level programming languages such as C/C++ or Java. LabView allows a user to control a device using graphical means such as buttons, sliders and dials represented on a front panel. This can be thought of as analogous to a GUI (Graphical User Interface) of the device being controlled. From a development perspective, LabView has a different approach to code creation than other programming languages using the unique V language (visual language). This operates like a flow diagram requiring the developer to link boxes up with wires. The boxes may contain operators such as a simple multiplication or may link to a sub-flow diagram with other functions. Some boxes connect to the front panel, where the user can manipulate certain aspects (i.e. provide an input) to the program. These programs are usually referred to as Virtual Instruments or VIs, and each VI can have multiple sub-VIs within it (this can be thought of as like a subroutine in traditional programming).

The wires linking the boxes carry information around the program and only certain coloured wires can carry certain types of data (such as Boolean, double precision, integer, floating point or text) much like variables created in Java or C/C++ can only contain certain types of data. Unlike other programming languages LabView allows one to 'cluster' different types of data together to allow for more organised code creation. This can be thought of as like bunching a group of different wires and tying them together to keep them organised. This is done using a bundle/un-bundle operator and the new wire created is pink and slightly thicker in colour (see figure 2.3).

More advanced operations can be carried out using for or while loops. These are represented using large boxes that only operate certain amounts of code depending

on the state of their operand. This way, all aspects of Java and C/C++ can be recreated in a graphical form within LabView.

Figure 2.3 shows both these types of boxes in more detail, showing the larger if/else operator (known also as a case structure) encompassing the smaller stacked sequence structure. Looking at the if/else box, we notice there is an operator connected to the small green question mark on the left hand side. This connects to an option box feeding either true or false to the operator (the green wire carries Boolean data) from a cluster (see earlier). The case structure then flicks to the appropriate page and only operates the instructions within that particular one. For example, inputting true causes no operations to happen (the lower structure has no operations inside it – it only passes the inputs straight to the outputs.

The smaller structure/box contained in the 'false' part of the case structure is the stacked sequence structure. This has no operator and instead operates one page of instructions at a time in sequence. The box drawn underneath the main diagram shows what instructions are next in sequence.

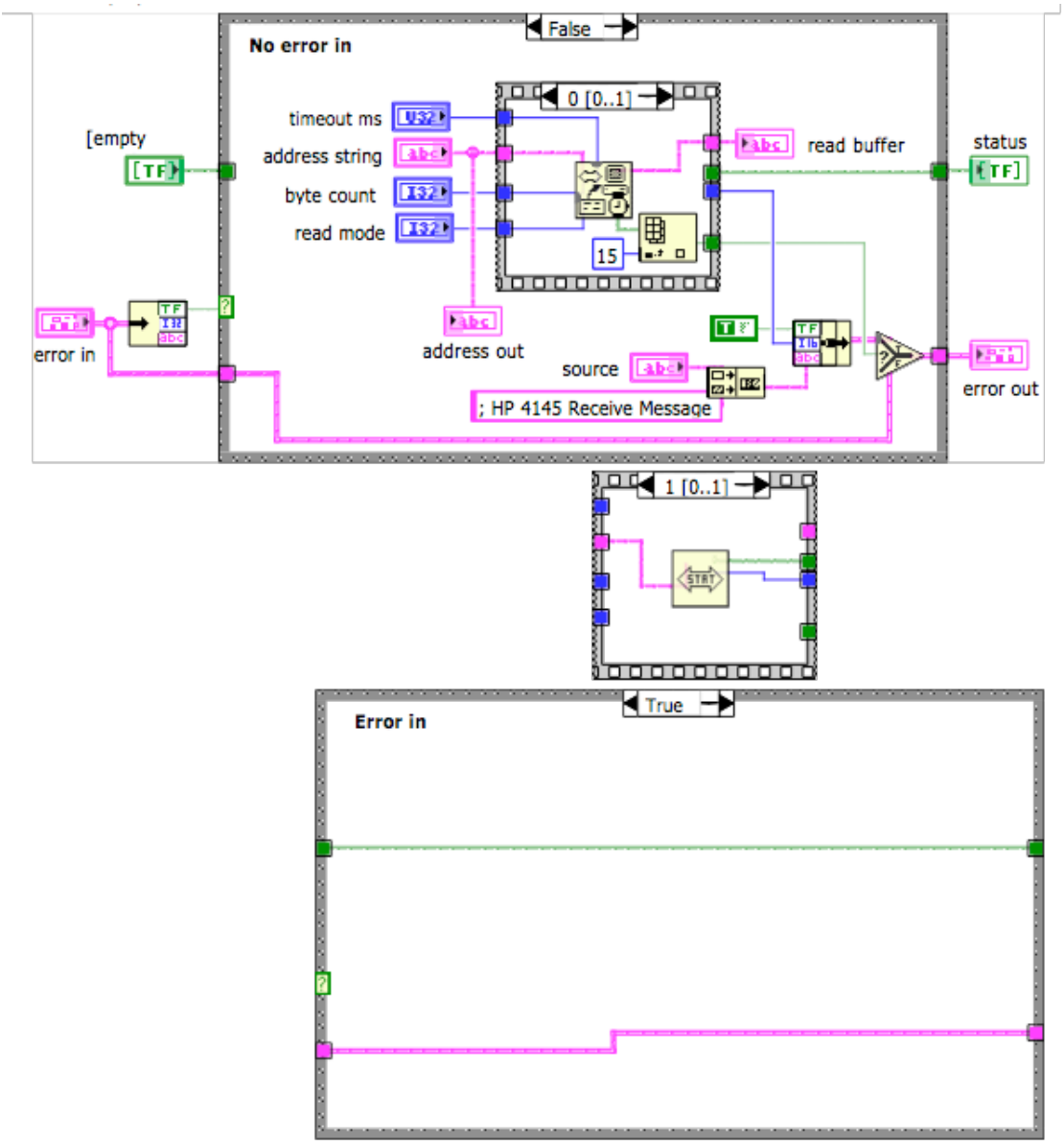

Figure 2.3 – *Stacked (smaller box) and case (larger box) structures*[own screenshot]

2.3 – Problems with the existing solutions

National Instruments provides a VI that was designed for use with the HP4145B (a later model) semiconductor parameter analyser and was originally programmed using LabView version 2[5]. The HP4145B was the model created by Hewlett Packard after the HP4145A, and shares a lot of characteristics with the older model. The major differences were the inclusion of a Floppy Disc (A) drive and the

increased number of measurement points the newer version could handle (1140[4] compared to 575[3]).

The VI created by National Instruments uses individual GUI panes to allow the user to set-up each individual part of the measurement being carried out by the device. Each 'page' the HP4145A has (see section 2.1) is recreated with its own VI. This means a typical user has to open up many windows on their desktop to control each aspect of the measurement process, cluttering the users desktop and making the control of each aspect hard to manage, as they need to flick between different windows regularly. A far better solution for most situations would be to use one single window to control all the major aspects of the HP4145A so that measurement/testing of a component can be done relatively simply.

Testing of the VI available from National Instruments showed that no graph could be obtained using the 'graphics plot.vi' (the program used to obtain graph data from the HP4145A). No data was being sent back to the computer along the GPIB bus (after checking GPIB activity using National Instruments GPIB analyser). Other aspects of the VI (Virtual Instrument) worked correctly, such as setting up the measurement channels, configuring measurements and setting up the type of result obtained (such as list, matrix or graph). As mentioned previously, these parts were controlled within their own individual VI.

In terms of aesthetics, the Virtual Instrument uses no design inspiration from the device itself. It was made clear in the specification that it would be useful to have the VI look as well as operate as close to the actual device as possible. This current solution doesn't do either, requiring the user to learn how to use both the HP4145A itself and the Virtual Instrument separately.

Other solutions that are available that do not use LabView include a piece of software from Metrics™ technology (http://www.metricstech.com/) called win4145©. This offers the user the ability to control the HP4145 using Microsoft™ Windows™ style icons and menus. However, the software does cost $1750 per licence (the National Instruments™ solution is free with a valid account and a full LabView software licence) and is compatible with only a limited number of GPIB cards.

3.0 –Project Specification

The specification from the project plan, written at the beginning of the project, has been modified slightly into the following specification. The points listed below were drawn up by consultation with the presiding academic (Professor PA Mawby).

- Must recreate the look of the front panel of the machine itself with all the functionality buttons the same (except for text and number entry as these will be easier to use if the keyboard of the computer is utilised)
- VI should allow control the vast majority of the HP4145A's functions including measurement and channel setup (e.g. voltage/current variation/sweep and constants), measurement itself (number of data points, repeating measurements, appending measurement), displaying measurement results (accurately display graphs, lists and/or matrices as they appear on the device as well as exporting to spreadsheet programs), setting the type of measurement result (e.g. graph, list, matrix).
- Should operate efficiently – i.e. should operate quickly on a users machine with no hanging or freezing.
- Must limit the number of LabView files that need to be opened in order to reduce cluttering of the computer workstation screen. This will require integration of all functionality into one complete virtual instrument and this should follow the operation of the device itself as closely as possible.
- The virtual instrument should operate as close to the original machine as possible. This means following the following steps for analysis of a device:
    - Definition of the Channels SMU1-3 – e.g. voltage, current etc.
    - Definition of the variables/parameters to be measured
    - Set-up of graph scales

    - Acquisition of data and display of graph and table/matrix of data
- Display the data in a graph that allows the user to zoom into particular regions, resize it on the screen and export the data to a file that can be opened using any spreadsheet application. The data should be properly formatted – i.e. results should be in columns next to the variable.

4.0 – Literature Review

This section will discuss texts and other research material that were consulted whilst completing this project. Most cover the subject of LabView software development. Prior to conducting the project, LabView needed to be researched, as this was a development environment I had not encountered before. The texts used to do this include "LabVIEW® for everyone"; Travis, Jeffrey; 2002; Prentice Hall and this book was a good introduction to LabVIEW®, although it deals with version 6 (this project used 7.1), but offers a good grounding in the basics. This book was particularly useful when reviewing the National Instruments™ VI available for the HP4145A, and was the prime source of research for the project. It goes through many basic LabView® programming examples that are available in the LabView® help database (no need for extra downloads) and this was extremely useful for learning LabView® for the first time as it offered another explanation on top of the LabView help files. It contains lots of introductory literature on the history of LabView® and the associated standards which you learn about, giving a thorough background knowledge. The book is also written in an informal manner, meaning it is much easier to read in long spells compared to the other books (see later). It also details some basic programming of GPIB devices, although this is very general and the book itself recommends that other resources be used for aiding creation of Virtual Instruments for these devices. This book offered an easy quick reference guide whilst programming with short explanations of many of the function blocks and wire types. In particular, it was useful for understanding the different case structures and the flow of data between different stacked sequence structures (Chapter 6, page 161-179) as these proved fundamental to the VI created. Also important in chapter 6 was the understanding of shift-registers and when to use these within for loops (to allow data

flow continually out from the function). Another major research point with this book was being able to understand the role of sub-VIs and the VI hierarchy (chapter 3.9). This was important, as without sub-VIs an overall VI to control all aspects of the HP4145A's functions could not have been produced.

Another text used for the early part of the project (when the National Instruments™ VI was being examined – see research methodology) was "Learning with LabVIEW® 6i"; Bishop, Robert H; 2001; Prentice Hall. This book was not as easy to use from a beginner's perspective, but did contain lots of useful examples and was a good back-up resource for the above book. This edition does deal with LabVIEW® version 6, but all skills learnt were easily transferable to the new version (7.1). As in the previous text, there were explanations of the examples included with the LabView help database. Important topics from this text included working with arrays and this helped in the understanding of how the original VI would build an array suitable for the graph plotting function. This was also important to grasp, as the array created would need to be siphoned off to allow output of the data in a correct format to the output file. Overall, this text provided a good companion to the above text.

Another main point of reference were the LabView® help files themselves, as they provided examples of each function block being used, as well as fairly detailed explanations on the functions of each. This was also backed-up by the National Instruments™ website and developer zone (www.ni.com) with some forum discussion on development for the HP4145A (http://forums.ni.com/ni/board/message?board.id=170&message.id=193466&view=by_date_ascending&page=1). These forum posts however can sometimes be brief and difficult to follow, and so only supply a good secondary research medium.

A major research tool was the manual for the HP4145A device, as this provided thorough background knowledge on the functionality of the HP4145A as well as how to program for external control of the device (i.e. what codes to send to conduct each task – Chapter 3-88). This text was also fairly easy to follow, with good explanations of each of the devices functions and excellent information on GPIB.

4.1 – Research and Production Methodology

The project was conducted by reviewing the National Instruments™ VI first[5], and then attempting to use as much of that original code in this solution to save on development time. To do this, LabView development was researched whilst looking through the code (using the texts identified above) and attempting to understand the functionality of each part of the VI. Once the problems with the original VI had been identified, solutions to these were built into the file 'Graphics Plot.vi' and then the other functions were added to it, such as channel definition, source setup as well as minor controls such as measurement type and integration time. This eventually led to an amalgamated Virtual Instrument that the user could control the entire measurement and analysis/results process from. As much as possible, the original VI functions were used as sub-VIs within this solution (if they operated correctly) to save time creating new ones. If these could not be used, then research was done using the HP4145A manual as a guide to try and instruct the device to do what was required. One example of this was in the creation of the integration time control for each measurement (see section 5 below and figure 5.3 (c)) by sending IT1, IT2 or IT3 commands depending on the type of integration time required.

Once this had been completed, testing was carried out using different measurement scenarios and the results of these used to improve general performance

of the VI. Also, as discussed in section 6, further testing has been carried out by publishing the VI on the National Instruments website.

5.0 – The New Virtual Instrument

This section details how the new Virtual Instrument works and the functions that it offers to the user. This is useful to describe as it can explain how much of the specification has been achieved (see the conclusion sections (6)).

5.1 – The Front Panel and overall aesthetic design

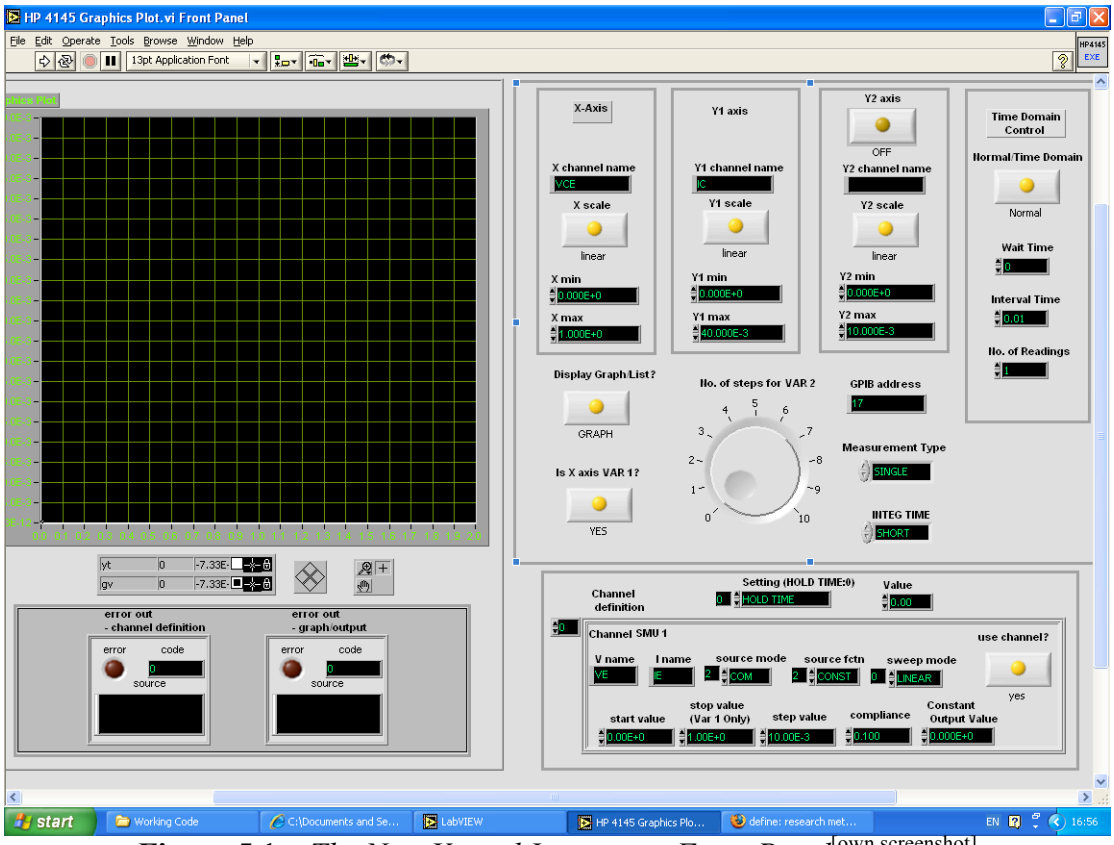

Figure 5.1 – *The New Virtual Instrument Front Panel* [own screenshot]

The front panel is what the user sees upon launching the new VI. It allows control of all measurement aspects and can also display graph results on its screen as well as output lists and matrices to an output file. To control the device, the user can use the method in figure 2.2 but does not need to move to other screens as the front panel in figure 5.1 can control almost everything (see specification). The front panel is divided into different sections using dark borders to 'mask off' the different sections, easily allowing the user to differentiate between the different parts of the

measurement process. The sections below will detail each part of the front panel in more detail, of which there are 4 basic sections; channel definition, output and measurement control, graph plotter and error code output. The colour scheme and overall design has taken cues form the HP4145A itself, using light grey for the main instrument panel and a darker grey around the graph screen and error output section. The buttons attempt to represent those used on the device with a yellow LED in the centre informing the user if they are on or off. The text/number inputs in the channel definition pane (see section 5.2 for more detail) use the same green font and black background that the HP4145A uses. The dial on the front of the HP4145A has been recreated on this front panel but is now used to control the number of VAR2 (used of you have a second variable) data points. The graph controls seen underneath the display are part of LabView's built in graph plotter and were added here to allow the user more analysis options within the VI to check measurements and data points. They allow the user to add makers and therefore obtain accurate read-offs from the graph without having to output the data to a stand-alone spreadsheet application.

5.2 – Channel Definition Pane

To start using the VI the user sets each channel they want to use by using the mouse to click through all the available channels in the Channel Definition Pane (see figure 5.2 and explanation of how to work the HP4145A in section 2). Each channel has control for entering 'V Name' and 'I Name' (if the channel is either a voltage/current) and a selector switch for setting the source type (e.g. if the channel is a voltage source, current source or common – used for constants usually). The user can also set the 'source function' so the channel is a variable or a constant (again using a selector switch). Also present are number entry fields for setting sweep lengths (start value, stop value, step value) as well as the compliance of the channel

(if the channel is a voltage source, the compliance will be the current through the channel and visa versa for a current source). The user can also set the sweep mode and the output value if the channel is a constant (or is set as a Vs channel) and there is a large push on-push off button for setting the channel as either on or off. Also present here is a control that allows the user to set the Hold Time (time between each VAR2 measurement), the Delay Time (time between each VAR1 measurement), the VAR1' Ratio (the ratio between Var1 and Var1') and VAR1' offset (if the user wants VAR1' to be offset from the variable VAR1). (Note: - VAR1' is a variable that changes in relation to the main VAR1 variable). Not all parts of the channel definition may be needed, for example a constant doesn't need any maximum or minimum defined, and so these fields can be left blank. Figure 5.2 shows the channel definition pane in more detail.

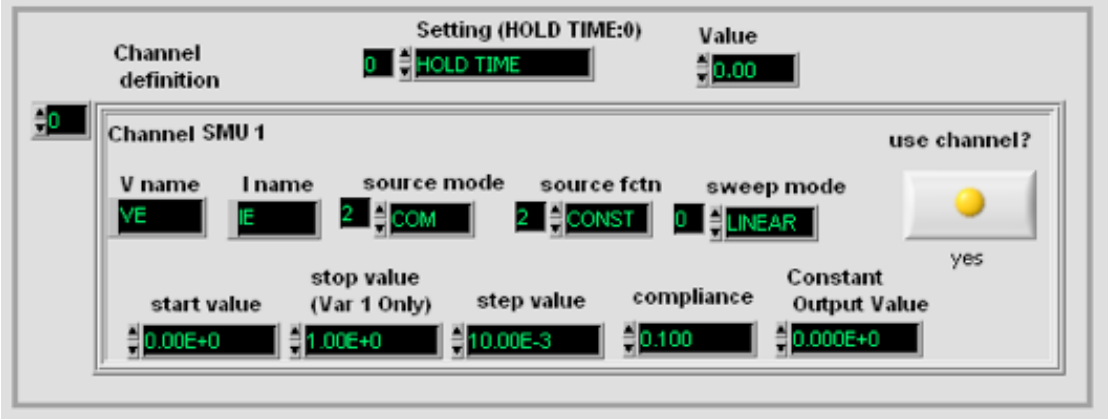

Figure 5.2 – *Channel Definition part of the front panel*[own screenshot]

5.3 – The Output and Measurement Control Pane

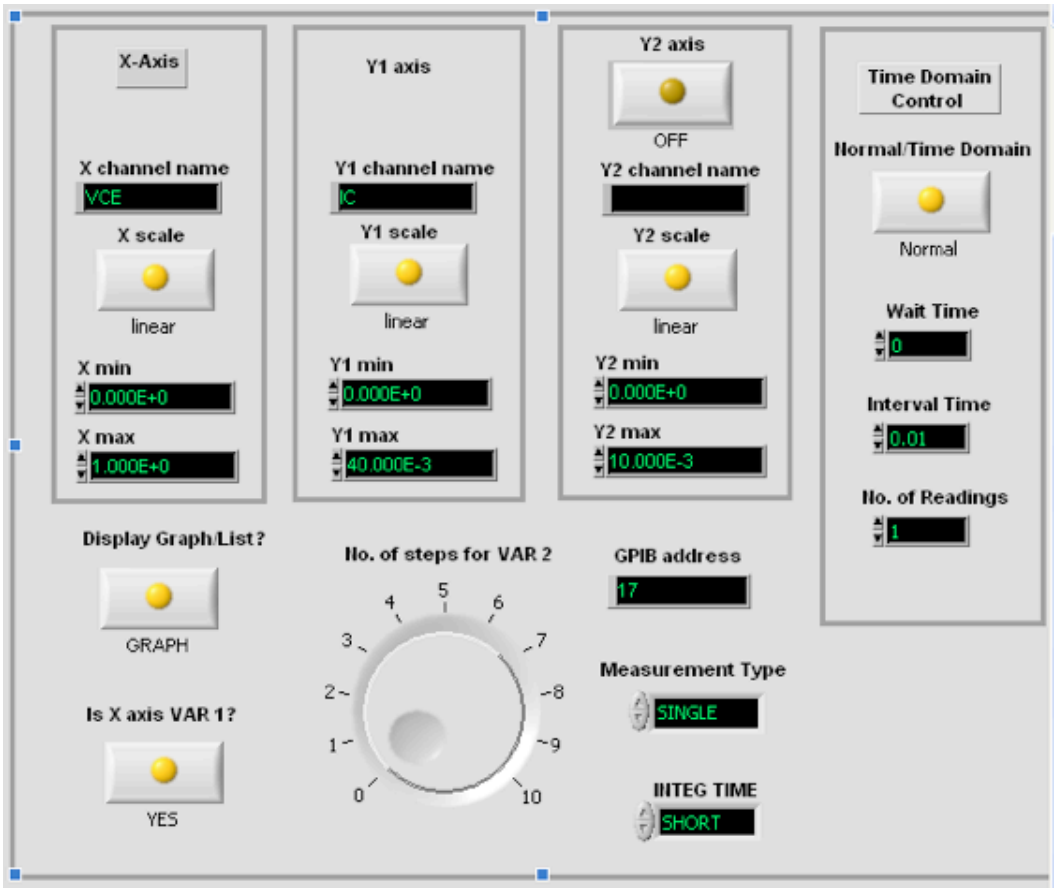

Figure 5.4 – *Output Control Pane from front panel*[own screenshot]

The next step the user takes in setting up the measurement is to organise the output wanted. The VI by default outputs a file that can be opened in a spreadsheet application with the measurement results on it and the user can select whether they wish to have a graph output or a list (see Figure 5.4 for a more detailed view of the output controls). If graph is selected, the VI will create a graph on the screen and write the graph data to a file that can be opened using another application (e.g. a spreadsheet application). A List will simply instruct the machine to record all the results onto a list and then output this to a file (no graph will be displayed on the VI). Also present in the output control pane are text entry boxes for naming each axis or list item (X, Y and Y2 (if used) – see section 2 for how to set-up the HP4145A

properly) and defining the scales for the graph. The graph on the Virtual Instrument itself is set to automatically adjust it's scale to best show the data but scales still need to be defined for the HP4145A to operate correctly (these also must be the same range of values selected in the channel definition pane for the output file created to be formatted properly). If the user only wants the graph on the VI (no output file required), these scales can be set to any range desired.

Other controls here include the setting the number of steps Variable 2 (VAR2) has, setting the measurement type (Single, Repeat, Append) and the Integration time (time spent on each measurement). The user can also decide to have a time domain measurement by using the controls on the left of the screen.

5.4 – The Graphic Screen

This part of the front panel displays graph data and allows the user to read off data points from it. Figure 5.5 shows the graph screen in more detail. The graph is set to automatically choose the appropriate scale to best show all the data points, but this can be turned off by the user and the scales adjusted manually (by right clicking and accessing the graph options). The graph uses the same green font and black background to mimic the HP4145A as much as possible, except for multiple variables (as in the example depicted in figure 5.5) where different colours are used to best show all the data points clearly. Again, this can be changed to the users liking by accessing the options menu.

Underneath the main graph are the tools where the user can add markers to better read-off data from the graph. Other tools can be added or removed again by accessing the graph's menu items (by right clicking).

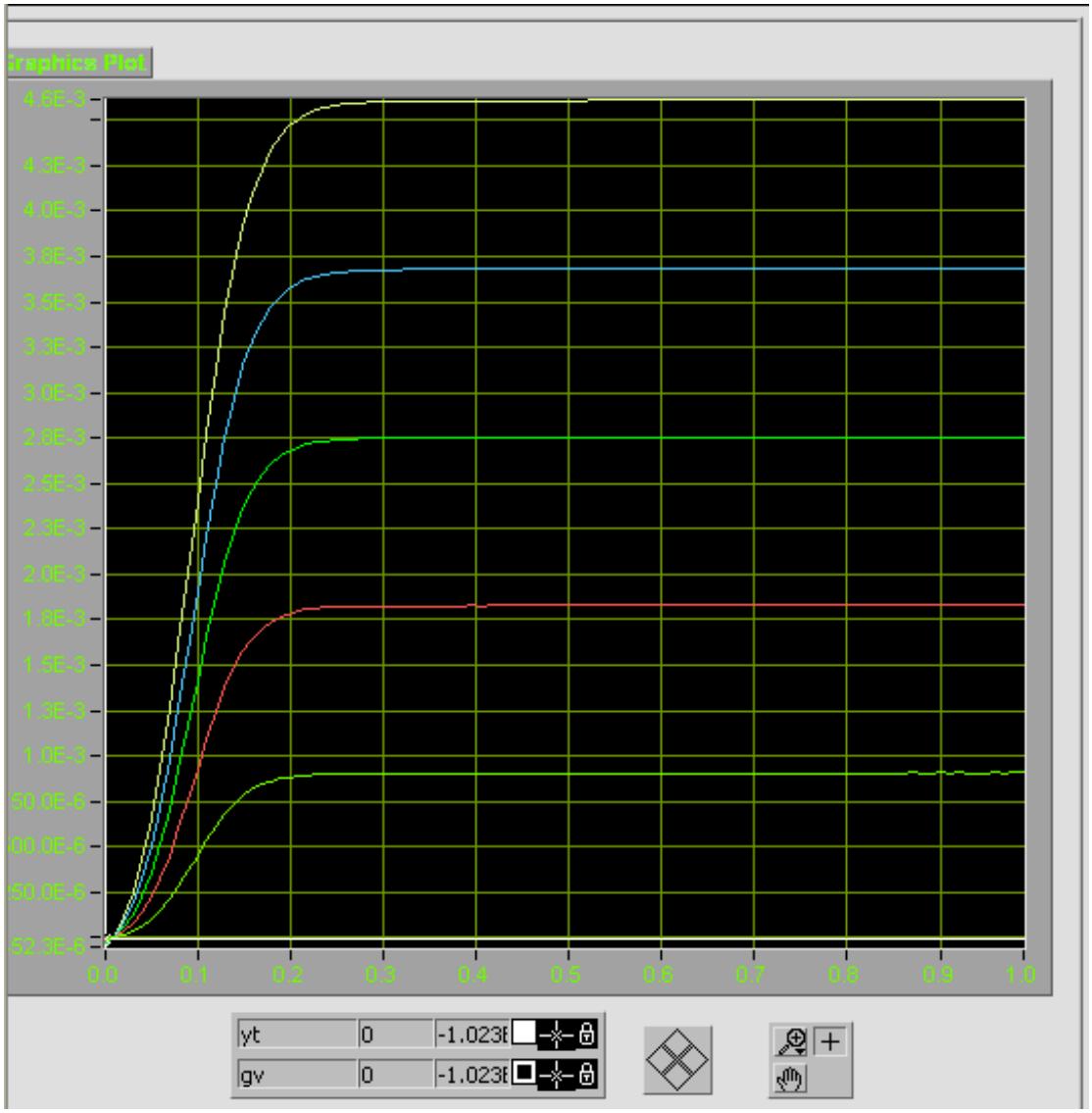

Figure 5.5 – *Graph screen showing transistor analysis graph*[own screenshot]

5.5 – Error Output

This section is located underneath the graph screen and is used to inform the operator of any errors encountered during the last test. The two boxes represent the two major parts of the device; channel definition and data output. When an error occurs in one of these areas the red LED in the corresponding box will illuminate and a code (if one exists for the type of error) will appear in the text field. A description of which sub-VI, or process, encountered the error will appear in the larger text box at the bottom. Figure 5.6 shows the error output boxes in more detail.

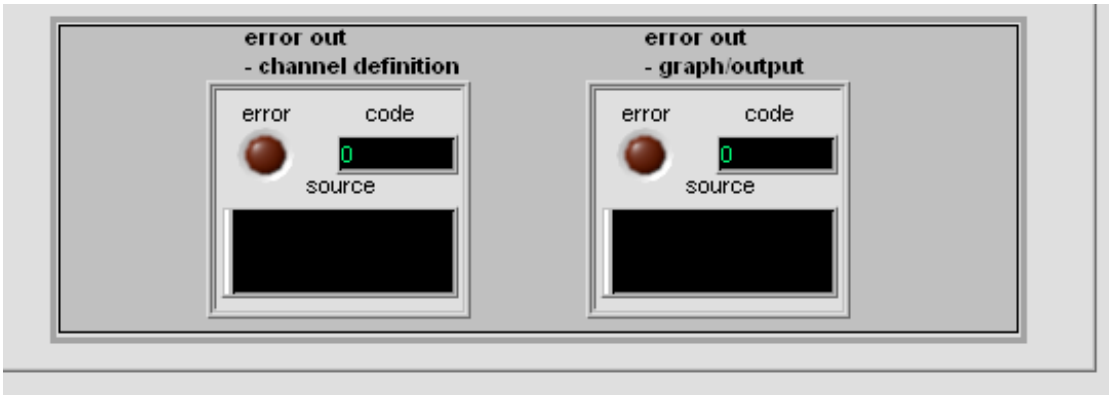

Figure 5.6 – *Error output boxes*[own screenshot]

5.6 – The Block Diagram

Each part of the front panel is programmed using a block diagram (as described in section 2). This can be thought of as analogous to a flow diagram, and this is the programming language used by LabView (called V).

Figure 5.3 shows the main block diagram (i.e. this is the top level of the program) showing all the sub-VIs and the contents of the different stacked sequence structures and if/else operators (both of these are represented by large grey boxes, the stacked sequence has a number control at the top, while the if/else box has true/false). The print out shows all the possible states of each box to better show here all the possible combinations. The largest stacked sequence structure in the diagram is the main control and operates the HP4145A in the order required by figure 2.2, that is it sends the user's channel definition data first (sequence number 0), then sets up the type of measurement required on the next page (sequence number 1) before finally sending the information required to setup outputs (sequence number 2) such as graph, list or matrix. The different coloured boxes on the far left are the inputs from the front panel and are mainly for the graphics setup. The exception being 'GPIB address' which takes the number entered by the user and supplies it to the functions that need to know the address of the device in the GPIB bus. Another is the dial on the front

panel controlling the number of steps of VAR2 and the 'error in' cluster. The error in/out system allows all the functions to output errors and allows the VI to identify which function returns an error.

Figure 5.7 (a) in the Appendix shows the block diagram for channel definition in more detail. Within here, a 'for' loop is utilised that will take each channel in turn (SMU1, SMU2, SMU3, SMU4, VS1, VS2, VM1 and VM2) and send the data to the HP4145A. Each channel's data (e.g. Channel ON/OFF, V Name, I Name, Sweep type etc) has been clustered together by LabView from the Channel Definition Pane on the Front Panel. The data cluster (represented by the thick pink line) is 'unbundled' using the 'unbundle' block (yellow tall block) into each separate element (see section 2 for more information on clusters). These are then fed to each of the function blocks (white blocks), which handle setup of the different channels. Each channel's data is unbundled in turn depending on which number the 'for' loop is on – so SMU1 is n=0, then SMU2 is n=1 and so on till n=7 (VM2).

The white function blocks were copied from the National Instruments™ VI and each one contains it's own front panel and subsequent block diagram that can be manipulated if needed (they are not opened when this VI is run). The white function block not surrounded by a black border is called 'HP4145 DEFINE CHANNEL' and this is used to populate the first page of the HP4145A's display with the channels being utilised (this is shown in figure 5.7(b) in the appendix). The other white function blocks present in the block diagram are (from left to right) 'VAR1 SOURCE SETUP', 'VAR2 SOURCE SETUP' and 'CONSTANT SOURCE SETUP' and these populate the 'Source Setup' page of the HP4145A (shown in figures 5.7(c), (d) and (e) in the appendix). The dark borders around these functions are 'Case Structures' and used to determine which is run and work the same way as an 'if' statement in

Java or C/C++ programming (see section 2 for more explanation on the if/else structures). Their operator is a Boolean variable created by comparing the 'Source Function' field to a constant. Figure 5.2 shows that each option in the 'Source Function' field is numbered 0 to 2 and so it is these numbers that are the ones compared to the constants. For example, if the user set channel SMU1 to be VAR1 (variable 1) then Source Function would be 0. This number is compared to 0, 1 and 2 for each case structure (from left to right) and a 'TRUE' value is returned only at the first one. This allows the function 'HP4145 VAR1 SOURCE SETUP' to operate while stopping all the others in this example.

Looking back to the main block diagram (figure 5.3), the next part of the main stacked sequence that is operated (n=1) controls the Hold/Delay time and Var1' offset/ratio and this can be seen in more detail in figure 5.8 (a). Only one of these can be operated at any one time (in line with the function of the HP4145A – see figure 2.2) and so makes use of a selector ring (user cycles through each available option with a mouse click). This is then used to select the appropriate command to send to the machine (using a 'Pick Line' function) before being sent as a string (along with the value required) to the device using the 'Send Message' function (copied from the National Instruments™ Virtual Instrument) and this can be seen in more detail in figure 5.8(b).

Referring back to figure 5.3, the final stage in the main stacked sequence structure controls the data output options – that is setting up a list, graph or matrix output as well as writing data to an output file or setting up a time domain measurement. There are lots of sub sequence and case structures such as one

containing the time domain setup function block as well as one for a list and graphics display, and all these are selected depending on the state of the buttons pushed on the front panel (case structures controlled by Boolean controls). There is also a send message function, which sends the integration type (short, medium or long) as well as 'Take Measurement' which instructs the machine to begin measuring (this was taken from the National Instruments™ VI).

The nested sequence structure (the structure on the right hand side of figure 5.3 and in more detail in figure 5.10) has three stages, each performed in sequence (shown in figure 5.3 and parts (b) and (c)). The first contains a function block called 'DATA OUTPUT' (figure 5.9 (a)) and this sends an instruction to the HP4145A to tell it to output the data from the required SMU channel along the GPIB to the computer using command 'DO'. Prior to this there is another instance of the 'Take Measurement' (shown in more detail in figure 5.9 (b)) block that tells the device to halt its measurements (by sending the command ME4), but this is only used to stop the device when the user has selected 'repeat' as the measurement type. There are several time delays included between the take measurement and the data output blocks as sending the data output command before the machine has completed the measurements causes no data to be returned at all (this was one of the problems with the original Virtual Instrument from National Instruments™ and was the cause of no data being sent along the GPIB – see section 2). The Time delays are all set to different amounts depending on what type of measurements the user has selected, so a large integration time will trigger a dedicated time delay so that sufficient time is given to the HP4145A to make the measurement before the VI sends the data output command. The same is done for 'hold/delay' times and short or medium integration

times. All these different delays are selected using the extensive set of if/else case structures (within the large stacked sequence structure shown in figure 5.3).

On the left hand side of figure 5.3 there is the case structure shown in figure 5.9 (c). This is what controls the command to the HP4145A telling it to either display a graph or list on its display. Figure 5.9 (c) shows the commands used to setup a list output. A graph is setup using a sub-VI called 'HP4145A Display Graphics'.

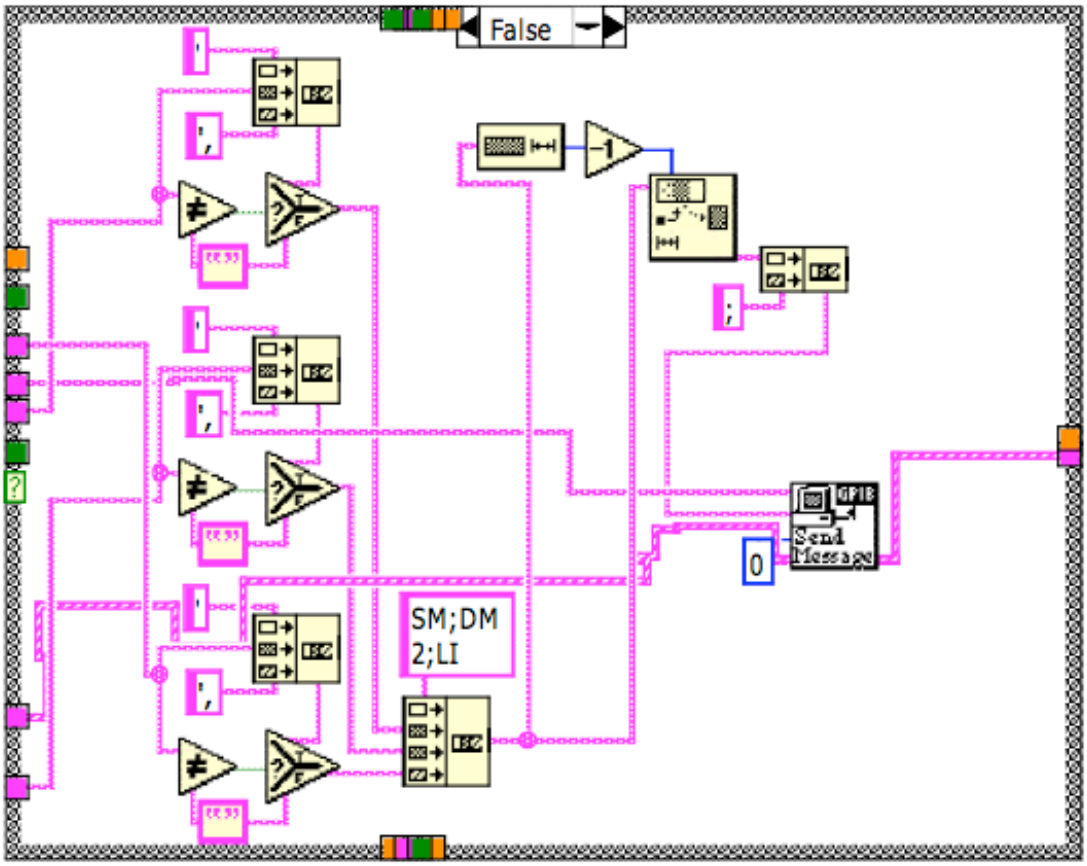

Figure 5.9 (c): *Nested case structure within the main block diagram*[own screenshot]

The 'DATA OUTPUT' block receives the data from the GPIB in the form of a string. The data returned from the machine is contained within a 2D array of values returned from the 'DATA OUTPUT' function block. After this, the x-axis is built using the data entered by the user (max value, min value) and this is contained within another array (see figure 5.3 (a)). Figure 5.3 (b) shows how this data is written to a

file using the ‘Write To Spreadsheet File’ function block from LabView’s Library. This is after the data (in a 2-D array) is combined together with the x-axis 1-D array into one array before being written to the file

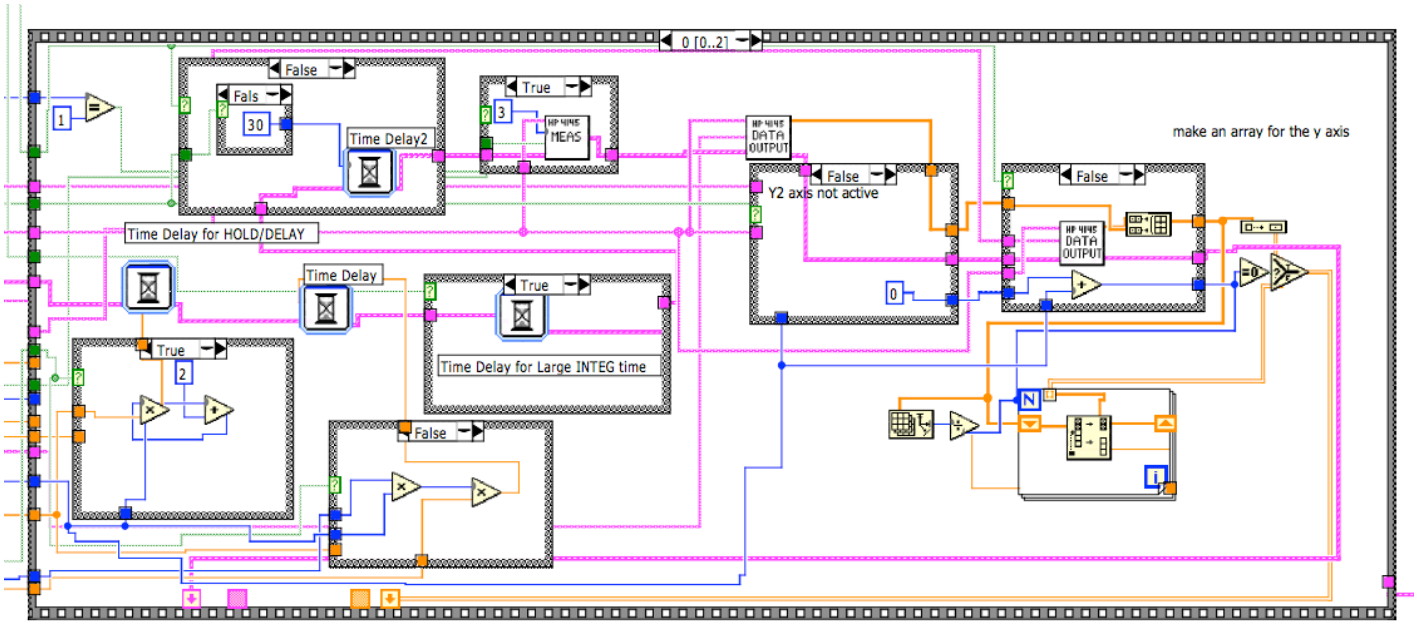

Figure 5.10: *First part of nested Sequence Structure from figure 5.3*[own screenshot]

6.0 – Analysis and Conclusions

This section analyses the Virtual Instrument and draws conclusions on how successful the project has been, both in terms of student learning and advancement Engineering profession as a whole.

6.1 – How much of the specification has been achieved

At this time the software has been completed and almost all of the required specification (see section 3) has been achieved. The front panel of the virtual instrument (the graphics the user interacts with to control the device) has been recreated to try and match the look of the device itself (see figure 2.1 for a picture of the actual HP4145A and figure 5.1 shows the front panel of the LabView Virtual Instrument).

As for functionality, the user can control almost all of the HP4145A's functions with the exception of a Schmoo display. Schmoo display can be used to show sweep dependent measurement results on a 3 dimensional graph (X,Y and Z domain). Unfortunately, this was not implemented in this Virtual Instrument. The original LabView VI[5] contains some function blocks that allow the set-up of the Schmoo graph, but this has not been integrated into this solution. This was going to be added before the project completion date, but this has not yet been done due to the development time needed to implement it. The major problem has been trying to add functionality to the block diagram to build 3D arrays – requiring a more complex algorithm than the 2D one seen in the VI. The graph display on the front panel can only display 2D graphs, so this would need to be replaced manually depending on the users requirements. List output can be used to acquire the Schmoo data, but a graph

needs to be created manually using a spreadsheet application. Discussed in the project plan was the possible inclusion of ActiveX® controls to allow easy linking to other applications. It was decided early in the project that it would have required much more programming and research into the subject in order to implement, and so was not worth adding to the final build. The VI currently writes result data to a text file that can be opened using generic text readers such as Microsoft® notepad, Apple Mac® TextEdit (LabView can run on both Windows™ and Mac OSX™) as well as word processors (it has been tested using OpenOffice™ version 2 and Microsoft® Word) and spreadsheet applications OpenOffice™ version 2 and Microsoft® Excel™).

Overall, the Virtual Instrument offers a more simplified solution to control the HP4145A from a Windows™ PC compared to the original VI[5]. It obtains data back from the device and displays this in a graph inside LabView and allows the user to export this data to a text file that can be opened by many different types of spreadsheet application. The specification required the VI to operate as closely to the device as possible, but this solution does not as everything has to be set at once before sending all the commands to the instrument. The areas on the VI to do each part are separated into distinct sections, so still offers a usable interface that any operator familiar with the HP4145A will be able to use.

6.1.2 - Testing

Since the publication of this project's progress report in week 21, the VI has been listed on the LabView (National Instruments) website (http://community.ni.com/examples/gui-for-hp4145a-semiconductor-parameter-analyser). This was to allow other people to test it and leave comments with how to

improve the software. Thus far (last checked on the 5th November 2007) there have been 100 downloads of the software, but no comments left on the webpage about problems or improvements that could be made. It is therefore hard to determine if this VI works correctly on other machines and setups or not as people are not communicating back their experiences with the software or any ideas of how to improve it. This VI has not been fully tested outside a controlled software development environment, only on the development machine. This was because it was difficult to locate a computer that had the necessary GPIB card and National Instrument drivers installed.

Another problem with the VI is that a list output can only show 3 variables (the HP4145A itself is capable of measuring six) as code has not been written to properly format the data before outputting it as a file (i.e. data is in one continuous column). Data is still returned, but it needs to be formatted manually. This can also happen if the user selects only two variables to be displayed on a list (although all the data is returned, it needs to be manually re-organised into separate columns). This was going to be added to the VI before the project completion date but has not been implemented due to time constraints.

The performance of the VI could also be questioned. At present it has only been tested on one PC with a 730MHz processor and 384Mb of RAM. It can be slow at running the VI after data has been returned back (sometimes taking as long as 2-3 minutes after the data has returned from the HP4145A), and this can probably be blamed on the operations used to create the arrays for that data as this requires large amounts of number manipulation. More testing on other systems is needed to see if the performance is the PC or the code, but this has not been carried out. In terms of memory usage, the VI uses approximately 437 Kbytes of space (according to

LabView's built in test) and so does not take up much of the systems resources. When run, task manager in Windows™ reports that CPU usage can be as high as 100% when the VI is running, and this occurs when data has been returned from the HP4145A and it is being manipulated into arrays. As this VI has been created in LabView, it may be able to run on other operating systems capable of operating the LabView environment, but this still needs to be tested.

At present, most problems encountered still return data to the user that simply needs to be re-organised to match the results shown on the HP4145A's screen, meaning they are not considered for this project to be critical bugs in the software.

6.2 – Future Work

There are some functions that could still be added to the VI if more time was given to the project. As discussed in the previous section, there are still some bugs present in the software (although not of a critical nature), and most are to do with timing and output of the data to a spreadsheet file. Timing makes sure that the VI does not send the data output command before the HP4145A has completed the measurement, and this is achieved using the time delay functions in the block diagram (see section 5 and figure 5.3 in the appendix). At present, if the user selects 'Delay Time' the VI can sometimes start to send the data output command before the device has finished measuring. One quick solution would be to include a time delay that switches on if the user selects Delay time and delays by 575[3] (the maximum number of allowed measurement points) multiplied by the delay time entered before sending the data output command. This would work, but the time from the measurement ending to the data being returned could potentially be massive and so a less crude solution will need to be found. Currently, the part of the VI that sets up the output (see figure 5.3 in appendix) cannot access the number of measurement points entered by

the user without them inputting it twice on the VI (one time in the channel definition pane and once in the data output pane – see section 5.3). If this could be rectified, the output control part could calculate the exact time delay required without having to use the extensive array of if/else case structures seem in figure 5.3.

The data written from the VI to the output file can sometimes be formatted wrongly and this seems to occur if the user selects list output and only measures two variables. This is probably due to the VI trying to setup the data so it can be used in a graph (as list output was added after the graph output had been developed) and so this will have to be looked at for improvement. Another problem is that if the user instructs the HP4145A to conduct a measurement and the number of data points is greater than 575[3] (i.e. the output buffer is saturated) the VI shows data incorrectly on its graph and the output file is wrongly formatted. This could be improved by including code that instructs the VI to count the number of data points and not divide them into equal columns of data, as one should be shorter than the rest (as it was cut short by a lack of memory).

In terms of the code itself (i.e. the block diagram), the different combinations of delay times (shown in more detail in figure 5.10 and in 5.3 (appendix)) could be combined together to offer more efficient code. As mentioned previously, the VI seems to run slowly with large amounts of output data and this must be due to the code run after the data is returned (figure 5.3). This may be due to the computer being used for development (it has a fairly low spec by modern standards – see section 6.1), but may indicate that the code used here is not that efficient and should be better engineered to accommodate slower computers.

A feature that could be improved is when the VI instructs the machine to conduct a repeat measurement, the user should be able to input into the VI how many

measurements are required and the VI will automatically stop the measurements once this is done.

Another improvement could be allowing the user to load in profiles from the HP4145A's floppy disk, rather than having to setup a measurement from scratch each time. When using the HP4145A directly, the operator can instruct the device to load in pre-programmed measurement setup parameters for commonly tested components (such as BJT or FET). It would be useful if this could be added to the VI so that the user can simply load a pre-programmed profile from the VI, which will then configure the device and allow measurements to begin much more quickly.

6.3 – Project costs and value

The project cost has two parts. Student working time is the largest contributor followed by academic meeting times. At the beginning of this project a PHD student was consulted briefly who gave information on why the previous VI was insufficient to meet the departments needs and the work he had undertaken to try and correct the faults. This was a brief 15-minute meeting and only happened once.

The project leader (professor PA Mawby) was consulted at various points during the year, and was consulted a total of 4 times (see project logbook – 6th October, 17the October, 23rd November and 27th January) during the project, and once when the project logbook was being marked (week 21 to discuss the project presentation). Each meeting was between 20 and 40 minutes in length. In total about 3 hours has been used in academic consultation (as a rough estimate).

As for student working time, this has been estimated at 300 hours approximately as this was the recommended time that a student should spend on the project. There were no other costs associated with this project. Below is a full breakdown of the costs involved:

Cost	Time (Hrs, Minutes)	Total Cost (£)
Student (£15 per hour)	300 Hours	4,500
Academic (including PHD student) (£50 per hour)	3 Hours	150
	Total Cost:	**4,650**

Table 6.1: *Cost breakdown for the project (costs per hour taken from project handbook – University of Warwick)*

This total cost is quite minimal compared to other projects as the student working time should not have been much shorter than 300 hours. Academic consultation was also minimal, reducing the costs further. There were also no consumables or other overheads used in this project as only the HP4145A and the development computer (with a full copy of LabView) were needed. The computer included a working version and licence of LabView and so these costs were not added to cost report as the University supplied them.

6.4 – Final Conclusions

Overall the project has been a success, with the main parts of the specification being achieved. The VI created allows the user to interactively control the HP4145A from a computer, and is very close to the actual device in terms of functionality and aesthetics. The VI displays a graph of the measurement results obtained by the HP4145A, allowing the user to acquire results straight from the VI. These can also be exported to a spreadsheet file and manipulated using dedicated spreadsheet applications. The VI operates well, although with a few data formatting bugs, and always returns results to the user. So far, the VI developed has been downloaded 100 times from the National Instruments website by other engineers using the HP4145A. There are still some areas where the VI could be improved if needed, such as allowing

the user to specify how many measurements are done for a repeated measurement. These can be added quite easily by other developers however, as LabView allows quick access to the source code of the VI.

Glossary and Definitions

HP = Hewlett Packard® Company
HP4145A = Semiconductor Parameter Analyser developed by Hewlett Packard Company
GPIB/HPIB = General Purpose Interface Bus/Hewlett-Packard Interface Bus –An interface bus developed by Hewlett Packard that became the standard interface bus recognised by the Institute of Electrical and Electronic Engineers (IEEE)[2].
VI = Used in this report to describe a Virtual Instrument. This is a piece of software that allows control of external devices using National Instruments™ LabView® software. This abbreviation comes from the file extension used for these files (.vi)

References

[1] = http://www.hit.bme.hu/~papay/edu/GPIB/tutor.htm
[2] = www.wrightcolorgraphics.com/g.htm
[3] = Hewlett Packard 4145A semiconductor Parameter Analyser operation and service manual; Copyright: Yokogawa-Hewlett-Packard LTD, 1982; Printed Jan 1983; Manual Part Number: 04145-90000
[4] = http://www.labx.com/v2/adsearch/detail3.cfm?adnumb=334529
[5] = Original driver (LabView 1994): http://sine.ni.com/apps/utf8/niid_web_display.download_page?p_id_guid=E3B19B3E948F659CE034080020E74861
[6] = http://www.eng.warwick.ac.uk/staff/elh/elh_es261_notes/notes1_m_2000_files/image064.gif

Appendix