1-1-1983

An examination of the use of an HP9836 desktop computer in microelectronic device characterization experiments.

F Matthew Rhodes

AN EXAMINATION OF THE USE OF AN HP9836 DESKTOP COMPUTER IN MICROELECTRONIC DEVICE CHARACTERIZATION EXPERIMENTS

by

F. Matthew Rhodes

A Thesis
Presented to the Graduate Committee
of Lehigh University
in Candidacy for the Degree of
Master of Science
in the Department of
Electrical and Computer Engineering

Lehigh University
May 1983

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

May 3, 1983
Date

Professor in Charge

Chairman of Department

ACKNOWLEDGEMENTS

The author is deeply indebted to Dr. Marvin H. White for both his guidence and support as well as the excellent research tools provided. In addition, the help of the graduate students at Sherman-Fairchild Laboratory has been greatly appreciated, especially the help of Anant K. Agarwal.

TABLE OF CONTENTS

	PAGE
Title Page	i
Certificate of Approval	ii
Acknowledgements	iii
List of Tables	vi
List of Figures	vii
Abstract	1
Introduction	2
Part 1 - Pulsed C-t Experiment	
Theory of the Pulsed C-t Experiment	7
Pulsed C-t Experimental Procedure	19
Discussion of Pulsed C-t Automation	31
Part 2 - Admittance Measurements	
Theory of Admittance Measurements	38
Admittance Measurements Procedure	53
Discussion of Admittance Measurement Automation	67
Conclusions	69
Appendix A - Derivation of Zerbst Analysis Technique	71
Appendix B - Admittance Between Conductance Band and Interface Trap	74
Appendix C - Admittance of a Continuum of States	77

Appendix D - Statistical Admittance Model 79

Appendix E - Pulsed C-t Automation Program 81

Appendix F - Admittance Measurements Data Gathering Program 122

Appendix G - Admittance Measurements Analysis Program 150

References 180

Vita 181

LIST OF TABLES

	PAGE
Table 1.1 - Device Parameters for Measured MOS Capacitor	24
Table 1.2 - Data Analysis for Device Pulsed from Accumulation to Inversion	25
Table 1.3 - Data Analysis for Device Pulsed from Inversion to Heavy Inversion	26
Table 2.1 - Comparison of Predicted and Measured Values of σ_s	66

LIST OF FIGURES

FIGURE		PAGE
Schematic of Measurement Setup		5
1.1	Schematic Representation of Pulsed C-t Experiment	14
1.2	Equivalent Circuit of Lehovec & Slobodsky	15
1.3	Simplified Equivalent Circuit	16
1.4	Equivalent Circuit Considered by Hofstein	17
1.5	Schematic Representation of Hofstein's Experiment	17
1.6	Device Considerations of Nathanson & Schroder	18
1.7	Experimental Configuration of Pulsed C-t Experiment	23
1.8	Microphotograph of Circular MOS Capacitor	24
1.9	Capacitance vs. Time Curve Accumulation to Inversion	27
1.10	Zerbst Plot - Accumulation to Inversion	28
1.11	Capacitance vs. Time Curve - Inversion to Inversion	29
1.12	Zerbst Plot - Inversion to Inversion	30
2.1	Schematic of MOS System and Band Diagram	46
2.2	Schematic of Trapping and Emission Processed	47

2.3	Equivalent Circuit of MOS Device	47
2.4	Extended Equivalent Circuit to Include Hole Effects	48
2.5	Alternate Equivalent Circuit	49
2.6	Plots of $(C_p-C_d)/C_t$ and $G_p/\omega C_t$ vs. freq.	50
2.7	Equivalent Circuit for Continuum of Traps	51
2.8	Comparison of Conductance Curves for Each Theory	52
2.9	Admittance Measurement Experimental Setup	59
2.10	Measured Circuit	60
2.11	Measured Quasi-static C-V Curve	61
2.12	Curve of D_{it} vs. Energy	61
2.13	Curve of Surface Potential vs. Gate Bias	62
2.14	Admittance Curves Measured at 1kHz	63
2.15	Measured and Fitted Admittance vs. Frequency Curves	64
2.16	Admittance vs. Frequency Curves for Various Charge States of MNOS Device	65

ABSTRACT

This thesis examines the use of an HP9836 desktop computer in two microelectronic device characterization experiments. The first is the Pulsed C-t experiment utilizing the Zerbst analysis technique. From this experiment values of surface recombination velocity and recombination lifetime are derived. The second experiment is an admittance measurement with the analysis based on the theory of Nicollian and Goetzburger. The theory of the measurements are examined. A discussion of automation techniques and advantages is included.

INTRODUCTION

Today's society is increasingly affected by the advent of the computer. It is utilized to save man the drudgery and boredom of well-defined repetitious tasks and speed the results of tedious and often untenuable numeric calculations. It has entered the home, the business office and in the case of current concern, the experimental laboratory. The question addressed by this thesis is, "What is the role of the computer in an experimental laboratory?". To resolve this question, the paper will examine the use of an HP9836 desktop computer in the control and analysis of several microelectronic device characterization experiments. Through these experiments the advantages and limitations of the computer's use will be apparent.

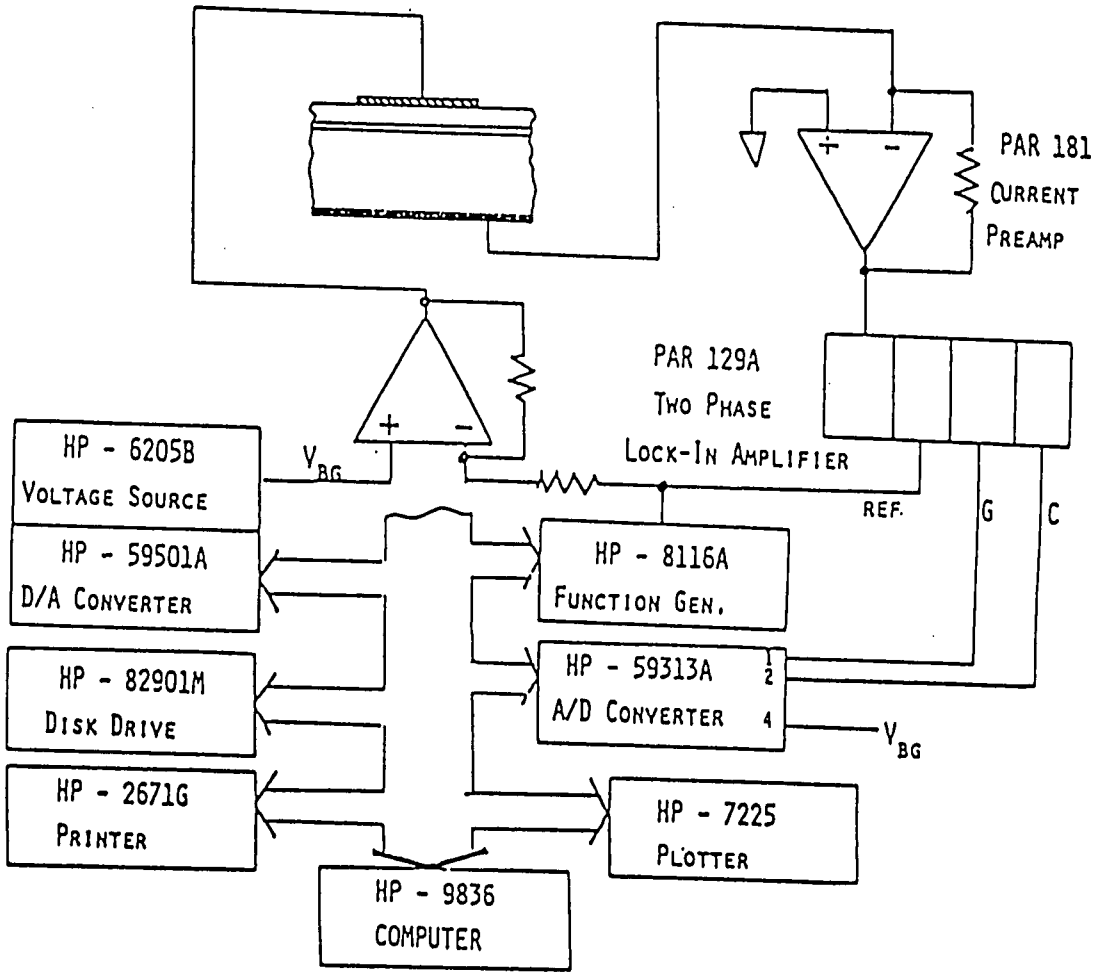
Each experiment begins with a discussion of the device physics involved followed by experiment documentation. The last part of each section includes a discussion of the programming techniques and their advantages.

The thesis concentrates on two experiments. The first experiment is the non-equilibrium or 'Pulsed' C-t experiment. Here the computer applies a voltage pulse across the terminals of a MOS capacitor and the

resulting capacitance transient is monitored. This experiment is especially interesting because it demonstrates the measurement of time by the computer. The data in this experiment is analyzed with the Zerbst technique. The second experiment measures the admittance of a MOS capacitor as it is swept through various equilibrium bias states. In this experiment the time factor is not a problem; however, there will be much more instrumental control involved. The data is analyzed with a theory developed by Nicollian and Goetzberger. This experiment is more concerned with the interaction between the user and the computer, since in this case control is not completely automatic. Instead some compromise between automatic and manual control causes some special programming considerations.

The equipment available in the laboratory for automation includes the HP9836 computer and several peripheral instruments which the computer communicates with over a Hewlett-Packard Interface Bus (HP-IB). The measurement equipment includes an A/D converter with four measurement channels for data acquisition, three D/A converters for DC voltage control and a programmable function generator to create measurement signals. A

schematic of this setup is included on the following page. This equipment and the software packages developed form the basis of this thesis.



Schematic of Laboratory Set-up in Device Characterization Lab

PART 1

PULSED C-t EXPERIMENT

THEORY OF THE PULSED C-t EXPERIMENT

The problem, in general, is to study how a MOS device comes to thermal equilibrium after an external electrical disturbance. More specifically, a MOS device is pulsed from some equilibrium bias level to a non-equilibrium deep depletion level. The device capacitance is monitored as the charges redistribute until a new equilibrium state occurs. The mechanisms of charge redistribution will be discussed and several proposed methods of analysis will be presented. The analytical method of Zerbst /6/ will be discussed in detail with experimental results given. Further correction to this analysis will also be given.

When the device is pulsed into deep depletion the majority and minority carrier quasi-fermi levels (Imrefs) will separate. Because of this carriers will redistribute themselves until equilibrium is reached (Figure 1.1). There are basically three electrical paths through which charge can redistribute: 1) Carrier pair generation in the bulk, 2) Carrier generation through surface states, and 3) Diffusion of carriers through the bulk (minority carriers). Historically some sort of resistive network has been associated with each of these paths so that the

MOS capacitor can represented by an A.C. equivalent circuit. This equivalent circuit was first proposed by Lehovec and Slobodsky /7/ and was fully discussed by Hofstein and Warfield /2/. The equivalent circuit is shown in Figure 1.2.

The equivalent circuit proposed is quite cumbersome and some simplifying assumptions must be made before it can be analyzed. First, the modulation of the inversion layer width is small and so the inversion capacitance can be considered infinite. The capacitance associated with the surface states is small and can be neglected. Applying these approximations the equivalent circuit becomes three resistive paths to ground in parallel with the depletion capacitance all in series with the oxide capacitance. The relative importance of these paths was carefully considered by Hofstein and Warfield /2/. They concluded that under the conditions of heavy inversion the resistance associated with the pair generation will dominate the response. The resistive path through the surface states may also be effective in devices with low bulk generation rates (high lifetimes). See Figure 1.3 for the equivalent circuit.

As a note, if there is a finite conductance along the surface of the device, then there will be an RC

network attached to the circuit of Figure 1.3 which will invalidate the analysis. The experiment therefore must be designed to minimize this surface conductance.

Two methods of analysis will be considered, one by Hofstein /1/ in which he considered only generation in the bulk and one by Zerbst /6/ in which bulk generation and surface generation were considered.

The analysis by Hofstein considers the equivalent circuit shown in Figure 1.4. For this circuit it was shown by Sah et al. /3/ that:

$$R_g = \tau_o \left(\frac{V_{sp}}{q n_i d} \right) \qquad (1.1)$$

where V_{sp} = voltage across space charge region

τ_o = generation lifetime

d = depletion layer width

With this expression for resistance the RC time constant of this circuit is:

$$\tau_r = \tau_o \left(\frac{V_{sp}}{q n_i d} \right) * (C_d + C_{ox}) \qquad (1.2)$$

Utitlizing the relations:

$$d = \frac{\varepsilon_s}{C_d} \qquad \& \qquad C_m = \frac{C_{ox}C_d}{C_{ox}+C_d} \qquad (1.3)$$

Equation (1.2) can be solved to $_o$ to give:

$$\tau_o = \tau_r \left(\frac{V_{sp}}{qn_i\varepsilon_s}\right) \frac{C_{ox}{}^3C_m}{(C_{ox}-C_m)^2} \qquad (1.4)$$

Hofsein used this result to experimentally measure the recombination lifetime. To make sure his analysis was valid Hofstein superimposed a small step voltage (∿25mV) over a bias voltage which kept his device in heavy inversion. This insured low surface recombination velocity since in heavy inversion most of the surface states are full. With his experimental setup he was able to obtain reasonable values of lifetime for his devices. Hofstein's experiment is represented schematically in Figure 1.5.

The analytical method of Zerbst is more complicated than that of Hofstein's in that it considers both surface effects and bulk effects. As shown in Appendix A, a relationship can be derived between the capacitances

measured in a non-equilibrium experiment:

$$-\frac{d}{dt}\left(\frac{C_{ox}}{C}\right)^2 = \frac{2C_{ox}}{N_d C_f}\left[\frac{n_i}{2\tau_g}\left(\frac{C_f}{C} - 1\right) + \frac{n_i C_f S}{\varepsilon_s}\right] \qquad (1.5)$$

If the bulk generation lifetime and the surface recombination velocity are constant, then a plot of $\frac{d}{dt}\left(\frac{C_{ox}}{C}\right)^2$ versus $\frac{C_f}{C} - 1$ will yield a straight line. Looking at the converse of this statement, a straight line fit to a good portion of a plot of these two variables provides a slope and intercept which yield the generation lifetime and the surface recombination velocity. This analysis technique is carried out in the experimental section.

A consideration due to Nathanson and Schroder /3/ is that upon application of a deep depleting voltage step a lateral spread of the space charge region occurs in addition to the longitudinal penetration. This lateral portion can be extremely important because the exposed surface does not invert leaving many surface states unoccupied and therefore maintains a high surface recombination velocity. The return to equilibrium then occurs due to three processes:

(i) - Bulk generation with lifetime τ_g

(ii) - Surface generation of the lateral surface with recombination velocity S_o (depleted surface)

(iii) - Surface generation of the surface under the gate with recombination velocity S.

Re-examining equation (A-3) in Appendix A in the Zerbst analysis in conjuction with Figure 1.6 the expression for $\frac{dQ_i}{dt}$ can be written as:

$$\frac{dQ_i}{dt} = \frac{qn_i}{2\tau_g}(W-W_f) + qn_i S\frac{A_g}{A} + n_i S_o(W-W_f)\frac{P}{A} \qquad (1.6)$$

where A_g = area of the gate = πr^2

A = area of the gate & lateral surface

P = peripheral length of the gate

For a circular gate of diameter d Equation (1.6) becomes:

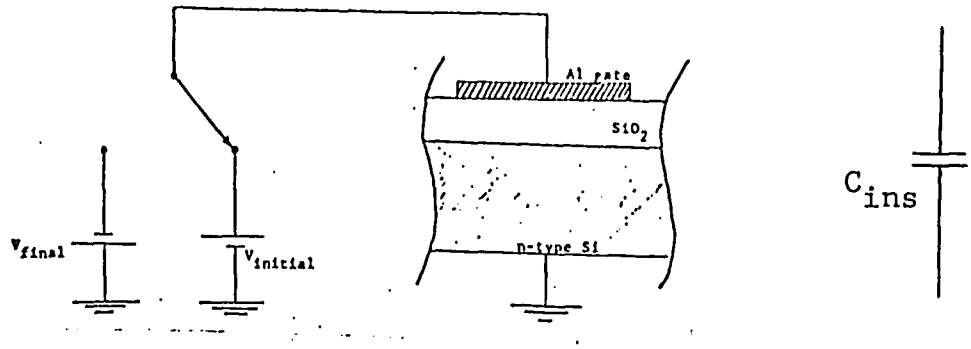
$$\frac{dQ_i}{dt} = qn_i(W-W_f)\left[\frac{1}{\tau_g} + \frac{4S}{d} + \frac{4S_o}{d}\right] + n_i S \qquad (1.7)$$

For the linear portion of the Zerbst plot $S<<S_o$ so that equation (1.7) can be written as:

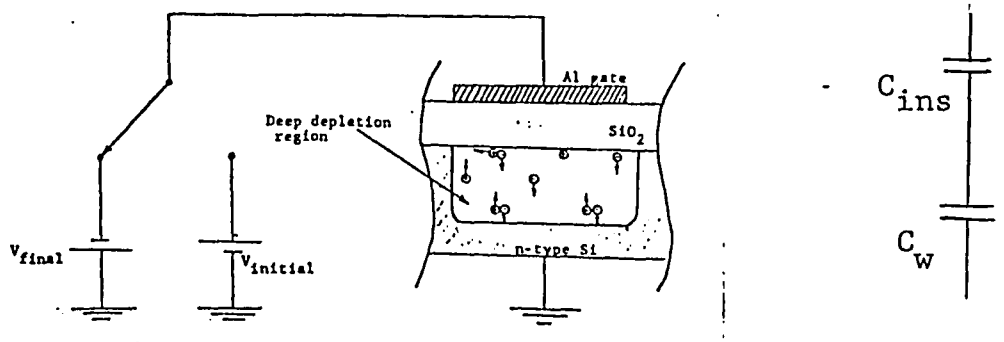
$$\frac{dQ_i}{dt} = qn_i\frac{(W-W_f)}{\tau_g'} + n_i S \qquad (1.8)$$

where $\tau_g' = \left(\frac{1}{\tau_g} + \frac{4S_o}{d}\right)^{-1}$

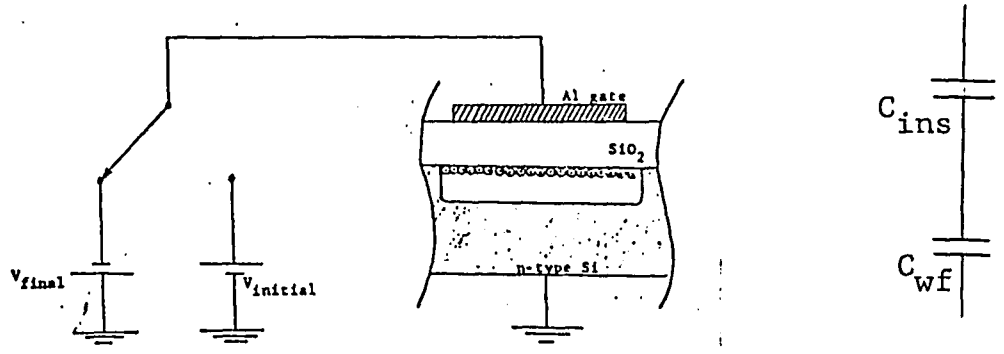
Thus according to Schroder and Nathanson the lifetime that is measured must be corrected because it includes a component of surface generation from a non-inverted surface. This additional correction is not considered in the analysis of the experimental section.



(a) Device set to initial equilibrium bias (accumulation on n-type Si shown). Equivalent circuit is shown.

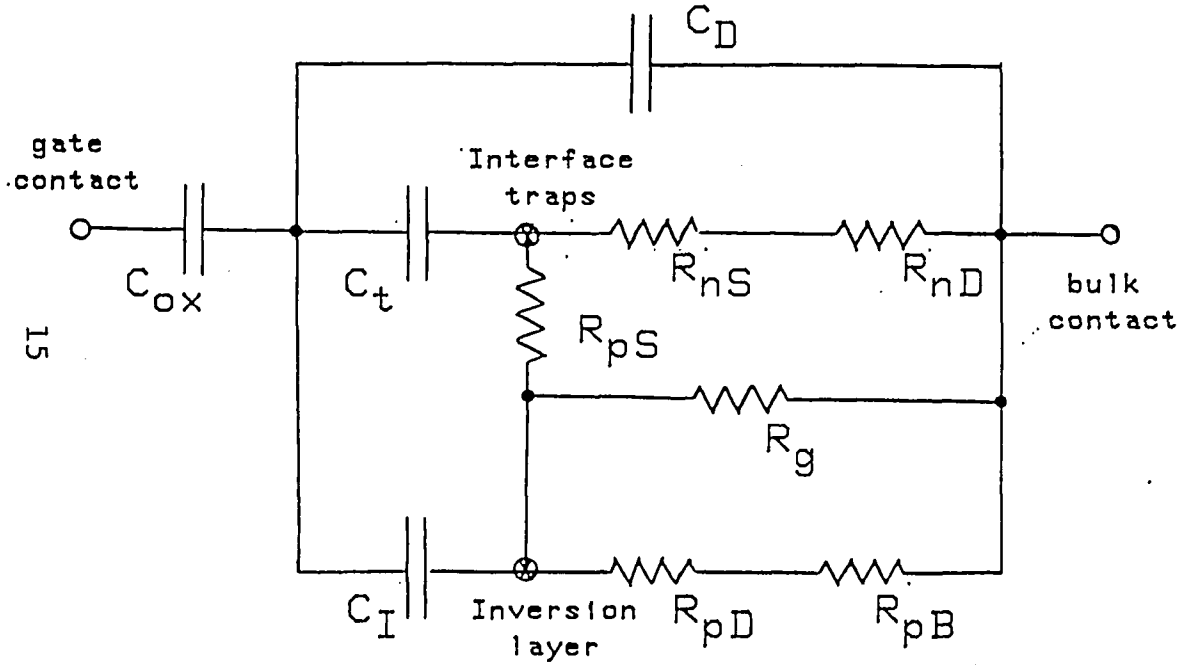


(b) Device immediately after voltage pulse. Non-equilibrium deep depletion region has formed. Pair generation occurring in depletion region and at surface to bring device back to equilibrium.



(c) Device after long time has returned to equilibrium. Inversion layer has formed at surface.

Fig. 1.1 - Schematic Representation of Pulsed C-t Experiment



C_{ox} – Oxide Capacitance

C_D – Depletion layer Capacitance

C_I – Inversion layer Capacitance

C_t – Trap Capacitance

R_{pD} – Hole dep. layer resistance

R_{pB} – Hole bulk resistance

R_{pS} – Hole trapping resistance

R_g – Pair Generation resistance

R_{nS} – Electron trapping resistance

R_{nD} – Electron dep. layer resistance

Figure 1.2 - Equivalent Circuit of Lehovec & Slobodsky

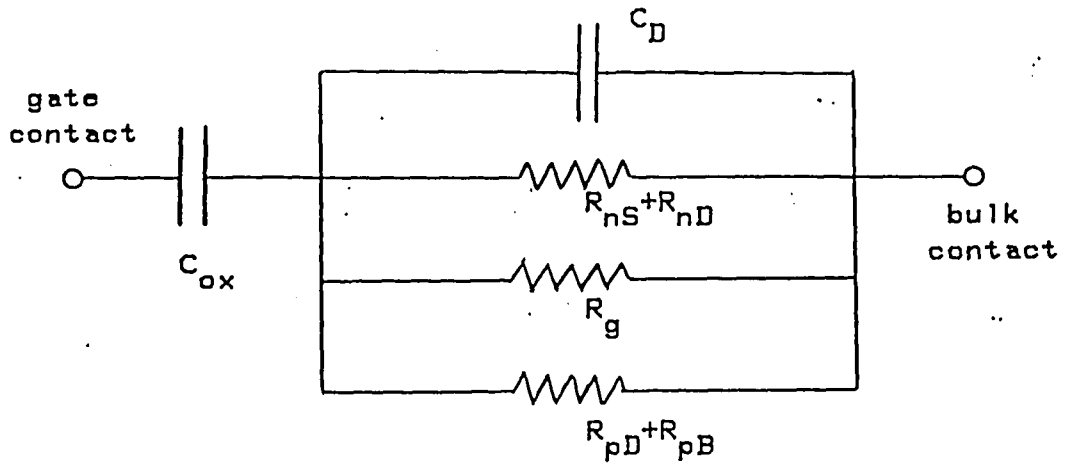


Figure 1.3 - Simplified Equivalent Circuit

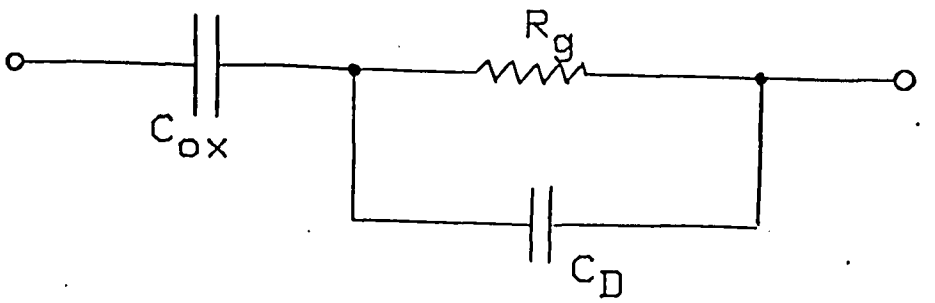


Figure 1.4 - Equivalent Circuit Considered by Hofstein

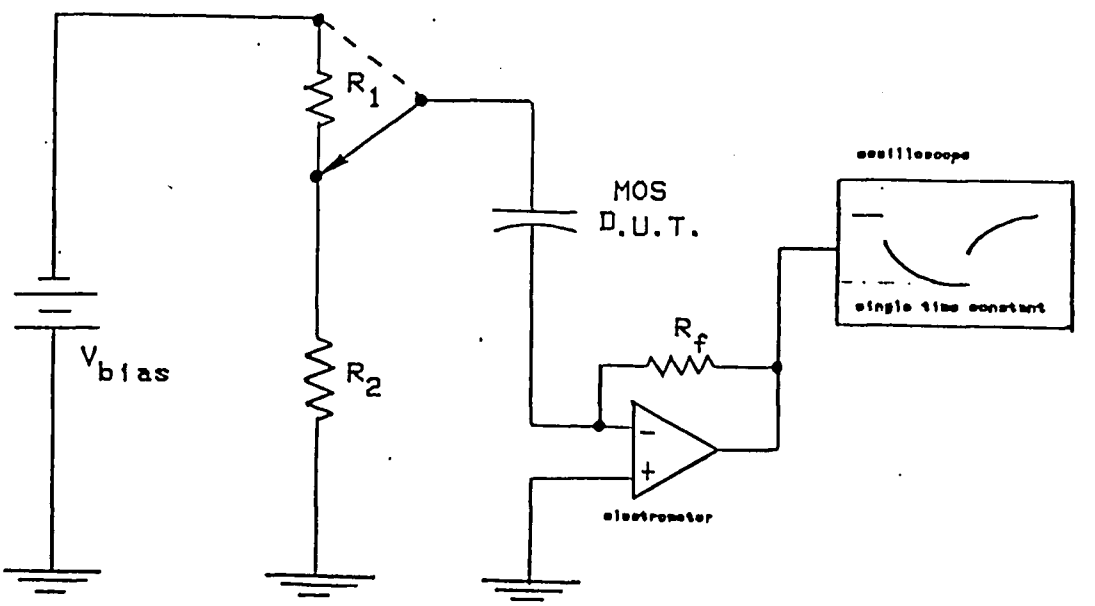


Figure 1.5 - Schematic Representation of Hofstein's Experiment

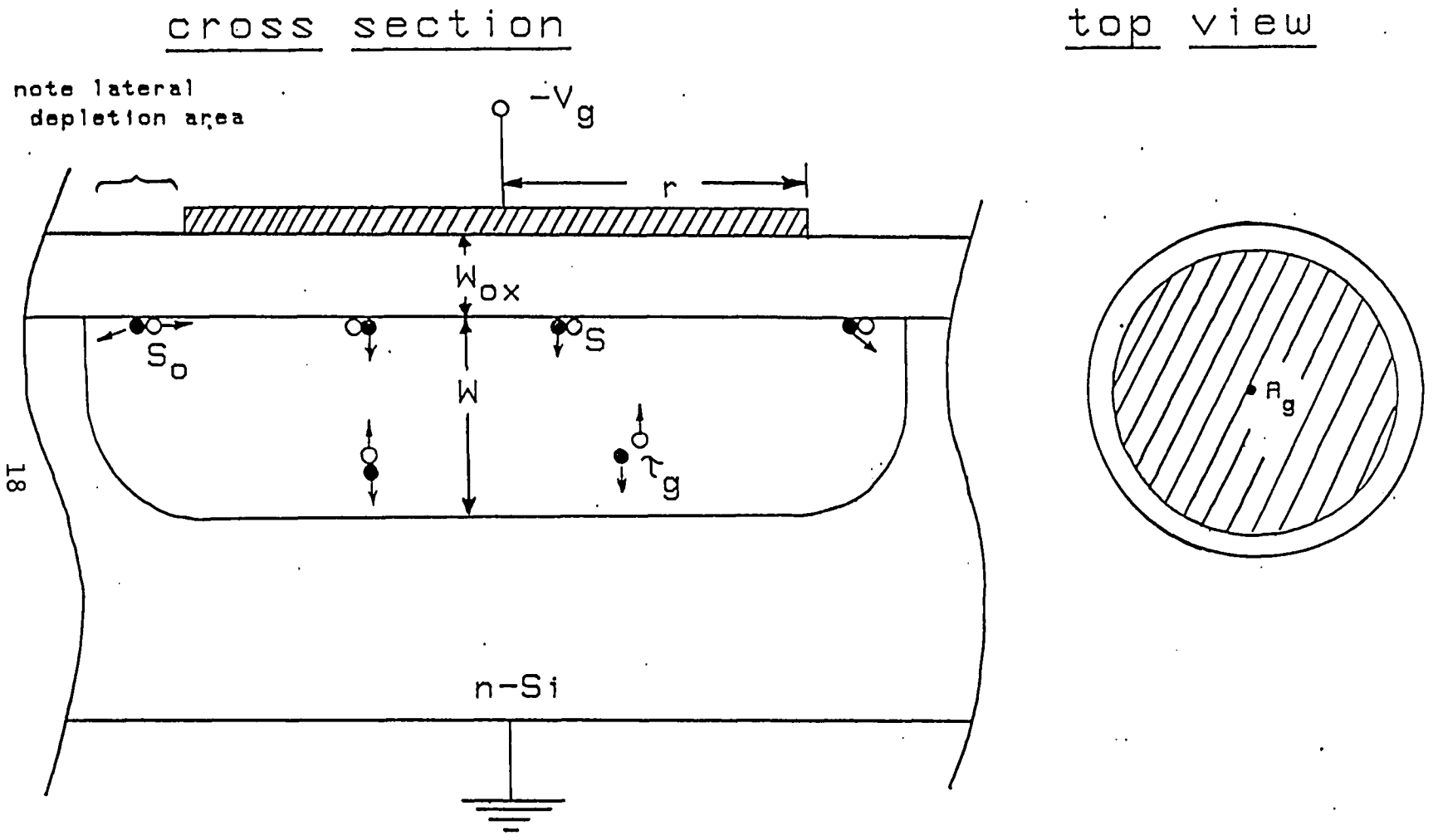


Figure 1.6 - Device Considerations of Nathanson & Schroder

PULSED C-t EXPERIMENTAL PROCEDURE

The pulsed C-t experiment is performed in the device characterization laboratory under the control of the HP9836 computer with loaded control program. The control program is documented in Appendix E; the experimental configuration is shown in Figure 1.7.

Briefly, at the beginning of the experiment, the device is set to a desired bias level. The control program monitors the device capacitance by taking measurements at the selected pacing rate of the A/D converter. When a software key is pressed a voltage pulse is applied to the device. The 9836 continues to monitor the capacitance while the device returns to equilibrium. The experiment completes when the computer has gathered 1000 data points at which time data analysis can occur.

The measurements presented in this section are taken on a simple MOS .75mm circular dot capacitor. A microphotograph of the device is shown in Figure 1.8. The device was fabricated with a standard MOS capacitor process: The silicon was precleaned followed by a gate oxidation at 1100°C for 40 min in a dry oxygen atmosphere, which grew approximately 1000A of oxide. Subsequently,

the oxide was annealed in-situ for 15 min in nitrogen. To complete processing an aluminum gate was deposited by filament and photographically etched. Back contact was formed with an aluminum deposition followed by a "sinter" operation at 450°C in forming gas. A table of device parameters is shown in Table 1.1.

Two pulsed measurements were performed on this device. In the first, the device was set initially into strong accumulation by the application of +5V equilibrium bias on the gate. A -15V pulse was applied causing a final bias of -10V, well into inversion. A capacitance transient was observed which lasted approximately 1 minute. The capacitance versus time curve is shown in Figure 1.9. Data analysis was performed with the Zerbst technique as documented in Appendix E. The results of curve fitting are shown in Table 1.2. It is important to note that the best fits are found corresponding to the data at the end of the transient. At this point in the transient the surface is heavily inverted and therefore the surface recombination velocity calculated corresponds to a heavily inverted surface. In a heavily inverted surface, with most of the surface states filled and therefore a lack of recombination centers, the surface recombination velocity should be low. The results are shown

in the Zerbst plot of Figure 1.10.

A second set of measurements were performed on the same device in which the initial equilibrium bias was set at -3V, already in inversion. A -7V pulse was applied which again caused a final bias of -10V. The resulting capacitance transient is shown in Figure 1.11. The calculated recombination lifetime and surface recombination velocity are very consistent between the two measurements. Note that in the Zerbst plot of the second measurement (Figure 1.12) the fitted straight line follows the data for a much greater portion of the transient. This is consistent since the device was initially set into inversion and therefore the surface is heavily inverted for a much larger portion of the transient. The Zerbst assumptions of constant surface recombination velocity and recombination lifetime are valid for a larger portion of the transient in the second measurement and therefore the second measurement could be considered the better of the two. The consistency between measurements can be accounted for since as noted the data was fit at the end of the transient of the first measurement where the surface is heavily inverted.

These results point to the validity of the Zerbst

technique as applied in this program.

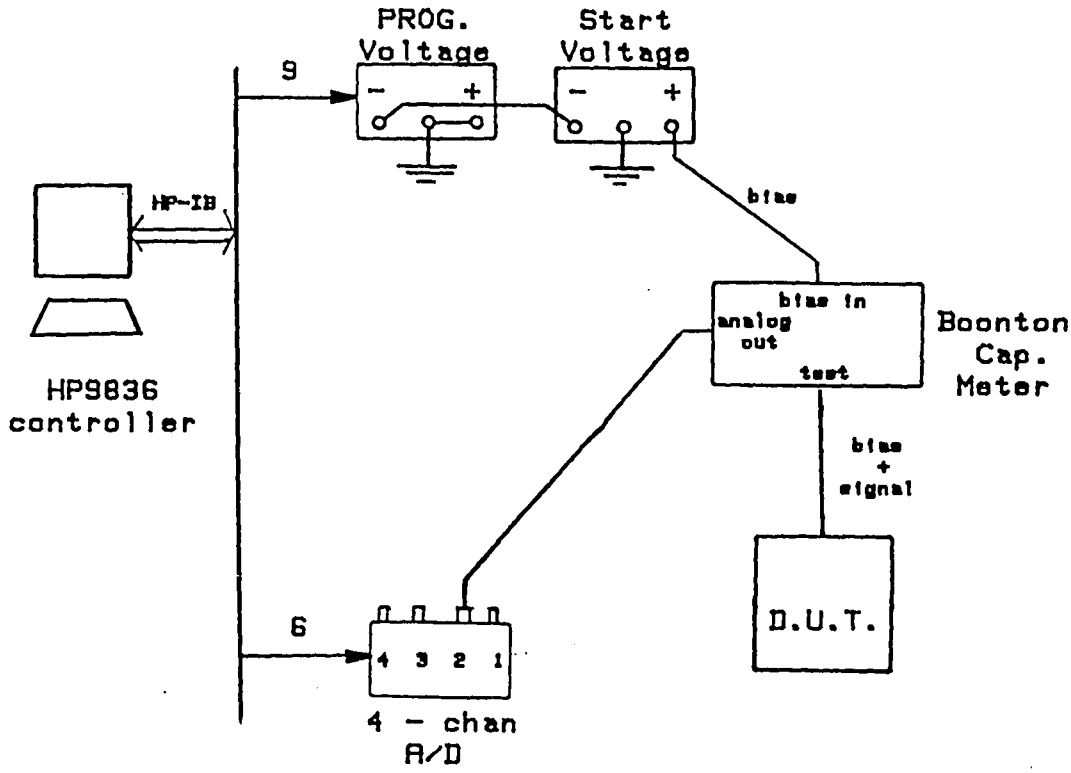


Figure 1.7 - Experimental Configuration of Pulsed C-t Experiment

Table 1.1 Device Parameters for Measured MOS Capacitor

Sample Name:	T2 N 100
Doping Type:	N
Doping Density:	$1x10^{15}/cm^3$
Gate Area:	$4.42x10^{-3}cm^2$
Oxide Thickness:	810 ± 10 Å
Oxide Capacitance:	185 ± 5 pF
Temperature:	295 deg C

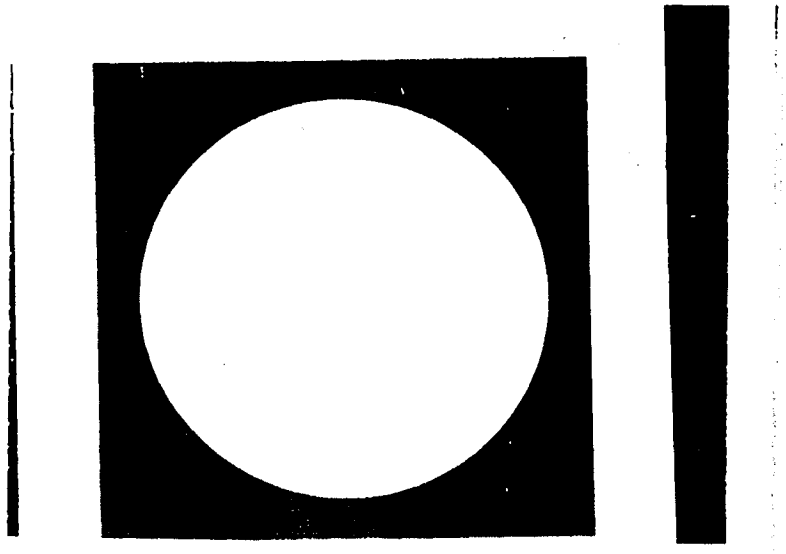


Figure 1.8 - Microphotograph of circular MOS capacitor.

Table 1.2 The results of the data analysis technique is shown for the device pulsed from accumulation to inversion. The results for the function coefficients from linear least square fit are shown at the top, a comparison of the actual and fitted data is also shown. Results of the regional linear least square fit is shown at the bottom.

FUNCTION COEFFICIANTS AND RESIDUAL VARIENCE FROM CURVE FIT

COEFFICIANTS:
A 1--1541.65690164
A 2- 6051.52507551
A 3--5976.48854744
A 4- 2521.98298402
A 5--924.209361768
A 6- 148.709140874
A 7--.12058494722
A 8- 57.9882341168
A 9--31.6000339468

RESIDUAL VARIENCE- 1.85079928367

COMPARISION OF SEVERAL DATA POINTS WITH FITTED CURVE

POINT NO.	DATA POINTS	FITTED POINTS
1	304.4194	304.4203
27	253.7788	247.6586
53	206.4575	207.8153
79	177.5193	175.7777
105	149.1655	149.3327
131	127.0996	127.2391
157	109.5910	108.6720
183	92.9702	93.0305
209	79.8626	79.8518
235	69.3435	68.7655
261	59.5009	59.4670
287	51.6144	51.7010
313	45.1979	45.2492
339	39.9075	39.9232
365	35.4945	35.5581
391	31.7749	32.0083
417	29.0331	29.1442
443	26.6314	26.8496
469	24.8506	25.0199
495	23.5514	23.5600
521	22.6428	22.3833

REPORT FROM LEAST SQUARE LINE FIT TO CURVE

TEST LINES:

SLOPE	INTERCEPT	RESIDUAL VAR
7.5258	-2.5055	.479477
6.7122	-1.5559	.175721
6.1693	-.9846	.080142
5.7602	-.6019	.041751
5.4532	-.3470	.025872
5.1965	-.1615	.017014
4.9834	-.0283	.012143
4.7971	.0693	.008171
4.6296	.1411	.004708
4.4806	.1928	.002131
4.3709	.2240	.000915
4.2968	.2404	.000446

FITTED SLOPE: 4.2968
FITTED INTER: .2404

Table 1.3 The Results of the Data Analysis Technique on Device Pulsed from Light to Heavy Inversion.

FUNCTION COEFFICIANTS AND RESIDUAL VARIENCE FROM CURVE FIT

COEFFICIANTS:
A 1- 209.432000148
A 2--25.4296337772
A 3- 105.900992832
A 4- 25.627512343
A 5--111.210591245
A 6- 31.2126212615
A 7--.0312139653478
A 8--.355070931589
A 9- 1.86298931607

RESIDUAL VARIENCE- 1.27421661449

COMPARISION OF SEVERAL DATA POINTS WITH FITTED CURVE

POINT NO.	DATA POINTS	FITTED POINTS
1	253.7788	253.7781
25	214.7983	213.5590
49	184.1550	180.1994
73	154.2648	153.6543
97	131.1025	131.9202
121	112.7906	113.7950
145	98.0635	98.5036
169	86.0432	85.5106
193	74.3247	74.4254
217	64.8467	64.9518
241	57.0726	56.8579
265	49.6483	49.9576
289	44.3798	44.0986
313	39.2283	39.1538
337	34.9243	35.0160
361	31.7749	31.5936
385	29.0331	28.8072
409	26.6314	26.5877
433	24.8506	24.8743
457	23.5514	23.6131
481	22.6428	22.7562

REPORT FROM LEAST SQUARE LINE FIT TO CURVE

TEST LINES:

SLOPE	INTERCEPT	RESIDUAL VAR
-2.3737	8.6368	560.650468
7.2276	-2.0890	.320473
6.5032	-1.2964	.183697
5.9181	-.7325	.095930
5.4688	-.3507	.051010
5.1260	-.0979	.025233
4.8679	.0661	.011139
4.6925	.1606	.004782
4.5704	.2149	.001813
4.5063	.2361	.000870
4.4778	.2394	.000584
4.5087	.2220	.000969
4.5892	.1927	.001741

FITTED SLOPE: 4.4778
FITTED INTER: .2394

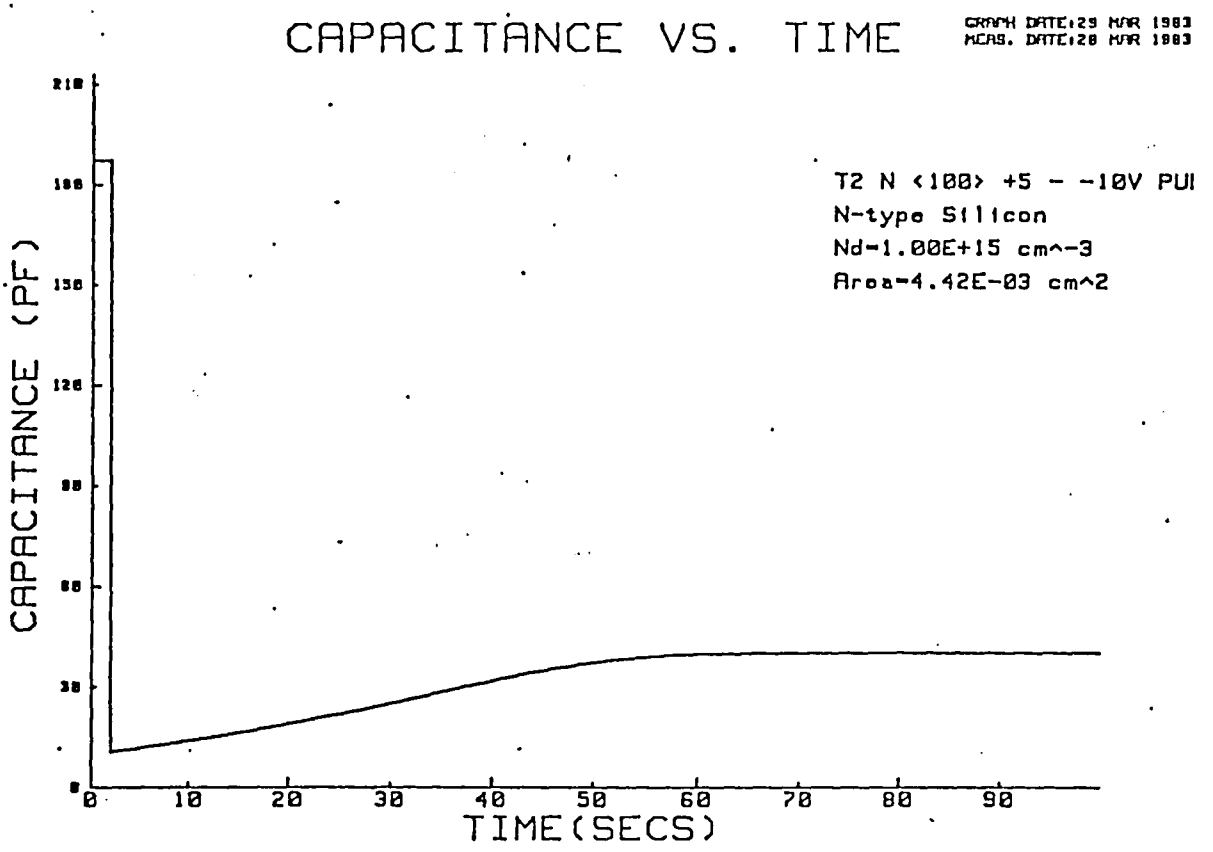


Figure 1.9 Capacitance vs. Time curve for MOS capacitor pulsed from accumulation into inversion. The initial capacitance is the accumulation capacitance which corresponds to the oxide capacitance. After pulse a non-equilibrium deep depletion capacitance is observed which eventually recovers to an equilibrium inversion capacitance.

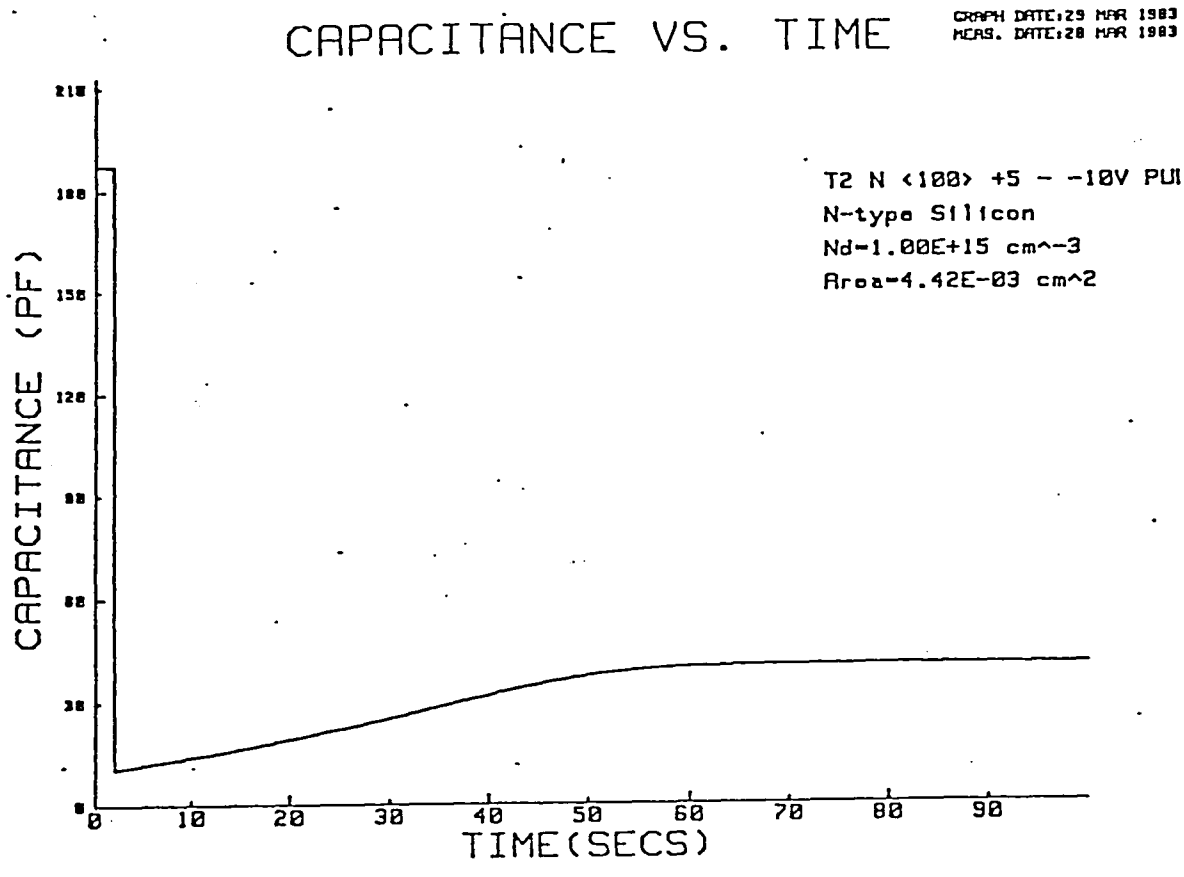


Figure 1.9 Capacitance vs. Time curve for MOS capacitor pulsed from accumulation into inversion. The initial capacitance is the accumulation capacitance which corresponds to the oxide capacitance. After pulse a non-equilibrium deep depletion capacitance is observed which eventually recovers to an equilibrium inversion capacitance.

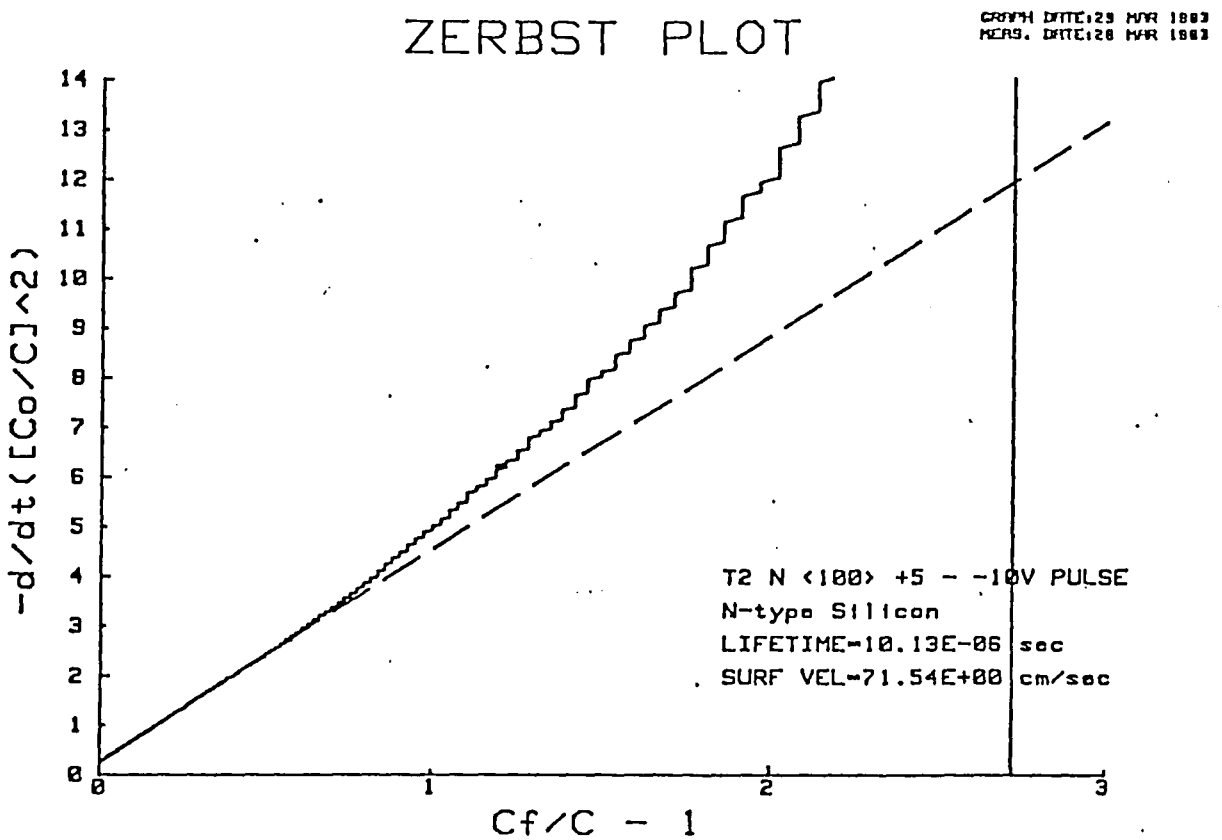


Figure 1.10 Zerbst plot of $d/dt(C_o/C)^2$ vs. $C_f/C - 1$ for capacitor pulsed from accumulation into inversion. Data points correspond to the solid line. Dotted line is best straight line fit to a portion of the graph. Note that fit occurs at low values of $C_f/C - 1$ which corresponds to the end of the transient. Values of lifetime and surface velocity shown are calculated from slope and y-intercept of fitted straight line.

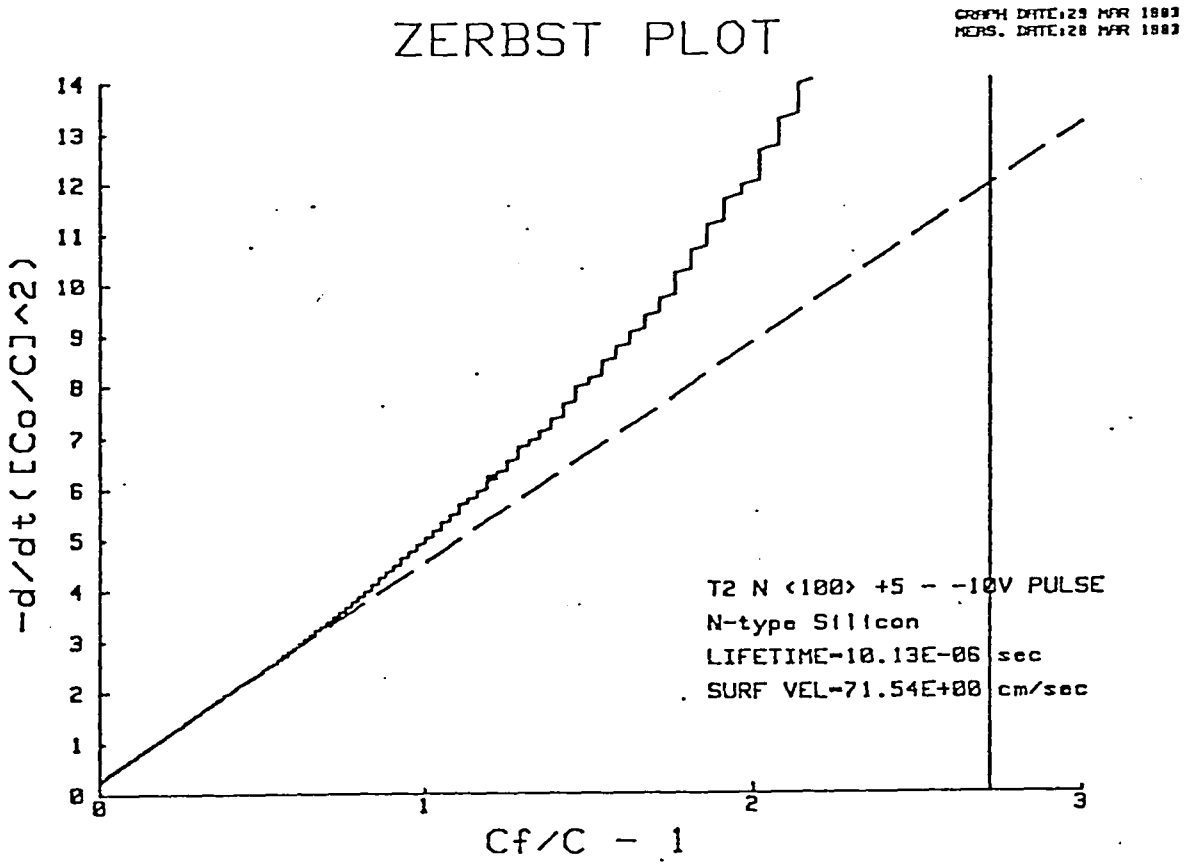


Figure 1.10 Zerbst plot of $d/dt(C_o/C)^2$ vs. $C_f/C - 1$ for capacitor pulsed from accumulation into inversion. Data points correspond to the solid line. Dotted line is best straight line fit to a portion of the graph. Note that fit occurs at low values of $C_f/C - 1$ which corresponds to the end of the transient. Values of lifetime and surface velocity shown are calculated from slope and y-intercept of fitted straight line.

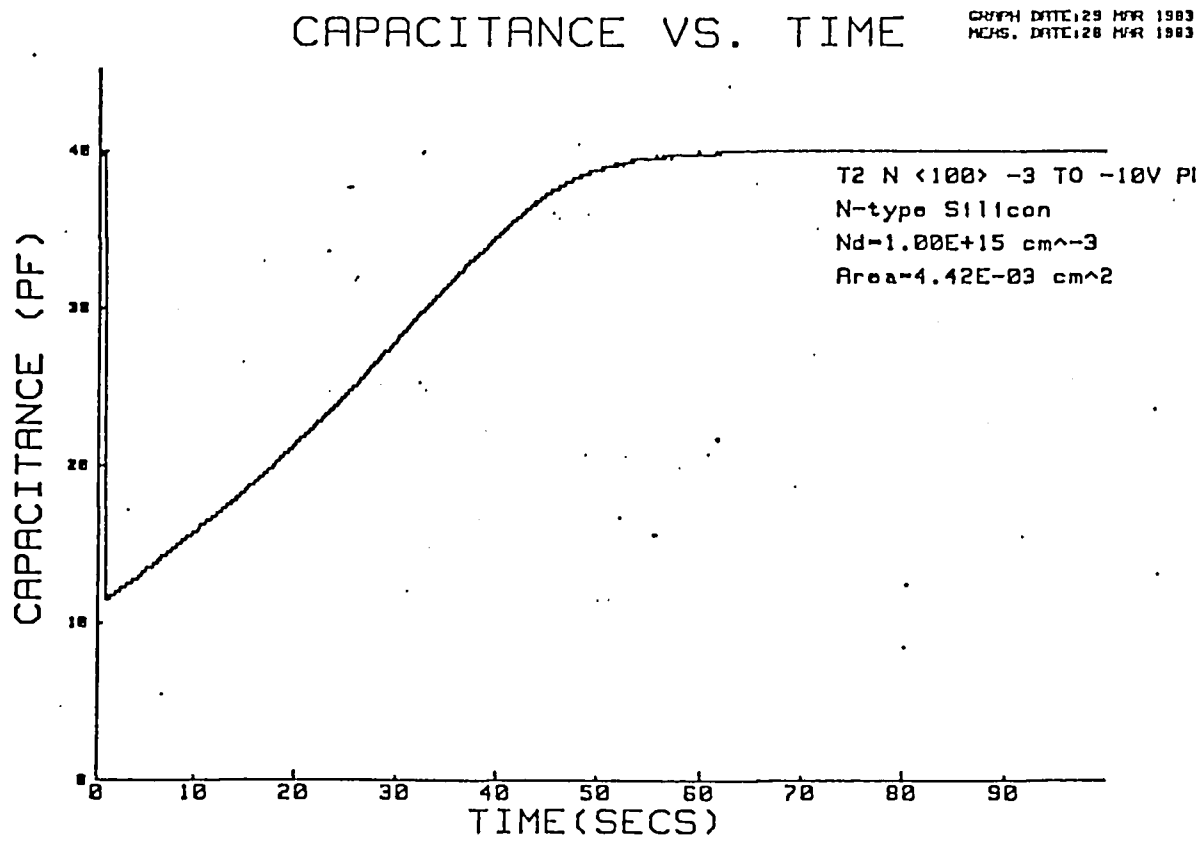


Figure 1.11 Capacitance vs. Time curve for MOS capacitor pulsed from light to heavy inversion. Both initial and final capacitance correspond to equilibrium inversion capacitance. Non-equilibrium deep depletion values of capacitance occur immediately after device is pulsed.

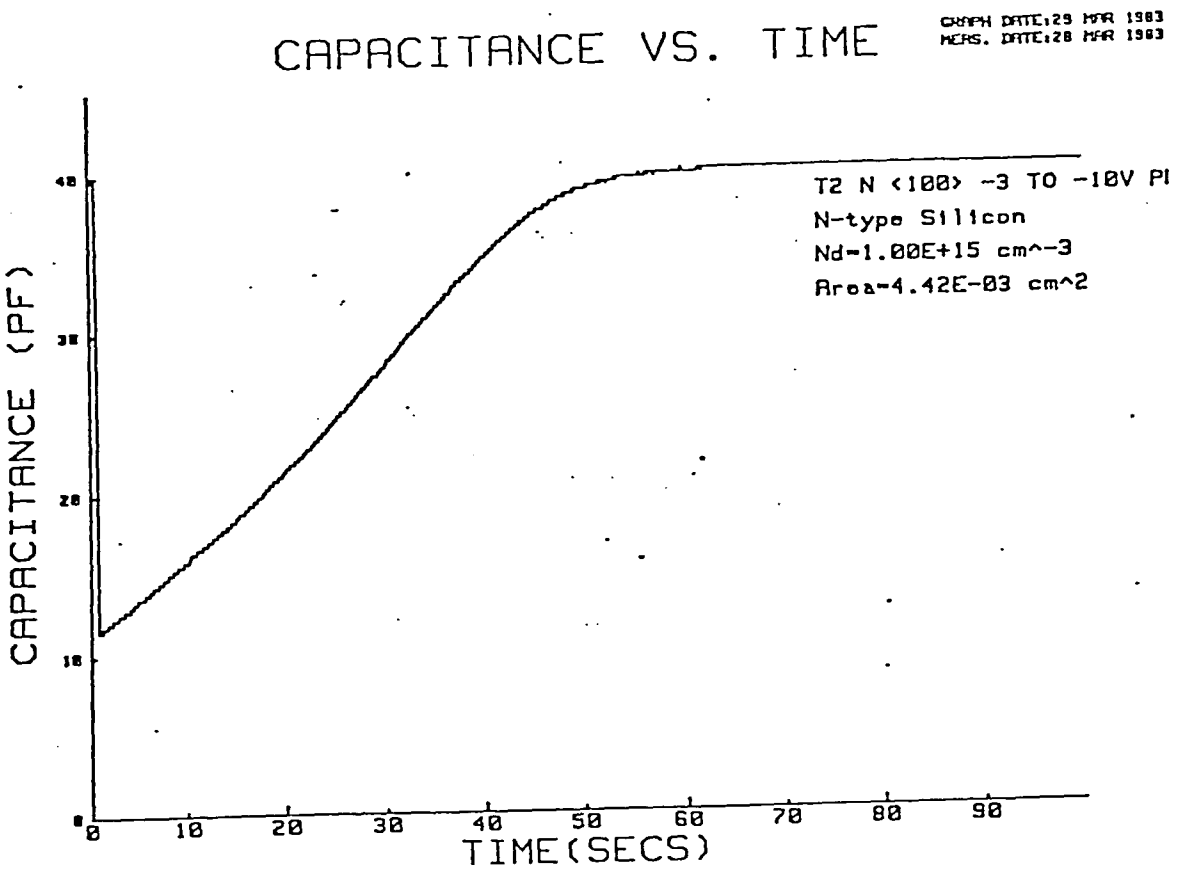


Figure 1.11 Capacitance vs. Time curve for MOS capacitor pulsed from light to heavy inversion. Both initial and final capacitance correspond to equilibrium inversion capacitance. Non-equilibrium deep depletion values of capacitance occur immediately after device is pulsed.

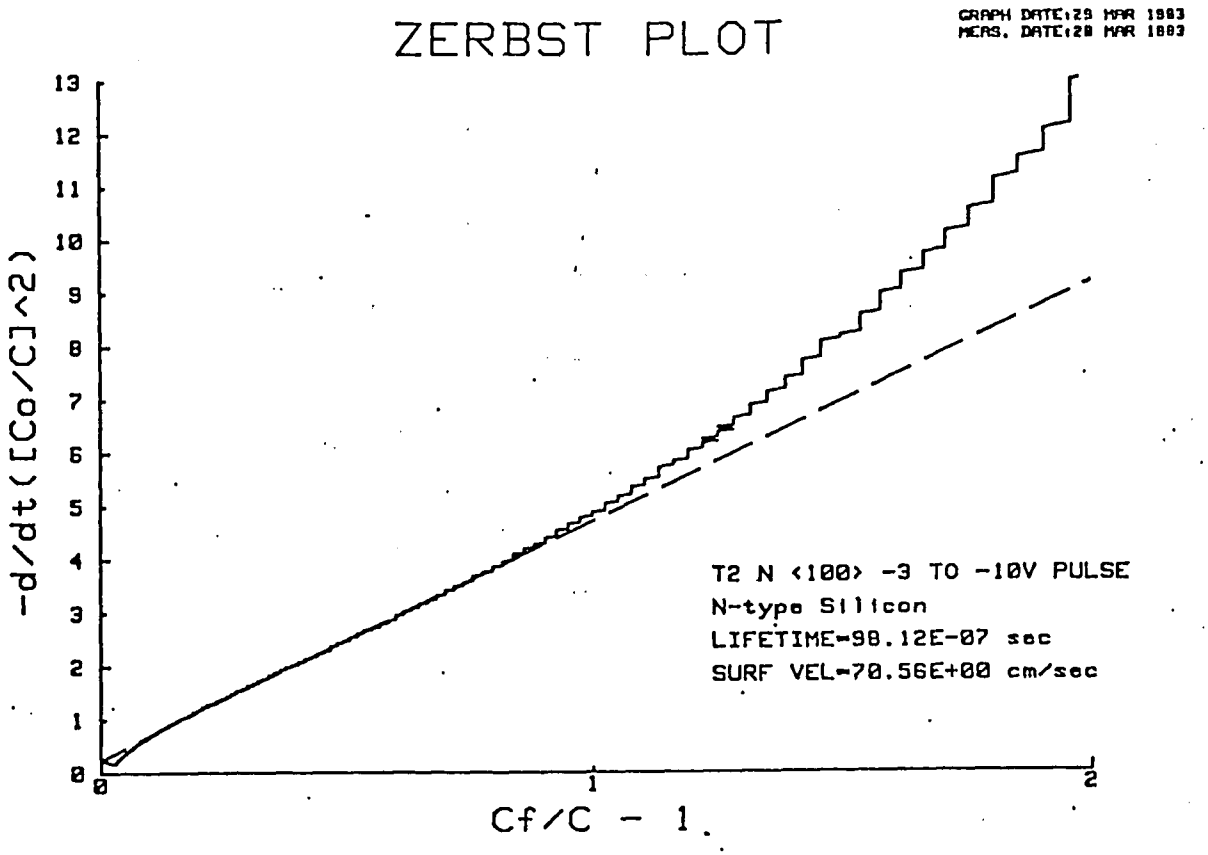


Figure 1.12 Zerbst plot for capacitor pulsed from light to heavy inversion. Data points are shown with solid line and straight line fit with dashed line. Note the large region over which the straight line fit is valid. This is caused by the constant surface condition of heavy inversion.

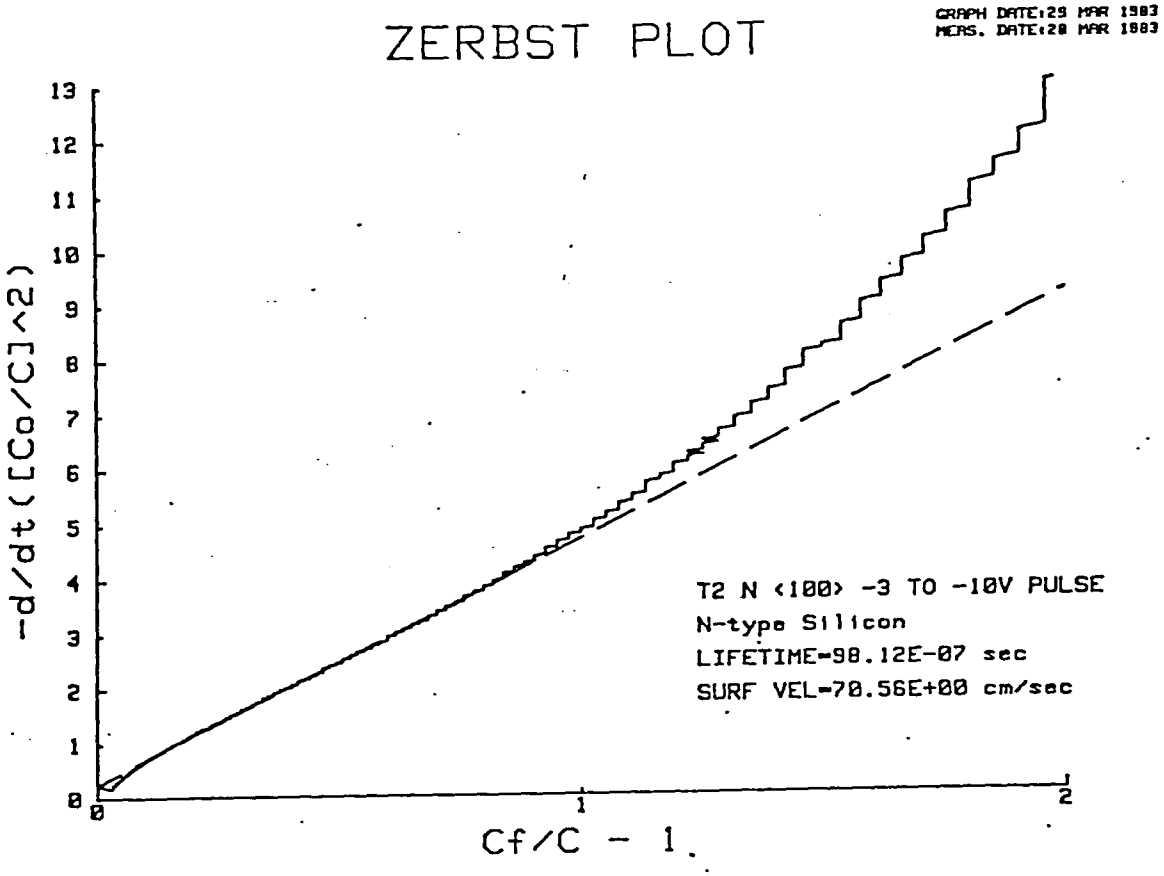


Figure 1.12 Zerbst plot for capacitor pulsed from light to heavy inversion. Data points are shown with solid line and straight line fit with dashed line. Note the large region over which the straight line fit is valid. This is caused by the constant surface dondition of heavy inversion.

DISCUSSION OF PULSED C-t AUTOMATION

The software program used in the control and analysis of this experiment needed to implement a variety of control and data gathering functions as well as data processing techniques. As can be seen by the program listing of Appendix E, the resulting code is well over 1000 lines. Though the documentation of the program should make the code clear there are several factors of the program that deserve discussion.

Though the code is long, it successfully simplifies the task of performing the pulsed C-t experiment. One very general reason that this software package could be considered successful is due to the use of a modular design scheme. The term modular design means that the program is not defined procedurally by instead certain subtasks are programmed which are then made available to the user. In this way the user can define the experimental procedure without reprogramming of the software. For example, the program might have been written such that the user initially provided input parameters followed by data acquisition, and finally analysis. Instead, there are three sparate subroutines written to perform each of these tasks, SETUP to get input parameters and

re-initialize the experiment, MEASURE to gather data points, and subroutine ZERBST to perform the Zerbst type analysis on the gathered data. Each of these subroutines (and several others) are then attached to a software key under user control. In this way the user can perform the SETUP and MEASURE subroutines and at this point, if there are initial indications that the data is not good, then the MEASURE routine can be rerun. This type of flexibility seems essential in an experimental environment where procedural need and results can not always be anticipated. This programming philosophy is also apparent in the admittance software documented in this thesis and is suggested for any future program development in the laboratory.

In contrast to the generalized program structure, a software problem specific to the C-t experiment was the monitoring of an experimental parameter with respect to time. The obvious solution to this problem (and the method first attempted) was to obtain the monitored parameter from the data bus line and then obtain the time from the processor's internal clock. The problem in this scheme is that there is a finite amount of time which elapses between the time when the data is gathered and the time when the system clock is checked. If this time

lapse is long compared to the time resolution required, then the time marked for the data will be useless for analysis purposes. An analogy to this scheme of processing would be a single experimenter taking data by hand, say recording a voltage level as recorded on a DVM with respect to time. The experimenter would look at the readout of the DVM and mark its value and then look at the clock and mark the time. If the time necessary to perform this task is longer than any meaningful transients in the voltage level then those transients would be lost.

The problem is solved in this case by the use of an off-line dedicated processor in the form of an A/D converter. With this instrument the control unit (the 9836) presets the A/D by giving it a measurement pacing rate. The A/D is then triggered by some start instruction and then at the given pacing rate the data is presented to the 9836 controller. The analogy here would be an experimenter with an instrument that would only measure at say 5 second intervals. At these times the instrument provides data readouts which the experimenter records. He would then know that the first point occured at time zero, the second at time 5 seconds, etc.. Because there is less work for the experimenter to do he can record much quicker

transients. This holds true for the computer as well and with this scheme the computer was able to mark the capacitance transient with down to a 5 millisecond resolution.

There are limits to this scheme as well however in that the central controller must be able to accept the data as quickly as it is gathered and because of the generality of the CPU it is not always extremely fast. To record even faster transients the use of dedicated processors with buffered memory is needed. In this way an offline instrument could be instructed to gather data at a given pacing rate and started. At this time the instrument could measure the data and store it in internal memory without trying to communicate with the central controller. At the end of some meaningful observation time the controller would stop the instrument and pull the data from the memory as convient. The analogy in this case would be an experimentor starting some similar instrument and going for a cup of coffee. At the end of the observation time he could return and stop the instrument and at his leisure pull the data values from the internal memory. This scheme is especially attractive if there is more than one parameter to be monitored. With the use of the offline processor doing the work the number of parameters that can be monitored is limited only by the number of instruments available

instead of the cycling time of the CPU.

Along with the control problem of monitoring time, this experiment raised an interesting data processing problem, the problem of taking the derivative of a set of gathered data points with respect to some other parameter. In this case, the Zerbst analysis requires the time derivative of the created array of $(C_{ox}/C)^2$. This was accomplished in the documented program by performing a least squares fit of the created array to nine analytical functions. Using the extracted function coefficiants then, the derivative was taken analytically. This may not, unfortunately, be the best method of attacking this problem. Because of the number of data points and the good resolution of the measurement system it might be better to take the derivative numerically through a differencing technique. If the derivative is found to be not well smoothed due to the digitizing of the data it could at this point be fit in a least square sense. In this way, the unique features of the data would be retained through the derivation process.

The use of this software package is, in general, quite easy and any user can now quickly get excellant estimates of the generation lifetime and surface recombination

velocity. For example, the measurements presented here were all performed in one afternoon. The use of the program however does not necessarily give a better understanding of the physics behind carrier generation and unique devices may not be well characterized by this program due to the many assumptions that go into the Zerbst analysis. For the person who wants a quick measurement of lifetime in a simple device the program is an excellant working tool.

PART 2

ADMITTANCE MEASUREMENTS

THEORY OF ADMITTANCE MEASUREMENTS

Historically, investigators have demonstrated the usefulness of the capacitance technique in probing the properties of the MOS system. These techniques have some serious limitations as discussed in the paper of Zaininger and Warfield /8/. A more general technique, and the technique of present concern, is to measure the overall admittance of the MOS system. This technique provides both the conductance and the capacitance of the system, and as is shown, the conductance part of the measurement is a much more sensitive indicator of the trapping properties in the MOS system.

Briefly, in this technique a bias is applied between the bulk and the gate of an MOS capacitor which determines the band bending at the Si-SiO_2 interface. This sets the position of the equilibrium Fermi level at some point in the Silicon bandgap. If an AC measurement signal is superimposed on top of this bias, then we can disturb the equilibrium Fermi level slightly. This will cause movement of the charge carriers between the traps and the bands. This population and depopulation of the traps will be detected by the measured admittance. The nature of the interface can be characterized by measuring the admittance

through a complete range of bias and measurement frequency.

To interpret these measurements the admittance of the MOS capacitor is derived assuming Shockley, Read, Hall type traps exist at the interface. This admittance is related to an equivalent circuit of the type proposed by Lehovec /7/. Qualitative observations are made and the theory extended to fit the measurements. In the present work, only majority carrier contributions are analyzed. Where appropriate an indication of how to include minority carrier effects is shown. A complete treatment of this theory is given by Nicollian and Brews in their book, MOS Physics and Technology /10/.

A schematic of the MOS system is shown in Figure 2.1. The physical device is represented in Figure 2.1a and the energy band diagram is shown in Figure 2.1b. The present analysis assumes a good ohmic contact to a perfectly conducting metallic gate which is completely DC isolated from the silicon by the oxide layer. A finite trap distribution will be assumed at the $Si-SiO_2$ interface. These traps will communicate with both the conduction and valence bands by capture and emission processes. Through donor (n) type doping the electrons in the conduction band far outnumber the holes in the valence band, therefore

electrical connection through detailed balance is made to the conduction band only.

Consider a single level trap located somewhere within the silicon bandgap at the Si-SiO_2 interface. As shown in Figure 2.2 there are four separate processes by which this trap communicates with the conduction and valence band. For a n-type device, in either accumulation or depletion (i.e. n>>p), only the electron capture and emission processes will be of concern. Considering the effect of these processes on the current flowing in the conduction band caused by a small AC signal, it is shown in Appendix C that the small signal admittance of this process in the silicon is given by:

$$Y_s = j\omega \frac{q^2 N_{it} f_o (1-f_o)}{kT\{1 + j\omega f_o/(c_n n_o)\}} \qquad (2.1)$$

where f_o is the equilibrium Fermi function given by:

$$f_o = \frac{1}{1 + \exp \frac{E-E_F}{kT}} \qquad (2.2)$$

Looking again at the MOS device in Figure 2.1 a simple equivalent circuit for this system can be proposed. The SiO_2 insulator can be replaced by a simple capacitor, C_{ox}.

Assuming the device is biased in depletion there will be some depletion capacitance between the back of the device and the Si-SiO_2 interface. With the assumed trap distribution at the interface there will be additional charge storage at the interface which can be represented by some trap capacitance, C_t, and fed by a resistive mechanism given by G_n. Considering the gate, the Si-SiO_2 interface and the silicon back contact to be equipotential surfaces the equivalent circuit of the MOS system is given by Figure 2.3. Analyzing this circuit, the silicon admittance (i.e. ignoring the oxide capacitance) is given by:

$$Y_s' = j\omega C_d + \frac{j\omega C_t G_n}{G_n + j\omega C_t} \qquad (2.3)$$

Comparing the second term of this equation to the admittance derived in equation (2.1), G_n and C_t can be defined as follows:

$$C_t = \frac{q^2}{kT} N_{it} f_o(1-f_o) \qquad (2.4a)$$

$$G_n = \frac{q^2}{kT} N_{it} c_n n_{so}(1-f_o) \qquad (2.4b)$$

An extension of this equivalent circuit motivation to include minority carrier effects is shown in Figure 2.4.

For explanation purposes consider the circuit of Figure 2.5 for the device admittance. This circuit represents the silicon system as some conductance, G_p and some capacitance, C_p in parallel. Comparing this circuit to the admittance given by equation (2.3), G_p and C_p are defined as:

$$\frac{G_p}{\omega} = \frac{C_t \omega\tau}{1 + (\omega\tau)^2} \qquad \text{where } \tau = \frac{C_t}{G_n} \qquad (2.5a)$$

$$C_p = \frac{C_t}{1 + (\omega\tau)^2} + C_d \qquad (2.5b)$$

Plots of the dimensionless quantities $(G_p/\omega C_t)$ and $(C_p-C_d)/C_t$ vs. frequency are shown in Figure 2.6.

Note the behavior of the $(C_p-C_d)/C_t$ curve. At low frequencies the curve approximately equals 1, i.e. $C_p=C_d+C_t$. This means that the trap is responding immediately to the AC signal. As the frequency increases the traps no longer respond and the trap capacitance does not enter into the circuit, note: $(C_p-C_d)/C_t \rightarrow 0$ or $C_p \rightarrow C_d$.

Note also the behavior of the $G_p/(\omega C_t)$ curve. At low frequencies the trap responds in phase with the signal, therefore there is no enrgy loss and the equations show $G_p/(\omega C_t) \rightarrow 0$. As frequency increases the curve peaks at the point where $\omega\tau=1$. As is increased further the traps no longer respond and the curve again goes toward zero.

Though the theoretical curves developed thus far show the same qualitative response as measured curves, the theory is not yet general enough to produce exact fits. So far a simple theory has been developed which indicates how a single energy level interface trap responds to an AC measurement signal. This theory must be extended two ways before the desired fits can be achieved. First, it is a gross simplification to assume a single trapping level at the interface. There is in fact a continuous energy distribution of interface traps. For each trap level there would be a parallel addition to the equivalent circuit consisting of a trap conductance and an electron capacitance as shown in Figure 2.7. To express this mathematically a continuous trap distribution D_{it} is assumed and the admittance as given by equation (2.1) is integrated over the bandgap. This is shown in Appendix D and the result is given by:

$$\frac{G_p}{\omega} = \frac{qN_{it}}{2\omega\tau_n} \ln(1 + \omega^2\tau_n{}^2) \qquad (2.6a)$$

where $\tau_n = \frac{1}{c_n n_{so}}$

$$C_p = \frac{qN_{it}}{\omega\ \tau_n} \tan^{-1}(\omega\tau_n) \qquad (2.6b)$$

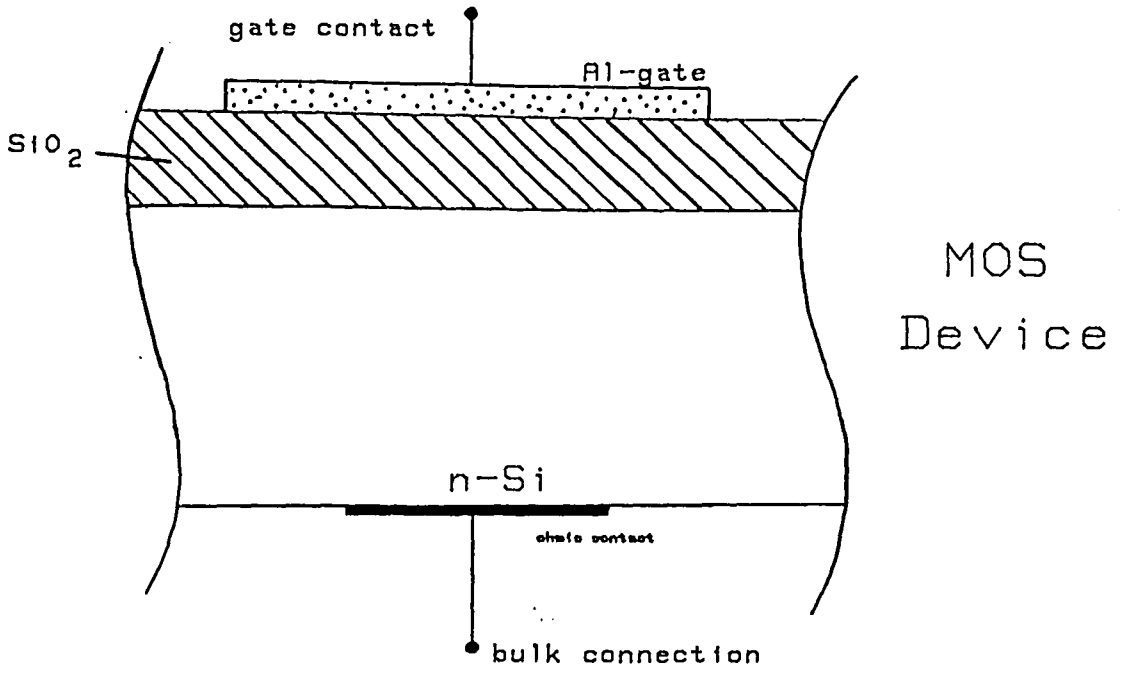
The second theoretical extension is due to the discrete nature of the fixed oxide charge and other processing effects causing chemical inhomogeneities. This causes a lateral non-uniformity in the location of the Fermi level within the bandgap. To mathematically account for this effect, a Gaussion approximation to a Poisson distributed surface potential will be assumed and the conductance and capacitance as given above in equation (2.6) will be integrated over this distribution. The resulting admittance as shown in Appendix E is given by:

$$\frac{G_p}{\omega} = \frac{qD_{it}}{2\omega\tau_n(2\pi\sigma_s)^{\frac{1}{2}}} \int_{\infty}^{\infty} \exp\left[-\frac{\eta^2}{2\sigma_s{}^2}\right] \exp(-\eta)\ \ln(1+\omega^2\tau_n{}^2\exp 2\eta)\, d\eta \qquad (2.7a)$$

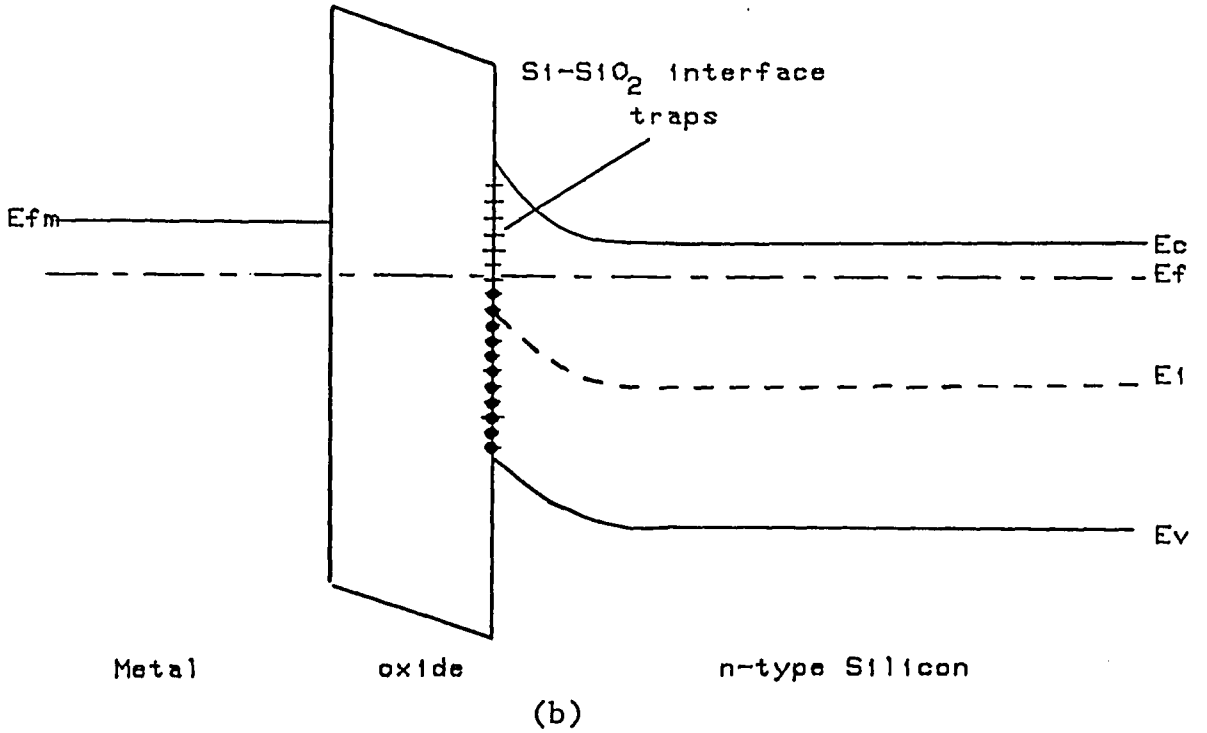
$$C_p = C_d + \frac{qD_{it}}{\omega\tau_n(2\pi\sigma_s)^{\frac{1}{2}}} \int_{\infty}^{\infty} \exp\left[-\frac{\eta^2}{2\sigma_s{}^2}\right]\ \exp(-\eta)\tan^{-1}(\omega\tau_n\exp\eta)\, d\eta \qquad (2.7b)$$

where $\eta = u_s - \overline{u_s}$

The resulting equations given equation (2.7) are shown to fit experimental data. A comparison of the conductance curves produced by each of the theories is shown in Figure 2.8.



(a)



(b)

Figure 2.1 - (a) Schematic of MOS System; (b) Band Diagram of MOS System

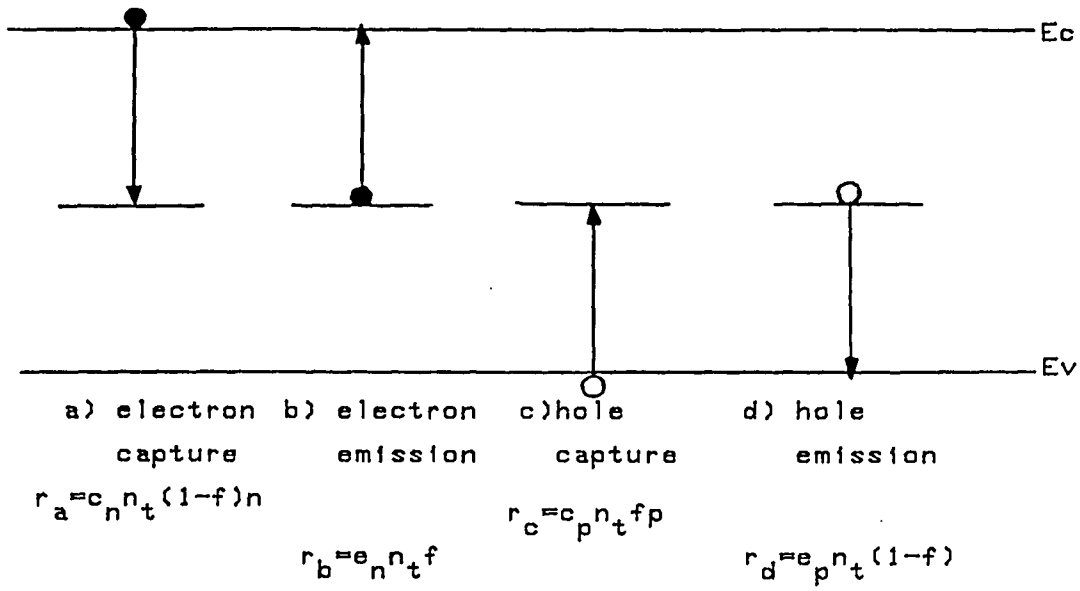


Figure 2.2 - Schematic of Trapping and Emission Processes

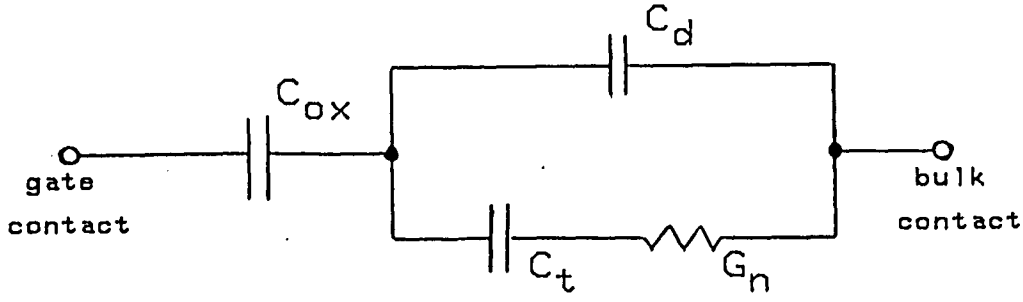


Figure 2.3 - Equivalent Circuit of MOS Device

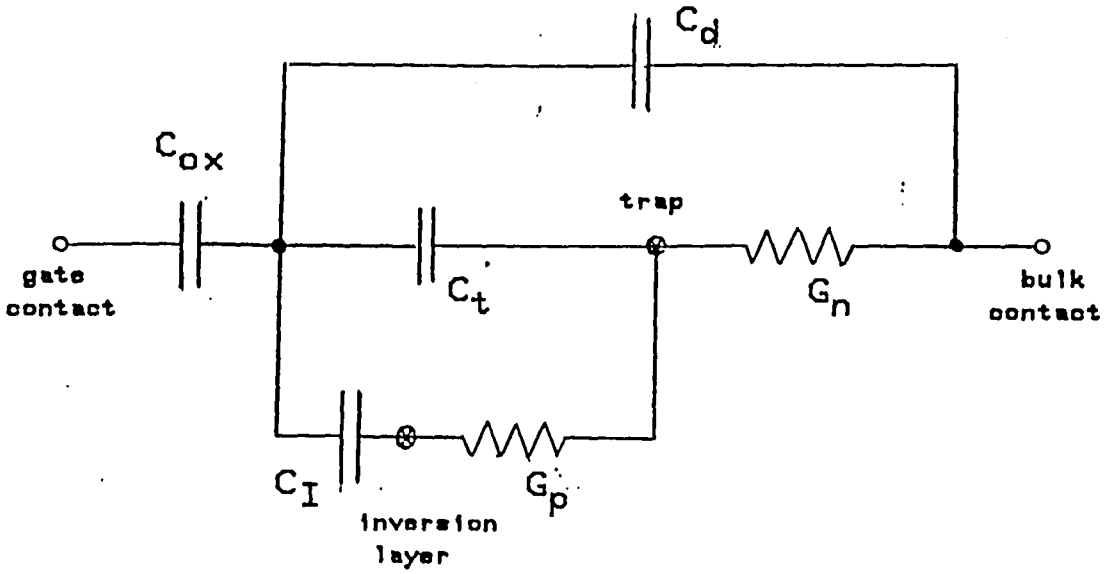


Figure 2.4 - Extended Equivalent Circuit to include Hole Effects

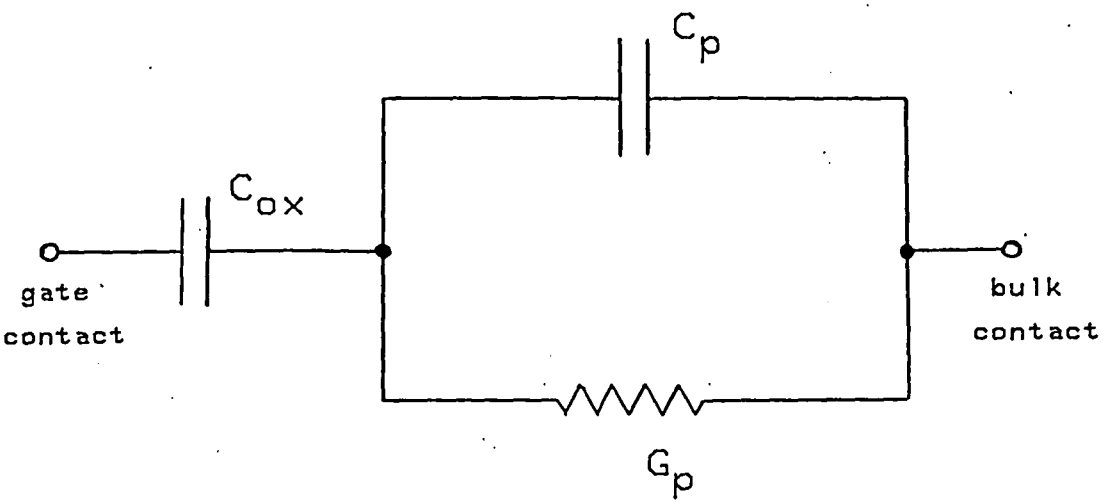


Figure 2.5 - Circuit which considers Silicon as equivalent parallel Conductance and Capacitance

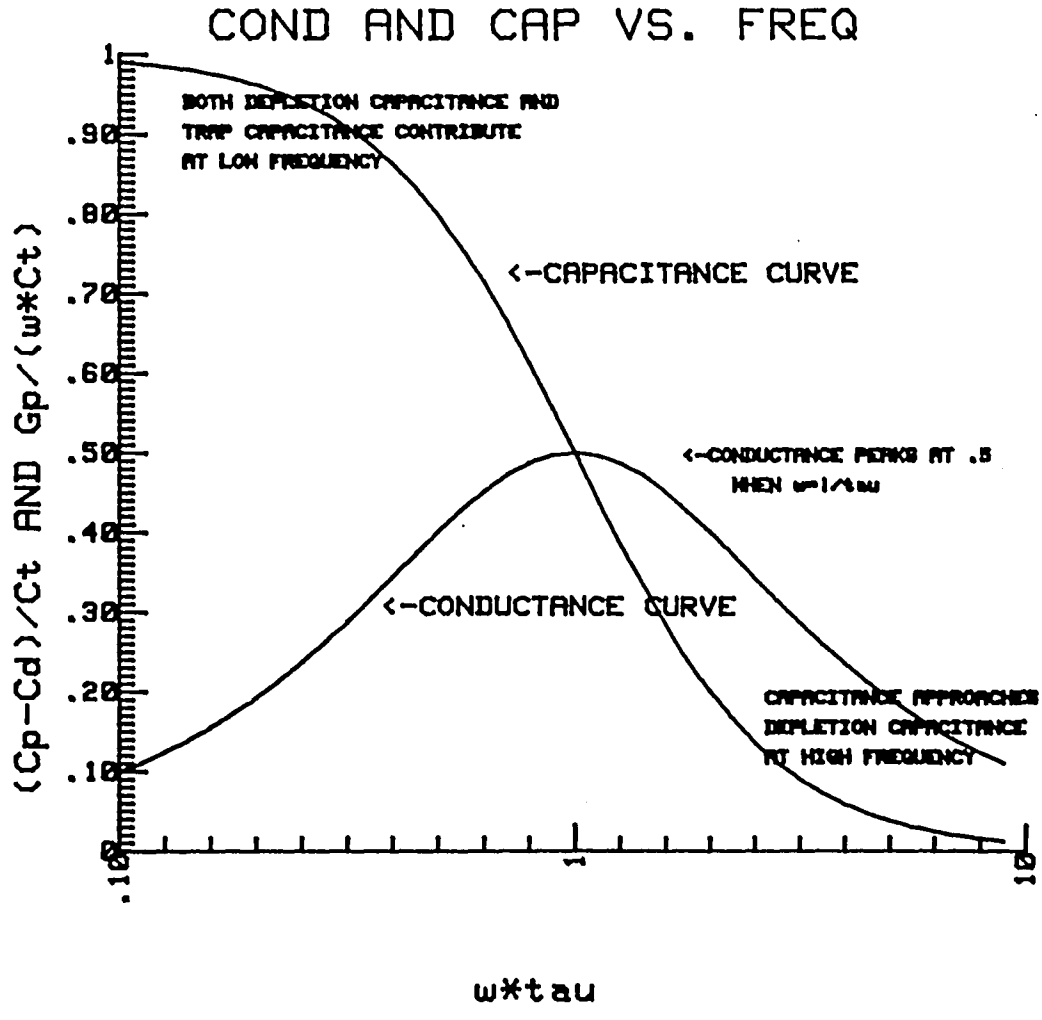


Figure 2.6 - Plots of dimensionless quanities $(C_p-C_t)/C_t$ and $G_p/\omega C_t$ versus frequency

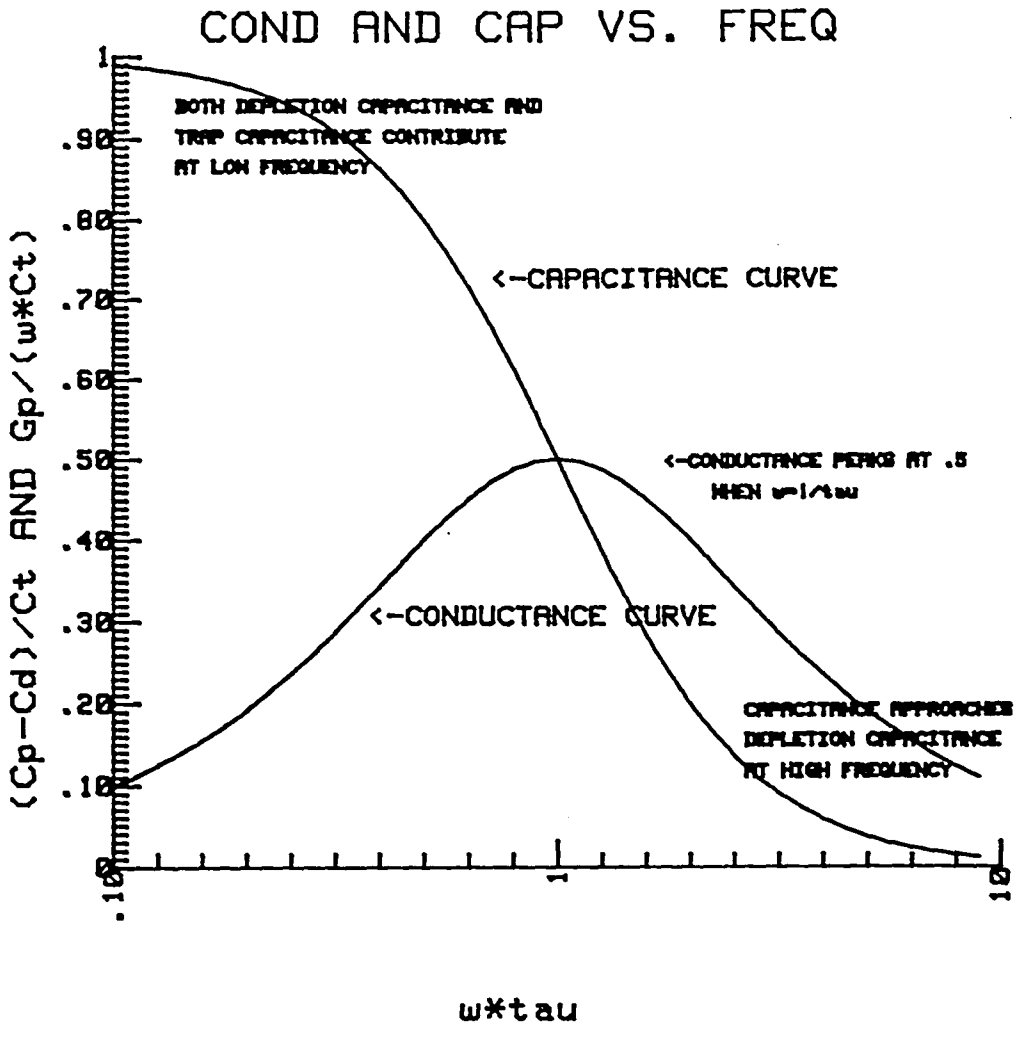


Figure 2.6 - Plots of dimensionless quanities $(C_p - C_t)/C_t$ and $G_p/\omega C_t$ versus frequency

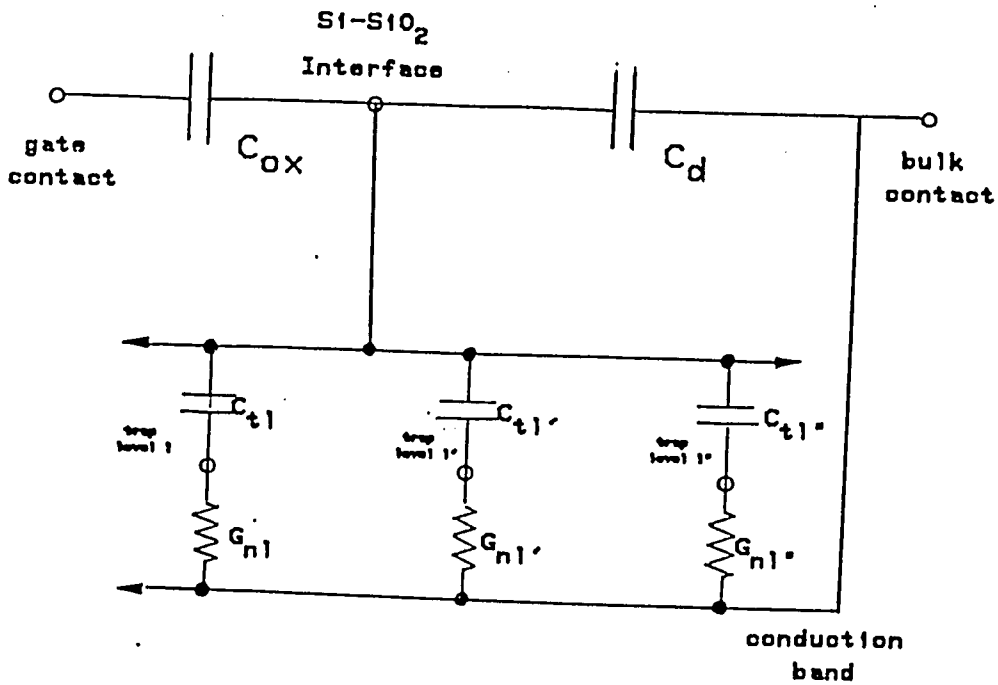


Figure 2.7 - Equivalent Circuit for a Continuum of Traps

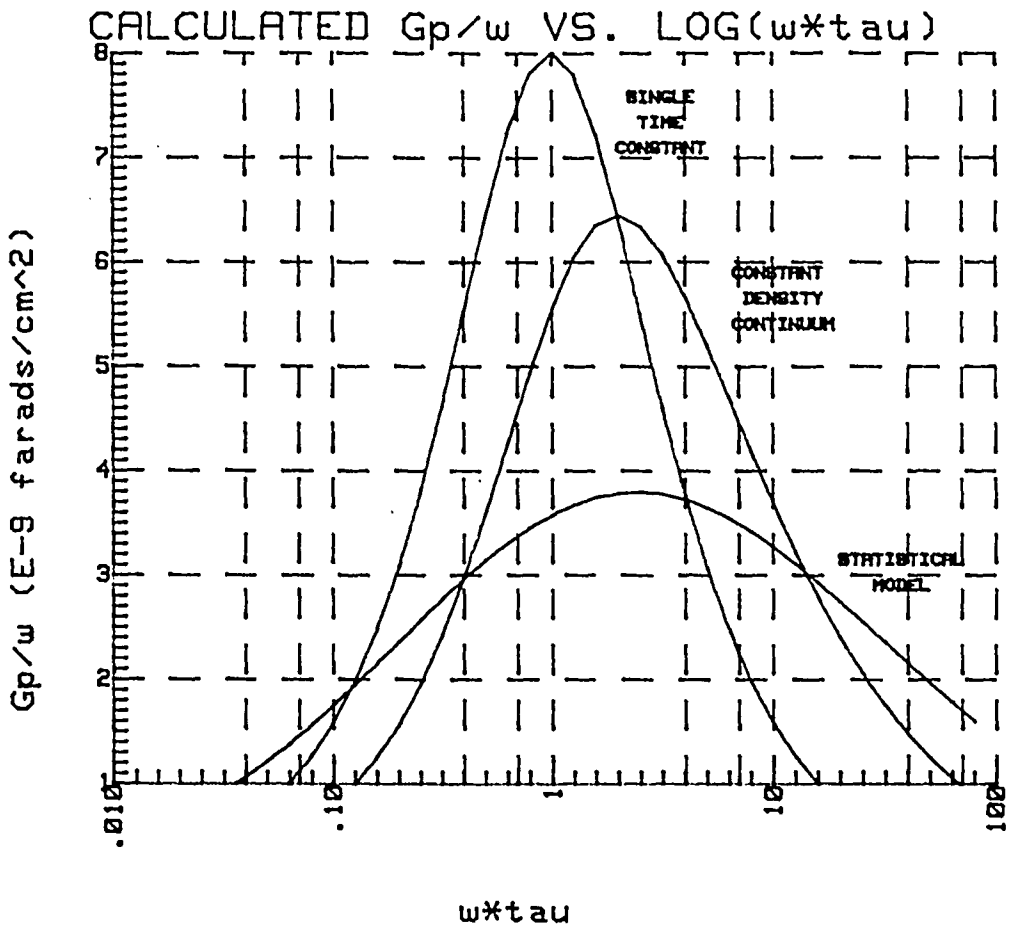


Figure 2.8 - Comparison of Conductance Curves produced by each Theory

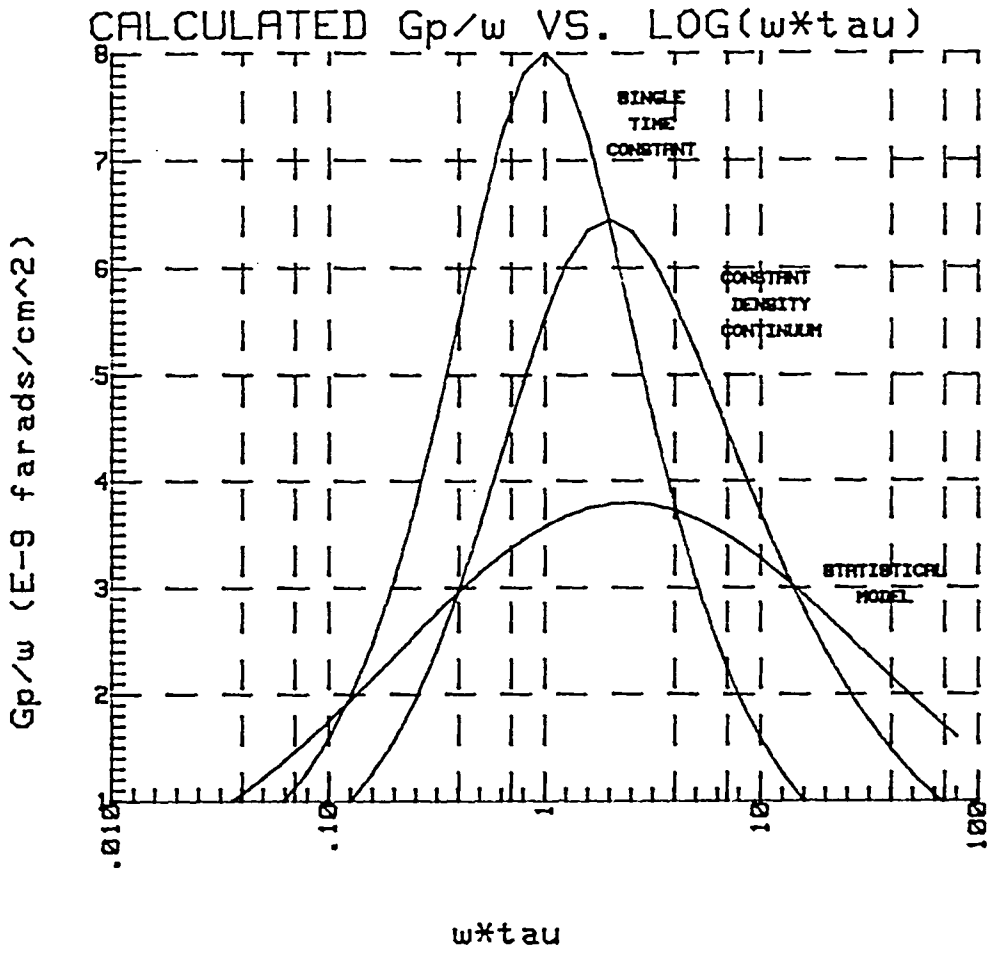


Figure 2.8 - Comparison of Conductance Curves produced by each Theory

ADMITTANCE MEASUREMENTS PROCEDURE

The experimental setup used to perform the admittance measurements is shown in Figure 2.9. Figure 2.9a shows a schematic representation of the instrumentation and Figure 2.9b shows the electrical equivalent of the measurement circuit. As is shown, a signal frequency is mixed with a bias voltage and applied to the device. The signal, after passing through the device under test and a current preamplifier arrives at the input of the lock-in amplifier. At this point, it has suffered some phase shift which is characteristic of the measurement circuit and the D.U.T. The shifted signal is compared to the original signal by the lock-in and resulting in-phase and quadrature component levels are indicated. To perform measurements the experiment is first calibrated by placing some known capacitor in the D.U.T. position and adjusting the phase compensation on the lock-in such that there is no deflection in the in-phase component meter. Next, the actual device is put in place and measurements are performed. The measurements in this case are performed under the control of a software program which sets the bias and gathers the data at each point. The program documentation is given in Appendix F.

Note, the circuit being measured, as shown in Figure 2.10, is some capacitance and conductance in parallel. However, the circuit on which the theoretical calculations are based is the oxide capacitance in series with a silicon parallel capacitance and conductance (Figure 2.5). To extract the parallel capacitance and conductance from the measured data the impedance of the oxide capacitance must be subtracted from the expression for measured impedance and the real and imaginary parts of the resulting admittances must be equated. The resulting formulas are given by:

$$\frac{G_p}{\omega C_{ox}} = \frac{G_m / C_{ox}}{\{G_m/\omega C_{ox}\}^2 + \{1 - C_m/C_{ox}\}^2} \qquad (2.8a)$$

$$\frac{C_p}{C_{ox}} = \frac{1 - C_m/C_{ox}}{\{G_m / C_{ox}\}^2 + \{1 - C_m/C_{ox}\}^2} - 1 \qquad (2.8b)$$

The formulas are applied to the measured data before fits are attempted. The fitting program used is documented in Appendix G.

Measurements were performed on the identical device of part 1 of this thesis, the .75mm circular dot capacitor. Preliminary information was obtained about the device

with the Quasi-static C-V technique. The measured C-V curve is shown in figure 2.11. From this curve a plot of D_{it} versus energy within the bandgap was obtained which yielded the plot of Figure 2.12. This plot shows a D_{it} of slightly more than $1x10^{11}/cm^2eV$. These curves were also integrated to obtain a plot of surface potential versus gate bias as shown in Figure 2.13.

Admittance measurements were performed on this device at twelve discrete frequencies by sweeping the bias between -2.5V and 0V. The measured curve at 1kHz is shown in Figure 2.14. From the measured curves a bias in depletion was chosen and the data was "cross sectioned" to obtain conductance and capacitance versus frequency curves. The bias chosen was -1.05V which from Figure 2.13 corresponds to a surface potential of about -.21V. Since the bulk Fermi potential is about -.29V this is near the center of the depletion biases. The measured and fitted curves are shown in Figure 2.15.

The fits for the curves are especially good and agree well with the results of the Quasi-static C-V measurement. The Quasi-static C-V technique indicated a D_{it} of slightly over $1x10^{11}/cm^2eV$, whereas the conductance measurements were fit with a D_{it} of $9x10^{10}/cm^2eV$.

From the fitting parameter τ_n, the trap response time, the electron capture cross section can be calculated from the equation:

$$\sigma_n = \frac{1}{v_t \tau_n N_d} \exp -\left(\frac{q\psi_s}{kT}\right) \qquad (2.9)$$

where v_t is the electron thermal velocity (about 10^7 at room temperature). A surface potential of -.2V yields a capture cross section of about $7\text{x}10^{-16}\text{cm}^2$. Because of the exponential dependance of the cross section on surface potential this can at best be considered an order of magnitude calculation. The overall consistancy of the data points to the validity of the technique.

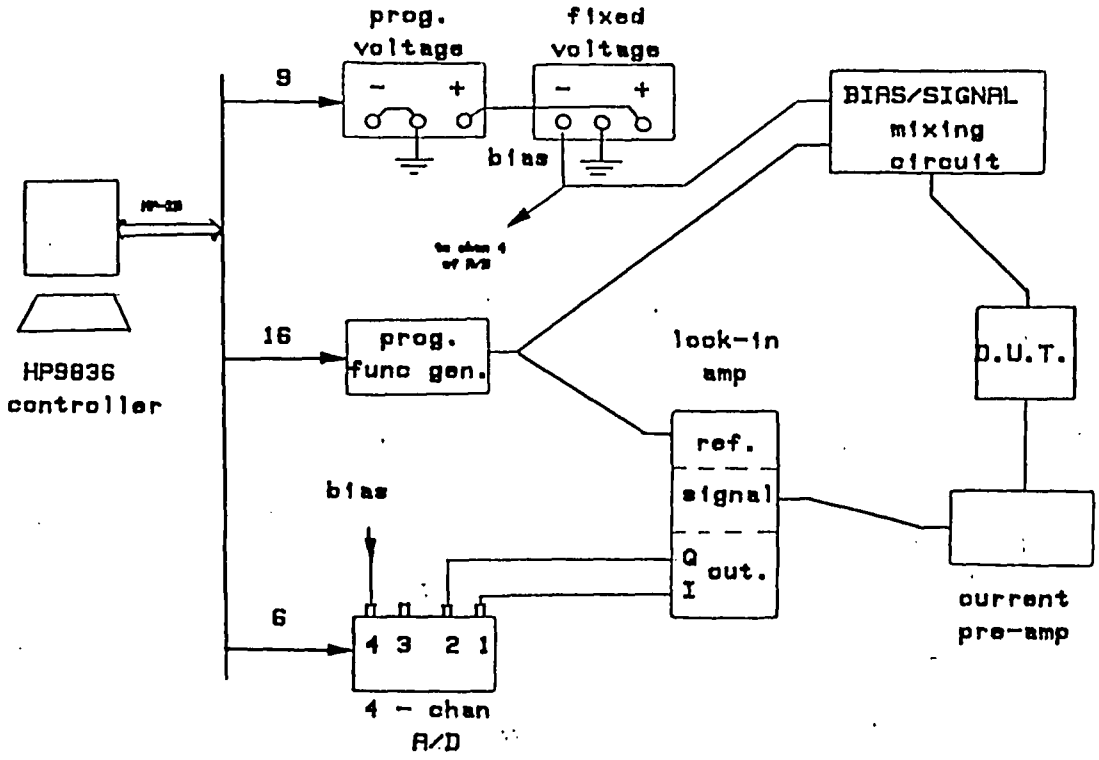
A second experiment was performed to investigate the fitting parameter σ_s. The theory states that due to the discrete nature of the electron point charges, the surface potential exhibits a granularity which follows the point charge distribution. Thus, a variation in the interface charge distribution alters the character of the surface potential fluctuations. This postulate was first investigated by Castagne and Vapaille /11/, who drifted charged lithium ions through the oxide layer to a region near the interface. Their measurements showed a qualitative agreement with theory.

In this experiment, an MNOS capacitor was used as the MIS system. With this device the point charge distribution near the interface may be electrically altered by tunneling various amounts of charge into the nitride-oxide layer. To apply the admittance technique to the MNOS device requires special consideration to avoid the read-disturb effect caused by changes in the gate bias. The measurements are performed by allowing the device to settle into a "quasi-equilibrium" condition at a particular gate bias corresponding to some average surface potential. This potential is indicated by a peak in the conductance versus bias curve at a specified search frequency. At this potential, the admittance is measured with respect to frequency only. The experimental technique is to first characterize the admittance of a virgin (electrically unperturbed) MNOS device with respect to frequency. Next, the device is perturbed by the application of a large positive or negative gate voltage, thereby altering the charge near the interface. The admittance is again measured at the same average surface potential, indicated by the peak in the conductance curve at the search frequency. Finally, the measurements are fit to theory to obtain the parameter which represents the varience in surface potential, σ_s.

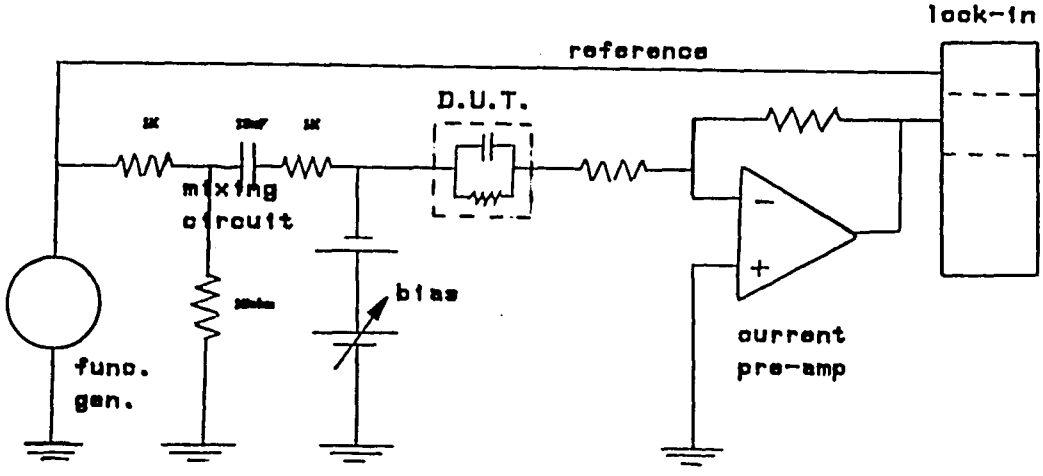
Reasonable fits with theory may be obtained with this technique and, as predicted by theory there is an increase in the surface potential fluctuation when the amount of charge near the interface is increased, independant of the sign of the charge. The data and fits are shown in Figure 2.16. Theory predicts /10/ that the square of the varience is proportional to the interface charge (i.e. $\sigma_s^2 \sim Q_i$). If the flatband voltage is proportional to the interface charge, then the following relationship can be derived:

$$\sigma_s^2 = K|V_{fb}| \qquad (2.10)$$

where K is a proportionality constant. Using V_{fb} of -2.5V and σ_s of 2.4 for a virgin device, K solves to 2.3. Using this value of K and equation (2.10) predicted values of σ_s can be obtained and compared with measured values. The results are shown in Table 2.1. These results further verify the validity of the admittance technique and show evidence of the physical reality of the postulated potential fluctuations.



(a)



(b)

Figure 2.9 - Admittance Measurement Experimental Setup
(a) Schematic ; (b) Electrical Equivalent

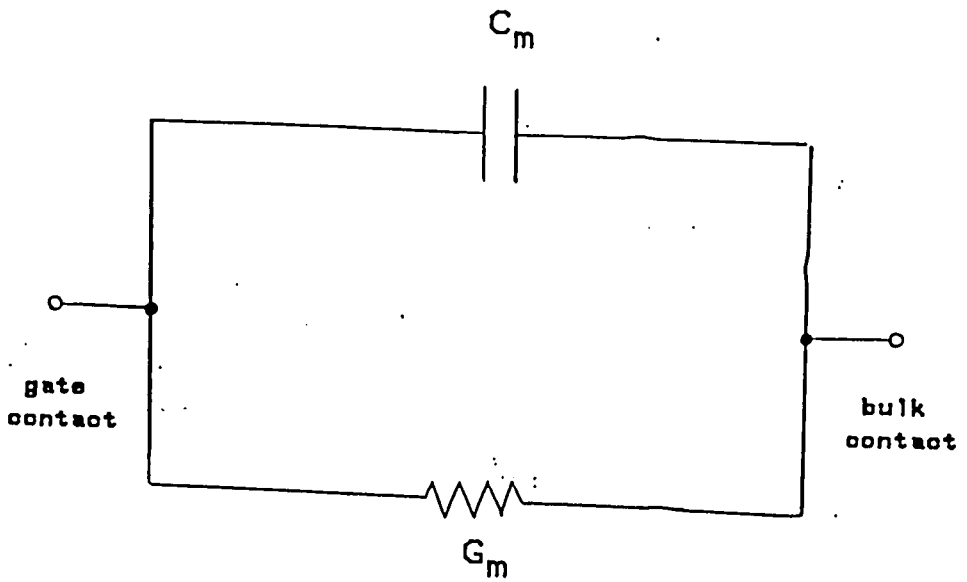


Figure 2.10 - Measured Circuit

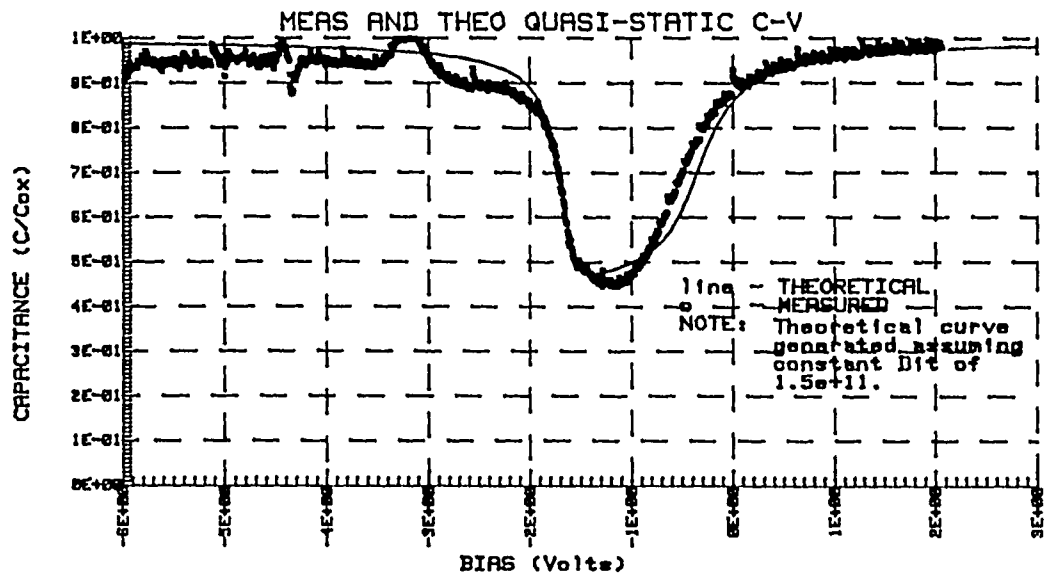


Figure 2.11 - Measured Quasi-static C-V Curve

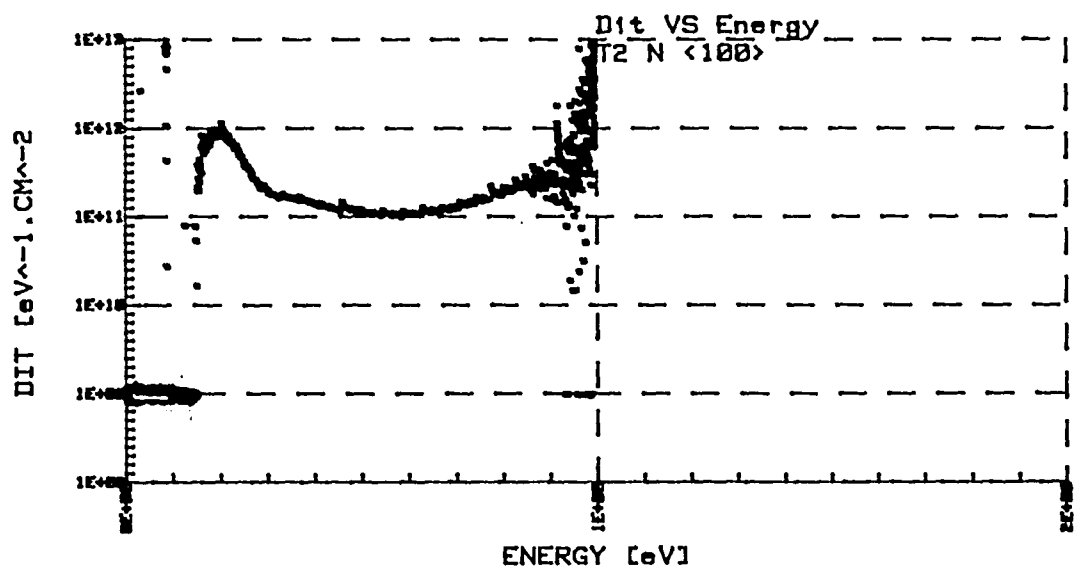


Figure 2.12 - Curve of D_{it} vs. Energy extract from QSCV

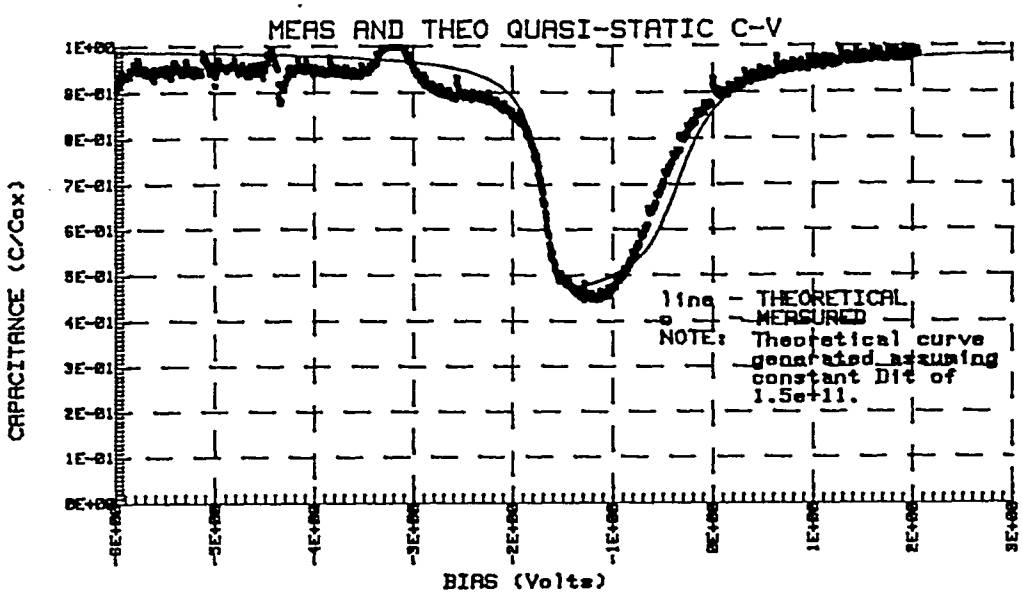


Figure 2.11 - Measured Quasi-static C-V Curve

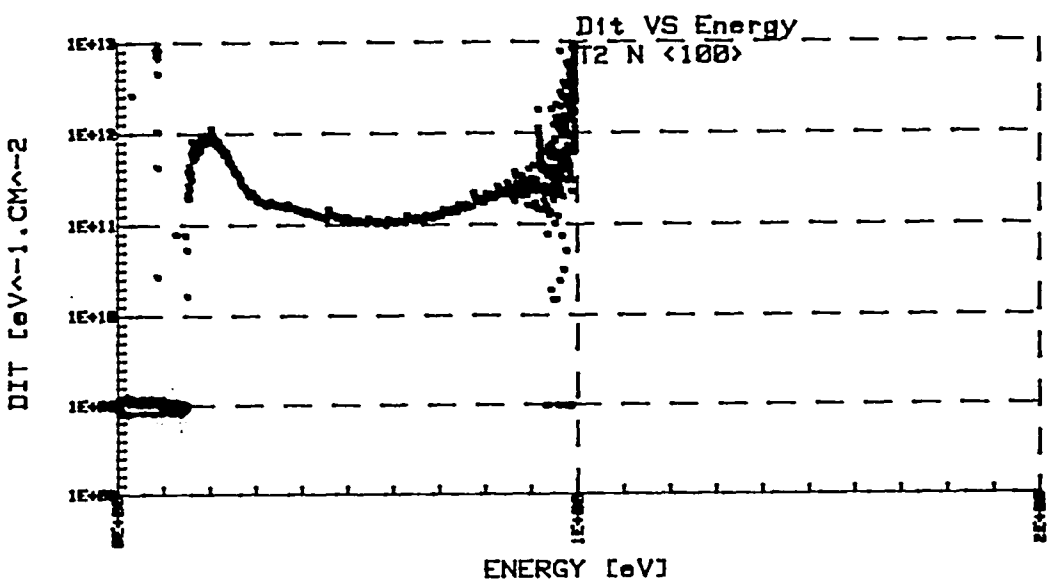


Figure 2.12 - Curve of D_{it} vs. Energy extract from QSCV

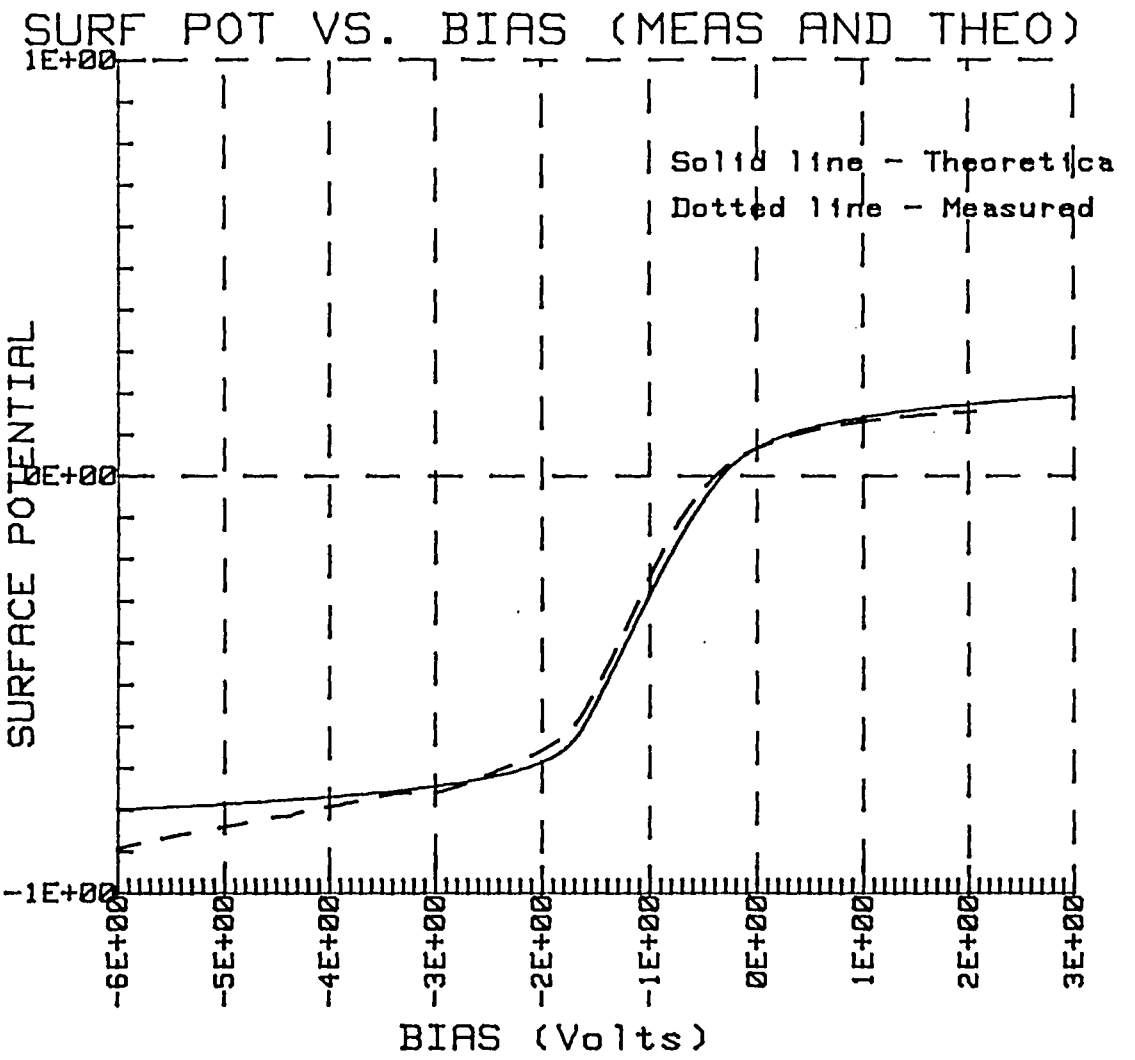


Figure 2.13 - Curve of Surface Potential vs. Gate Bias Extracted from QSCV

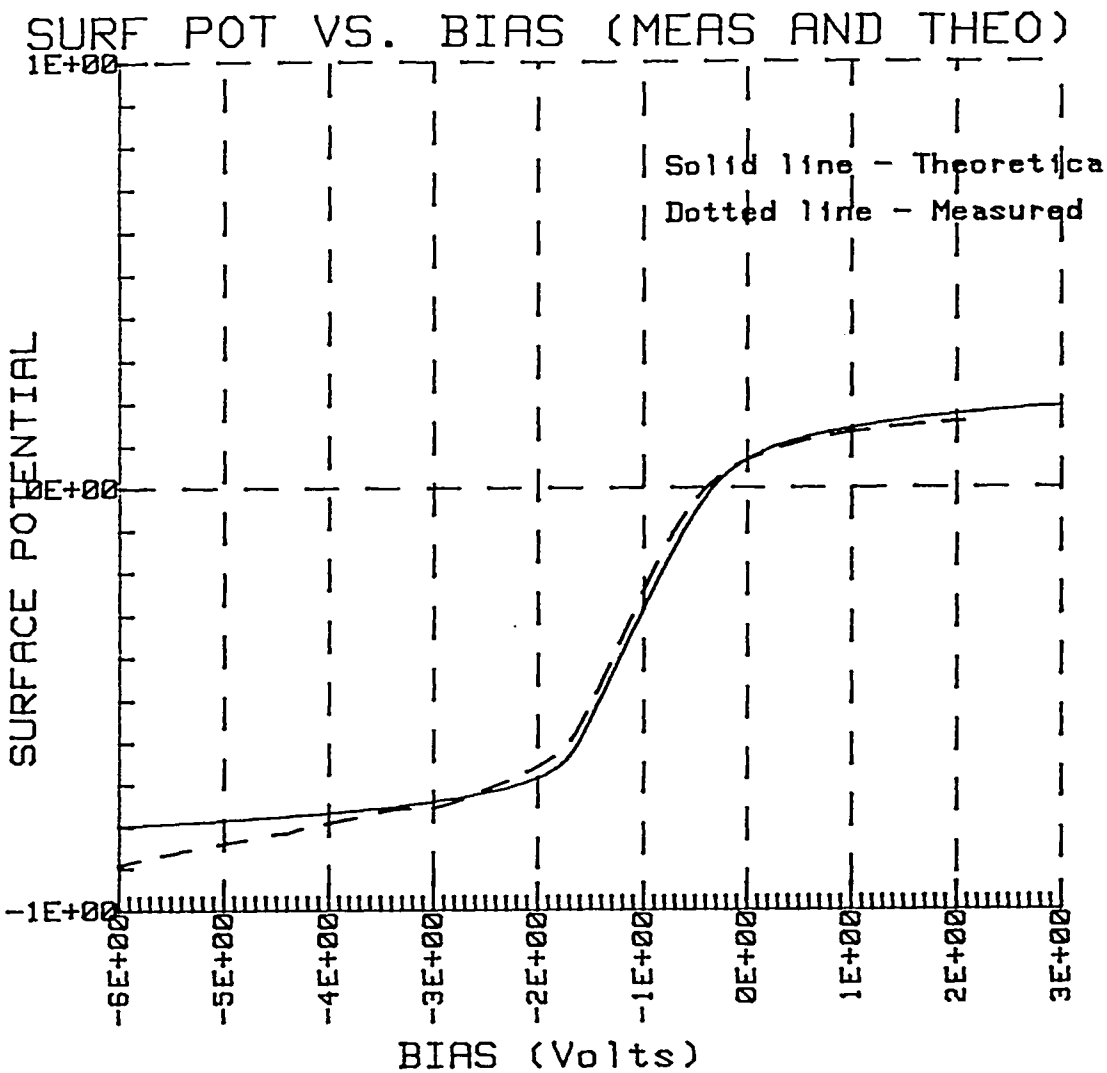


Figure 2.13 - Curve of Surface Potential vs. Gate Bias Extracted from QSCV

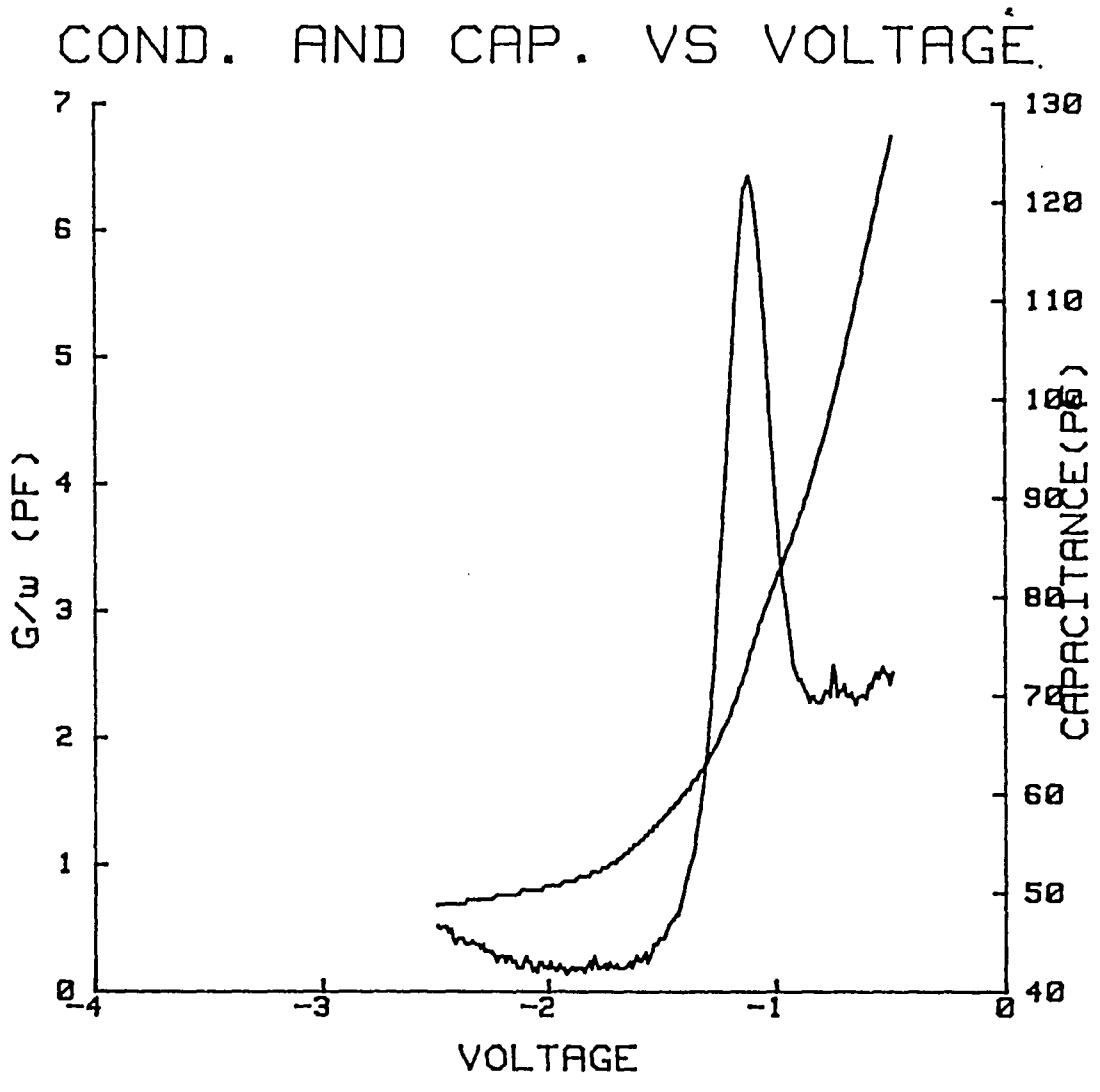


Figure 2.14 - Admittance Curve Measured at 1KHz

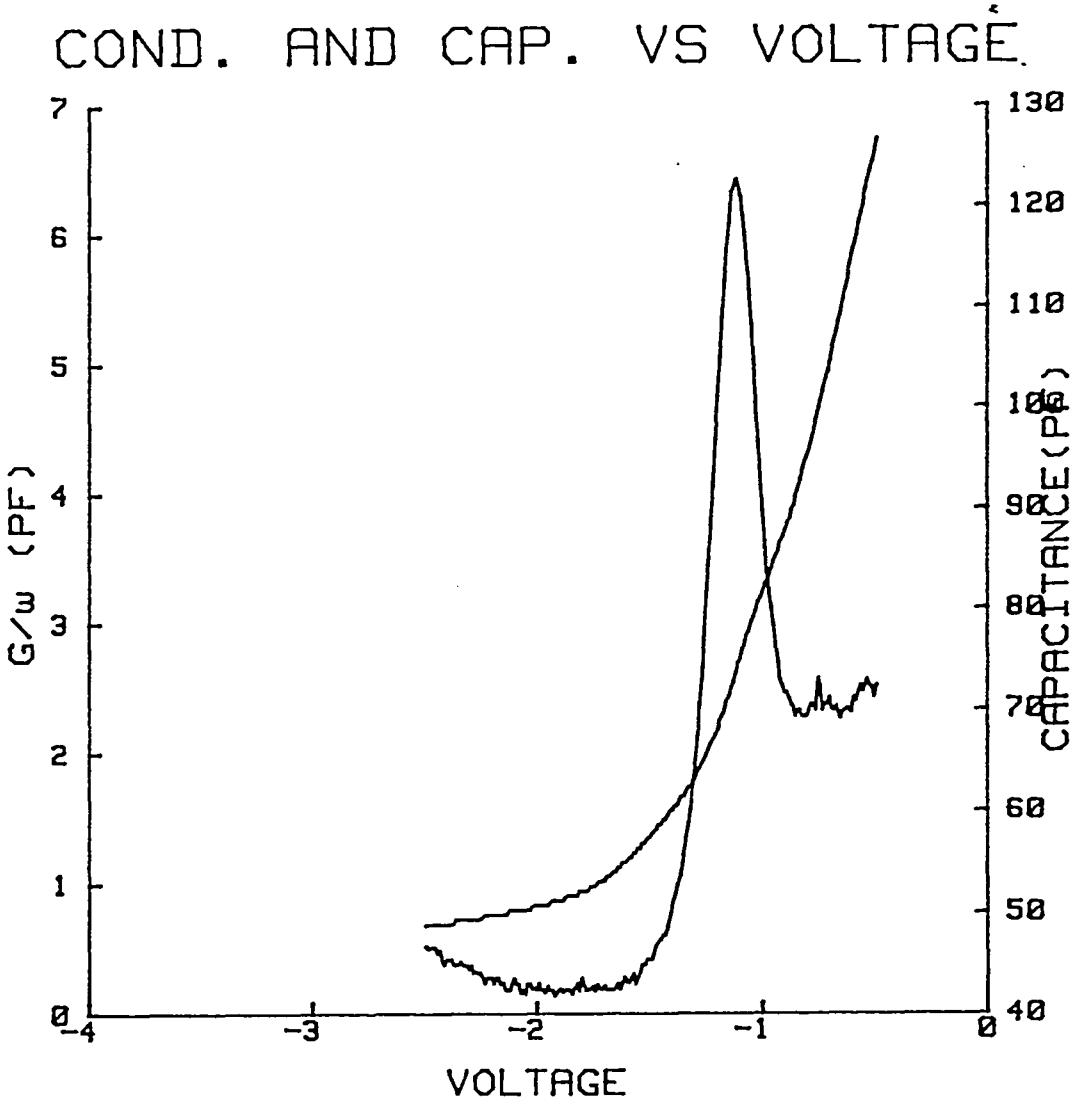
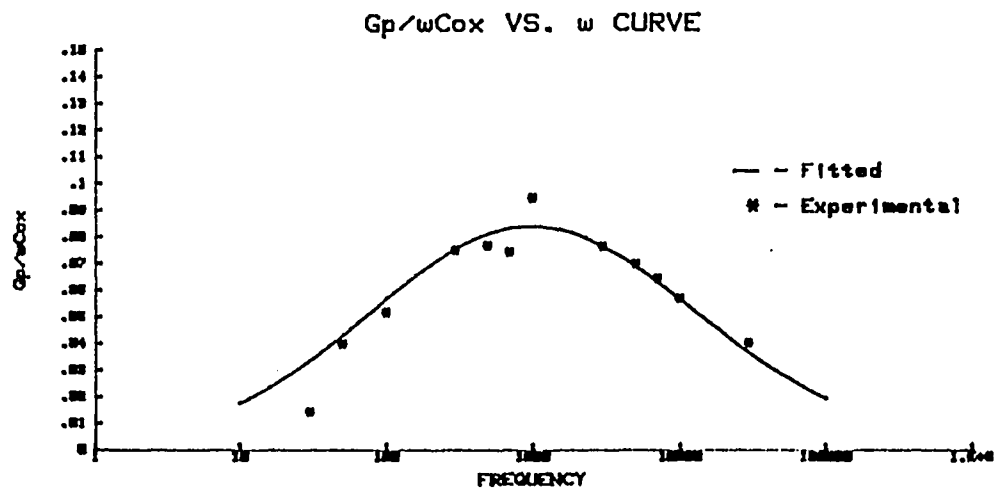


Figure 2.14 - Admittance Curve Measured at 1KHz



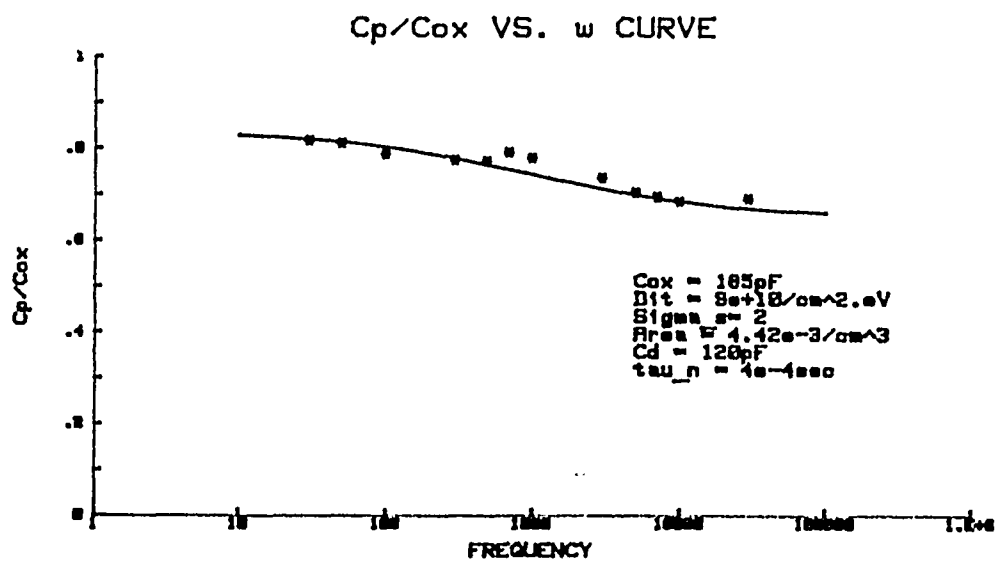
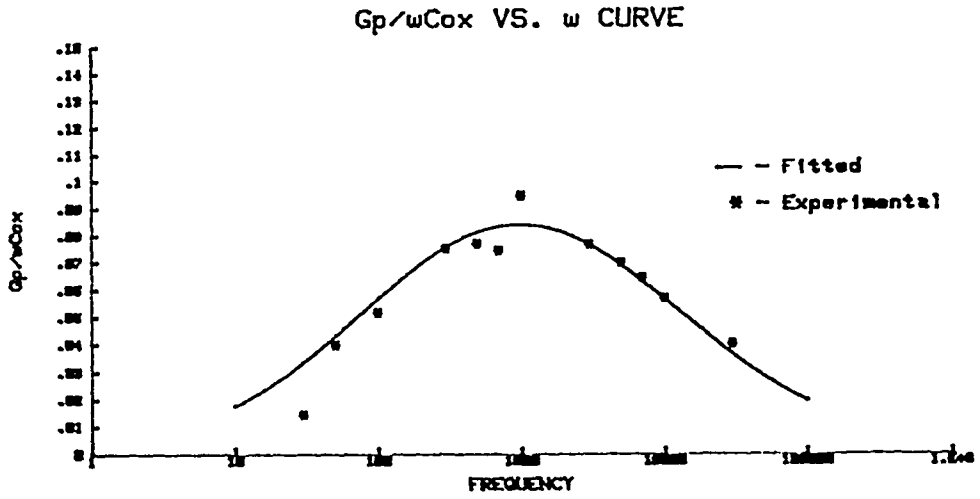


Figure 2.15 - Measured and Fitted Admittance vs. Frequency Curves



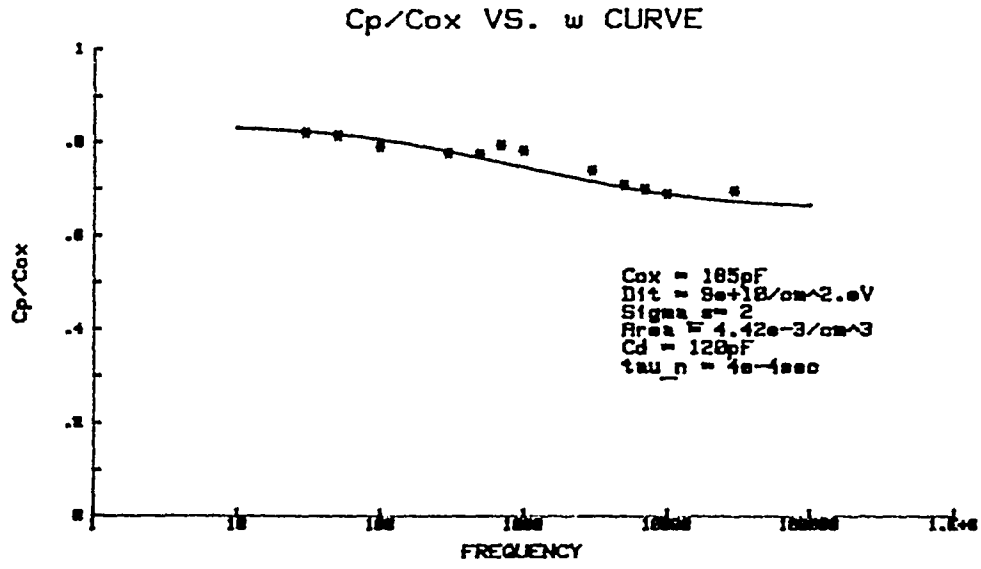


Figure 2.15 - Measured and Fitted Admittance vs. Frequency Curves

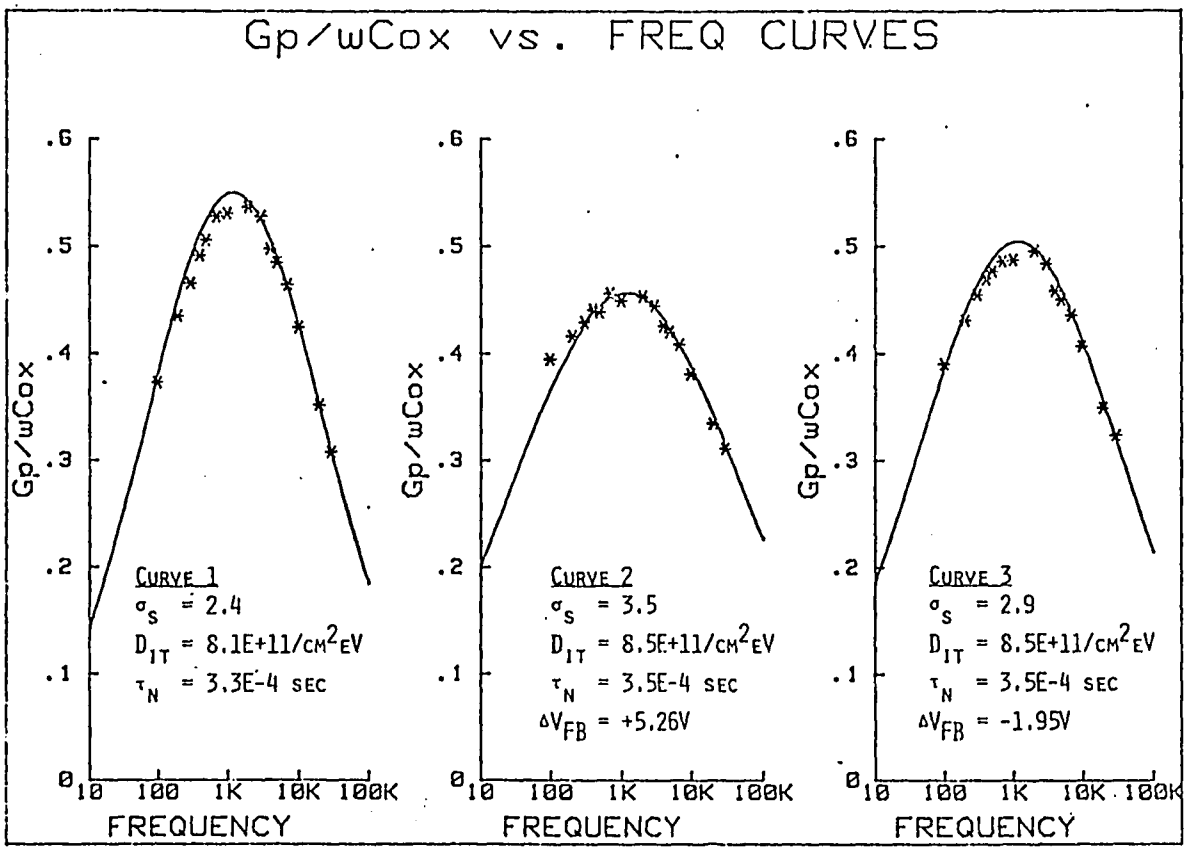


Figure 2.16 Results of measurements on MNOS devices.

Table 2.1 - Comparison of Predicted and Measured Values of σ_s.

Device History	V_{fb}	Predicted σ_s	Measured σ_s
Virgin	-2.5V	-	2.4
Virgin written w/ +21.2V	+2.75V	2.5	2.9
Virgin written w/ -21.2V	-4.5V	3.2	3.5

DISCUSSION OF ADMITTANCE MEASUREMENT AUTOMATION

The admittance software package is an excellent example of the advantagious use of an automated control system in an experimental laboratory. This program again used the modular design technique as discussed in the C-t program but to an even greater extent. The tasks programed here were often simpler and more limited, therefore allowing even greater program flexibility. This is expecially true in the analysis program where the steps of the analysis were left to the user instead of procedurally automated. The results is a compromise between automatic and manual control. The advantage gained is that the user of this program will obtain some understanding of the analysis technique because he is guiding the analysis through the use of the software keys and numeric inputs. This is a much better scheme than the kind of black box approach where data goes into analysis and the answer comes out the other end. This is especially true in a research enviroment where understanding of where the answers come from rather than the answers themselves is often the important factor.

The experimental results presented here were taken in approximately two days time. This amount of data

if taken by conventional techniques, would probably have taken at least a week. Considering the amount of information extracted from these measurements this is an extreme experimental advantage.

CONCLUSIONS

From the experiments presented here the advantages to the user of the computer in the experimental laboratory can be seen. The computer saves work and frees the researcher from time consuming data gathering tasks. These advantages are best exploited through the use of a modular programming scheme where simple, well-defined tasks are programmed for use in any appropriate procedural manner. With this type of programming scheme the use of the computer will not hide the complexities of analytical techniques by producing answers out of a "black box" type of environment. It is suggested that all future software be developed in such a manner and the automation of long, complex analytical techniques be avoided. The automation of the Zerbst technique may be an example of the automation of too complex a task. The user who sits down and uses this program will probably not get a good feel for the device physics involved, instead he will get numbers for generation lifetime and surface recombination velocity which may or may not be relevant.

Some more functional conclusions that can be drawn from the automation performed are one, the time monitoring

of experimental parameters can be best accomplished by dedicated processors and two, taking the derivative of computer gathered data may be best performed numerically instead of analytically. Further discussion of these points are included in the C-t experiment.

From the experience gained in using the Zerbst technique it can be concluded that its use is valid. The results obtained are good and with use of initial condition which inforce the analytical assumptions (i.e. starting in inversion) consistent results are obtained.

From the use of the Admittance technique it can be concluded that it is an excellent candidate for automation. The data measured fits well to theoretical calculations. Investigations into the fitting parameter σ_s also show evidence of its physical reality.

APPENDIX A

Derivation of the Zerbst Analysis Technique

Consider the MOS Capacitor in which the total charge on the device is equal to zero. Mathematically stated:

$$Q_g + Q_i + Q_d = 0 \qquad (A.1)$$

Q_g = Charge on Gate
Q_i = Charge in Inversion Layer
Q_d = Charge in Depletion Layer

The current in the circuit external to the device is:

$$I_g = \frac{dQ_g}{dt} = -\left(\frac{dQ_i}{dt} + \frac{dQ_d}{dt}\right) \qquad (A.2)$$

Now inversion charge is supplied by generation current in the depletion region (non-equilibrium) and also by generation in the surface states. Therefore:

$$\frac{dQ_i}{dt} = \frac{qn_i}{2\tau_g}(W-W_f) + qn_iS \qquad (A.3)$$

where τ_g = generation lifetime
S = surface recombination velocity
W_f = final or equilibrium depletion layer width

Also:

$$\frac{dQ_d}{dt} = qN_d\left(\frac{dW}{dt}\right) \qquad (A.4)$$

Substituting equations (A.3) & (A.4) into equation (A.1):

$$I_g = -q \frac{n_i}{2\tau_g}(W-W_f) + n_i S + N_d\left(\frac{dW}{dt}\right) \qquad (A.5)$$

Now consider the expression for gate voltage:

$$V_g = \frac{Q_g}{C_{ox}} + \psi_s = \frac{Q_g}{C_{ox}} - \frac{qN_d W^2}{2\varepsilon_s} \qquad (A.6)$$

With non-varying gate voltage then:

$$\frac{dV_g}{dt} = 0 = \frac{I_g}{C_{ox}} - \frac{qN_d W}{\varepsilon_s}\left(\frac{dW}{dt}\right) \qquad (A.7)$$

Substituting expression for I_g from (A.5) into equation (A.7):

$$0 = \frac{1}{C_{ox}} \frac{n_i}{2\tau_g}(W-W_f) + n_i S - \frac{N_d}{C_{ox}} \frac{dW}{dt} - \frac{N_d W}{\varepsilon_s}\left(\frac{dW}{dt}\right) \qquad (A.8)$$

Solving equation (A.8) for $\frac{dW}{dt}$:

$$\frac{dW}{dt} = - \frac{\frac{n_i}{2\tau_g}(W-W_f) + n_i S}{C_{ox}\left(\frac{N_d}{C_{ox}} + \frac{N_d W}{\varepsilon_s}\right)} \qquad (A.9)$$

Noting that for capacitances for the silicon and oxide in series $1/C = 1/C_{ox} + 1/C_s$, we can anticipate our result and write an expression for the time derivative

of $(C_{ox}/C)^2$:

$$\frac{C_{ox}}{C}^2 = \left(1 + \frac{C_{ox}W}{\varepsilon_s} \right)^2 \qquad (A.10)$$

$$\frac{d}{dt}\left(\frac{C_{ox}}{C}\right)^2 = 2\left(1 + \frac{C_{ox}W}{\varepsilon_s}\right)\left(\frac{C_{ox}}{\varepsilon_s}\right)\left(\frac{dW}{dt}\right) \qquad (A.11)$$

Substituting equation (A.9) into (A.11):

$$\frac{d}{dt}\left(\frac{C_{ox}}{C}\right)^2 = -2\left(1 + \frac{C_{ox}W}{\varepsilon_s}\right)\frac{\frac{n_i}{2\varepsilon_s\tau_g}(W-W_f) + \frac{n_i S}{\varepsilon_s}}{\frac{N_d}{C_{ox}} + \frac{N_d W}{\varepsilon_s}} \qquad (A.12)$$

Now note that:

$$\frac{W}{\varepsilon_s} = \frac{1}{C_s} = \frac{1}{C} - \frac{1}{C_{ox}} = \frac{C_{ox} - C}{C_{ox}C} \quad \text{and} \quad \frac{W_f}{\varepsilon_s} - \frac{C_{ox} - C_f}{C_{ox}C_f} \qquad (A.13)$$

Therefore by substituting equation (A.13) into (A.12) and reducing:

$$-\frac{d}{dt}\left(\frac{C_{ox}}{C}\right)^2 = \frac{2C_{ox}}{N_d C_f}\left(\frac{n_i}{2\tau_g}\left(\frac{C_f}{C} - 1\right) + \frac{n_i C_f S}{\varepsilon_s}\right) \qquad (A.14)$$

This is the final result of the Zerbst derivation.

APPENDIX B

Admittance Between Conduction Band and Interface Trap

In conjunction with figure 2.2 consider the current between the conduction band and a trap:

$$i_n(t) = qn_t c_n\{1-f(t)\}n(t) - qn_t e_n f(t) \qquad (B.1)$$

Seperate the fermi function and surface population function into AC and DC components

$$f(t) = f_o + \delta f(t) \qquad n(t) = n_o + \delta n(t) \qquad (B.2)$$

The equation for current becomes:

$$i_n = qn_t c_n\{(1-f_o)n_o + (1-f_o)\delta n - n_o\delta f\} - qe_n n_t(f_o + \delta f) \qquad (B.3)$$

Note the condition that the conduction band is in thermal equilibrium with the traps.

$$\text{i.e.} \qquad r_{ao} = r_{bo}$$

$$qn_t c_n(1-f_o)n_o = e_n n_t f_o \qquad (B.4)$$

Therefore

$$i_n = qn_t c_n\left((1-f_o)\delta n - n_o\frac{\delta f}{f_o}\right) \qquad (B.5)$$

Also by definition

$$i_n = qn_t\frac{\delta f}{t\delta t} \qquad (B.6)$$

Equating (B.5) and (B.6):

$$\frac{\delta f}{\delta t} = c_n(1-f_o)\delta n - c_n n_o \frac{\delta f}{f_o} \qquad (B.7)$$

Consider small signal fluctuations of the Fermi function of the form $\delta f = f_m \exp(j\omega t)$. Therefore:

$$\frac{\delta f}{\delta t} = j\omega\delta f$$

$$j\omega\delta f = c_n(1-f_o)\delta n - c_n n_o \frac{\delta f}{f_o}$$

$$\delta f = \frac{f_o(1-f_o)\delta n}{\{1 + j\omega f_o/(c_n n_o)\}n_o} \qquad (B.8)$$

Substituting equation (B.8) into (B.5):

$$i_n = \frac{j\omega q n_t f_o(1-f_o)\delta n}{1 + j\omega f_o/(c_n n_o)\delta n_o} \qquad (B.9)$$

Note that $n = \exp(u)$ where u surface band bending, so that $n = \exp(u)$ u. Therefore:

$$\frac{\delta n}{n} = \delta u = \frac{q}{kT}\delta\psi \qquad (B.10)$$

Then by substituting equation (B.10) into (B.9):

$$i_n = \frac{j\omega q^2 n_t f_o(1-f_o)\delta\psi}{kT\ 1 + j\omega f_o/(c_n n_o)} \qquad (B.11)$$

Since $i_n = Y_n \delta\psi$ the small signal admittance can be defined as:

$$Y_n = j\omega \frac{q^2 n_t f_o(1-f_o)}{kT\{1 + j\omega f_o/(c_n n_o)\}} \qquad (B.12)$$

APPENDIX C

Admittance of Continnuum of States

Integrating the expression given by equation (2.1) over a continuum of states:

$$Y_s' = j\omega \frac{q^2}{kT} \int_{-\infty}^{\infty} \frac{D_{it} f_o(1-f_o)}{1 + j\omega f_o/(c_n n_{so})} \, d\psi_s \qquad (C.1)$$

where now D_{it} is the interface state density per unit area per eV accross the bandgap.

With the state occupation function given by the fermi function of equation (2.2) it can be shown that:

$$f_o(1-f_o) = \frac{kT}{q}\left(\frac{df_o}{d\psi_s}\right) \qquad (C.2)$$

Substituting (C.2) into (C.1) and changing the limits of integration.

$$Y_s' = j\omega q \int_0^1 \frac{D_{it}}{1 + j\omega\tau_n f_o} \, df_o \qquad \text{where } \tau_n = \frac{1}{c_n n_{so}} \qquad (C.3)$$

Since the magnitude of the function being integrated peaks over a fairly small range and D_{it} is a slowly changing function the integral of equation (C.3) can be rearranged as:

$$Y_s' = j\omega D_{it} q \int_0^1 \frac{df_o}{1 + \omega^2 \tau_n^2 f_o^2} + j\omega D_{it} q \int_0^1 \frac{j\omega\tau_n f_o \, df_o}{1 + \omega^2 \tau_n^2 f_o^2} \qquad (C.4)$$

Evaluating the integrals.

$$\int_0^1 \frac{df_o}{1 + \omega^2 \tau_n^2 f_o^2} = \frac{1}{\omega\tau_n} \tan^{-1}(\omega\tau_n) \qquad (C.5a)$$

$$\int_0^1 \frac{f_o df_o}{1 + \omega^2 \tau_n^2 f_o^2} = \frac{1}{2\omega^2 \tau_n^2} \ln(1 + \omega^2 \tau_n^2) \qquad (C.5b)$$

Substituting (C.5) back into (C.4)

$$Y_s' = \frac{jqD_{it}}{\tau_n} \tan^{-1}(\omega\tau_n) + \frac{qD_{it}}{2\tau_n} \ln(1 + \omega^2 \tau_n^2) \qquad (C.6)$$

From equation (C.6) the Conductance and Capacitance can be defined as:

$$\frac{G_p}{\omega} = \frac{qD_{it}}{2\omega\tau_n} \ln(1 + \omega^2 \tau_n^2) \qquad (C.7a)$$

$$C_p = \frac{qD_{it}}{\omega\tau_n} \tan^{-1}(\omega\tau_n) \qquad (C.7b)$$

APPENDIX D

Statistical Admittance Model

Assuming a Gaussian approximation for a Poission distributed surface potential and integrating the expressions for admittance as given by equation 2.6 for n-type silicon:

$$\frac{\langle G_p \rangle}{\omega} = \frac{q}{2} \int_{\infty}^{\infty} \frac{c_n n_{so} D_{it}}{\omega} \ln\left(1 + \frac{\omega^2}{c_n^2 n_{so}^2}\right) P(\psi_s) d\psi_s \qquad (D.1a)$$

$$\langle C_p \rangle = C_d + q \int_{\infty}^{\infty} \frac{c_n n_{so} D_{it}}{\omega} \tan^{-1}\left(\frac{\omega}{c_n n_{so}}\right) P(\psi_s) d\psi_s \qquad (D.1b)$$

$$\text{where } P(\psi_s) = (2\pi\sigma_s)^{-\frac{1}{2}} \exp -\frac{(\psi_s - \langle\psi_s\rangle)^2}{2\sigma_s^2}$$

Realizing the n_{so} can be written as $N_d \exp(-\psi_s)$ and assuming D_{it} and c_n to be approximately constant we can rewrite the conductance expression as:

$$\frac{G_p}{\omega} = \frac{qD_{it}}{2\omega} c_n N_d \exp(-\langle\psi_s\rangle)(2\pi\sigma_s)^{-\frac{1}{2}} \int_{\infty}^{\infty} \exp -\frac{(\psi_s - \langle\psi_s\rangle)^2}{2\sigma_s^2}$$

$$* \exp(\langle\psi_s\rangle - \psi_s) \ln\left[1 + \frac{\omega^2}{c_n^2 N_d^2 \exp(-2\langle\psi_s\rangle)}\right] \qquad (D.2)$$

$$* \exp(2\psi_s - 2\langle\psi_s\rangle) d\psi_s$$

$$\text{letting } \eta = \psi_s - \langle\psi_s\rangle \text{ and } n = \frac{1}{c_n N_d \exp(-\langle\psi_s\rangle)}$$

$$\frac{G_p}{\omega} = \frac{qD_{it}}{2\omega\tau_n(2\pi\sigma_s)^{\frac{1}{2}}} \int_{\infty}^{\infty} \exp\left(-\frac{\eta^2}{2\sigma_s^2}\right) \exp(-\eta)\ \ln(1+\omega^2\tau_n^2\exp 2\eta)\, d\eta$$

Similar substitution in the capacitance expression leads to the result of equation 2.7b.

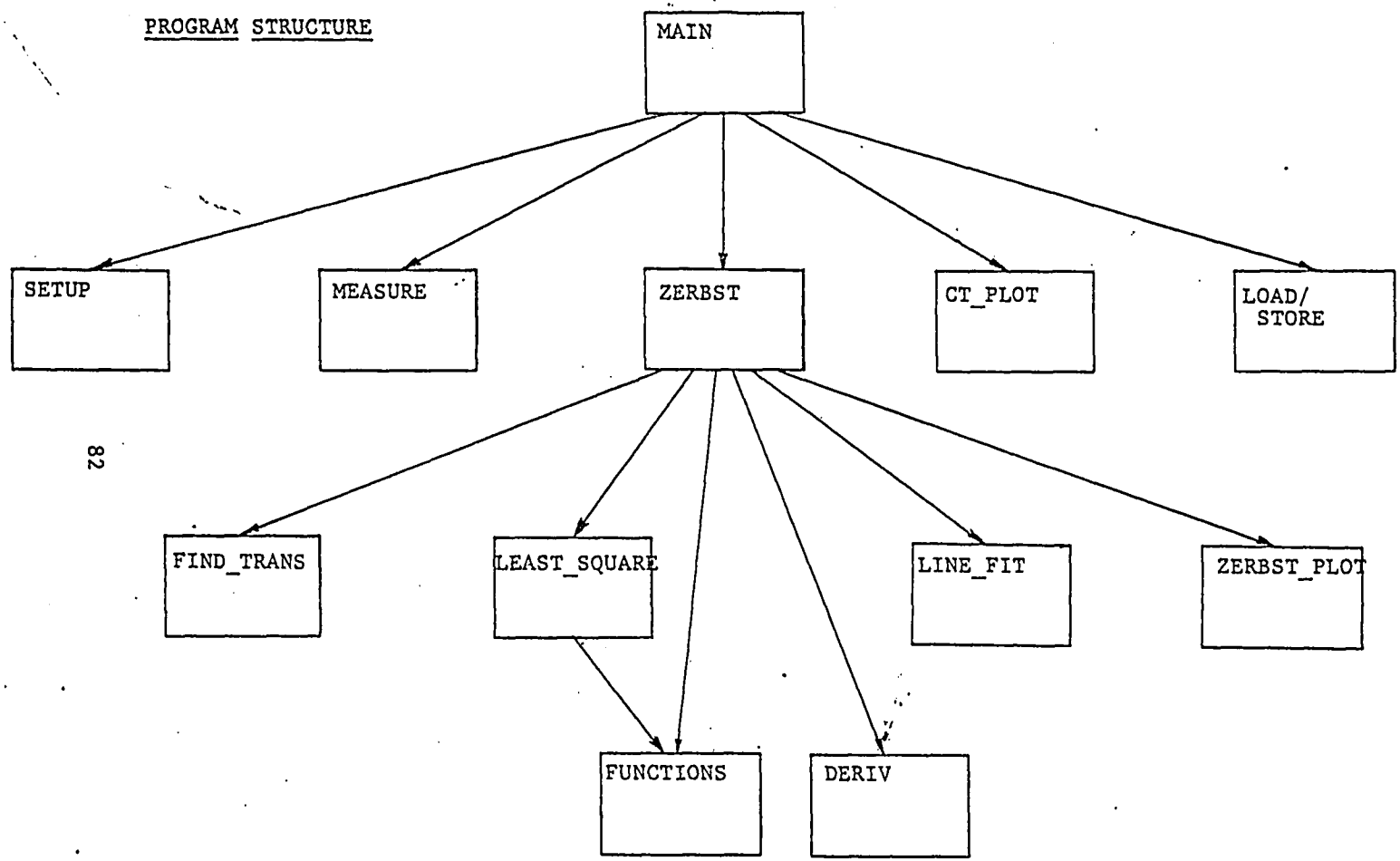
APPENDIX E

Pulsed C-t Automation Program

This program can be used to perform data gathering in a Pulsed C-t experiment and the subsequent Zerbst analysis. The program is designed in a structured fashion so that the programmed functions are available to the user through use of the software keys. The function available to the user are setup, measure graph, store, load, and analysis.

Typically, the user would first run SETUP which would prompt for needed input data. Following this task, the user would run MEASURE which would gather the data. Sometime during the measure routine (usually near the beginning) the user must press the "PULSE" key to apply the voltage pulse to the capacitor. When all the data is gathered, it can be graphed using the "GRAPH" function or stored for later use with the "STORE" function. Also at this point, the Zerbst analysis can be performed by pressing the "ZERBST" key. Previously measured data files are recalled with use of the "LOAD" key.

PROGRAM STRUCTURE



Variable Directory of COM Block
/PULSECT_DATA/

Variable	Description
NOTE$ -	(STRING) 30 characters in which user may enter any comments or identifier; set by subroutine SETUP.
GENDATE$ -	(STRING) Date fixed by measure subroutine indicating the date of measurement.
PACERATE$ -	(STRING) User input parameter in subroutine SETUP indicating sampling rate of A/D converter; one character variable which can be letters A-F.
DIGPULSE$ -	(STRING) Four character parameter generated by SETUP subroutine from user input of pulse height. Formatted output to D/A converter including scaler character (1 or 2) and 3 characters of digital counts.
DOPING_TYPE$ -	(STRING) One character user input parameter in subroutine SETUP indicating doping type of device under test.
SETUP_FLAG -	(INTEGER) Switch set to 1 (true) when program has completed SETUP subroutine.
DATA_FLAG -	(INTEGER) Switch set to 1 (true) when program has successfully completed MEASURE subroutine.
THEO_EXP -	(INTEGER) Unused
CURVE_FIT -	(INTEGER) Switch set to 1 (true) when program has successfully completed LEAST_SQUARE subroutine; reset by SETUP or MEASURE subroutines.
CAPACITANCE(1000) -	(REAL) measured data points of capacitance transient.
READTIME(1000) -	(REAL) measured data points of times when capacitance was read; filled by subroutine measure.
FUNC_COEF(10) -	(REAL) Fitted coefficiants of functions determined by subroutine LEAST_SQUARE.
FIT_SLOPE -	(REAL) Fitted straight line slope determined by subroutine LINE_FIT

FIT_INTERCEPT - (REAL) Fitted y-axis intercept for straight line determined by subroutine LINE_FIT.

DOPING_DENS - (REAL) User input parameter in subroutine SETUP indicating doping density of device under test.

THICK_OXIDE - (REAL) Parameter determined by subroutine FIND_TRANS indicating oxide thickness of device under test.

CINS - (REAL) Parameter determined by subroutine FIND_TRANS indicating capacitance of insulator in D.U.T.

CO - (REAL) Unused.

CF - (REAL) Parameter determined by subroutine FIND_TRANS indicating the final equilibrium capacitance of device after pulse has been applied.

KS - (REAL) User input parameter indicating the dielectric constant of silicon in the D.U.T.

KIN - (REAL) User input parameter in SETUP indicating the dielectric constant of the insulator in the D.U.T.

NI - (REAL) User input parameter in SETUP indicating the intrinsic carrier density in the D.U.T.

AREA - (REAL) User input parameter in SETUP indicating the area of the D.U.T.

DELTA - (REAL) Parameter determined in SETUP indicating the time interval between measurements determined from the sampling rate of the A/D converter.

LIFETIME - (REAL) Parameter determined by subroutine ZERBST indicating the measured generation lifetime of the D.U.T.

SURFACE_VEL - (REAL) Parameter determined by subroutine ZERBST indicating the measured surface recombination velocity of the D.U.T.

Subroutine Directory
Pulsed C-T Experiment Program

MEASURE

COM Blocks:

/PULSECT_DATA/

Function:

Subroutine monitors the capacitance across the D.U.T. at the pacing rate determined by the sampling rate of the A/D converter. This subroutine also applies the voltage pulse to the device when defined software key is pressed by user.

SETUP

COM Blocks

/PULSECT_DATA/

Function:

Subroutine re-initializes the data COM block to restart experiment. User input parameters are filled through user prompts.

CT_PLOT

COM Blocks:

/PULSECT_DATA/

/AXIS_FLAG/
AXIS_DRAWN - Switch indicating if axis drawn on graphics display.

Function:

This subroutine plots the data gathered by the measure subroutine in a plot of capacitance versus time.

GAUSS ELIM

Pass Parameters:

N - (INTEGER) Number of rows in A matrix.

A(*) - (REAL) Augmented matrix of N rows which will be diagonalized.

X(*) - (REAL) Array of N elements gining the result of the gauss elimination process.

Function:

This subroutine diagonalizes the A matrix by the method of Gauss Elimination. The X matrix or result matrix is determined by dividing the auxilary row of the augmented matrix by the diagonal element of the A matrix.

LEAST SQUARE

Pass Parameters:

X(*) - (REAL) Array of independant parameters of data points.

Y(*) - (REAL) Array of dependant parameter values at the X value data points; array to be fitted.

DATA_POINTS - (INTEGER) Number of points in data array

NO_OF_FUNCTIONS - (INTEGER) Number of functions to be fitted to the data.

INFO_FLAG - (INTEGER) Unused.

COEFFICIANTS(*) - (REAL) Coefficiants of functions which give best least square fit.

Function:

This subroutine creates the matrix to be solved by Gauss Elimination which is the set of equations which minimizes the squared error between the fit and the data.

FUNCTIONS

Pass Parameters:

X - (REAL) Independant variable value.

FUNC(*) - (REAL) Value of evaluated functions.

Function:

Evaluate programmed functions using given x value.

STORE DATA

COM Blocks:

/PULSECT_DATA/

Function:

This subroutine creates and writes a data disk file which holds the data from the COM Block /PULSECT_DATA/

LOAD DATA

COM Blocks:

/PULSECT_DATA/

Function:

This subroutine loads a data file from disk properly formatted to fill the COM Block /PULSECT_DATA/

ZERBST

COM Blocks:

/PULSECT_DATA/

Function:

This subroutine is the driver for the Zerbst method of data analysis of the capacitance transient. This subroutine extracts the recovery transient through a call to FIND_TRANS after which it does a linear least square fit to the data. The subroutine then does the analytical drivation of the fitted functions and uses them to create the independant and dependant variables of the Zerbst relationship. Using these variables an optimal straight line fit is extracted and from the fitted straight line the generation lifetime and surface recombination velocity are calculated.

FIND TRANS

COM Blocks:

/PULSECT_DATA/

Pass Parameters:

ZERX(*) - (REAL) Data array of the independant variable in the Zerbst relationship generated by this subroutine.

CURVE_TO_FIT(*) - (REAL) Data array of the extracted capacitance recovery transient.

TIME_TRAN(*) - (REAL) Data array of the corresponding times for the recovery transient.

DATA_POINTS - (INTEGER) No. of points in extracted data array.

ERROR_FLAG - (INTEGER) Switch indicating processing error in subroutine.

Function:

This subroutine takes the raw data as it exists in the data arrays in the COM Block /PULSECT_DATA/ and extracts the recovery transient. The subroutine finds

the beginning of the recovery transient by marking the end of the down transient. The end of transient is defined as the capacitance which has recovered to 97% of its final value. The transient is moved to the data array CURVE_TO_FIT and the Zerbst independant variable is calculated and saved in ZERX(*).

DERIV

Pass Parameters:

X - (REAL) Independant variable value.

DERIV(*) - (REAL) Value of evaluated derivatives calulated by subroutine.

Function:

This subroutines evaluates the programmed derivvatives using given X value.

LINE FIT

Pass Parameters:

ZERX(*) - (REAL) Independant variable of curve to be fitted.

ZERY(*) - (REAL) Dependant variable of curve to be fitted.

DATA_POINTS - (INTEGER) No. of data points in curve to fit.

FIT_SLOPE - (REAL) Extracted slope of optimally fitted line.

FIT_INTERCEPT - (REAL) Extracted y intercept of optimally fitted straight line.

Function:

This subroutine does a optimal straight line fit to the data in the ZERX(*) and ZERY(*) arrays. This is done by taking segments of the arrays and doing a least square error straight line fit to the segment. The optimal fit is chosen by finding the minimum residual varience of the fit.

ZERBST PLOT

Pass Parameters:

ZERX(*) - (REAL) x variable in Zerbst plot.

ZERY(*) - (REAL) y variable in Zerbst plot.

DATA_POINTS - (INTEGER) Number of points in data arrays.

Function:

This subroutine creates the plot of the final data created in the Zerbst analysis. The x variables is C_f/C - 1 and the y variable is the time derivative of (C_o/C) squared.

```
10    !*************************************************************************
20    !        PULSED CAPACITANCE-VOLTAGE MEASUREMENTS PROGRAM
30    !*************************************************************************
40    !
50    !      DECLARATIONS
60    !
61    COM /Pulsect_data/ Notes$[30]
70    COM /Pulsect_data/ Gendate$[11],Pacerate$[1],Digpulse$[4],Doping_type$[1]
80    COM /Pulsect_data/ INTEGER Setup_flag,Data_flag,Theo_exp,Curve_fit
90    COM /Pulsect_data/ REAL Capacitance(1000),Readtime(1000)
100   COM /Pulsect_data/ Function_coef(10),Fit_slope,Fit_intercept
110   COM /Pulsect_data/ Doping_dens,Thick_oxide,Cins,C0,Cf,Ks,Kin,Ni,Area,Delta
120   COM /Pulsect_data/ Lifetime,Surface_vel
130   !
140   DIM Blanks$[60]
150   Blanks$=" "
160   !
170   !      DEFINE KEY FUNCTIONS
180   !
190   ON KEY 0 LABEL "HELP",1 GOSUB Dummy
200   ON KEY 1 LABEL "SETUP",7 CALL Setup
210   ON KEY 2 LABEL "MEASURE",6 CALL Measure
220   ON KEY 3 LABEL "PULSE",10 GOSUB Dummy
230   ON KEY 4 LABEL "CTPLOT",3 CALL Ct_plot
240   ON KEY 5 LABEL "ZERBST",4 CALL Zerbst
250   ON KEY 6 LABEL "THEORY",5 GOSUB Dummy
260   ON KEY 7 LABEL "LOAD",8 CALL Load_data
270   ON KEY 8 LABEL "STORE",3 CALL Store_data
280   ON KEY 9 LABEL "EXIT",2 GOTO Ending
290   !
300   !     SET MACHINE IN IDLE LOOP WHILE WAITING FOR KEY INTERRUPTS
310   !
320   WHILE 1=1
330      DISP FNTime$(TIMEDATE)&Blanks$&FNDate$(TIMEDATE)
340   END WHILE
350   !
360   !     DUMMY SUBROUTINE
370 Dummy:   !
380   RETURN
390   !
400   !     EXIT PROGRAM
410   !
420 Ending:  !
430   END!
```

```
440   SUB Measure
450!
460!  SUBROUTINE MEASURE TAKES DATA FROM PULSED CAPACITANCE-VOLTAGE EXPERIMENT
470!
480!
490!       DECLARATIONS
500 !
501     COM /Pulsect_data/ Notes$[30]
510     COM /Pulsect_data/ Gendate$[11],Pacerate$[1],Digpulse$[4],Doping_type$[1]
520     COM /Pulsect_data/ INTEGER Setup_flag,Data_flag,Theo_exp,Curve_fit
530     COM /Pulsect_data/ REAL Capacitance(1000),Readtime(1000)
540     COM /Pulsect_data/ Function_coef(10),Fit_slope,Fit_intercept
550     COM /Pulsect_data/ Doping_dens,Thick_oxide,Cins,C0,Cf,Ks,Kin,Ni,Area,Delt
a
560     COM /Pulsect_data/ Lifetime,Surface_vel
570   !
580      INTEGER Done,Readings(1000)
590      DIM Blanks$[60]
600      Blanks$="                                                            "
610!
620   !   CHECK TO SEE THAT SET UP IS COMPLETED FOR EXPERIMENT
630   !
640   IF Setup_flag=0 THEN
650      OUTPUT 1;CHR$(12)
660      BEEP
670      OUTPUT 1;"ERROR: EXPERIMENT HAS NOT COMPLETED SETUP"
680      OUTPUT 1;"PRESS ""SET UP"" BEFORE MEASURE"
690      GOTO Endsub
700   ELSE
710   !
720   !   INITIALIZATION
730   !
740      OUTPUT 1;CHR$(12)
750      OUTPUT 1;"***************** M E A S U R E M E N T   P H A S E **********
**************"
760      OUTPUT 1;CHR$(10);CHR$(10);"PRESS ""PULSE"" KEY TO APPLY VOLTAGE PULSE T
O DEVICE"
770      J=0
780      Done=0
790      ASSIGN @Hpib TO 7
800      SEND @Hpib;UNL LISTEN 6 MTA DATA "H2L"&Pacerate$  ! RESET A/D CONVERTOR
810      ON KEY 3 LABEL "PULSE",10 GOSUB Pulse             ! ENABLE PULSE KEY
820   !
830   !   BEGIN TAKING READINGS FROM A/D CONVERTOR
840   !
850      ON INTR 7,13 GOSUB Reading
860      SEND @Hpib;UNL LISTEN 6 MTA DATA "J"              ! START A/D CONVERTOR
870      SEND @Hpib;CMD "?5F"                              ! UNL TALK 6 MLA
880      ENABLE INTR 7;2
890   !
900   !   PLACE MACHINE IN IDLE LOOP WHILE WAITING FOR DATA FROM A/D
910   !
920      WHILE Done=0
930         DISP FNTime$(TIMEDATE)
940      END WHILE
950   !
960   !   EXPERIMENT COMPLETE SET D/A TO ZERO
970   !
980      SEND @Hpib;UNL LISTEN 7 MTA DATA "2000"
```

```
990  !
1000 !   CONVERT READINGS TO CAPACITANCE AND TIME
1010 !
1020    FOR I=1 TO 1000
1030       Capacitance(I)=Readings(I)/4
1040       SELECT Pacerate$
1050          CASE "B"
1060             Readtime(I)=(I*5.E-3)-(5.E-3)
1070          CASE "C"
1080             Readtime(I)=(I*1.0E-2)-(1.0E-2)
1090          CASE "D"
1100             Readtime(I)=(I*2.0E-2)-(2.0E-2)
1110          CASE "E"
1120             Readtime(I)=(I*5.0E-2)-(5.0E-2)
1130          CASE "F"
1140             Readtime(I)=(I*1.00E-1)-(1.00E-1)
1150          CASE "G"
1160             Readtime(I)=(I*2.00E-1)-(2.00E-1)
1170       END SELECT
1180    NEXT I
1190 !
1200 !      END OF PULSED C-T EXPERIMENTAL READINGS
1210 !
1220    Data_flag=1
1230    Theo_exp=1
1240    Curve_fit=0
1250    FOR I=1 TO 10
1260       Function_coef(I)=0
1270    NEXT I
1280    Fit_slope=0
1290    Fit_intercept=0
1300    Genda$=FNDate$(TIMEDATE)
1310    OUTPUT 1;CHR$(12)
1320    GOTO Endsub
1330 END IF
1340 !
1350 !*************************************************************************
1360 !    SERVICE SUBROUTINES FOR INTERRUPT EVENTS
1370 !*************************************************************************
1380 !
1390 !      PULSE: APPLY VOLTAGE PULSE TO DEVICE
1400 !
1410 Pulse: !
1420   DISABLE
1430   SEND @Hpib;CMD "?U'" DATA Digpulse$              ! UNL MTA LISTEN 7 DATA
1440   SEND @Hpib;CMD "?5F"                             ! UNL TALK 6 MLA
1450   ENABLE
1460   RETURN
1470 !
1480 !      READING: TAKE READING FROM A/D CONVERTOR
1490 !
1500 Reading: !
1510   J=J+1
1520   ENTER @Hpib USING "#,W";Readings(J)
1530   IF J<1000 THEN
1540      ENABLE INTR 7;2
1550   ELSE
1560      SEND @Hpib;UNL LISTEN 6 MTA DATA "I"
1570      Done=1
1580   END IF
1590   RETURN
```

```
1600!
1610!      END OF SUBROUTINE MEASURE
1620!
1630 Endsub: !
1640  SUBEND!
```

```
1650  SUB Setup
1660  !
1670  !*************************************************************************
1680  !  SUBROUTINE SETUP GETS PARAMETERS FROM USER TO RUN EXPERIMENT
1690  !*************************************************************************
1700  !
1710  !   DECLARATIONS
1720  !
1721  COM /Pulsect_data/ Notes$[30]
1730  COM /Pulsect_data/ Gendate$[11],Pacerate$[1],Digpulse$[4],Doping_type$[1]
1740  COM /Pulsect_data/ INTEGER Setup_flag,Data_flag,Theo_exp,Curve_fit
1750  COM /Pulsect_data/ REAL Capacitance(1000),Readtime(1000)
1760  COM /Pulsect_data/ Function_coef(10),Fit_slope,Fit_intercept
1770  COM /Pulsect_data/ Doping_dens,Thick_oxide,Cins,C0,Cf,Ks,Kin,Ni,Area,Delta
1780  COM /Pulsect_data/ Lifetime,Surface_vel
1790  !
1800  REAL Pulse_voltage
1810  !
1820  !   SANITY CHECK ON USER
1830  !
1840  OUTPUT 1;CHR$(12)
1850  IF Data_flag=1 THEN
1860     BEEP
1870     PRINT "WARNING: INTERNAL DATA FILE ABOUT TO BE RE-INITIALIZED, PRESS ""
STORE"" TO SAVE DATA ON DISK"
1880     WAIT 10
1890  END IF
1900  !
1910  !
1920  OUTPUT 1;CHR$(12)
1930  OUTPUT 1;"************************  S E T - U P   P H A S E  *************
**************"
1940  Gendate$=" "
1950  Pacerate$=" "
1960  Digpulse$=" "
1970  Doping_type$=" "
1980  Setup_flag=0
1990  Data_flag=0
2000  Theo_exp=0
2010  Curve_fit=0
2020  FOR I=1 TO 1000
2030     Capacitance(I)=0
2040     Readtime(I)=0
2050  NEXT I
2060  FOR I=1 TO 10
2070     Function_coef(I)=0
2080  NEXT I
2090  Fit_slope=0
2100  Fit_intercept=0
2110  Doping_dens=0
2120  Thick_oxide=0
2130  Cins=0
2140  C0=0
2150  Cf=0
2160  Ks=0
2170  Kin=0
2180  Ni=0
2190  Area=0
2200  Delta=0
2210  Lifetime=0
```

```
2220  Surface_vel=0
2230  !
2240  !     INPUT PACING RATE CHARACTER TO BE OUTPUT TO A/D CONVERTOR
2250  !
2260  OUTPUT 1;"*** EXPERIMENT PARAMETERS ***"
2270  OUTPUT 1;CHR$(10)
2280 Get_pace:   !
2290  OUTPUT 1;" ",
2300  OUTPUT 1;"ENTER PACING RATE CHARACTER FOR A/D CONVERTOR(B-G):"
2310  INPUT Pacerate$
2320  !
2330  IF Pacerate$<"B" OR Pacerate$>"G" THEN
2340     OUTPUT 1;"INVALID PACERATE ENTRY, MUST BE ONE OF LETTERS B,C,D,E,F, OR
G"
2350     GOTO Get_pace
2360  END IF
2370  !
2380  !  RETRIEVE THE TIME INTERVAL OF MEASUREMENT FROM THE PACERATE
2390  !
2400  SELECT Pacerate$
2410     CASE "B"
2420        Delta=5.0E-3
2430     CASE "C"
2440        Delta=1.00E-2
2450     CASE "D"
2460        Delta=2.0E-2
2470     CASE "E"
2480        Delta=5.0E-2
2490     CASE "F"
2500        Delta=1.00E-1
2510     CASE "G"
2520        Delta=2.00E-1
2530  END SELECT
2540  !
2550  !    GET VOLTAGE PULSE HEIGHT AND CHANGE TO DIGITAL NUMBER
2560  !        (ASSUME FULL SCALE OF D/A SET TO 10 VOLTS)
2570  !
2580 Get_volt:   !
2590  OUTPUT 1;" "
2600  OUTPUT 1;"ENTER VOLTAGE PULSE HEIGHT:"
2610  INPUT Pulse_voltage
2620  !
2630  IF Pulse_voltage>0 AND Pulse_voltage<10 THEN
2640     Digpulse=Pulse_voltage*1000/10
2650  ELSE
2660     PRINT "INVALID VOLTAGE ENTRY, MUST BE POSITIVE LESS THAN 40"
2670     GOTO Get_volt
2680  END IF
2690  Digpulse$="2"&VAL$(Digpulse)
2700  !
2710  !    GET D.U.T. PARAMETERS
2720  !
2721  OUTPUT 1
2722  OUTPUT 1
2730  OUTPUT 1;"*** DEVICE PARAMETERS ***"
2731  !
2732  OUTPUT 1;"ENTER DOPING TYPE OF DEVICE UNDER TEST(N OR P):"
2733  INPUT Doping_type$
2734  !
2740  OUTPUT 1;"ENTER DOPING DENSITY OF SILICON(CM^-3):"
```

```
2750  INPUT Doping_den
2760  !
2770  OUTPUT 1;"ENTER INTRINSIC CARRIER CONC(CM^-3):"
2780  INPUT Ni
2790  !
2800  OUTPUT 1;"ENTER DIELECTRIC CONSTANT OF SILICON:"
2810  INPUT Ks
2820  !
2830  OUTPUT 1;"ENTER DIELECTRIC CONSTANT OF INSULATOR:"
2840  INPUT Kin
2850  !
2860  OUTPUT 1;"ENTER AREA OF CAPACITOR(ANGSTROMS^2):"
2870  INPUT Area
2871  !
2872  OUTPUT 1;"ENTER COMMENTS:"
2873  INPUT Notes$
2880  !
2890  !    SET FLAG TO INDICATE THAT SETUP PHASE HAS BEEN COMPLETED
2900  !
2910  OUTPUT 1;CHR$(12)
2920  Setup_flag=1
2930  !
2940  SUBEND!
```

```
2950  SUB Ct_plot
2960  !
2970  !*************************************************************************
2980  !         SUBROUTINE CT_PLOT DRAW CAPACITANCE VS. TIME PLOT
2990  !*************************************************************************
3000  !
3010  !       DECLARATIONS
3020  !
3021  COM /Pulsect_data/ Notes$[30]
3030  COM /Pulsect_data/ Gendate$[11],Pacerate$[1],Digpulse$[4],Doping_type$[1]
3040  COM /Pulsect_data/ INTEGER Setup_flag,Data_flag,Theo_exp,Curve_fit
3050  COM /Pulsect_data/ REAL Capacitance(1000),Readtime(1000)
3060  COM /Pulsect_data/ Function_coef(10),Fit_slope,Fit_intercept
3070  COM /Pulsect_data/ Doping_dens,Thick_oxide,Cins,C0,Cf,Ks,Kin,Ni,Area,Delta
3080  COM /Pulsect_data/ Lifetime,Surface_vel
3090  !
3100  COM /Axis_flag/ Axes_drawn
3110  !
3120  DIM Label$[30]
3130  !
3140  !       CLEAR DISPLAY SCREEN
3150  !
3160  OUTPUT 1;CHR$(12)
3170  !
3180  IF Data_flag=0 THEN
3190     BEEP
3200     OUTPUT 1;"ERROR:  NO DATA AVAILABLE TO PLOT"
3210     GOTO Endsub
3220  END IF
3230  !
3240  !       SEE IF AXES ARE MULTIPLE PLOT AXES
3250  !
3260  IF Axes_drawn=1 THEN
3270     OUTPUT 1;"SHOULD DATA BE PLOTTED ON SAME AXIS (ANSWER YES OR NO):"
3280     INPUT Answer$
3290     IF Answer$<>"YES" THEN Axes_drawn=0
3300  END IF
3310  !
3320  !       BEGIN DRAWING PLOT
3330  !
3340  OUTPUT 1;CHR$(12)
3350  OUTPUT 1;"*************** CAPACITANCE VS. TIME PLOT BEING GENERATED ******
*****"
3360  !
3370  !      INITIALIZE GRAPHICS DISPLAY
3380  !
3390  DEG
3400  CLIP OFF
3410  PIVOT 0
3420  PEN 1
3430  LINE TYPE 1,5
3440  LORG 1
3450  CSIZE 5,.6
3460  LDIR 0
3470  MOVE 0,0
3480  !
3490  !     DRAW AXES IF NEEDED
3500  !
3510  IF Axes_drawn=0 THEN
```

```
3520      GCLEAR
3530      !
3540      !   DEFINE GRAPHICS UNITS
3550      !
3560      Fullheight=Capacitance(1)/.75
3570      Fullwidth=Readtime(1000)/.85
3580      Left=-.075*Fullwidth
3590      Right=Left+Fullwidth
3600      Bottom=-.075*Fullheight
3610      Top=Bottom+Fullheight
3620      WINDOW Left,Right,Bottom,Top
3630      !
3640      !   TITLE GRAPH
3650      !
3660      LORG 6
3670      CSIZE 6,.6
3680      MOVE Left+Fullwidth/2,Top
3690      LABEL "CAPACITANCE VS. TIME"
3700      !
3710      !   DATE GRAPH
3720      !
3730      LORG 9
3740      CSIZE 2,.6
3750      MOVE Right,Top
3760      LABEL "GRAPH DATE:"&FNDate$(TIMEDATE)
3770      !
3780      MOVE Right,Top-.02*(Top-Bottom)
3790      LABEL "MEAS. DATE:"&Gendate$
3800      !
3810      !   LABEL AXES
3820      !
3830      LORG 4
3840      LDIR 0
3850      CSIZE 5,.6
3860      MOVE Left+Fullwidth/2,Bottom
3870      LABEL "TIME(SECS)"
3880      !
3890      LORG 6
3900      LDIR 90
3910      MOVE Left,Bottom+Fullheight/2
3920      LABEL "CAPACITANCE (PF)"
3930      !
3940      !    PLACE KEY ON GRAPH
3950      !
3960      LORG 2
3970      LDIR 0
3980      CSIZE 3,.6
3990      !
4000      MOVE Left+.70*(Right-Left),Bottom+.80*(Top-Bottom)
4010      LABEL Notes$
4020      !
4030      MOVE Left+.70*(Right-Left),Bottom+.76*(Top-Bottom)
4040      LABEL Doping_type$&"-type Silicon"
4050      !
4060      OUTPUT Label$ USING "K,D.DDESZZ,K";"Nd=",Doping_dens," cm^-3"
4070      MOVE Left+.70*(Right-Left),Bottom+.72*(Top-Bottom)
4080      LABEL Label$
4090      !
4140      OUTPUT Label$ USING "K,D.DDESZZ,K";"Area=",Area," cm^2"
4150      MOVE Left+.70*(Right-Left),Bottom+.68*(Top-Bottom)
4160      LABEL Label$
```

```
4170      !
4180      !    NUMBER AXES
4190      !
4200      LORG 6
4210      LDIR 0
4220      CSIZE 3,.6
4230      FOR I=1 TO 1000 STEP 100
4240         MOVE Readtime(I),0
4250         LABEL Readtime(I)
4260      NEXT I
4270      !
4280      LORG 8
4290      CSIZE 2,.6
4300      Top=(INT(Capacitance(1)/88)+1)*100
4310      FOR I=0 TO Top STEP Top/10
4320         IF Capacitance(1)/.88>I THEN
4330            MOVE 0,I
4340            LABEL I
4350         END IF
4360      NEXT I
4370      !
4380      !    DRAW AXES
4390      !
4400      CLIP 0,Readtime(1000),0,Capacitance(1)/.88
4410      AXES Readtime(101),Top/10,0,0,2,2
4420      CLIP OFF
4430      Axes_drawn=1
4440   END IF
4450   !
4460   !  **********  PLOT DATA POINTS ************
4470   !
4480   CLIP 0,Readtime(1000),0,Capacitance(1)/.88
4490   PENUP
4500   MOVE Readtime(1),Capacitance(1)
4510   FOR I=1 TO 1000
4520      DRAW Readtime(I),Capacitance(I)
4530   NEXT I
4540   PENUP
4550   CLIP OFF
4560   OUTPUT 1;CHR$(12)
4570   !
4580 Endsub: !
4590   SUBEND!
```

```
4600  SUB Gauss_elim(INTEGER N,REAL A(*),X(*))
4610  !
4620  !*************************************************************************
4630  !   SUBROUTINE GAUSS_ELIM SOLVES A SET OF N SIMULTANEOUS EQUATIONS
4640  !              BY THE METHOD OF GAUSSIAN ELIMINATION
4650  !*************************************************************************
4660  !
4670  !    DECLARATIONS
4680  !
4690  REAL Matrix(10,11),Biggest,Temp,Sum,Quot
4700  INTEGER I,J,K,Jj
4710  !
4720  !    ****** BEGIN GAUSSIAN ELIMINATION PROCESS ******
4730  !
4740  FOR K=1 TO N
4750     !
4760     !   SEARCH FOR LARGEST ELEMENT IN COLUMN
4770     !
4780     Jj=K
4790     Biggest=ABS(A(K,K))
4800     FOR I=K+1 TO N
4810        IF ABS(A(I,K))>Biggest THEN
4820           Biggest=ABS(A(I,K))
4830           Jj=I
4840        END IF
4850     NEXT I
4860     !
4870     !   ROW INTERCHANGE IF NECESSARY
4880     !
4890     IF Jj<>K THEN
4900        FOR J=K TO N+1
4910           Temp=A(Jj,J)
4920           A(Jj,J)=A(K,J)
4930           A(K,J)=Temp
4940        NEXT J
4950     END IF
4960     !
4970     !   CALCULATE NEW MATRIX
4980     !
4990     FOR I=K+1 TO N
5000        Quot=A(I,K)/A(K,K)
5010        FOR J=K+1 TO N+1
5020           A(I,J)=A(I,J)-Quot*A(K,J)
5030        NEXT J
5040     NEXT I
5050     !
5060     !   ZERO APPROPRIATE COLUMN
5070     !
5080     FOR I=K+1 TO N
5090        A(I,K)=0
5100     NEXT I
5110  NEXT K
5120  !
5130  !     BEGIN BACK SUBSTITUTION
5140  !
5150  X(N)=A(N,N+1)/A(N,N)
5160  FOR I=N-1 TO 1 STEP -1
5170     Sum=0
5180     FOR J=I+1 TO N
5190        Sum=Sum+A(I,J)*X(J)
```

```
5200      NEXT J
5210      X(I)=(A(I,N+1)-Sum)/A(I,I)
5220    NEXT I
5230    !
5240    !   EXIT
5250    !
5260    SUBEND!
```

```
5270  SUB Least_square(REAL X(*),Y(*),INTEGER Data_points,No_of_functions,Info_f
lag,REAL Coefficiants(*))
5280  !
5290  !*************************************************************************
5300  !   SUBROUTINE LEAST_SQUARE DOES LEAST SQUARE CURVE FITTING BY FORMING
5310  !      MATRIX REPRESENTING SIMULTANEOUS EQUATIONS WHICH CAN SOLVE
5320  !         FOR COEFFICIANTS OF USER-DEFINED FUNCTIONS
5330  !*************************************************************************
5340  !
5350  !   DECLARATIONS
5360  !
5370  REAL F(10),Matrix(10,11)
5380  !
5390  !   FILL MATRIX
5400  !
5410  FOR K=1 TO Data_points
5420     CALL Functions(X(K),F(*))
5430     !
5440     FOR I=1 TO No_of_functions
5450        FOR J=I TO No_of_functions
5460           Matrix(I,J)=F(I)*F(J)+Matrix(I,J)
5470           Matrix(J,I)=Matrix(I,J)
5480        NEXT J
5490        Matrix(I,No_of_functions+1)=Matrix(I,No_of_functions+1)+F(I)*Y(K)
5500     NEXT I
5510  NEXT K
5520  !
5530  !   SOLVE SYSTEM OF SIMULTANEOUS EQUATIONS
5540  !
5550  CALL Gauss_elim(No_of_functions,Matrix(*),Coefficiants(*))
5560  !
5570  !   EXIT
5580  !
5590  SUBEND!
```

```
5600   DEF FNTime$(Now)    ! GIVEN 'SECONDS' RETURN 'HH:MM:SS'
5610     !
5620     Now=INT(Now) MOD 86400
5630     H=Now DIV 3600
5640     M=Now MOD 3600 DIV 60
5650     S=Now MOD 60
5660     OUTPUT T$ USING "#,ZZ,K";H,":",M,":",S
5670     RETURN T$
5680   FNEND!


5690   !
5700   DEF FNTime(T$)    ! GIVEN 'HH:MM:SS' RETURN 'SECONDS'
5710     !
5720     ON ERROR GOTO Err
5730     ENTER T$;H,M,S
5740     RETURN (3600*H+60*M+S) MOD 86400
5750 Err:OFF ERROR
5760     RETURN TIMEDATE MOD 86400
5770   FNEND!


5780  DEF FNDate$(Seconds) !GIVEN 'SECONDS' RETURN "DD MMM YYYY"
5790    !
5800    DATA JAN,FEB.MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC
5810    DIM Month$(1:12)[3]
5820    READ Month$(*)
5830    !
5840    Julian=Seconds DIV 86400-1721119
5850    Year=(4*Julian-1) DIV 146097
5860    Julian=(4*Julian-1) MOD 146097
5870    Day=Julian DIV 4
5880    Julian=(4*Day+3) DIV 1461
5890    Day=(4*Day+3) MOD 1461
5900    Day=(Day+4) DIV 4
5910    Month=(5*Day-3) DIV 153
5920    Day=(5*Day-3) MOD 153
5930    Day=(Day+5) DIV 5
5940    Year=100*Year+Julian
5950    IF Month<10 THEN
5960       Month=Month+3
5970    ELSE
5980       Month=Month-9
5990       Year=Year+1
6000    END IF
6010    OUTPUT D$ USING "#,ZZ,X,3A,X,4Z";Day,Month$(Month),Year
6020    RETURN D$
6030  FNEND!


6040  !
6050  DEF FNDate(Dmy$)  ! GEVEN 'DD MMM YYYY' RETURN 'SECONDS'
6060    !
6070    DATA JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC
6080    DIM Month$(1:12)[3]
6090    READ Month$(*)
6100    !
6110    ON ERROR GOTO Err
6120    I$=Dmy$&"    "
```

```
6130     ENTER I$ USING "DD,4A,5D";Day,M$,Year
6140     IF Year<100 THEN Year=Year+1900
6150     FOR I=1 TO 12
6160     IF POS(M$,Month$(I)) THEN Month=I
6170     NEXT I
6180     IF Month=0 THEN Err
6190     IF Month>2 THEN
6200        Month=Month-3
6210     ELSE
6220        Month=Month+9
6230        Year=Year-1
6240     END IF
6250     Century=Year DIV 100
6260     Remainder=Year MOD 100
6270     Julian=146097*Century DIV 4+1461*Remainder DIV 4+(153*Month+2) DIV 5+Day
+1721119
6280     Julian=Julian*86400
6290     IF Julian<2.08662912E+11 OR Julian>=2.143252224E+11 THEN Err
6300     RETURN Julian      ! RETURN JULIAN DATE IN SECONDS
6310 Err:    OFF ERROR
6320     RETURN TIMEDATE    ! RETURN CURRENT DATE
6330   FNEND!


6340   SUB Functions(X,Func(*))
6350   !
6360   !*************************************************************************
6370   !  SUBROUTINE FUNCTIONS EVALUATES THE FUNCTIONS AT THE GIVEN X VALUES
6380   !*************************************************************************
6390   !
6400   !   DECLARATIONS
6410   !
6411   COM /Pulsect_data/ Notes$[30]
6420   COM /Pulsect_data/ Gendate$[11],Pacerate$[1],Digpulse$[4],Doping_type$[1]
6430   COM /Pulsect_data/ INTEGER Setup_flag,Data_flag,Theo_exp,Curve_fit
6440   COM /Pulsect_data/ REAL Capacitance(1000),Readtime(1000)
6450   COM /Pulsect_data/ Function_coef(10),Fit_slope,Fit_intercept
6460   COM /Pulsect_data/ Doping_dens,Thick_oxide,Cins,C0,Cf,Ks,Kin,Ni,Area,Delta
6470   COM /Pulsect_data/ Lifetime,Surface_vel
6480   !
6490   !   EVALUATE FUNCTIONS
6500   !
6510   Func(1)=1.0
6520   Func(2)=X^.125
6530   Func(3)=X^.25
6540   Func(4)=X^.5
6550   Func(5)=X^.75
6560   Func(6)=X
6570   Func(7)=X^2
6580   Func(8)=(5.0*Delta)/(X+.1*Delta)
6590   Func(9)=(10.0*Delta)/(X+.3*Delta)
6600   !
6610   !    EXIT FUNCTION EVALUATING SUBROUTINE
6620   !
6630   SUBEND!
```

```
6640  SUB Store_data
6650  !
6660  !*************************************************************************
6670  !  SUBROUTINE STORE_DATA STORES THE INTERNAL DATA FILE INTO A DISK
6680  !*************************************************************************
6690  !
6700  !   DECLARATIONS
6710  !
6711  COM /Pulsect_data/ Notes$[30]
6720  COM /Pulsect_data/ Gendate$[11],Pacerate$[1],Digpulse$[4],Doping_type$[1]
6730  COM /Pulsect_data/ INTEGER Setup_flag,Data_flag,Theo_exp,Curve_fit
6740  COM /Pulsect_data/ REAL Capacitance(1000),Readtime(1000)
6750  COM /Pulsect_data/ Function_coef(10),Fit_slope,Fit_intercept
6760  COM /Pulsect_data/ Doping_dens,Thick_oxide,Cins,C0,Cf,Ks,Kin,Ni,Area,Delta
6770  COM /Pulsect_data/ Lifetime,Surface_vel
6780  !
6790  DIM File_specifier$[30],File_name$[10],Answer$[1]
6800  !
6810  !   DEFINE PROGRAM STATE TO USER
6820  !
6830  OUTPUT 1;CHR$(12)
6840  OUTPUT 1;"****************** STORAGE OF INTERNAL DATA FILE ON DISK *******
************"
6850  !
6860  !   GET NAME OF FILE FROM USER
6870  !
6880  OUTPUT 1;"ENTER FILE NAME:"
6890  INPUT File_name$
6900  !
6910  !   LEFT OR RIGHT HAND INTERNAL DISK DRIVE?
6920  !
6930  OUTPUT 1;"SPECIFY WHETHER STORAGE DISK IS MOUNTED IN LEFT OR RIGHT HAND DR
IVE"
6940  OUTPUT 1;"ENTER L OR R:"
6950  INPUT Answer$
6960  IF Answer$="L" THEN
6970     File_specifier$=File_name$&":INTERNAL,4,1"
6980  ELSE
6990     File_specifier$=File_name$&":INTERNAL,4,0"
7000  END IF
7010   !
7020   !   CREATE DISK FILE
7030   !
7040   CREATE BDAT File_specifier$,6,4096
7050   ASSIGN @File TO File_specifier$;FORMAT OFF
7060   !
7070   !   PLACE COM VARIABLE IN FILE
7080   !
7081   OUTPUT @File:Notes$
7090   OUTPUT @File;Gendate$:Pacerate$:Digpulse$:Doping_type$
7100   OUTPUT @File;Setup_flag:Data_flag;Theo_exp,Curve_fit
7110   OUTPUT @File;Capacitance(*)
7120   OUTPUT @File;Readtime(*)
7130   OUTPUT @File;Function_coef(*);Fit_slope:Fit_intercept
7140   OUTPUT @File;Doping_dens;Thick_oxide;Cins;C0;Cf;Ks;Kin;Ni;Area;Delta
7150   OUTPUT @File;Lifetime;Surface_vel
7160   !
7170   !  CLOSE I/O PATH
7180   !
7190   ASSIGN @File TO *
```

```
7200    OUTPUT 1;CHR$(12)
7210    !
7220    !  EXIT SUBROUTINE STORE_DATA
7230    !
7240    SUBEND!
```

```
7250  SUB Load_data
7260  !
7270  !*************************************************************************
7280  !  SUBROUTINE LOAD_DATA RETRIEVES AN INTERNAL DATA FILE FROM DISK
7290  !*************************************************************************
7300  !
7310  !   DECLARATIONS
7320  !
7321  COM /Pulsect_data/ Notes$[30]
7330  COM /Pulsect_data/ Gendate$[11],Pacerate$[1],Digpulse$[4],Doping_type$[1]
7340  COM /Pulsect_data/ INTEGER Setup_flag,Data_flag,Theo_exp,Curve_fit
7350  COM /Pulsect_data/ REAL Capacitance(1000),Readtime(1000)
7360  COM /Pulsect_data/ Function_coef(10),Fit_slope,Fit_intercept
7370  COM /Pulsect_data/ Doping_dens,Thick_oxide,Cins,C0,Cf,Ks,Kin,Ni,Area,Delta
7380  COM /Pulsect_data/ Lifetime,Surface_vel
7390  !
7400  DIM File_specifier$[30],File_name$[10],Answer$[1]
7410  !
7420  !   DEFINE PROGRAM STATE TO USER
7430  !
7440  OUTPUT 1;CHR$(12)
7450  OUTPUT 1;"****************** LOAD OF INTERNAL DATA FILE FROM DISK ********
**********"
7460  !
7470  !   GET NAME OF FILE FROM USER
7480  !
7490  OUTPUT 1;"ENTER FILE NAME:"
7500  INPUT File_name$
7510  !
7520  !   LEFT OR RIGHT HAND INTERNAL DISK DRIVE?
7530  !
7540  OUTPUT 1;"SPECIFY WHETHER STORAGE DISK IS MOUNTED IN LEFT OR RIGHT HAND DR
IVE"
7550  OUTPUT 1:"ENTER L OR R:"
7560  INPUT Answer$
7570  IF Answer$="L" THEN
7580     File_specifier$=File_name$&":INTERNAL,4,1"
7590  ELSE
7600     File_specifier$=File_name$&":INTERNAL,4,0"
7610  END IF
7620   !
7630   !   CREATE DISK FILE
7640   !
7650   ASSIGN @File TO File_specifier$;FORMAT OFF
7660   !
7670   !   PLACE COM VARIABLE IN FILE
7680   !
7681   ENTER @File;Notes$
7690   ENTER @File;Gendate$;Pacerate$;Digpulse$;Doping_type$
7700   ENTER @File;Setup_flag;Data_flag;Theo_exp;Curve_fit
7710   ENTER @File;Capacitance(*)
7720   ENTER @File;Readtime(*)
7730   ENTER @File;Function_coef(*);Fit_slope;Fit_intercept
7740   ENTER @File;Doping_dens;Thick_oxide;Cins,C0,Cf,Ks,Kin,Ni,Area,Delta
7750   ENTER @File;Lifetime;Surface_vel
7760   !
7770   !  CLOSE I/O PATH
7780   !
7790   ASSIGN @File TO *
7800   OUTPUT 1;CHR$(12)
7810   !
```

```
7820    !  EXIT SUBROUTINE STORE_DATA
7830    !
7840    SUBEND!
```

```
7850  SUB Zerbst
7860  !
7870  !*************************************************************************
7880  !   SUBROUTINE ZERBST DOES ZERBST DATA ANALYSIS OF CAPACITANCE TRANSIENT
7890  !*************************************************************************
7900  !
7910  !   DECLARATIONS
7920  !
7921  COM /Pulsect_data/ Notes$[30]
7930  COM /Pulsect_data/ Gendate$[11],Pacerate$[1],Digpulse$[4],Doping_type$[1]
7940  COM /Pulsect_data/ INTEGER Setup_flag,Data_flag,Theo_exp,Curve_fit
7950  COM /Pulsect_data/ REAL Capacitance(1000),Readtime(1000)
7960  COM /Pulsect_data/ Function_coef(10),Fit_slope,Fit_intercept
7970  COM /Pulsect_data/ Doping_dens,Thick_oxide,Cins,C0,Cf,Ks,Kin,Ni,Area,Delta
7980  COM /Pulsect_data/ Lifetime,Surface_vel
7990  !
8000  INTEGER Data_points,No_of_functions,Error_flag,Line_fit
8010  REAL Zerx(1000),Zery(1000),Curve_to_fit(1000),Time_tran(1000),F(10)
8020  REAL Sum,Squ,Mean,Var,Yfit
8021  REAL Lifetime_cor,Sur_vel(1000)
8030  !
8040  !   INITIALIZATION
8050  !
8060  IF Curve_fit=1 THEN Line_fit=1
8070  No_of_functions=9
8080  Error_flag=0
8090  OUTPUT 1;CHR$(12)
8100  !
8110  !   SANITY CHECK
8120  !
8130  IF Data_flag=0 THEN
8140     BEEP
8150     OUTPUT 1;"ERROR--NO DATA AVAILABLE TO ANALYZE, MEASUREMENTS MUST BE TAK
EN"
8160  ELSE
8170     OUTPUT 1;"**************** DATA ANALYSIS OF PULSED CT DATA ************
*****"
8180     !
8190     !    FIND CAPACITANCE TRANSIENT
8200     !
8210     CALL Find_trans(Zerx(*),Curve_to_fit(*),Time_tran(*),Data_points,Error_
flag)
8220     IF Error_flag=0 THEN
8230     !
8240     !    FIT CURVE IF NOT ALREADY DONE
8250     !
8260        IF Curve_fit=0 THEN
8270           OUTPUT 1;"LEAST SQUARE CURVE FIT BEING PERFORMED"
8280           CALL Least_square(Time_tran(*),Curve_to_fit(*),Data_points,No_of_
functions,Error_flag,Function_coef(*))
8290        !
8300        !   LEAST SQUARE CURVE FIT REPORT
8310        !
8320           OUTPUT 708:CHR$(12)
8330           OUTPUT 708;"FUNCTION COEFFICIANTS AND RESIDUAL VARIENCE FROM CURV
E FIT"
8340           FOR I=1 TO 2
8350              OUTPUT 708;CHR$(10)
8360           NEXT I
8370           !
```

```
8380            !  OUTPUT DERIVED FUNCTION COEFFICIANTS
8390            !
8400            OUTPUT 708;"COEFFICIANTS:"
8410            FOR I=1 TO No_of_functions
8420               OUTPUT 708;"A";I;"=";Function_coef(I)
8430            NEXT I
8440            !
8450            !   CALCULATE RESIDUAL VARIENCE OF CURVE FIT
8460            !
8470            Sum=0
8480            Squ=0
8490            FOR I=1 TO Data_points
8500               Yfit=0
8510               CALL Functions(Time_tran(I),F(*))
8520               FOR J=1 TO No_of_functions
8530                  Yfit=Yfit+F(J)*Function_coef(J)
8540               NEXT J
8550               Diff=Curve_to_fit(I)-Yfit
8560               Squ=Squ+Diff^2
8570               Sum=Sum+Diff
8580            NEXT I
8590            Mean=Sum/Data_points
8600            Var=(1/(Data_points-1))*(Squ-Data_points*Mean^2)
8610            OUTPUT 708;CHR$(10)
8620            OUTPUT 708;"RESIDUAL VARIENCE=";Var
8630            !
8640            !   COMPARE SEVERAL POINTS ALONG CURVE
8650            !
8660            OUTPUT 708;CHR$(10)
8670            OUTPUT 708;CHR$(10)
8680            OUTPUT 708;"COMPARISION OF SEVERAL DATA POINTS WITH FITTED CURVE"
8690            OUTPUT 708;CHR$(10)
8700            OUTPUT 708 USING "K,10X,K,18X,K";"POINT NO.","DATA POINTS","FITTE
D POINTS"
8710            FOR I=1 TO Data_points STEP INT(Data_points/20)
8720                Yfit=0
8730                CALL Functions(Time_tran(I),F(*))
8740                FOR J=1 TO No_of_functions
8750                   Yfit=Yfit+F(J)*Function_coef(J)
8760                NEXT J
8770                OUTPUT 708 USING "3X,DDD,13X,DDDD.DDDD,20X,DDDD.DDDD";I,Curve
_to_fit(I),Yfit
8780            NEXT I
8790            !
8800            !   SET LOGIC FLAGS
8810            !
8820            Line_fit=0
8830            Curve_fit=1
8840         END IF
8850         !
8860         !   EXTRACT Y DATA FOR ZERBST PLOT
8870         !
8880         OUTPUT 1;"Y-AXIS DATA BEING EXTRACTED FOR ZERBST PLOT"
8890         !
8900         FOR I=1 TO Data_points
8910            CALL Deriv(Time_tran(I),F(*))
8920            FOR J=1 TO No_of_functions
8930               Zery(I)=Zery(I)-F(J)*Function_coef(J)
8940            NEXT J
8950         NEXT I
```

```
8960        !
8970        !   GET BEST STRAIGHT LINE FIT FOR ZERBST PLOT
8980        !
8990        IF Line_fit=0 THEN
9000           OUTPUT 1;"STRAIGHT LINE FIT BEING PERFORMED ON ZERBST PLOT"
9010           CALL Line_fit(Zerx(*),Zery(*),Data_points,Fit_slope,Fit_intercept
)
9020        END IF
9030        !
9040        !   CALCULATE LIFETIME AND SURFACE VELOCITY
9050        !
9060        IF Doping_dens<>0 AND Cf<>0 AND Fit_slope<>0 AND Ni<>0 AND Cins<>0 T
HEN
9070           Lifetime=(Ni*Cins)/(Doping_dens*Cf*Fit_slope)
9080           Surface_vel=(Fit_intercept*Doping_dens*Ks*8.85418E-14)/(2*Ni*Cins
*1.E-12)
9081           !
9082           !   CALCULATE CORRECTED VALUE OF LIFETIME (according to D.K.
9083           !   Schroder and H.C. Nathanson SCIENTIFIC PAPER 69-1F6CARTS-P2)
9084           !
9085           Lifetime_cor=1/(1/Lifetime+4*Surface_vel/((Area/PI)^.5/2))
9086           PRINT Lifetime_cor
9087           !
9088           !   CALCULATE SURFACE VELOCITY AS A FUNCTION OF TIME
9089           !
9090           FOR I=1 TO Data_points
9091              Sur_vel(I)=((Doping_dens/(2*Ni))*Zery(I)-(C0/Cf)*(Zerx(I)/Life
time_cor))*((Ks*Thick_oxide)/Kin)
9092           NEXT I
9093           !
9094           !   PLOT SURFACE VELOCITY VS. TIME CURVE
9095           !
9097           CALL Survel_plot(Readtime(*),Sur_vel(*),Data_points)
9098           PAUSE
9099        ELSE
9101           BEEP
9110           OUTPUT 1;"DATA FILE NOT COMPLETE, LIFETIME AND SURFACE VEL CANNOT
 BE CALCULATED"
9120        END IF
9130        !
9140        !   CREATE ZERBST PLOT
9150        !
9160        CALL Zerbst_plot(Zerx(*),Zery(*),Data_points)
9170     END IF
9180     OUTPUT 1;CHR$(12)
9190  END IF
9200  !
9210  !   EXIT SUBROUTINE ZERBST
9220  !
9230  SUBEND!
```

```
9240  SUB Find_trans(REAL Zerx(*),Curve_to_fit(*),Time_tran(*),INTEGER Data_poin
ts,Error_flag)
9250  !
9260  !*************************************************************************
9270  !    SUBROUTINE FIND_TRAN EXTRACT THE RECOVERING CAPACITANCE TRANSIENT
9280  !                  FROM THE MEASURED DATA
9290  !*************************************************************************
9300  !
9310  !   DECLARATION
9320  !
9321  COM /Pulsect_data/ Notes$[30]
9330  COM /Pulsect_data/ Gendate$[11],Pacerate$[1],Digpulse$[4],Doping_type$[1]
9340  COM /Pulsect_data/ INTEGER Setup_flag,Data_flag,Theo_exp,Curve_fit
9350  COM /Pulsect_data/ REAL Capacitance(1000),Readtime(1000)
9360  COM /Pulsect_data/ Function_coef(10),Fit_slope,Fit_intercept
9370  COM /Pulsect_data/ Doping_dens,Thick_oxide,Cins,C0,Cf,Ks,Kin,Ni,Area,Delta
9380  COM /Pulsect_data/ Lifetime,Surface_vel
9390  !
9400  INTEGER Done,I,Start,Finish,Index
9410  !
9420  !    FIND INSULATOR AND FINAL CAPACITANCE AND INSULATOR THICKNESS
9430  !
9440  Cins=183.2
9450  Cf=Capacitance(1000)
9460  Thick_oxide=(Kin*8.85418E-14*Area)/(Cins*1.E-12)
9470  !
9480  !    FIND START OF TRANSIENT
9490  !
9500  OUTPUT 1;"CAPACITANCE TRANSIENT BEING EXTRACTED FROM DATA"
9510  !
9520  Done=0
9530  I=1
9540  WHILE Done=0
9550     IF Capacitance(I)<.90*Capacitance(1) THEN
9560     !
9570     !    THROW OUT DOWN TRANSIENT AS EXPERIMENTAL ERROR
9580     !
9590        WHILE Capacitance(I+1)<Capacitance(I)
9600           I=I+1
9610        END WHILE
9620        Done=1
9630     END IF
9640     !
9650     !   INCREMENT COUNTER IF NOT DONE
9660     !
9670     IF Done=0 THEN I=I+1
9680     IF I=1000 THEN
9690        BEEP
9700        OUTPUT 1;"ERROR--NO PULSE TRANSIENT FOUND IN DATA"
9710        Error_flag=1
9720        Done=1
9730     END IF
9740  END WHILE
9750  !
9760  Start=I
9770  !
9780  !    FIND END OF CAPACITANCE TRANSIENT
9790  !
9800  Done=0
```

```
9810  I=Start
9820  IF Error_flag=0 THEN
9830     WHILE Done=0
9840        IF Capacitance(I)>.97*Cf THEN
9850           Done=1
9860        END IF
9870        IF Done=0 THEN I=I+1
9880     END WHILE
9890     !
9900     !    MARK END OF TRANSIENT AND MOVE TRANSIENT POINTS INTO TRAN ARRAYS
9910     !
9920     Finish=I
9930     Data_points=Finish-Start+1
9940     FOR I=Start TO Finish
9950        Index=I-Start+1
9960        Zerx(Index)=(Cf/Capacitance(I)-1)
9970        Curve_to_fit(Index)=(Cins/Capacitance(I))^2
9980        Time_tran(Index)=Readtime(I)-Readtime(Start)
9990     NEXT I
10000 END IF
10010 !
10020 !   EXIT FIND_TRAN SUBROUTINE
10030 !
10040 SUBEND!
```

```
10050 SUB Deriv(X,Deriv(*))
10060 !
10070 !*************************************************************************
10080 !  SUBROUTINE DERIV CALCULATES THE DERIVATIVE OF THE FUNCS IN SUB FUNCS
10090 !*************************************************************************
10100 !
10110 !    DECLARATIONS
10120 !
10121 COM /Pulsect_data/ Notes$[30]
10130 COM /Pulsect_data/ Gendate$[11],Pacerate$[1],Digpulse$[4],Doping_type$[1]
10140 COM /Pulsect_data/ INTEGER Setup_flag,Data_flag,Theo_exp,Curve_fit
10150 COM /Pulsect_data/ REAL Capacitance(1000),Readtime(1000)
10160 COM /Pulsect_data/ Function_coef(10),Fit_slope,Fit_intercept
10170 COM /Pulsect_data/ Doping_dens,Thick_oxide,Cins,C0,Cf,Ks,Kin,Ni,Area,Delta
10180 COM /Pulsect_data/ Lifetime,Surface_vel
10190 !
10200 !    EVALUATE DERIVATIVES
10210 !
10220 Deriv(1)=0
10230 IF X<>0 THEN
10240    Deriv(2)=.125*X^(-.875)
10250    Deriv(3)=.25*X^(-.75)
10260    Deriv(4)=.5*X^(-.5)
10270    Deriv(5)=.75*X^(-.25)
10280 ELSE
10290    Deriv(2)=0
10300    Deriv(3)=0
10310    Deriv(4)=0
10320    Deriv(5)=0
10330 END IF
10340 Deriv(6)=1
10350 Deriv(7)=2*X
10360 Deriv(8)=(5.0*Delta)/(X+.1*Delta)^2
10370 Deriv(9)=(10.0*Delta)/(X+.3*Delta)^2
10380 !
10390 !   EXIT DERIVATIVE EVALUATING SUBROUTINE
10400 !
10410 SUBEND!
```

```
10420 SUB Line_fit(REAL Zerx(*),Zery(*),INTEGER Data_points,REAL Fit_slope,Fit_i
ntercept)
10430 !
10440 !*************************************************************************
10450 !  SUBROUTINE LINE_FIT FITS A STRAIGHT LINE TO THE MOST LINEAR PART
10460 !                        OF THE ZERBST PLOT
10470 !*************************************************************************
10480 !
10490 !      DECLARATIONS
10500 !
10510 INTEGER Points_to_fit,Jump_interval,I,J
10520 REAL Xj,Yj,Xj2,Xjyj
10530 REAL Test_slope,Test_intercept,Squ,Sum,Yfit,Diff,Mean,Test_var,Low_var
10540 !
10550 !      INITIALIZATION
10560 !
10570 Low_var=1.E+36
10580 Points_to_fit=INT(Data_points/2)
10590 Jump_interval=INT(Data_points/24)
10600 !
10610 !      FORMAT REPORT
10620 !
10630 OUTPUT 708;CHR$(12)
10640 OUTPUT 708;"REPORT FROM LEAST SQUARE LINE FIT TO CURVE"
10650 OUTPUT 708;CHR$(10);CHR$(10);"TEST LINES:"
10660 OUTPUT 708 USING "3X,K,10X,K,10X,K";"SLOPE","INTERCEPT","RESIDUAL VAR"
10670 !
10680 !      SEARCH ALONG CURVE FOR GOOD STRAIGHT LINE FIT
10690 !
10700 FOR I=1 TO Data_points-Points_to_fit STEP Jump_interval
10710    !
10720    !      SUM THE MATRIX ELEMENTS NEEDED FOR A LEAST SQUARE FIT
10730    !
10740    Xj=0
10750    Yj=0
10760    Xj2=0
10770    Xjyj=0
10780    FOR J=I TO I+Points_to_fit
10790       Xj=Xj+Zerx(J)
10800       Yj=Yj+Zery(J)
10810       Xj2=Xj2+Zerx(J)^2
10820       Xjyj=Xjyj+Zerx(J)*Zery(J)
10830    NEXT J
10840    !
10850    !     DETERMINE TEST FIT SLOPE AND FIT INTERCEPT
10860    !
10870    Test_slope=(Yj*Xj-Points_to_fit*Xjyj)/(Xj^2-Points_to_fit*Xj2)
10880    Test_intercept=(Yj*Xj2-Xjyj*Xj)/(Points_to_fit*Xj2-Xj^2)
10890    !
10900    !     FIND RESIDUAL VARIENCE OF FITTED LINE
10910    !
10920    Squ=0
10930    Sum=0
10940    FOR J=I TO I+Points_to_fit
10950       Yfit=Test_slope*Zerx(J)+Test_intercept
10960       Diff=Zery(J)-Yfit
10970       Squ=Squ+Diff^2
10980       Sum=Sum+Diff
10990    NEXT J
```

```
11000     !
11010     Mean=Sum/Points_to_fit
11020     Test_var=(Squ-Points_to_fit*Mean^2)/(Points_to_fit-1)
11030     !
11040     !    OUTPUT PARAMETERS FROM FITTED LINE TO REPORT
11050     !
11060     IF Test_var<1.E+6 THEN
11070        OUTPUT 708 USING "DDDD.DDDD,10X,DDDD.DDDD,8X,DDDDDD.DDDDDD";Test_slo
pe,Test_intercept,Test_var
11080     END IF
11090     !
11100     !    TEST TO SEE IF OPTIMAL STRAIGHT LINE FIT ALONG CURVE
11110     !
11120     IF Test_var<Low_var THEN
11130        Low_var=Test_var
11140        Fit_intercept=Test_intercept
11150        Fit_slope=Test_slope
11160     END IF
11170     !
11180     !    MOVE ON TO NEXT CURVE SEGMENT
11190     !
11200  NEXT I
11210  !
11220  !   REPORT RESULTS OF LINE FIT
11230  !
11240  OUTPUT 708;CHR$(10)
11250  OUTPUT 708 USING "K,DDDD.DDDD";"FITTED SLOPE:";Fit_slope
11260  OUTPUT 708 USING "K,DDDD.DDDD";"FITTED INTER:";Fit_intercept
11270  !
11280  !   EXIT SUBROUTINE LINE_FIT
11290  !
11300  SUBEND!
```

```
11310 SUB Zerbst_plot(REAL Zerx(*),Zery(*),INTEGER Data_points)
11320 !
11330 !*************************************************************************
11340 !   SUBROUTINE ZERBST PLOT CREATES PLOT FROM EXTRACTED DATA
11350 !*************************************************************************
11360 !
11370 !   DECLARATIONS
11380 !
11381 COM /Pulsect_data/ Notes$[30]
11390 COM /Pulsect_data/ Gendate$[11],Pacerate$[1],Digpulse$[4],Doping_type$[1]
11400 COM /Pulsect_data/ INTEGER Setup_flag,Data_flag,Theo_exp,Curve_fit
11410 COM /Pulsect_data/ REAL Capacitance(1000),Readtime(1000)
11420 COM /Pulsect_data/ Function_coef(10),Fit_slope,Fit_intercept
11430 COM /Pulsect_data/ Doping_dens,Thick_oxide,Cins,C0,Cf,Ks,Kin,Ni,Area,Delta
11440 COM /Pulsect_data/ Lifetime,Surface_vel
11450 !
11460 DIM Label$[30]
11470 INTEGER High_x,High_y
11480 REAL Left,Right,Bottom,Top
11490 !
11500 !   RE-INITIALIZE GRAPHICS DISPLAY
11510 !
11520 DEG
11530 CLIP OFF
11540 PIVOT 0
11550 PEN 1
11560 LINE TYPE 1,5
11570 LORG 1
11580 CSIZE 5,.6
11590 LDIR 0
11600 MOVE 0,0
11610 GCLEAR
11620 !
11630 !   DEFINE GRAPHICS UNITS
11640 !
11650 High_x=INT(Zerx(INT(Data_points/10))+1)
11660 High_y=INT(Zery(INT(Data_points/10))+1)
11670 Left=-.10*High_x
11680 Right=1.1*High_x
11690 Bottom=-.10*High_y
11700 Top=1.1*High_y
11710 WINDOW Left,Right,Bottom,Top
11720 !
11730 !    TITLE GRAPH
11740 !
11750 LORG 6
11760 CSIZE 7,.6
11770 MOVE (Left+Right)/2,Top
11780 LABEL "ZERBST PLOT"
11790 !
11800 !    DATE GRAPH
11810 !
11820 LORG 9
11830 CSIZE 2,.6
11840 MOVE Right,Top
11850 LABEL "GRAPH DATE:"&FNDate$(TIMEDATE)
11860 !
11870 MOVE Right,Top-.02*(Top-Bottom)
11880 LABEL "MEAS. DATE:"&Gendate$
11890 !
```

```
11900 !    CREATE GRAPH KEY
11910 !
11920 LORG 2
11930 CSIZE 3,.6
11940 LDIR 0
11950 !
11960 MOVE Left+.60*(Right-Left),Bottom+.32*(Top-Bottom)
11970 LABEL Notes$
11980 !
11990 MOVE Left+.60*(Right-Left),Bottom+.28*(Top-Bottom)
12000 LABEL Doping_type$&"-type Silicon"
12010 !
12020 OUTPUT Label$ USING "K,DD.DDESZZ,K";"LIFETIME=",Lifetime," sec"
12030 MOVE Left+.60*(Right-Left),Bottom+.24*(Top-Bottom)
12040 LABEL Label$
12050 !
12060 OUTPUT Label$ USING "K,DD.DDESZZ,K";"SURF VEL=",Surface_vel," cm/sec"
12070 MOVE Left+.60*(Right-Left),Bottom+.20*(Top-Bottom)
12080 LABEL Label$
12090 !
12100 !    LABEL AXES
12110 !
12120 LORG 4
12130 LDIR 0
12140 CSIZE 5,.6
12150 MOVE (Left+Right)/2,Bottom
12160 LABEL "Cf/C - 1"
12170 !
12180 LORG 6
12190 LDIR 90
12200 MOVE Left,(Bottom+Top)/2
12210 LABEL "-d/dt([Co/C]^2)"
12220 !
12230 !    NUMBER AXES
12240 !
12250 LORG 6
12260 LDIR 0
12270 CSIZE 3,.6
12280 FOR I=0 TO High_x
12290    MOVE I,0
12300    LABEL I
12310 NEXT I
12320 !
12330 LORG 8
12340 FOR I=0 TO High_y
12350    MOVE 0,I
12360    LABEL I
12370 NEXT I
12380 !
12390 !    DRAW AXES
12400 !
12410 CLIP 0,High_x,0,High_y
12420 AXES 1,1,0,0,2,2
12430 !
12440 !    DRAW ZERBST PLOT
12450 !
12460 FOR I=1 TO Data_points
12470    PLOT Zerx(I),Zery(I)
12480 NEXT I
12490 !
12500 !    DRAW FITTED LINE
```

```
12510 !
12520 PENUP
12530 LINE TYPE 5
12540 FOR X=0 TO High_x STEP High_x
12550     DRAW X,Fit_slope*X+Fit_intercept
12560 NEXT X
12570 !
12580 !   EXIT SUBROUTINE ZERBST PLOT
12590 !
12600 SUBEND!
```

```
12610 SUB Survel_plot(REAL Readtime(*),Surface_vel(*),INTEGER Data_points)
12620 GCLEAR
12630 WINDOW 0,100,0,200
12631 MOVE Readtime(1),Surface_vel(1)
12640 FOR I=1 TO Data_points
12650     DRAW Readtime(I),Surface_vel(I)
12660 NEXT I
12670 SUBEND
```

APPENDIX F

Admittance Measurement Data Gathering Program

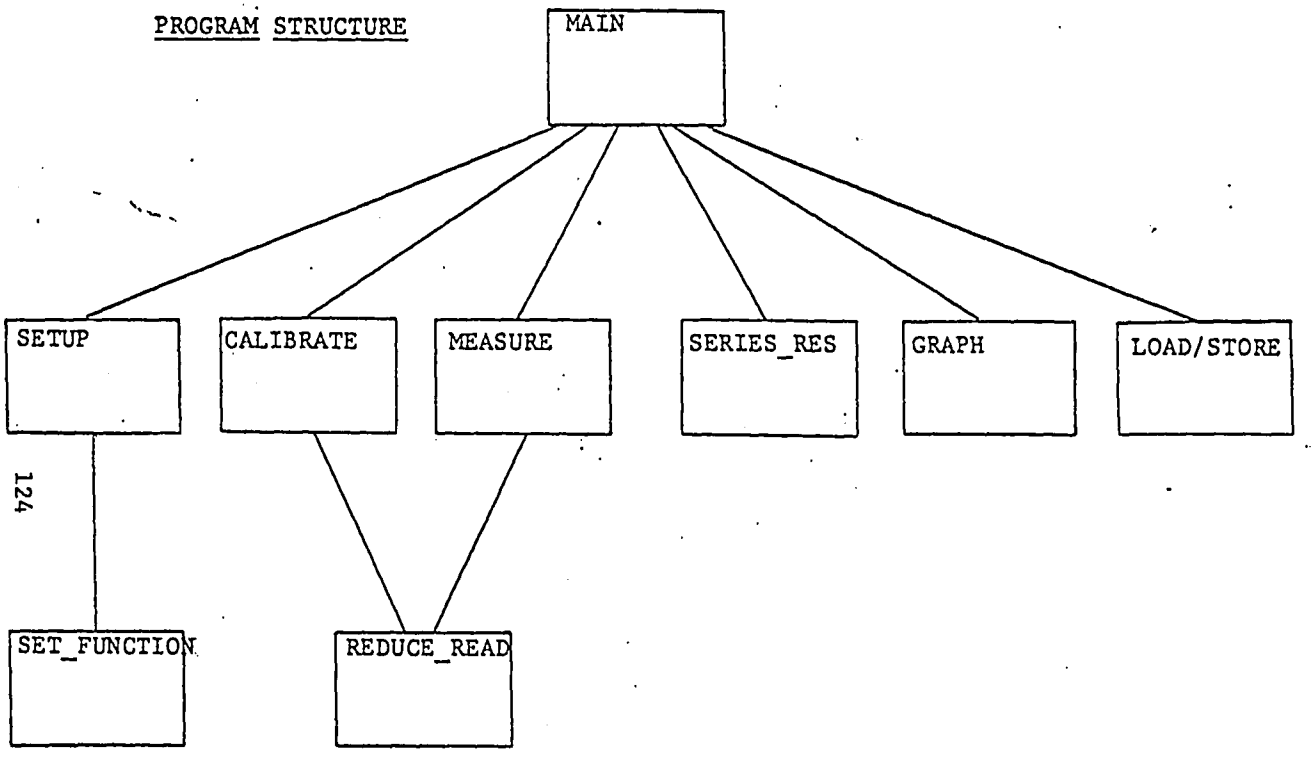
This program was written to gather admittance data from a lock-in amplifier. The program is written in a structured manner so that the program functions are available to the user with use of the software keys. The functions available are setup, calibrate, measure, graph, load and store.

The program uses the function generator, a D/A, and the A/D converter in control and data gathering. The function generator is set to the desired measurement frequency. The program controls the D/A to create the bias sweep across the device. Data is gathered from three channels of the A/D Converter, channel 1 is used to measure conductance, channel 2 capacitance, and channel 4 the bias across the device.

Typically, a user, after setting up the measurement instruments would run the setup phase of the program to enter input data and experimental parameters. At this point a known capacitor would be place in the D.U.T. position and the CALIBRATE subroutine would be run. After calibration the device can be put in place and measurements

taken. When data gathering is complete it can be graphed and stored. Old data files can be retrieved with the LOAD key. An additional processing feature allows the user to correct measured data for series resistance problems be pressing the SER_RES key.

PROGRAM STRUCTURE



Variable Directory of COM Block
/CONDUCT_DATA/

Variable	Description
NOTES$ -	(STRING) 30 characters in which user may enter any comments or identifier; set by subroutine SETUP.
GENDATE$ -	(STRING) string variable containing data of data creation. Set by subroutine MEASURE
DOPING_TYPE$ -	(STRING) One character user input parameter in subroutine SETUP indicating doping type of device under test.
CALIBRATE_FLAG -	(INTEGER) Switch indicating whether or not subroutine CALIBRATE has been executed in current run.
DATA_FLAG -	(INTEGER) Switch indicating that the data arrays have been filled. Set by subroutine MEASURE.
SETUP_FLAG -	(INTEGER) Switch set to true (1) when program has successfully completed subroutine SETUP.
SER_RES -	(INTEGER) Switch set to true (1) when program has successfully completed subroutine SERIES_RES.
HIGH_VOLT -	(INTEGER) Variable indicating digital counts of D/A converter that highest output voltage corresponds to. Set by subroutine SETUP.
POINTS -	(INTEGER) Varible indicating number of points in data arrays. Set by subroutine MEASURE.
BIAS_VOLT(1000) -	(REAL) Measured data points of set bias voltages

CAPACITANCE(1000) - (REAL) Measured data points of capacitance across device under test.

CONDUCTANCE(1000) - (REAL) Measured data points of conductance across device under test.

CAPAC_CONST - (REAL) Variable set by subroutine CALIBRATE which is multiplier for digital reading from A/D to obtain capacitance value.

CONDUCT_CONST - (REAL) Variable set by subroutine CALIBRATE which is multiplier for conductance reading from A/D to obtain conductance value.

CAPAC_OXIDE - (REAL) Variable set by subroutine SETUP which is the oxide capacitance across D.U.T.

MEAS_FREQ - (REAL) Variable set by subroutine SETUP which indicates the measurement frequency.

SAMPLE_CODE$ - (STRING) Variable set by subroutine SETUP which indicates identifier for D.U.T.

DEV_AREA - (REAL) User input parameter set by subroutine SETUP indicating the area of the D.U.T.

V_SOURCE_BULK - (REAL) User input parameter set by subroutine SETUP indicating the source to bulk voltage in 3 terminal measurements.

TEMPERATURE - (REAL) User input parameter set by subroutine SETUP indicating the temperature of the measurements

Subroutine Directory
Admittance Measurement Data Gathering Program

SETUP

Function:

a) set function generator to measurement frequency & amplitude

b) get experimental parameters from user
- -sweep amplitude
- -source-bulk voltage (3 term meas)
- -temperature
- -sample id
- -device area
- -comments

CALIBRATE

Function:

a) get multiplication factors on capacitance & conductance scale

b) get value of standard capacitor

c) read values on conductance & capacitance scale until conductance is zero and so capacitance reading corresponds to standard cap.

MEASURE

Function:

a) control bias on device through sweep in 1mV steps

b) read capacitance & conductance at each bias & store in data array

SERIES RES

Function:

a) correct data array for series resistance according to furmulas given in MOS Physics and Tech /10/ pg. 220

GRAPH

Function:

a) Find upper & lower limits of data in data array

b) graph conductance & capacitance on same screen

LOAD/STORE

Function:

a) put data on disk file for later analysis

```
10    !*************************************************************************
20    !  NICOLLIAN AND GOETZBERGER TYPE CONDUCTANCE MEASUREMENT PROGRAM
30    !*************************************************************************
40    !
50    !   DECLARATIONS
60    !
70    OPTION BASE 1
80    !
90    COM /Conduct_data/ Notes$[30]
100   COM /Conduct_data/ Gendate$[11],Doping_type$[1]
110   COM /Conduct_data/ INTEGER Calibrate_flag,Data_flag,Setup_flag,Ser_res
120   COM /Conduct_data/ INTEGER High_volt,Points
130   COM /Conduct_data/ REAL Bias_volt(1000),Capacitance(1000),Conductance(1000
)
140   COM /Conduct_data/ Capac_const,Conduct_const,Capac_oxide,Meas_freq
150   COM /Conduct_data/ Sample_code$[10],Dev_area,V_source_bulk,Temperature
160   !
170   COM /Config/ @Hpib,@A_to_d,@D_to_a,@Func_gen,@Printer,@Plotter
180   !
190   DIM Blanks$[60]
200   Blanks$="                                                            "
210   !
220   !   ASSIGN I/O PATHS
230   !
240   ASSIGN @Hpib TO 7
250   ASSIGN @A_to_d TO 706
260   ASSIGN @D_to_a TO 707
270   ASSIGN @Func_gen TO 716
280   ASSIGN @Printer TO 708
290   ASSIGN @Plotter TO 705
300   !
310   !   SET UP SOFTWARE KEYS
320   !
330   ON KEY 0 LABEL "SETUP" CALL Setup
340   ON KEY 1 LABEL "CALIBRATE" CALL Calib
350   ON KEY 2 LABEL "MEASURE" CALL Measure
360   ON KEY 3 LABEL "GRAPH" CALL Graph
370   ON KEY 4 LABEL "ANALYZE" GOSUB Dummy
380   ON KEY 5 LABEL "LOAD" CALL Load
390   ON KEY 6 LABEL "STORE" CALL Store
400   ON KEY 7 LABEL "FREQ FILE" CALL Freq_file
410   ON KEY 8 LABEL "REDUCE/SER RES" CALL Reduce
420   ON KEY 9 LABEL "EXIT" GOTO Ending
421   ON KEY 18 LABEL "SER RES" CALL Series_res
430   !
440   !   SET MACHINE IN IDLE LOOP WAITING FOR INPUT
450   !
460   WHILE 1=1
470      DISP FNTime$(TIMEDATE)&Blanks$&FNDate$(TIMEDATE)
480   END WHILE
490   !
500   !   DUMMY SUBROUTINE
510   !
520 Dummy:   !
530   RETURN
540   !
550   !   EXIT
560   !
570 Ending:  !
580   END
590   SUB Calib
```

```
600   !
610   !*************************************************************************
620   !   SUBROUTINE CALIBRATE GETS CALIBRATION CONSTANTS FOR CONVERSION OF
630   !   DIGITAL READINGS FROM CHANNEL 1&2 OF A/D TO CAPAC AND CONDUCT
640   !*************************************************************************
650   !
660   !   DECLARATIONS
670   !
680   OPTION BASE 1
690   !
700   COM /Conduct_data/ Notes$[30]
710   COM /Conduct_data/ Gendate$[11],Doping_type$[1]
720   COM /Conduct_data/ INTEGER Calibrate_flag,Data_flag,Setup_flag,Ser_res
730   COM /Conduct_data/ INTEGER High_volt,Points
740   COM /Conduct_data/ REAL Bias_volt(1000),Capacitance(1000),Conductance(1000
)
750   COM /Conduct_data/ Capac_const,Conduct_const,Capac_oxide,Meas_freq
760   COM /Conduct_data/ Sample_code$,Dev_area,V_source_bulk,Temperature
770   !
780   COM /Config/ @Hpib,@A_to_d,@D_to_a,@Func_gen,@Printer,@Plotter
790   !
800   REAL Known_capac,Capac_read,Conduct_read
810   INTEGER Read_cap(10),Read_cond(10),Cond_mult,Cap_mult
820   INTEGER Cap_chan_read
830   !
840   !    CHECK TO SEE IF SET UP COMPLETE
850   !
860   IF Setup_flag=0 THEN
870      OUTPUT 1;CHR$(12);"ERROR -- SETUP PHASE NOT COMPLETED, MUST BE DONE BEF
ORE CALIBRATION"
880      BEEP
890      GOTO Ending
900   END IF
910   !
920   !    SET UP SCREEN
930   !
940   OUTPUT 1;CHR$(12);"****************** C A L I B R A T I O N   P H A S E *
******************"
950   OUTPUT 1 USING 960;CHR$(10),CHR$(10),"CHANNEL","R1","R2","R3","R4","R5","R
6","R7","R8","R9","R10","VALUE"
960   IMAGE A,A,7A,5X,2A,4X,2A,4X,2A,4X,2A,4X,2A,4X,2A,4X,2A,4X,2A,4X,2A,3X,3A,5
X,5A
970   !
980   !   GET VALUE OF KNOWN CAPACITOR FROM USER
990   !
1000  INPUT "ENTER VALUE OF KNOWN CAPACITOR IN PICOFARADS:",Known_capac
1010  !
1020  !   GET SCALUE MULTIPLIER IN CONDUCTANCE CHANNEL
1030  !
1040  Cond_mult=0
1050  WHILE (Cond_mult<>1 AND Cond_mult<>10 AND Cond_mult<>100)
1060     INPUT "ENTER MULTIPLICATION FACTOR ON CONDUCTANCE SCALE:",Cond_mult
1070  END WHILE
1080  !
1090  !   GET SCALE MULTIPLIER IN CAPACITANCE CHANNEL
1100  !
1110  Cap_mult=0
1120  WHILE (Cap_mult<>1 AND Cap_mult<>10 AND Cap_mult<>100)
1130     INPUT "ENTER MULTIPLICATION FACTOR ON CAPAPCITANCE SCALE:",Cap_mult
1140  END WHILE
```

```
1150  !
1160  !   SET UP SOFTWARE KEYS SO USER CAN DEFINE WHEN CONDUCT AND CAPACITANCE
1170  !                CALIBRATIONS SHOULD TAKE PLACE
1180  !
1190  OFF KEY
1200  ON KEY 0 LABEL "CONDUCTANCE",10 GOSUB Conductance
1210  ON KEY 1 LABEL "CAPACITANCE",9 GOSUB Capacitance
1220  ON KEY 9 LABEL "EXIT",8 GOTO Complete
1230  !
1240  ENABLE
1250  Cap_chan_read=0
1260  !
1270 Idle:  !
1280  DISP FNTime$(TIMEDATE)
1290  GOTO Idle
1300  !
1310  !    CONDUCTANCE CHANNEL READ ROUTINE
1320  !
1330 Conductance: !
1340  !
1350  !   GET 10 CONDUCTANCE READINGS FROM CHANNEL 1
1360  !
1370  FOR I=1 TO 10
1380     OUTPUT @A_to_d USING "4A";"H1AJ"
1390     ENTER @A_to_d USING "#,W";Read_cond(I)
1400     WAIT .2
1410  NEXT I
1420  !
1430  !   REDUCE READINGS TO SINGLE VALUE
1440  !
1450  CALL Reduce_read(Read_cond(*),Conduct_read,10)
1460  !
1470  OUTPUT 1 USING 1480;"CONDUCT:";Read_cond(*);Conduct_read
1480  IMAGE 8A,2X,10(DDDDD,1X),1X,DDDDD.DD
1490  !
1500  RETURN
1510  !
1520  !   CAPACITANCE CALIBRATION SUBROUTINE
1530  !
1540 Capacitance: !
1550  !
1560  !   GET 10 READINGS FROM CHANNEL 2 OF A TO D
1570  !
1580  FOR I=1 TO 10
1590     OUTPUT @A_to_d USING "#,4A";"H2AJ"
1600     ENTER @A_to_d USING "#,W";Read_cap(I)
1610     WAIT .2
1620  NEXT I
1630  !
1640  !   REDUCE READINGS TO SINGLE VALUE
1650  !
1660  CALL Reduce_read(Read_cap(*),Capac_read,10)
1670  !
1680  OUTPUT 1 USING 1690;"CAPAC:",Read_cap(*),Capac_read
1690  IMAGE 6A,4X,10(DDDDD,1X),1X,DDDDD.DD
1700  !
1710  RETURN
1720  !
1730 Complete:!
1740  !
1750  !  GET CAPACITANCE CALIBRATION CONSTANT
```

```
1760  !
1770  IF Capac_read<>0 THEN
1780     Capac_const=Known_capac/Capac_read
1790     Conduct_const=Capac_const*Cap_mult/Cond_mult
1800     Calibrate_flag=1
1801     OUTPUT 1;CHR$(12)
1810  ELSE
1820     OUTPUT 1;"NO CAPACITANCE VALUE READ, CALIBRATION NOT COMPLETED"
1830  END IF
1840  !
1850 Ending: !
1860  SUBEND
1870  SUB Reduce_read(INTEGER Readings(*),REAL Read_value,INTEGER No_of_read)
1880  !*************************************************************************
1890  ! SUBROUTINE TAKES BIVEN READINGS THROWS OUT HIGH AND LOW AND AVERAGES
1900  !                THE REMAINING TO GET VALUE OF READ
1910  !*************************************************************************
1920  !
1930  !   DECLARATIONS
1940  !
1950  REAL Low_read,High_read,Sum_of_reads
1960  INTEGER Low_spot,High_spot
1970  !
1980  !   FIND LOW READING
1990  !
2000  Low_read=Readings(1)
2010  Low_spot=1
2020  !
2030  FOR I=1 TO No_of_read
2040     IF Readings(I)<Low_read THEN
2050        Low_read=Readings(I)
2060        Low_spot=I
2070     END IF
2080  NEXT I
2090  !
2100  !   FIND HIGH READING
2110  !
2120  High_read=Readings(No_of_read)
2130  High_spot=No_of_read
2140  !
2150  FOR I=1 TO No_of_read
2160     IF Readings(I)>High_read AND I<>Low_spot THEN
2170        High_read=Readings(I)
2180        High_spot=I
2190     END IF
2200  NEXT I
2210  !
2220  !   SUM ALL READINGS EXCEPT LOW AND HIGH
2230  !
2240  Sum_of_reads=0
2250  FOR I=1 TO No_of_read
2260     IF I<>Low_spot AND I<>High_spot THEN
2270        Sum_of_reads=Sum_of_reads+Readings(I)
2280     END IF
2290  NEXT I
2300  !
2310  !   FIND AVERAGE VALUE FROM SUM
2320  !
2330  Read_value=Sum_of_reads/(No_of_read-2)
2340  !
```

```
2350  !   EXIT
2360  !
2370  SUBEND
2380   DEF FNTime$(Now)   ! GIVEN 'SECONDS' RETURN 'HH:MM:SS'
2390     !
2400     Now=INT(Now) MOD 86400
2410     H=Now DIV 3600
2420     M=Now MOD 3600 DIV 60
2430     S=Now MOD 60
2440     OUTPUT T$ USING "#,ZZ,K";H,":",M,":",S
2450     RETURN T$
2460   FNEND
2470   !
2480   DEF FNTime(T$)   ! GIVEN 'HH:MM:SS' RETURN 'SECONDS'
2490     !
2500     ON ERROR GOTO Err
2510     ENTER T$;H,M,S
2520     RETURN (3600*H+60*M+S) MOD 86400
2530 Err:OFF ERROR
2540     RETURN TIMEDATE MOD 86400
2550   FNEND
2560  DEF FNDate$(Seconds) !GIVEN 'SECONDS' RETURN "DD MMM YYYY"
2570    !
2580    DATA JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC
2590    DIM Month$(1:12)[3]
2600    READ Month$(*)
2610    !
2620    Julian=Seconds DIV 86400-1721119
2630    Year=(4*Julian-1) DIV 146097
2640    Julian=(4*Julian-1) MOD 146097
2650    Day=Julian DIV 4
2660    Julian=(4*Day+3) DIV 1461
2670    Day=(4*Day+3) MOD 1461
2680    Day=(Day+4) DIV 4
2690    Month=(5*Day-3) DIV 153
2700    Day=(5*Day-3) MOD 153
2710    Day=(Day+5) DIV 5
2720    Year=100*Year+Julian
2730    IF Month<10 THEN
2740       Month=Month+3
2750    ELSE
2760       Month=Month-9
2770       Year=Year+1
2780    END IF
2790    OUTPUT D$ USING "#,ZZ,X,3A,X,4Z";Day,Month$(Month),Year
2800    RETURN D$
2810  FNEND
2820  !
2830  DEF FNDate(Dmy$)  ! GEVEN 'DD MMM YYYY' RETURN 'SECONDS'
2840    !
2850    DATA JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC
2860    DIM Month$(1:12)[3]
2870    READ Month$(*)
2880    !
2890    ON ERROR GOTO Err
2900    I$=Dmy$&"    "
2910    ENTER I$ USING "DD,4A,5D";Day,M$,Year
2920    IF Year<100 THEN Year=Year+1900
2930    FOR I=1 TO 12
2940    IF POS(M$,Month$(I)) THEN Month=I
2950    NEXT I
```

```
2960     IF Month=0 THEN Err
2970     IF Month>2 THEN
2980        Month=Month-3
2990     ELSE
3000        Month=Month+9
3010        Year=Year-1
3020     END IF
3030     Century=Year DIV 100
3040     Remainder=Year MOD 100
3050     Julian=146097*Century DIV 4+1461*Remainder DIV 4+(153*Month+2) DIV 5+Day
+1721119
3060     Julian=Julian*86400
3070     IF Julian<2.08662912E+11 OR Julian>=2.143252224E+11 THEN Err
3080     RETURN Julian      ! RETURN JULIAN DATE IN SECONDS
3090 Err:    OFF ERROR
3100     RETURN TIMEDATE    ! RETURN CURRENT DATE
3110  FNEND
3120  SUB Setup
3130  !*************************************************************************
3140  !   SUBROUTINE SETUP GET EXPERIMENTAL PARAMETER FROM USER TO PERFORM
3150  !                     CONDUCTANCE MEASUREMENTS
3160  !*************************************************************************
3170  !
3180  !   DECLARATIONS
3190  !
3200  OPTION BASE 1
3210  !
3220  COM /Conduct_data/ Notes$[30]
3230  COM /Conduct_data/ Gendate$[11],Doping_type$[1]
3240  COM /Conduct_data/ INTEGER Calibrate_flag,Data_flag,Setup_flag,Ser_res
3250  COM /Conduct_data/ INTEGER High_volt,Points
3260  COM /Conduct_data/ REAL Bias_volt(1000),Capacitance(1000),Conductance(1000
)
3270  COM /Conduct_data/ Capac_const,Conduct_const,Capac_oxide,Meas_freq
3280  COM /Conduct_data/ Sample_code$,Dev_area,V_source_bulk,Temperature
3290  !
3300  COM /Config/ @Hpib,@A_to_d,@D_to_a,@Func_gen,@Printer,@Plotter
3310  !
3320  INTEGER Voltage_sweep
3330  !
3340  !   SET FUNCTION GENERATOR
3350  !
3360  OUTPUT 1;CHR$(12);"********************   SET FUNCTION GENERATOR   ********
**********"
3370  CALL Set_func_gen
3380  !
3390  !   GET EXPERIMENTAL PARAMETERS FROM USER
3400  !
3410  OUTPUT 1;CHR$(12);"*****************  ENTER EXPERIMENTAL PARAMETERS   ****
**********"
3420  !
3430  Voltage_sweep=0
3440  WHILE (Voltage_sweep<1 OR Voltage_sweep>10)
3450     INPUT "ENTER NUMBER OF VOLTS TO BE SWEPT (SET START VOLT MANUALLY):",Vo
ltage_sweep
3460  END WHILE
3470  !
3480  High_volt=Voltage_sweep*100
3481  !
3482  !
```

```
3483  INPUT "ENTER SOURCE TO BULK VOLTAGE IN VOLT:",V_source_bulk
3484  !
3485  INPUT "ENTER TEMPERATURE IN DEG KELVIN:",Temperature
3490  !
3500  !   GET DEVICE PARAMETERS FROM USER
3510  !
3520  OUTPUT 1;CHR$(12);"***************  ENTER DEVICE PARAMETERS & COMMENTS  *
**********"
3530  !
3540  Doping_type$=""
3550  WHILE (Doping_type$<>"N" AND Doping_type$<>"P")
3560     INPUT "ENTER DOPING TYPE OF DEVICE (N OR P):",Doping_type$
3570  END WHILE
3580  !
3581  !
3582  INPUT "ENTER SAMPLE IDENTIFICATION CODE:",Sample_code$
3583  !
3584  INPUT "ENTER DEVICE AREA IN CM^2:",Dev_area
3585  !
3590  INPUT "ENTER COMMENTS:",Notes$
3600  !
3610  !   EXIT SUBROUTINE SETUP
3620  !
3630  OUTPUT 1;CHR$(12)
3640  Setup_flag=1
3650  SUBEND
3660  SUB Set_func_gen
3670  !
3680  !*************************************************************************
3690  !   SUBROUTINE TO SET FUNCTION GENERATOR TO APPLY SIGNAL TO DEVICE
3700  !*************************************************************************
3710  !
3720  !   DECLARATIONS
3730  !
3731  OPTION BASE 1
3732  !
3733  COM /Conduct_data/ Notes$[30]
3734  COM /Conduct_data/ Gendate$[11],Doping_type$[1]
3735  COM /Conduct_data/ INTEGER Calibrate_flag,Data_flag,Setup_flag,Ser_res
3736  COM /Conduct_data/ INTEGER High_volt,Points
3737  COM /Conduct_data/ REAL Bias_volt(1000),Capacitance(1000),Conductance(1000
)
3738  COM /Conduct_data/ Capac_const,Conduct_const,Capac_oxide,Meas_freq
3739  COM /Conduct_data/ Sample_code$,Dev_area,V_source_bulk,Temperature
3740  !
3742  COM /Config/ @Hpib,@A_to_d,@D_to_a,@Func_gen,@Printer,@Plotter
3750  !
3760  REAL Frequency,Device_voltage,Set_voltage
3770  DIM Func_gen_com$[100],Error_message$[20]
3780  !
3790  !   SETUP SERVICE ROUTINE FOR FUNCTION GENERATOR IF SERVICE REQUEST RAISED
3800  !
3810  ON INTR 7,10 GOSUB Serv_rout
3820  ENABLE INTR 7;2
3830  !
3840  !   INPUT SIGNAL FREQUENCY
3850  !
3860 Get_func:  !
3870  INPUT "ENTER SIGNAL FREQUENCY IN HERTZ",Frequency
3871  Meas_freq=Frequency
3880  !
```

```
3890  !   INPUT SIGNAL AMPLITUDE THAT IS APPLIED TO DEVICE (.01X GEN SIGNAL)
3900  !
3910  INPUT "ENTER SIGNAL AMPLITUDE IN MV (AS APPLIED TO DEVICE)",Device_voltage
3920  Set_voltage=Device_voltage*1.0E-3*1.0E+2/2
3930  !
3940  !   SET-UP COMMAND LINE FOR FUNCTION GENERATOR
3950  !
3960  OUTPUT Func_gen_com$ USING "K,K,K,K,K";"M1,CT0,T0,H0,W1,FRQ ",Frequency,"
HZ,AMP ",Set_voltage," V,OFS,0V,L0,C0,D0,A0"
3970  !
3980  !   SEND COMMAND LINE TO FUNCTION GENERATOR
3990  !
4000  Done=1
4010  !
4020  REMOTE @Func_gen
4030  OUTPUT @Func_gen USING "K";Func_gen_com$
4040  !
4050  IF Done=1 THEN
4060     SUBEXIT
4070  ELSE
4080     GOTO Get_func
4090  END IF
4100  !
4110  !   SERVICE ROUTINE FOR SERVICE REQUEST FROM FUNCTION GEN
4120  !
4130 Serv_rout: !
4140  OUTPUT @Func_gen USING "K";"IERR"
4150  ENTER @Func_gen USING "K";Error_message$
4160  A=SPOLL(@Func_gen)
4170  OUTPUT 1;"ERROR RECIEVED FROM FUNCTION GENERATOR:";Error_message$;" ";A
4180  ENABLE INTR 7;2
4190  Done=0
4200  RETURN
4210  !
4220  !   EXIT
4230  !
4240 Ending: !
4250  SUBEND
4260  SUB Measure
4270  !*************************************************************************
4280  !  SUBROUTINE MEASURE TAKE CAPACITANCE AND CONDUCTANCE READINGS AT
4290  !                          DIFFERENT BIASES
4300  !*************************************************************************
4310  !
4320  !   DECLARATIONS
4330  !
4340  OPTION BASE 1
4350  !
4360  COM /Conduct_data/ Notes$[30]
4370  COM /Conduct_data/ Gendate$[11],Doping_type$[1]
4380  COM /Conduct_data/ INTEGER Calibrate_flag,Data_flag,Setup_flag,Ser_res
4390  COM /Conduct_data/ INTEGER High_volt,Points
4400  COM /Conduct_data/ REAL Bias_volt(1000),Capacitance(1000),Conductance(1000
)
4410  COM /Conduct_data/ Capac_const,Conduct_const,Capac_oxide,Meas_freq
4420  COM /Conduct_data/ Sample_code$,Dev_area,V_source_bulk,Temperature
4430  !
4440  COM /Config/ @Hpib,@A_to_d,@D_to_a,@Func_gen,@Printer,@Plotter
4450  !
4460  INTEGER Read_volt(50),Read_cap(50),Read_cond(50)
4470  REAL Dig_volt,Dig_cap,Dig_cond
```

```
4480  !
4490  !    CHECK TO SEE IF CALIBRATION COMPLETE
4500  !
4510  IF Calibrate_flag=0 THEN
4520     BEEP
4530     OUTPUT 1;CHR$(12);"ERROR -- EXPERIMENT NOT CALIBRATED, MEASUREMENTS CAN
NOT BE PERFORMED"
4540  ELSE
4550  !
4560  !    PERFORM MEASUREMENTS
4570  !
4580  OUTPUT 1;CHR$(12);"**********   MEASUREMENT PHASE   **********
**********"
4590  Points=0
4600  !
4610  FOR Bias=0 TO High_volt
4620     !
4630     !   SET BIAS VOLTAGE
4640     !
4650     OUTPUT @D_to_a USING "ZZZZ";Bias+2000
4651     WAIT .3
4660     !
4670     !   READ BIAS VOLTAGE
4680     !
4690     FOR I=1 TO 8
4691        WAIT .01
4700        OUTPUT @A_to_d USING "#,4A";"H8AJ"
4710        ENTER @A_to_d USING "#,W";Read_volt(I)
4720     NEXT I
4730     !
4740     Dig_volt1=.02*Read_volt(1)+.10*Read_volt(2)+.16*Read_volt(3)+.23*Read_v
olt(4)
4741     Dig_volt2=.23*Read_volt(5)+.16*Read_volt(6)+.10*Read_volt(7)+.02*Read_v
olt(8)
4742     Dig_volt=Dig_volt1+Dig_volt2
4750     !
4760     !   READ CONDUCTANCE
4770     !
4780     FOR I=1 TO 8
4781        WAIT .01
4790        OUTPUT @A_to_d USING "#,4A";"H1AJ"
4800        ENTER @A_to_d USING "#,W";Read_cond(I)
4810     NEXT I
4820     !
4830     Dig_cond1=.02*Read_cond(1)+.10*Read_cond(2)+.16*Read_cond(3)+.23*Read_c
ond(4)
4831     Dig_cond2=.23*Read_cond(5)+.16*Read_cond(6)+.10*Read_cond(7)+.02*Read_c
ond(8)
4832     Dig_cond=Dig_cond1+Dig_cond2
4840     !
4850     !   READ CAPACITANCE
4860     !
4870     FOR I=1 TO 8
4871        WAIT .01
4880        OUTPUT @A_to_d USING "#,4A";"H2AJ"
4890        ENTER @A_to_d USING "#,W";Read_cap(I)
4900     NEXT I
4910     !
4920     Dig_cap1=.02*Read_cap(1)+.10*Read_cap(2)+.16*Read_cap(3)+.23*Read_cap(4
)
4921     Dig_cap2=.02*Read_cap(8)+.10*Read_cap(7)+.16*Read_cap(6)+.23*Read_cap(5
```

```
)
4922      Dig_cap=Dig_cap1+Dig_cap2
4930      !
4940      !   ENTER POINT INTO DATA FILE
4950      !
4960      Points=Points+1
4970      Bias_volt(Points)=Dig_volt/100
4980      Capacitance(Points)=Dig_cap*Capac_const
4990      Conductance(Points)=Dig_cond*Conduct_const
5000      !
5010      !   MOVE TO NEXT BIAS
5030    NEXT Bias
5031    !
5032    !  EXTRACT OXIDE CAPACITANCE
5033    !
5034    Capac_oxide=FNMax(Capacitance(*),Points)
5040    !
5050    !  END OF MEASUREMENTS
5060    !
5070    OUTPUT @D_to_a USING "ZZZZ";2000
5080    OUTPUT 1;CHR$(12)
5081    BEEP
5082    BEEP
5090    END IF
5100    !
5110    !   EXIT MEASUREMENT SUBROUTINE
5120    !
5130    SUBEND
5140    SUB Store
5150    !*****************************************************************************
5160    !     SUBROUTINE "STORE" CREATES A DATA FILE AND STORES THE RELEVENT
5170    !                                    DATA IN IT
5180    !*****************************************************************************
5190    !
5200    !    DECLARATIONS
5210    !
5220    OPTION BASE 1
5230    COM /Conduct_data/ Notes$[30]
5240    COM /Conduct_data/ Gendate$[11],Doping_type$[1]
5250    COM /Conduct_data/ INTEGER Calibrate_flag,Data_flag,Setup_flag,Ser_res
5260    COM /Conduct_data/ INTEGER High_volt,Points
5270    COM /Conduct_data/ REAL Bias_volt(1000),Capacitance(1000),Conductance(1000
)
5280    COM /Conduct_data/ Capac_const,Conduct_const,Capac_oxide,Meas_freq
5290    COM /Conduct_data/ Sample_code$,Dev_area,V_source_bulk,Temperature
5300    !
5310    DIM Answer$[1],Data_file$[10],File_specifier$[30]
5320    !
5330    !   DEFINE PROGRAM STATE TO USER
5340    !
5350    OUTPUT 1;CHR$(12);"*************** STORAGE OF INTERNAL DATA FILE DISK ****
**************"
5380    !
5390    !   GET DATA FILE NAME AND LOCATION FROM USER
5400    !
5450    INPUT "ENTER DATA FILENAME:",Data_file$
5460    INPUT "SPECIFY LEFT OR RIGHT HAND DRIVE",Answer$
5470    IF Answer$="L" THEN
5480       File_specifier$=Data_file$&":INTERNAL,4,1"
5490    ELSE
5500       File_specifier$=Data_file$&":INTERNAL,4,0"
```

```
5510  END IF
5520  !
5530  !   CREATE THE FILE AND STORE
5540  !
5550  CREATE BDAT File_specifier$,6,4096
5560  ASSIGN @Path1 TO File_specifier$;FORMAT OFF
5570  !
5580  !
5590  OUTPUT @Path1;Notes$
5600  OUTPUT @Path1;Gendate$,Doping_type$
5610  OUTPUT @Path1;High_volt,Points
5620  OUTPUT @Path1;Bias_volt(*),Capacitance(*),Conductance(*)
5630  OUTPUT @Path1;Capac_const,Conduct_const,Capac_oxide,Meas_freq
5640  OUTPUT @Path1;Sample_code$,Dev_area,V_source_bulk
5650  OUTPUT @Path1;Temperature
5660  !
5670  !   CLOSE I/O PATH AND EXIT
5680  !
5681  OUTPUT 1;CHR$(12)
5690  ASSIGN @Path1 TO *
5700  !
5710  SUBEND
5720  !
5730  SUB Load
5740  !*****************************************************************************
5750  !   SUBROUTINE "LOAD" RETRIEVES A DATA FILE FROM THE DISK
5760  !*****************************************************************************
5770  !
5780  !    DECLARATIONS
5790  !
5800  OPTION BASE 1
5810  !
5820  COM /Conduct_data/ Notes$[30]
5830  COM /Conduct_data/ Gendate$[11],Doping_type$[1]
5840  COM /Conduct_data/ INTEGER Calibrate_flag,Data_flag,Setup_flag,Ser_res
5850  COM /Conduct_data/ INTEGER High_volt,Points
5860  COM /Conduct_data/ REAL Bias_volt(1000),Capacitance(1000),Conductance(1000
)
5870  COM /Conduct_data/ Capac_const,Conduct_const,Capac_oxide,Meas_freq
5880  COM /Conduct_data/ Sample_code$,Dev_area,V_source_bulk,Temperature
5890  !
5900  DIM Answer$[1],File_specifier$[30],Data_file$[10]
5910  !
5920  !   DEFINE PROGRAM STATE TO USER
5930  !
5940  OUTPUT 1;CHR$(12);"***************** LOAD OF INTERNAL FILE FROM DISK *****
**************"
5950  !
5960  !   GET FILE NAME AND LOCATION
5970  !
5980  INPUT "ENTER THE FILENAME:",Data_file$
5990  INPUT "SPECIFY LEFT OR RIGHT HAND DRIVE(L OR R):",Answer$
6000  IF Answer$="L" THEN
6010     File_specifier$=Data_file$&":INTERNAL,4,1"
6020  ELSE
6030     File_specifier$=Data_file$&":INTERNAL,4,0"
6040  END IF
6050  !
6060  !         LOAD THE DATA FILE
6070  !
6080  ASSIGN @Path2 TO File_specifier$;FORMAT OFF
```

```
6090  !
6100  !
6110  ENTER @Path2;Notes$
6120  ENTER @Path2;Gendate$,Doping_type$
6130  ENTER @Path2;High_volt,Points
6140  ENTER @Path2;Bias_volt(*),Capacitance(*),Conductance(*)
6150  ENTER @Path2;Capac_const,Conduct_const,Capac_oxide,Meas_freq
6160  ENTER @Path2;Sample_code$,Dev_area,V_source_bulk
6170  ENTER @Path2;Temperature
6171  !
6172  !   SET PROGRAM STATE FLAGS
6173  !
6174  Calibrate_flag=0
6175  Setup_flag=0
6176  Data_flag=1
6177  Ser_res=0
6180  !
6190  !   CLOSE I/O PATH AND EXIT
6200  !
6201  OUTPUT 1;CHR$(12)
6210  ASSIGN @Path2 TO *
6220  !
6230  SUBEND
6240  !
6250  SUB Graph
6260  !*****************************************************************************
6270  !       SUBROUTINE GRAPH PLOTS TO C-V CURVES AND THE G-V CURVES
6280  !*****************************************************************************
6290  !
6300  !   DECLARATIONS
6310  !
6320  OPTION BASE 1
6330  !
6340  COM /Conduct_data/ Notes$[30]
6350  COM /Conduct_data/ Gendate$[11],Doping_type$[1]
6360  COM /Conduct_data/ INTEGER Calibrate_flag,Data_flag,Setup_flag,Ser_res
6370  COM /Conduct_data/ INTEGER High_volt,Points
6380  COM /Conduct_data/ REAL Bias_volt(1000),Capacitance(1000),Conductance(1000
)
6390  COM /Conduct_data/ Capac_const,Conduct_const,Capac_oxide,Meas_freq
6400  COM /Conduct_data/ Sample_code$[10],Dev_area,V_source_bulk,Temperature
6410  !
6420  COM /Config/ @Hpib,@A_to_d,@D_to_a,@Func_gen,@Printer,@Plotter
6430  !
6440  COM /Graphics/ INTEGER Axes_drawn
6450  COM /Graphics/ Left,Right,Top_cap,Bottom_cap,Top_cond,Bottom_cond
6460  COM /Graphics/ High_cap,Low_cap,High_cond,Low_cond,Low_voltage,High_voltag
e
6470  COM /Graphics/ INTEGER Cond_range,Cap_range
6480  !
6490  DIM Answer$[3]
6500  !
6510  !   RE-INITIALIZE GRAPHICS DISPLAY
6520  !
6530  DEG
6540  CLIP OFF
6550  PIVOT 0
6560  PEN 1
6570  LINE TYPE 1,5
6580  LORG 1
```

```
6590  CSIZE 5,.6
6600  LDIR 0
6610  MOVE 0,0
6620  !
6630  IF Axes_drawn=1 THEN
6640     INPUT "SHOULD DATA BE PLOTTED ON SAMES AXES(YES OR NO)?",Answer$
6650  END IF
6660  !
6670  !  SET FLAG TO SEE WHETHER TO DRAW AXES
6680  !
6690  Init=((Axes_drawn=1 AND Answer$="NO") OR Axes_drawn=0)
6700  IF Init THEN
6710     GCLEAR
6720     !
6730     !   DEFINE GRAPHING LIMITS BY FINDING MAX'S AND MINS OF PARMS
6740     !
6750     High_cap=FNMax(Capacitance(*),Points)
6760     Low_cap=FNMin(Capacitance(*),Points)
6770     High_cond=FNMax(Conductance(*),Points)
6780     Low_cond=FNMin(Conductance(*),Points)
6790     !
6800     !   RE-DEFINE HIGH CAP AND COND ACCORDING TO GRAPHING LIMITS
6810     !
6820     Cond_range=-28
6830     WHILE (10^(Cond_range+1)<High_cond-Low_cond)
6840        Cond_range=Cond_range+1
6850     END WHILE
6860     !
6870     Cap_range=-28
6880     WHILE (10^(Cap_range+1)<High_cap-Low_cap)
6890        Cap_range=Cap_range+1
6900     END WHILE
6910     !
6920     !
6930     High_cap=INT(High_cap/10^Cap_range+1)*10^Cap_range
6940     Low_cap=INT(Low_cap/10^Cap_range)*10^Cap_range
6950     High_cond=INT(High_cond/10^Cond_range+1)*10^Cond_range
6960     Low_cond=INT(Low_cond/10^Cond_range)*10^Cond_range
6970     Low_voltage=INT(Bias_volt(Points))
6980     High_voltage=INT(Bias_volt(1)+1)
6990     !
7000     !   DEFINE GRAPHING LIMITS
7010     !
7020     Bottom_cap=-.1*(ABS(High_cap-Low_cap))+Low_cap
7030     Top_cap=1.1*(ABS(High_cap-Low_cap))+Low_cap
7040     Bottom_cond=-.1*(ABS(High_cond-Low_cond))+Low_cond
7050     Top_cond=1.1*(ABS(High_cond-Low_cond))+Low_cond
7060     Left=-.1*(ABS(High_voltage-Low_voltage))+Low_voltage
7070     Right=1.1*(ABS(High_voltage-Low_voltage))+Low_voltage
7080     !
7090     Axes_drawn=1
7100  END IF
7110  !
7120  !   SET UP GRAPHICS DISPLAY FOR CONDUCTANCE CURVES
7130  !
7140  WINDOW Left,Right,Bottom_cond,Top_cond
7150  !
7160  !   DRAW TITLE AND AXES IF INITIAL
7170  !
7180  IF Init THEN
7190     !
```

```
7200      !   TITLE GRAPH
7210      !
7220      CSIZE 6,.6
7230      MOVE (Left+Right)/2,Top_cond
7240      LORG 6
7250      LABEL "COND. AND CAP. VS VOLTAGE"
7260      !
7270      !   LABEL AXES
7280      !
7290      LDIR 90
7300      CSIZE 4,.6
7310      MOVE Left,(Bottom_cond+Top_cond)/2
7320      LORG 6
7330      LABEL "G/w (PF)"
7340      !
7350      LDIR 0
7360      MOVE (Left+Right)/2,Bottom_cond
7370      LORG 4
7380      LABEL "VOLTAGE"
7390      !
7400      CSIZE 3,.6
7410      LORG 8
7420      FOR I=Low_cond TO High_cond STEP 10^Cond_range
7430          MOVE Low_voltage,I
7440          LABEL I
7450      NEXT I
7460      !
7470      LORG 6
7480      FOR I=Low_voltage TO High_voltage
7490          MOVE I,Low_cond
7500          LABEL I
7510      NEXT I
7520  END IF
7530  !
7540  CLIP Low_voltage,High_voltage,Low_cond,High_cond
7550  IF Init THEN
7560     AXES 1,10^Cond_range,Low_voltage,Low_cond
7570  END IF
7580  !
7590  PENUP
7600  FOR I=1 TO Points
7610     PLOT Bias_volt(I),Conductance(I)
7620  NEXT I
7630  !
7640  CLIP OFF
7650  !
7660  !   MOVE TO CAPACITANCE GRAPH
7670  !
7680  WINDOW Left,Right,Bottom_cap,Top_cap
7690  !
7700  IF Init THEN
7710     !
7720     !   LABEL AXES
7730     !
7740     LDIR 90
7750     LORG 4
7760     CSIZE 4,.6
7770     MOVE Right,(Bottom_cap+Top_cap)/2
7780     LABEL "CAPACITANCE(PF)"
7790     !
```

```
7800      LDIR 0
7810      LORG 2
7820      CSIZE 3,.6
7830      FOR I=Low_cap TO High_cap STEP 10^Cap_range
7840         MOVE High_voltage,I
7850         LABEL I
7860      NEXT I
7870      !
7880   END IF
7890   !
7900   CLIP Low_voltage,High_voltage,Low_cap,High_cap
7910   IF Init THEN
7920      AXES 1,10^Cap_range,High_voltage,Low_cap
7930   END IF
7940   !
7950   PENUP
7960   FOR I=1 TO Points
7970      PLOT Bias_volt(I),Capacitance(I)
7980   NEXT I
7990   !
8000   CLIP OFF
8010   !
8020   !   EXIT
8030   !
8040   SUBEND
8050   DEF FNMax(REAL Array(*),INTEGER Elements)
8060   !*************************************************************************
8070   !   SUBROUTINE FIND THE MAXIMUM VALUE OF AN ARRAY OF REAL NUMBERS
8080   !*************************************************************************
8090   !
8100   !   DECLARATION
8110   !
8120   REAL Maximum
8130   !
8140   Maximum=1.E-99
8150   FOR I=1 TO Elements
8160      IF Array(I)>Maximum THEN Maximum=Array(I)
8170   NEXT I
8180   !
8190   RETURN Maximum
8200   FNEND
8210   DEF FNMin(REAL Array(*),INTEGER Elements)
8220   !*************************************************************************
8230   !   SUBROUTINE FINDS THE MINIMUM VALUE OF AN ARRAY OF REAL NUMBERS
8240   !*************************************************************************
8250   !
8260   !   DECLARATIONS
8270   !
8280   REAL Minimum
8290   Minimum=1.E+99
8300   !
8310   FOR I=1 TO Elements
8320      IF Array(I)<Minimum THEN Minimum=Array(I)
8330   NEXT I
8340   !
8350   RETURN Minimum
8360   FNEND
8370   SUB Freq_file
8380   !*************************************************************************
8390   !   THIS SUBROUTINE ADDS A DATA POINT TO A CONDUCTANCE VS. FREQUENCY
8400   !                     DATA FILE ON DISK
```

```
8410  !*************************************************************************
8420  !
8430  !   DECLARATIONS
8440  !
8450  OPTION BASE 1
8460  !
8470  COM /Cond_freq_data/ Bias_voltage,Sample_id$[10],Temp,Dop_type$[1]
8480  COM /Cond_freq_data/ Pnts,Meas_fr(50),Conduct(50),Capac(50)
8490  !
8500  COM /Conduct_data/ Notes$[30]
8510  COM /Conduct_data/ Gendate$[11],Doping_type$[1]
8520  COM /Conduct_data/ INTEGER Calibrate_flag,Data_flag,Setup_flag,Ser_res
8530  COM /Conduct_data/ INTEGER High_volt,Points
8540  COM /Conduct_data/ REAL Bias_volt(1000),Capacitance(1000),Conductance(1000
)
8550  COM /Conduct_data/ Capac_const,Conduct_const,Capac_oxide,Meas_freq
8560  COM /Conduct_data/ Sample_code$[10],Dev_area,V_source_bulk,Temperature
8570  !
8580  DIM Answer$[3],File_name$[10],File_specifier$[30]
8590  !
8600  INTEGER Data_point,Data_compatable,Count,Inc,Slot
8610  !
8620  !   DEFINE PROGRAM STATE TO USER
8630  !
8640  OUTPUT 1;CHR$(12);"**** STORAGE OF DATA POINT IN FREQ VS. CONDUCTANCE AND
CAPACITANCE FILE ******"
8650  !
8660  !   DECIDE IF OLD OR NEW DATA FILE
8670  !
8680  INPUT "ENTER ""NEW"" IF DATA FILE TO BE CREATED:",Answer$
8690  !
8700  IF Answer$="NEW" THEN
8710     INPUT "ENTER NEW FILE NAME:",File_name$
8720     INPUT "SPECIFY LEFT OR RIGHT HAND DRIVE(L OR R):",Answer$
8730     IF Answer$="L" THEN
8740        File_specifier$=File_name$&":INTERNAL,4,1"
8750     ELSE
8760        File_specifier$=File_name$&":INTERNAL,4,0"
8770     END IF
8780     !
8790     !   CREATE DISK FILE
8800     !
8810     CREATE BDAT File_specifier$,4,400
8820     ASSIGN @File TO File_specifier$;FORMAT OFF
8830     !
8840     !   GET BIAS VOLTAGE TO BE RECORDED
8850     !
8860     REPEAT
8870        INPUT "ENTER BIAS VOLTAGE TO BE RECORDED:",Bias_voltage
8880     UNTIL (Bias_voltage>Bias_volt(1) AND Bias_voltage<Bias_volt(Points)) OR
 (Bias_voltage<Bias_volt(1) AND Bias_voltage>Bias_volt(Points))
8890     !
8900     !   ASSIGN REMAING FILE PARAMETERS
8910     !
8920     Sample_id$=Sample_code$
8930     Temp=Temperature
8940     Dop_type$=Doping_type$
8950     Pnts=0
8960     FOR I=1 TO 50
8970        Capac(I)=0
```

```
8980          Conduct(I)=0
8990          Meas_fr(I)=0
9000       NEXT I
9010       !
9020       !   WRITE RECORD 1 OF FILE
9030       !
9040       OUTPUT @File,1;Bias_voltage,Sample_id$,Temp,Dop_type$,Pnts
9050       OUTPUT @File,2;Meas_fr(*)
9060       OUTPUT @File,3;Conduct(*)
9070       OUTPUT @File,4;Capac(*)
9080    ELSE
9090       !
9100       !   GET EXISTING DATA FILE FROM DISK
9110       !
9120       REPEAT
9130          File_name$=""
9140          INPUT "ENTER EXISTING FILE NAME:",File_name$
9150          !
9160          !   IF USER ENTERS BLANK FILE NAME EXIT SUBROUTINE
9170          !
9180          IF File_name$="" THEN SUBEXIT
9190          INPUT "SPECIFY LEFT OR RIGHT HAND DRIVE(L OR R):",Answer$
9200          IF Answer$="L" THEN
9210             File_specifier$=File_name$&":INTERNAL,4,1"
9220          ELSE
9230             File_specifier$=File_name$&":INTERNAL,4,0"
9240          END IF
9250          !
9260          !   POINT TO DISK FILE
9270          !
9280          ASSIGN @File TO File_specifier$;FORMAT OFF
9290          !
9300          !   ENTER FILE INTO COM AREA
9310          !
9320          ENTER @File,1;Bias_voltage,Sample_id$,Temp,Dop_type$,Pnts
9330          ENTER @File,2;Meas_fr(*)
9340          ENTER @File,3;Conduct(*)
9350          ENTER @File,4;Capac(*)
9360          !
9370          !   CHECK FILE FOR CONSISTENCY WITH DATA
9380          !
9390          Data_compatable=(Sample_id$=Sample_code$ AND Temp=Temperature AND Do
p_type$=Doping_type$)
9400          !
9410          IF NOT Data_compatable THEN
9420             BEEP
9430             OUTPUT 1;CHR$(10);"ERROR -- DATA NOT COMPATABLE WITH DISK FILE";C
HR$(10)
9440             OUTPUT 1 USING "20X,13A,12X,9A";"INTERNAL DATA","DISK DATA"
9450             OUTPUT 1 USING "12A,9X,10A,14X,10A";"SAMPLE CODE:",Sample_code$,S
ample_id$
9460             OUTPUT 1 USING "12A,9X,DDD.DD,18X,DDD.DD";"TEMPERATURE:",Temperat
ure,Temp
9470             OUTPUT 1 USING "12A,9X,1A,23X,1A";"DOPING TYPE:",Doping_type$,Dop
_type$
9480          END IF
9490       UNTIL Data_compatable
9500    END IF
9510    !
9520    !  VERIFY DATA FILE TO USER
9530    !
```

```
9540  IMAGE 20A,DDDDD.DD,2A
9550  IMAGE 20A,10A
9560  OUTPUT 1;CHR$(10);"THE DISK DATA FILE IS SPECIFIED BY THE FOLLOWING PARAME
TERS:";CHR$(10)
9570  OUTPUT 1 USING 9540;"BIAS VOLTAGE-",Bias_voltage
9580  OUTPUT 1 USING 9550;"SAMPLE CODE-",Sample_id$
9590  OUTPUT 1 USING 9540;"TEMPERATURE-",Temp
9600  OUTPUT 1 USING 9550;"DOPING TYPE-",Dop_type$
9610  !
9620  !   FIND DATA POINT IN INTERNAL FILE TO BE SAVED
9630  !
9640  IF Bias_volt(1)>Bias_volt(Points) THEN
9650     Count=Points
9660     Inc=-1
9670  ELSE
9680     Count=1
9690     Inc=1
9700  END IF
9710  !
9720  WHILE Bias_volt(Count)<Bias_voltage
9730     Count=Count+Inc
9740  END WHILE
9750  !
9760  !  EXTRACT CLOSEST POINT TO GIVEN BIAS VOLTAGE
9770  !
9780  IF ABS(Bias_volt(Count)-Bias_voltage)<ABS(Bias_volt(Count-Inc)-Bias_voltag
e) THEN
9790     Data_point=Count
9800  ELSE
9810     Data_point=Count-Inc
9820  END IF
9830  !
9840  !   FIND SLOT TO SAVE DATA POINT IN FREQUENCY FILE
9850  !
9860  IF Pnts=0 OR Meas_freq<=Meas_fr(1) THEN
9870     Slot=1
9880  ELSE
9881     Slot=2
9882     WHILE Meas_freq>Meas_fr(Slot) AND Meas_fr(Slot)<>0
9883        Slot=Slot+1
9890     END WHILE
9891   END IF
9900  !
9910  !   CHECK TO SEE THAT DATA NOT ALREADY ENTERED FOR MEAS FREQUENCY
9920  !
9930  IF Meas_fr(Slot)=Meas_freq THEN
9940     BEEP
9950     OUTPUT 1;CHR$(10);"ERROR -- THE FOLLOWING POINT ALREADY EXISTS IN FILE:
";CHR$(10)
9960     OUTPUT 1 USING "20X,10A,10X,11A,9X,11A";"FREQUENCY","CONDUCTANCE","CAPA
CITANCE"
9970     IMAGE 10A,10X,DDDDD.DD,"Hz",10X,DDDDD.DD,"pF",9X,DDDDD.DD,"pF"
9980     OUTPUT 1 USING 9970;"DATA FILE:",Meas_fr(Slot),Conduct(Slot),Capac(Slot
)
9990     OUTPUT 1 USING 9970;"INTL FILE:",Meas_freq,Conductance(Data_point),Capa
citance(Data_point)
9991     PAUSE
10000 ELSE
10010    !
10020    !  VERIFY DATA POINT ADDITION WITH USER
```

```
10030     !
10040     OUTPUT 1;CHR$(10);"THE FOLLOWING POINT WILL BE ADDED TO THE DATA FILE:"
;CHR$(10)
10050     IMAGE 15A,10X,DDDDD.DD,3A
10060     OUTPUT 1 USING 10050;"BIAS VOLTAGE:",Bias_volt(Data_point),"V"
10070     OUTPUT 1 USING 10050;"FREQUENCY-",Meas_freq,"Hz"
10080     OUTPUT 1 USING 10050;"CONDUCTANCE-",Conductance(Data_point),"pF"
10090     OUTPUT 1 USING 10050;"CAPACITANCE-",Capacitance(Data_point),"pF"
10100     Answer$=""
10110     INPUT "HIT ENTER TO CONTINUE, ANYTHING ELSE WILL ABORT:",Answer$
10120     IF Answer$="" THEN
10130        !
10140        !   ENTER POINT IN FILE
10150        !
10160        FOR I=Pnts TO Slot STEP -1
10170           Meas_fr(I+1)=Meas_fr(I)
10180           Conduct(I+1)=Conduct(I)
10190           Capac(I+1)=Capac(I)
10200        NEXT I
10210        !
10220        Meas_fr(Slot)=Meas_freq
10230        Conduct(Slot)=Conductance(Data_point)
10240        Capac(Slot)=Capacitance(Data_point)
10250        !
10260        Pnts=Pnts+1
10270        !
10280        !   RE-WRITE DATA FILE
10290        !
10300        OUTPUT @File,1;Bias_voltage,Sample_id$,Temp,Dop_type$,Pnts
10310        OUTPUT @File,2;Meas_fr(*)
10320        OUTPUT @File,3;Conduct(*)
10330        OUTPUT @File,4;Capac(*)
10340        !
10350        !    NORMAL EXIT OF FREQ FILE SUBROUTINE
10360        !
10370        OUTPUT 1;CHR$(12)
10380     END IF
10390  END IF
10400  !
10410  !   EXIT SUBROUTINE FREQ FILE
10420  !
10430  SUBEND
10440  SUB Series_res
10450  !*****************************************************************************
10460  !   SUBROUTINE SERIES_RES MAKES CORRECTION FOR SERIES RESITANCE
10470  !   ACCORDING TO MODEL OF NICOLLIAN AND BREWS IN MOS PHYSICS AND TECH
10480  !                BOOK PAGE 220
10490  !*****************************************************************************
10500  !
10510  !   DECLARATION
10520  !
10530  OPTION BASE 1
10540  !
10550  COM /Conduct_data/ Notes$[30]
10560  COM /Conduct_data/ Gendate$[11],Doping_type$[1]
10570  COM /Conduct_data/ INTEGER Calibrate_flag,Data_flag,Setup_flag,Ser_res
10580  COM /Conduct_data/ INTEGER High_volt,Points
10590  COM /Conduct_data/ REAL Bias_volt(1000),Capacitance(1000),Conductance(1000
)
10600  COM /Conduct_data/ Capac_const,Conduct_const,Capac_oxide,Meas_freq
10610  COM /Conduct_data/ Sample_code$,Dev_area,V_source_bulk,Temperature
```

```
10620 !
10630 INTEGER Accum_point
10640 REAL Series_resist,Num_factor,Den_factor,A
10641 IF NOT Data_flag THEN
10642    BEEP
10643    OUTPUT 1;"ERROR -- NO DATA TAKEN TO CORRECT FOR SERIES RESISTANCE"
10644    SUBEXIT
10645 END IF
10646 !
10647 !   DEFINE PROGRAM STATE TO USER
10648 !
10649 OUTPUT 1;CHR$(12);"**************  DATA FILE BEING CORRECTED FOR SERIES
RESITANCE  ***********"
10651 !
10660 !   DECIDE WHICH END BIAS POINT IS TAKEN IN ACCUMULATION
10670 !
10680 IF ABS(Capacitance(1)-Capac_oxide)<ABS(Capacitance(Points)-Capac_oxide) TH
EN
10690    Accum_point=1
10700 ELSE
10710    Accum_point=Points
10720 END IF
10730 !
10820 !   GET SERIES RESISTANCE AND OXIDE CAPACITANCE
10830 !
10840 Series_resist=Conductance(Accum_point)/((Conductance(Accum_point))^2+(Capa
citance(Accum_point))^2)
10850 !
10860 Capac_oxide=Capacitance(Accum_point)*(1+(Conductance(Accum_point)/(Capacit
ance(Accum_point)))^2)
10870 !
10880 !    CORRECT DATA FILE FOR SERIES RESISTANCE
10890 !
10900 FOR I=1 TO Points
10910    Num_factor=Conductance(I)^2+(Capacitance(I))^2
10920    A=Conductance(I)-Num_factor*Series_resist
10930    Den_factor=A^2+(Capacitance(I))^2
10940    !
10950    Capacitance(I)=Num_factor*Capacitance(I)/Den_factor
10960    Conductance(I)=Num_factor*A/Den_factor
10970    !
10980 NEXT I
10990 !
11000 !   EXIT SERIES RESISTANCE SUBROUTINE
11010 !
11011 Ser_res=1
11012 OUTPUT 1;CHR$(12)
11013 !
11020 SUBEND
11030 SUB Reduce
11040 !*************************************************************************
11050 !   SUBROUTINE REDUCE CHANGES THE MEASURE CAPACITANCES AND CONDUCTANCES
11060 !     TO EQUIVALENT PARALLEL CAPACITANCE AND CONDUCTANCE
11070 !  REF: H. DEULING,E. KLAUSMANN AND A. GOETZBURGER,SOL ST ELEC,15,
11080 !           559-571(1972)
11090 !*************************************************************************
11100 !
11110 !   DECLARATIONS
11120 !
11130 OPTION BASE 1
```

```
11140 !
11150 COM /Conduct_data/ Notes$[30]
11160 COM /Conduct_data/ Gendate$[11],Doping_type$[1]
11170 COM /Conduct_data/ INTEGER Calibrate_flag,Data_flag,Setup_flag,Ser_res
11180 COM /Conduct_data/ INTEGER High_volt,Points
11190 COM /Conduct_data/ REAL Bias_volt(1000),Capacitance(1000),Conductance(1000
)
11200 COM /Conduct_data/ Capac_const,Conduct_const,Capac_oxide,Meas_freq
11210 COM /Conduct_data/ Sample_code$,Dev_area,V_source_bulk,Temperature
11220 !
11230 REAL Den_factor
11240 !
11250 !   GET w*Cox
11260 !
11280 FOR I=1 TO Points
11290    Den_factor=(Conductance(I)/Capac_oxide)^2+(1-Capacitance(I)/Capac_oxide
)^2
11300    !
11310    !   GET EQUIVALENT PARALLEL CONDUCTANCE
11320    !
11321    IF Den_factor<>0 THEN
11330       Conductance(I)=(Conductance(I)/Capac_oxide)/Den_factor
11331    END IF
11340    !
11350    !   GET EQUIVALENT PARALLEL CAPACITANCE
11360    !
11361    IF Den_factor<>0 THEN
11370      Capacitance(I)=((1-Capacitance(I)/Capac_oxide)/Den_factor)-1
11371    END IF
11380 NEXT I
11390 !
11400 !   EXIT
11410 !
11420 SUBEND
```

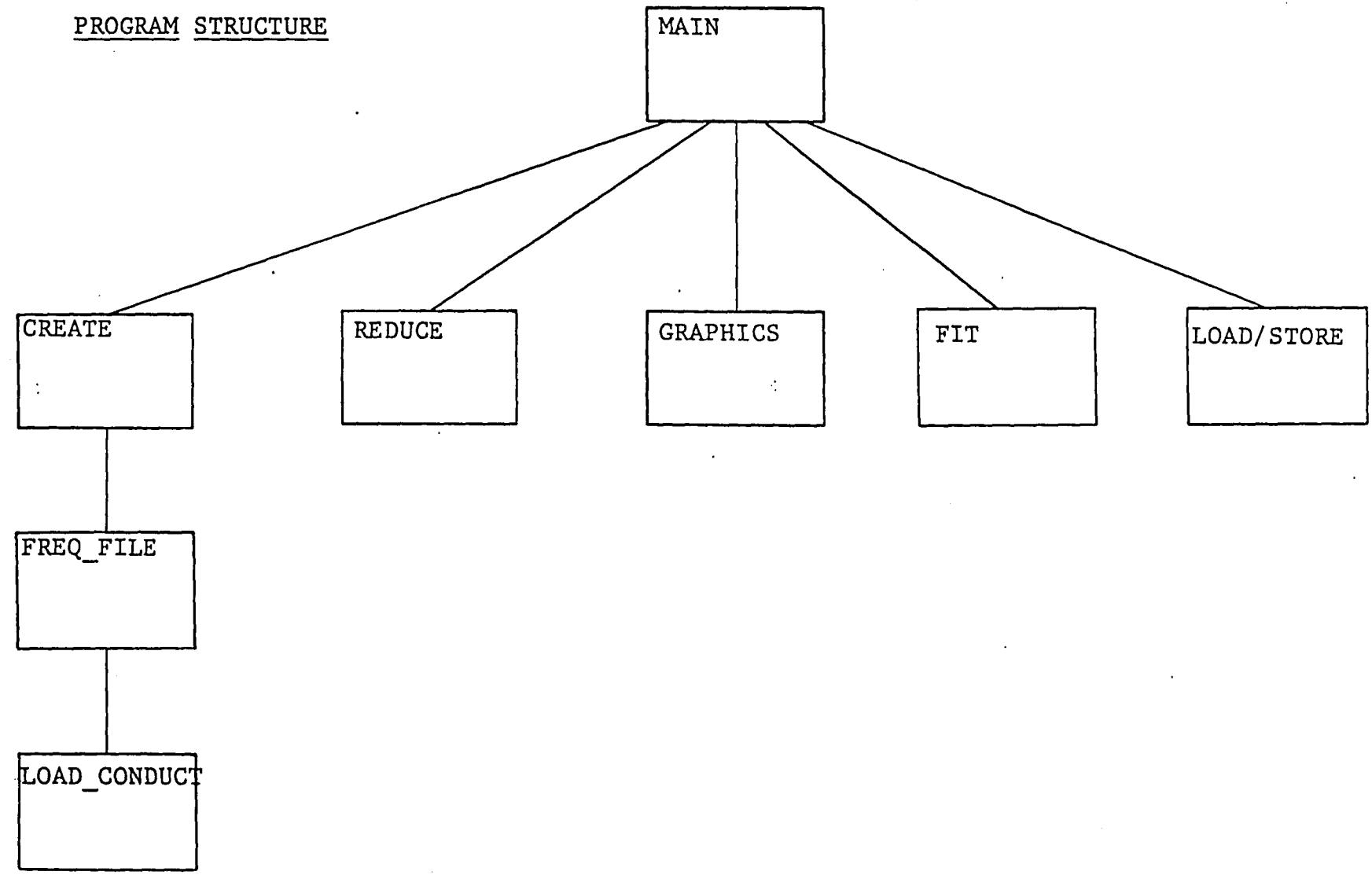
APPENDIX G

Admittance Measurement Analysis Program

This program was written to analyze admittance measurements according to the theory developed by Nicollian & Goetzburger. The program is written in a structured manner with file creation, reduction, fitting as well as graphics available to the user.

The program is typically used by first creating a Conductance versus Frequency file by use of the CREATE key. This function cross sections input files at a given bias level. Before the created data can be compared to theory it must be reduced to an equivalent parallel capacitance and conductance by use of the REDUCE key. Fits can then be attempted with the user providing input parameters. Comparison of measured and fitted data can be done graphically. Load and store functions are provided so that both fitted and measured data can be saved on disk.

PROGRAM STRUCTURE



Subroutine Directory
Admittance Measurement Analysis Program

CREATE

Function:

a) get bias voltage to be analyzed from user

b) get list of disk file names from user which contain conductance & capacitance vs. bias data taken at different frequencies on same device

c) pass list to subroutine frequency file which searches disk files & extracts conductance & capacitance at given bias

d) result is a conductance & capacitance vs. frequency file at a fixed surface potential.

REDUCE

Function:

a) takes measured conductance & capacitance data and reduces them to equivalent parallel conductance & capacitance according to formulas given by H. Deuling, E. Klausmann and A. Goetzburger, Sol. St. Elec., 15, 559-571, (1972)

GRAPHICS

Function:

a) maintains seperate graphics screens of conductance vs. frequency and capacitance vs. frequency (can either graph measured or theoretical data).

FIT

Function:

a) gets device parameters from user from which it generates theoretical curves of the conductance & capacitance vs. frequency.

LOAD/STORE

Function:

a) store/load data on disk for later analysis.

Variable Directory of COMMON Block
/COND_FREQ_DATA/

BIAS_VOLTAGE - User input parameter indicating the bias voltage to be examined. Set by subroutine CREATE.

SAMPLE_ID$ - (STRING) 10 Character input parameter in data file indicating sample identification code.

TEMP - (REAL) Parameter in data file indicating the temperature at measurement time.

DOP_TYPE$ - (STRING) 1 Character in data file indicating the doping type of sample.

PNTS - (REAL) Variable indicating the number of points in data file.

MEAS_FR(50)- (REAL) Data array of cross sectioned data indicating measurement frequency of data point.

CONDUCT_M(50) - (REAL) Data array of measured conductances at corresponding measurement frequency.

CAPAC_M(50) - (REAL) Data array of measured capacitances at corresponding measurement frequency.

CONDUCT_P(50) - (REAL) Created data array set by subroutine REDUCE which is the parallel conductance extracted from the measured conductance.

CAPAC_P(50) - (REAL) Created data array set by subroutine REDUCE which is the parallel capacitance extracted from the measured capacitance and conductance.

COND_P_FIT(100) - (REAL) Created data array set by subroutine FIT which is the created theoretical conductance data.

CAP_P_FIT(100) - (REAL) Created data array set by subroutine FIT which is the created theoretical capacitance data.

FREQ_FIT(100) - (REAL) Created data array of the frequencies which theoretical data is calculated for. Set by subroutine FIT

OX_CAP - (REAL) User input parameter in subroutine FIT which indicates the sample oxide capacitance.

SIGMA_S - (REAL) User input parameter used for fitting indicating the varience in the surface potential.

DIT - (REAL) User input parameter in subroutine FIT indicating the interface state density.

TAU_N - (REAL) User input parameter in subroutine FIT indicating the trap time constant.

AREA - (REAL) User input parameter in subroutine FIT indicating the device area.

CAL_POINTS - (REAL) Create variable in subroutine FIT indicating the number of theoretical data points calculated

CD - (REAL) User input parameter in subroutine FIT indicating the device depletion capacitance.

```
10    !*************************************************************************
20    !  PROGRAM FOR ANALYSIS OF CONDUCTANCE VS. FREQUENCY DATA
30    !*************************************************************************
40    !
50    !   DECLARATION
60    !
70    OPTION BASE 1
80    !
90    COM /Cond_freq_data/ Bias_voltage,Sample_id$[10],Temp,Dop_type$[1]
100   COM /Cond_freq_data/ Pnts,Meas_fr(50),Conduct_m(50),Capac_m(50)
110   COM /Cond_freq_data/ Conduct_p(50),Capac_p(50)
120   COM /Cond_freq_data/ Cond_p_fit(100),Cap_p_fit(100),Freq_fit(100)
130   COM /Cond_freq_data/ Ox_cap,Sigma_s,Dit,Tau_n,Area,Cal_points,Cd
140   !
150   ON KEY 0 LABEL "REDUCE" CALL Reduce
160   ON KEY 1 LABEL "VIEW G/C" CALL View_cond
161   ON KEY 11 CALL View_cap
170   ON KEY 2 LABEL "GRAPH G/C" CALL Graph
171   ON KEY 12 CALL Cap_graph
180   ON KEY 3 LABEL "ANALYZE" GOSUB Dummy
190   ON KEY 4 LABEL "CREATE" CALL Create_file
200   ON KEY 5 LABEL "LOAD" CALL Load
210   ON KEY 6 LABEL "STORE" CALL Store
220   ON KEY 7 LABEL "FIT" CALL Fit
221   ON KEY 8 LABEL "FIT LOAD/STORE" CALL Load_fit
222   ON KEY 18 CALL Store_fit
230   ON KEY 9 LABEL "EXIT" GOTO Ending
240   !
250   !  SET MACHINE IN IDLE LOOP
260   !
270   WHILE 1=1
280      DISP FNTime$(TIMEDATE)
290   END WHILE
300   !
310   !   DUMMY SUBROUTINE
320   !
330 Dummy:   !
340   RETURN
350   !
360 Ending:   !
370   END!
```

```
SUB Reduce
!*************************************************************************
!   SUBROUTINE REDUCE CHANGES THE MEASURE CAPACITANCES AND CONDUCTANCES
!      TO EQUIVALENT PARALLEL CAPACITANCE AND CONDUCTANCE
!  REF:  H. DEULING,E. KLAUSMANN AND A. GOETZBURGER,SOL ST ELEC,15,
!          559-571(1972)
!*************************************************************************
!
!    DECLARATIONS
!
OPTION BASE 1
!
COM /Cond_freq_data/ Bias_voltage,Sample_id$[10],Temp,Dop_type$[1]
COM /Cond_freq_data/ Pnts,Meas_fr(50),Conduct_m(50),Capac_m(50)
COM /Cond_freq_data/ Conduct_p(50),Capac_p(50)
COM /Cond_freq_data/ Cond_p_fit(100),Cap_p_fit(100),Freq_fit(100)
COM /Cond_freq_data/ Ox_cap,Sigma_s,Dit,Tau_n,Area,Cal_points,Cd
!
REAL Den_factor,Cap_ox
!
!    GET OXIDE CAPACITANCE
!
INPUT "ENTER OXIDE CAPACITANCE(PF):",Cap_ox
!
!    REDUCE MEASURED CAP AND COND TO EQUIVALENT PARALLEL CAP AND COND
!
FOR I=1 TO Pnts
   Den_factor=(Conduct_m(I)/Cap_ox)^2+(1-Capac_m(I)/Cap_ox)^2
   !
   IF Den_factor<>0 THEN
      Conduct_p(I)=(Conduct_m(I)/Cap_ox)/Den_factor
      Capac_p(I)=((1-Capac_m(I)/Cap_ox)/Den_factor)-1
   END IF
   !
NEXT I
!
!   EXIT
!
SUBEND!
```

```
770   SUB Load
780   !*************************************************************************
790   !   SUBROUTINE LOAD RETRIEVES DATA FROM DISK FILE
800   !*************************************************************************
810   !
820   OPTION BASE 1
830   !
840   COM /Cond_freq_data/ Bias_voltage,Sample_id$[10],Temp,Dop_typ$[1]
850   COM /Cond_freq_data/ Pnts,Meas_fr(50),Conduct_m(50),Capac_m(50)
860   COM /Cond_freq_data/ Conduct_p(50),Capac_p(50)
870   COM /Cond_freq_data/ Cond_p_fit(100),Cap_p_fit(100),Freq_fit(100)
871   COM /Cond_freq_data/ Ox_cap,Sigma_s,Dit,Tau_n,Area,Cal_points,Cd
890   !
900   DIM Answer$[3],File_name$[10],File_specifier$[30]
910   !
920   !  DEFINE PROGRAM STATE TO USER
930   !
940   OUTPUT 1;"****************** LOAD OF DATA FILE FROM DISK ****************
      ************"
950   !
960   !  GET FILE
970   !
980   INPUT "ENTER FILE NAME:",File_name$
990   INPUT "SPECIFY LEFT OR RIGHT HAND DRIVE(L OR R):",Answer$
1000  IF Answer$="L" THEN
1010     File_specifier$=File_name$&":INTERNAL,4,1"
1020  ELSE
1030     File_specifier$=File_name$&":INTERNAL,4,0"
1040  END IF
1050  !
1060  !   POINT TO DISK FILE
1070  !
1080  ASSIGN @File TO File_specifier$;FORMAT OFF
1090  !
1100  !   ENTER FILE INTO COM AREA
1110  !
1120  ENTER @File,1;Bias_voltage,Pnts
1130  ENTER @File,2;Meas_fr(*)
1140  ENTER @File,3;Conduct_m(*)
1150  ENTER @File,4;Capac_m(*)
1160  ENTER @File,5;Conduct_p(*)
1170  ENTER @File,6;Capac_p(*)
1180  !
1190  !   EXIT LOAD SUBROUTINE
1200  !
1210  SUBEND!
```

```
1220   DEF FNTime$(Now)   ! GIVEN 'SECONDS' RETURN 'HH:MM:SS'
1230     !
1240     Now=INT(Now) MOD 86400
1250     H=Now DIV 3600
1260     M=Now MOD 3600 DIV 60
1270     S=Now MOD 60
1280     OUTPUT T$ USING "#,ZZ,K";H,":",M,":",S
1290     RETURN T$
1300   FNEND
1310   !
1320   DEF FNTime(T$)   ! GIVEN 'HH:MM:SS' RETURN 'SECONDS'
1330     !
1340     ON ERROR GOTO Err
1350     ENTER T$;H,M,S
1360     RETURN (3600*H+60*M+S) MOD 86400
1370 Err:OFF ERROR
1380     RETURN TIMEDATE MOD 86400
1390   FNEND
1400  DEF FNDate$(Seconds) !GIVEN 'SECONDS' RETURN "DD MMM YYYY"
1410    !
1420    DATA JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC
1430    DIM Month$(1:12)[3]
1440    READ Month$(*)
1450    !
1460    Julian=Seconds DIV 86400-1721119
1470    Year=(4*Julian-1) DIV 146097
1480    Julian=(4*Julian-1) MOD 146097
1490    Day=Julian DIV 4
1500    Julian=(4*Day+3) DIV 1461
1510    Day=(4*Day+3) MOD 1461
1520    Day=(Day+4) DIV 4
1530    Month=(5*Day-3) DIV 153
1540    Day=(5*Day-3) MOD 153
1550    Day=(Day+5) DIV 5
1560    Year=100*Year+Julian
1570    IF Month<10 THEN
1580       Month=Month+3
1590    ELSE
1600       Month=Month-9
1610       Year=Year+1
1620    END IF
1630    OUTPUT D$ USING "#,ZZ,X,3A,X,4Z";Day,Month$(Month),Year
1640    RETURN D$
1650  FNEND
1660  !
1670  DEF FNDate(Dmy$)  ! GEVEN 'DD MMM YYYY' RETURN 'SECONDS'
1680    !
1690    DATA JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC
1700    DIM Month$(1:12)[3]
1710    READ Month$(*)
1720    !
1730    ON ERROR GOTO Err
1740    I$=Dmy$&"    "
1750    ENTER I$ USING "DD,4A,5D";Day,M$,Year
1760    IF Year<100 THEN Year=Year+1900
1770    FOR I=1 TO 12
1780    IF POS(M$,Month$(I)) THEN Month=I
1790    NEXT I
1800    IF Month=0 THEN Err
```

```
1810     IF Month>2 THEN
1820        Month=Month-3
1830     ELSE
1840        Month=Month+9
1850        Year=Year-1
1860     END IF
1870     Century=Year DIV 100
1880     Remainder=Year MOD 100
1890     Julian=146097*Century DIV 4+1461*Remainder DIV 4+(153*Month+2) DIV 5+Day
+1721119
1900     Julian=Julian*86400
1910     IF Julian<2.08662912E+11 OR Julian>=2.143252224E+11 THEN Err
1920     RETURN Julian      ! RETURN JULIAN DATE IN SECONDS
1930 Err:    OFF ERROR
1940     RETURN TIMEDATE    ! RETURN CURRENT DATE
1950  FNEND!
```

```
1960  SUB Graph
1970  !*************************************************************************
1980  !  SUBROUTINE GRAPH MAKES THE Gp/ω VS. ω CURVES FOR THE DATA FILE
1990  !*************************************************************************
2000  !
2010  !   DECLARATIONS
2020  !
2030  OPTION BASE 1
2040  !
2050  COM /Cond_freq_data/ Bias_voltage,Sample_id$[10],Temp,Dop_type$[1]
2060  COM /Cond_freq_data/ Pnts,Meas_fr(50),Conduct_m(50),Capac_m(50)
2070  COM /Cond_freq_data/ Conduct_p(50),Capac_p(50)
2080  COM /Cond_freq_data/ Cond_p_fit(100),Cap_p_fit(100),Freq_fit(100)
2081  COM /Cond_freq_data/ Ox_cap,Sigma_s,Dit,Tau_n,Area,Cal_points,Cd
2100  !
2110  COM /Graphics/ INTEGER Axes_drawn,Cond_screen(12480)
2111  COM /Graphics/ REAL Left,Right,Top,Bottom
2120  !
2130  DIM Answer$[3],Plot_fit$[3]
2140  !
2150  !  RE-INITIALIZE GRAPHICS DISPLAY
2160  !
2170  DEG
2180  CLIP OFF
2190  PIVOT 0
2200  PEN 1
2210  LINE TYPE 1,5
2220  LORG 1
2230  CSIZE 5,.6
2240  LDIR 0
2250  MOVE 0,0
2260  !
2270  IF Axes_drawn THEN
2271     GLOAD Cond_screen(*)
2280     INPUT "SHOULD DATA BE PLOTED ON SAME AXES(YES OR NO)?",Answer$
2290  END IF
2300  !
2310  !  SET FLAG TO SEE WHETHER TO DRAW AXES
2320  !
2330  Init=((Axes_drawn AND Answer$="NO") OR NOT Axes_drawn)
2340  IF Init THEN
2350     GCLEAR
2360     Axes_drawn=1
2370     !
2380     !   DEFINE GRAPHING LIMITS
2390     !
2400     Bottom=-.1
2410     Top=1.1
2420     Left=-.1*6
2430     Right=1.1*6
2440     !
2441  END IF
2442  !
2450  WINDOW Left,Right,Bottom,Top
2451  !
2452  IF Init THEN
2460     !
2470     !   TITLE GRAPH
2480     !
```

```
2490      CSIZE 6,.6
2500      MOVE (Left+Right)/2,Top
2510      LORG 6
2520      LABEL "Gp/ωCox VS. ω CURVE"
2530      !
2540      !   LABEL AXES
2550      !
2560      LDIR 90
2570      CSIZE 4,.6
2580      MOVE Left,(Bottom+Top)/2
2590      LORG 6
2600      LABEL "Gp/ωCox"
2610      !
2620      LDIR 0
2630      MOVE (Left+Right)/2,Bottom
2640      LORG 4
2650      LABEL "FREQUENCY"
2660      !
2670      CSIZE 3,.6
2680      LORG 8
2690      FOR I=0 TO 1 STEP .1
2700        MOVE 0,I
2710        LABEL I
2720      NEXT I
2730      !
2740      LORG 6
2750      FOR I=0 TO 6
2760         MOVE I,0
2770         LABEL 10^I
2780      NEXT I
2790      !
2800      !  DRAW AXES
2810      !
2820      CLIP 0,6,0,1
2830      AXES 1,.1,0,0
2840      !
2850    END IF
2860    !
2870    INPUT "DO YOU WANT TO PLOT FIT (YES OR NO)",Plot_fit$
2880    IF Plot_fit$="YES" THEN GOTO 2990
2890    !  DRAW DATA POINTS
2900    !
2910    PENUP
2920    CSIZE 3,.6
2930    FOR I=1 TO Pnts
2940       MOVE LGT(Meas_fr(I)),Conduct_p(I)
2950       LORG 5
2960       LABEL "*"
2970    NEXT I
2980    GOTO 3034
2990    !  PLOT THE FIT
3000    PENUP
3010    FOR I=1 TO Cal_points
3020        PLOT LGT(Freq_fit(I)),Cond_p_fit(I)
3030    NEXT I
3031    !
3032    !  STORE THE CONDUCTANCE GRAPHICS SCREEN
3033    !
3034    GSTORE Cond_screen(*)
3040    !
3050    SUBEND!
```

```
3060  SUB Load_conduct(File_specifier$,INTEGER Error_flag)
3070  !*************************************************************************
3080  !   SUBROUTINE "LOAD" RETRIEVES A DATA FILE FROM THE DISK
3090  !*************************************************************************
3100  !
3110  !    DECLARATIONS
3120  !
3130  OPTION BASE 1
3140  !
3150  COM /Conduct_data/ Notes$[30]
3160  COM /Conduct_data/ Gendate$[11],Doping_type$[1]
3170  COM /Conduct_data/ INTEGER Calibrate_flag,Data_flag,Setup_flag,Ser_res
3180  COM /Conduct_data/ INTEGER High_volt,Points
3190  COM /Conduct_data/ REAL Bias_volt(1000),Capacitance(1000),Conductance(1000
)
3200  COM /Conduct_data/ Capac_const,Conduct_const,Capac_oxide,Meas_freq
3210  COM /Conduct_data/ Sample_code$[10],Dev_area,V_source_bulk,Temperature
3220  !
3230  !         LOAD THE DATA FILE
3240  !
3250  ON ERROR GOSUB Error_hand
3260  ASSIGN @Path2 TO File_specifier$;FORMAT OFF
3270  OFF ERROR
3280  !
3290  !
3300  ENTER @Path2;Notes$
3310  ENTER @Path2;Gendate$,Doping_type$
3320  ENTER @Path2;High_volt,Points
3330  ENTER @Path2;Bias_volt(*),Capacitance(*),Conductance(*)
3340  ENTER @Path2;Capac_const,Conduct_const,Capac_oxide,Meas_freq
3350  ENTER @Path2;Sample_code$,Dev_area,V_source_bulk
3360  ENTER @Path2;Temperature
3370  !
3380  !   SET PROGRAM STATE FLAGS
3390  !
3400  Calibrate_flag=0
3410  Setup_flag=0
3420  Data_flag=1
3430  Ser_res=0
3440  !
3450  !   CLOSE I/O PATH AND EXIT
3460  !
3470  OUTPUT 1;CHR$(12)
3480  ASSIGN @Path2 TO *
3490  SUBEXIT
3500  !
3510  !  ERROR HANDLING SUBROUTINE
3520  !
3530 Error_hand:  !
3540  BEEP
3550  OUTPUT 1;CHR$(12),"ERROR WHILE OPENING INPUT FILE WITH ERROR:";ERRN
3560  Error_flag=1
3570  SUBEXIT
3580  RETURN
3590  !
3600  SUBEND!
```

```
3610  SUB Freq_file(Infile_spec$,REAL Bias,INTEGER New,Error_flag)
3620  !*************************************************************************
3630  !   THIS SUBROUTINE ADDS A DATA POINT TO A CONDUCTANCE VS. FREQUENCY
3640  !                    DATA FILE ON DISK
3650  !*************************************************************************
3660  !
3670  !   DECLARATIONS
3680  !
3690  OPTION BASE 1
3700  !
3710  COM /Cond_freq_data/ Bias_voltage,Sample_id$[10],Temp,Dop_type$[1]
3720  COM /Cond_freq_data/ Pnts,Meas_fr(50),Conduct_m(50),Capac_m(50)
3730  COM /Cond_freq_data/ Conduct_p(50),Capac_p(50)
3740  COM /Cond_freq_data/ Cond_p_fit(100),Cap_p_fit(100),Freq_fit(100)
3741  COM /Cond_freq_data/ Ox_cap,Sigma_s,Dit,Tau_n,Area,Cal_points,Cd
3760  !
3770  COM /Conduct_data/ Notes$[30]
3780  COM /Conduct_data/ Gendate$[11],Doping_type$[1]
3790  COM /Conduct_data/ INTEGER Calibrate_flag,Data_flag,Setup_flag,Ser_res
3800  COM /Conduct_data/ INTEGER High_volt,Points
3810  COM /Conduct_data/ REAL Bias_volt(1000),Capacitance(1000),Conductance(1000
)
3820  COM /Conduct_data/ Capac_const,Conduct_const,Capac_oxide,Meas_freq
3830  COM /Conduct_data/ Sample_code$[10],Dev_area,V_source_bulk,Temperature
3840  !
3850  INTEGER Data_point,Data_compatable,Count,Inc,Slot
3860  !
3870  !   RETRIEVE CONDUCTANCE FILE
3880  !
3890  CALL Load_conduct(Infile_spec$,Error_flag)
3900  IF NOT Error_flag THEN
3910     !
3920     !   DECIDE IF OLD OR NEW DATA FILE
3930     !
3940     IF New THEN
3950        !
3960        !  INITIALIZE INTERNAL FILE
3970        !
3980        Bias_voltage=Bias
3990        Sample_id$=Sample_code$
4000        Temp=Temperature
4010        Dop_type$=Doping_type$
4020        !
4030        Pnts=0
4040        FOR I=1 TO 50
4050           Capac_m(I)=0
4060           Conduct_m(I)=0
4070           Capac_p(I)=0
4080           Conduct_p(I)=0
4090           Meas_fr(I)=0
4100        NEXT I
4110        !
4120     ELSE
4130        !
4140        !   CHECK FILE FOR CONSISTENCY WITH DATA
4150        !
4160        Data_compatable=(Sample_id$=Sample_code$ AND Temp=Temperature AND Do
p_type$=Doping_type$)
4170        !
4180        IF NOT Data_compatable THEN
```

```
4190            BEEP
4200            OUTPUT 1;CHR$(10);"ERROR -- DATA NOT COMPATABLE WITH DISK FILE";C
HR$(10)
4210            OUTPUT 1 USING "20X,13A,12X,9A";"INTERNAL DATA","DISK DATA"
4220            OUTPUT 1 USING "12A,9X,10A,14X,10A";"SAMPLE CODE:",Sample_code$,S
ample_id$
4230            OUTPUT 1 USING "12A,9X,DDD.DD,18X,DDD.DD";"TEMPERATURE:",Temperat
ure,Temp
4240            OUTPUT 1 USING "12A,9X,1A,23X,1A";"DOPING TYPE:",Doping_type$,Dop
_type$
4250            Error_flag=1
4260            SUBEXIT
4270         END IF
4280      END IF
4290      !
4300      !   FIND DATA POINT IN INTERNAL FILE TO BE SAVED
4310      !
4320      IF Bias_volt(1)>Bias_volt(Points) THEN
4330         Count=Points
4340         Inc=-1
4350         !
4360         !   CHECK TO SEE THAT BIAS IS IN RANGE
4370         !
4380         IF Bias_voltage>Bias_volt(1) OR Bias_voltage<Bias_volt(Points) THEN
4390            BEEP
4400            OUTPUT 1;"ERROR -- BIAS TO BE RECORDED NOT INCLUDED IN FILE"
4410            OUTPUT 1;"CONDUCTANCE FILE BIAS FROM ";Bias_volt(Points);" TO ";B
ias_volt(1)
4420            Error_flag=1
4430            SUBEXIT
4440         END IF
4450      ELSE
4460         Count=1
4470         Inc=1
4480         !
4490         !   CHECK TO SEE THAT BIAS IS IN RANGE
4500         !
4510         IF Bias_voltage<Bias_volt(1) OR Bias_voltage>Bias_volt(Points) THEN
4520            BEEP
4530            OUTPUT 1;"ERROR -- BIAS TO BE RECORDED NOT INCLUDED IN FILE"
4540            OUTPUT 1;"CONDUCTANCE FILE BIAS FROM ";Bias_volt(1);" TO ";Bias_v
olt(Points)
4550            Error_flag=1
4560            SUBEXIT
4570         END IF
4580      END IF
4590      !
4600      WHILE Bias_volt(Count)<Bias_voltage
4610         Count=Count+Inc
4620      END WHILE
4630      !
4640      !  EXTRACT CLOSEST POINT TO GIVEN BIAS VOLTAGE
4650      !
4660      IF ABS(Bias_volt(Count)-Bias_voltage)<ABS(Bias_volt(Count-Inc)-Bias_vol
tage) THEN
4670         Data_point=Count
4680      ELSE
4690         Data_point=Count-Inc
4700      END IF
4710      !
```

```
4720      !   FIND SLOT TO SAVE DATA POINT IN FREQUENCY FILE
4730      !
4740      IF Pnts=0 OR Meas_freq<=Meas_fr(1) THEN
4750         Slot=1
4760      ELSE
4770         Slot=2
4780         WHILE Meas_freq>Meas_fr(Slot) AND Meas_fr(Slot)<>0
4790            Slot=Slot+1
4800         END WHILE
4810       END IF
4820      !
4830      !   CHECK TO SEE THAT DATA NOT ALREADY ENTERED FOR MEAS FREQUENCY
4840      !
4850      IF Meas_fr(Slot)=Meas_freq THEN
4860         BEEP
4870         OUTPUT 1;CHR$(10);"ERROR -- THE FOLLOWING POINT ALREADY EXISTS IN FI
LE:";CHR$(10)
4880         OUTPUT 1 USING "20X,10A,10X,11A,9X,11A";"FREQUENCY","CONDUCTANCE","C
APACITANCE"
4890         IMAGE 10A,10X,DDDDD.DD,"Hz",10X,DDDDD.DD,"pF",9X,DDDDD.DD,"pF"
4900         OUTPUT 1 USING 4890;"FREQ FILE:",Meas_fr(Slot),Conduct_m(Slot),Capac
_m(Slot)
4910         OUTPUT 1 USING 4890;"COND FILE:",Meas_freq,Conductance(Data_point),C
apacitance(Data_point)
4920         !
4930         !   CHECK FOR POINT REPLACEMENT
4940         !
4950         Answer$=""
4960         INPUT "HIT ENTER TO REPLACE POINT IN FREQ FILE, ANYTHING ELSE WILL A
BORT",Answer$
4970         !
4980         IF Answer$="" THEN
4990            !
5000            !   REPLACE POINT
5010            !
5020            Meas_fr(Slot)=Meas_freq
5030            Conduct_m(Slot)=Conductance(Data_point)
5040            Capac_m(Slot)=Capacitance(Data_point)
5050            !
5060         ELSE
5070            Error_flag=1
5080         END IF
5090      ELSE
5100         !
5110         !  VERIFY DATA POINT ADDITION WITH USER
5120         !
5130 !       OUTPUT 1;CHR$(10);"THE FOLLOWING POINT WILL BE ADDED TO THE DATA FIL
E:";CHR$(10)
5140 !       IMAGE 15A,10X,DDDDDD.DD,3A
5150 !       OUTPUT 1 USING 4760;"BIAS VOLTAGE:",Bias_volt(Data_point),"V"
5160 !       OUTPUT 1 USING 4760;"FREQUENCY-",Meas_freq,"Hz"
5170 !       OUTPUT 1 USING 4760;"CONDUCTANCE-",Conductance(Data_point),"pF"
5180 !       OUTPUT 1 USING 4760;"CAPACITANCE-",Capacitance(Data_point),"pF"
5190 !       Answer$=""
5200 !       INPUT "HIT ENTER TO CONTINUE, ANYTHING ELSE WILL ABORT:",Answer$
5210 !       IF Answer$="" THEN
5220            !
5230            !   ENTER POINT IN FILE
5240            !
5250            FOR I=Pnts TO Slot STEP -1
5260               Meas_fr(I+1)=Meas_fr(I)
```

```
5270              Conduct_m(I+1)=Conduct_m(I)
5280              Capac_m(I+1)=Capac_m(I)
5290           NEXT I
5300           !
5310           Meas_fr(Slot)=Meas_freq
5320           Conduct_m(Slot)=Conductance(Data_point)
5330           Capac_m(Slot)=Capacitance(Data_point)
5340           !
5350           Pnts=Pnts+1
5360           !
5370           !    NORMAL EXIT OF FREQ FILE SUBROUTINE
5380           !
5390           OUTPUT 1;CHR$(12)
5400 !      ELSE
5410 !         OUTPUT 1;"USER ABORT"
5420 !         Error_flag=1
5430 !      END IF
5440     END IF
5450  END IF
5460  !
5470  !   EXIT SUBROUTINE FREQ FILE
5480  !
5490  SUBEND!
```

```
5500  SUB Create_file
5510  !
5520  !*************************************************************************
5530  !  THIS SUBROUTINE CREATE A FREQUENCY FILE FROM INPUT CONDUCTANCE
5540  !                      DATA FILES
5550  !*************************************************************************
5560  !
5570  !   DECLARATIONS
5580  !
5590  OPTION BASE 1
5600  !
5610  COM /Hold_logic/ Infiles$[900],INTEGER No_of_files
5620  !
5630  REAL Bias_voltage
5640  INTEGER Create_flag,Pass_flag,Error_flag,Input_flag
5650  DIM Answer$[10],Infile_spec$[30],File_name$[10]
5660  DIM Drive_spec$[3]
5670  !
5680  !-------------------------------------------------------------------------
5690  !
5700  !   CHECK FOR RE-INITILIZATION OF INTERNAL COM AREA
5710  !
5720  INPUT "IS INTERNAL FILE TO BE RE-INITIALIZED(YES OR NO)?",Answer$
5730  !
5740  IF Answer$="Y" OR Answer$="YES" THEN
5750     Create_flag=1
5760     !
5770     !   GET BIAS VOLTAGE WHICH FILE IS TO BE CREATED AT
5780     !
5790     INPUT "ENTER BIAS VOLTAGE TO BE RECORDED",Bias_voltage
5800  ELSE
5810     Create_flag=0
5820  END IF
5830  !
5840  !    GET INPUT CONDUCTANCE FILES AND PROCESS
5850  !
5860  IF No_of_files<>0 THEN
5870     INPUT "USE LIST OF INPUT FILES FROM LAST CREATION",Answer$
5880     IF Answer$="YES" THEN
5890        Input_flag=0

5900        OUTPUT 1;CHR$(12);"FREQUENCY FILE WILL BE CREATED FROM FOLLOWING FIL
ES:"
5910        FOR I=1 TO No_of_files
5920           Start_byte=(I-1)*30+1
5930           End_byte=Start_byte+29
5940           OUTPUT 1;Infiles$[Start_byte,End_byte]
5950        NEXT I
5960        Answer$=""
5970        INPUT "HIT 'ENTER' TO CONTINUE,ANYTHING ELSE WILL ABORT",Answer$
5980        IF Answer$<>"" THEN SUBEXIT
5990     ELSE
6000        Input_flag=1
6010        Answer$=""
6020        INPUT "RE-INITIALIZE FILE LIST OR ADD NAMES TO EXISTING LIST(R OR A)
:",Answer$
6030        IF Answer$="R" THEN
6040           No_of_files=0
6050           Infiles$=""
```

```
6060        END IF
6070     END IF
6080  ELSE
6090     Input_flag=1
6100  END IF
6110  !
6120  IF Input_flag THEN
6130     REPEAT
6140        File_name$=""
6150        INPUT "ENTER FILE NAME (HIT 'ENTER' TO EXIT):",File_name$
6160        !
6170        IF File_name$<>"" THEN
6180           !
6190           !  PROCESS INPUT FILE
6200           !
6210           No_of_files=No_of_files+1
6220           Drive_spec$=""
6230           INPUT "SPECIFY WHERE FILE EXISTS(L,R,EL,ER):",Drive_spec$
6240           SELECT Drive_spec$
6250              CASE "L"
6260                 Infile_spec$=File_name$&":INTERNAL,4,1"
6270              CASE "EL"
6280                 Infile_spec$=File_name$&":HP82901,703,0"
6290              CASE "ER"
6300                 Infile_spec$=File_name$&":HP82901,703,1"
6310              CASE ELSE
6320                 Infile_spec$=File_name$&":INTERNAL,4,0"
6330           END SELECT
6340           !
6350           Error_flag=0
6360           Pass_flag=Create_flag AND (No_of_files=1)
6370           CALL Freq_file(Infile_spec$,Bias_voltage,Pass_flag,Error_flag)
6380           !
6390           IF Error_flag THEN
6400              No_of_files=No_of_files-1
6410              OUTPUT 1;"FILE ";File_name$;" NOT ADDED TO FREQUENCY FILE"
6420           ELSE
6430              List_flag=1
6440              Start_byte=(No_of_files-1)*30+1
6450              End_byte=Start_byte+29
6460              Infiles$[Start_byte,End_byte]=Infile_spec$
6470           END IF
6480           !
6490        END IF
6500        !
6510        !
6520     UNTIL (File_name$="")
6530  ELSE
6540     FOR I=1 TO No_of_files
6550        !
6560        Start_byte=(I-1)*30+1
6570        End_byte=Start_byte+29
6580        Infile_spec$=Infiles$[Start_byte,End_byte]
6590        Pass_flag=Create_flag AND (I=1)
6600        Error_flag=0
6610        CALL Freq_file(Infile_spec$,Bias_voltage,Pass_flag,Error_flag)
6620        IF Error_flag THEN
6630           OUTPUT 1;"FILE ";Infile_spec$;" NOT ADDED TO FREQ FILE"
6640        END IF
6650     NEXT I
6660  END IF
```

```
6670  !
6680  !   EXIT CREATE FILE SUBROUTINE
6690  !
6700  SUBEND!
```

```
6710  SUB Store
6720  !*************************************************************************
6730  !   SUBROUTINE STORE STORES DATA FROM INTL FILE TO DISK
6740  !*************************************************************************
6750  !
6760  OPTION BASE 1
6770  !
6780  COM /Cond_freq_data/ Bias_voltage,Sample_id$[10],Temp,Dop_typ$[1]
6790  COM /Cond_freq_data/ Pnts,Meas_fr(50),Conduct_m(50),Capac_m(50)
6800  COM /Cond_freq_data/ Conduct_p(50),Capac_p(50)
6810  COM /Cond_freq_data/ Cond_p_fit(100),Cap_p_fit(100),Freq_fit(100)
6811  COM /Cond_freq_data/ Ox_cap,Sigma_s,Dit,Tau_n,Area,Cal_points,Cd
6830  !
6840  DIM Answer$[3],File_name$[10],File_specifier$[30]
6850  !
6860  !  DEFINE PROGRAM STATE TO USER
6870  !
6880  OUTPUT 1;"****************** STORAGE OF DATA FILE ONTO DISK ***************
**************"
6890  !
6900  !  GET FILE NAME
6910  !
6920  INPUT "ENTER FILE NAME TO BE CREATED:",File_name$
6930  INPUT "SPECIFY LEFT OR RIGHT HAND DRIVE(L OR R):",Answer$
6940  IF Answer$="L" THEN
6950     File_specifier$=File_name$&":INTERNAL,4,1"
6960  ELSE
6970     File_specifier$=File_name$&":INTERNAL,4,0"
6980  END IF
6990  !
7000  !   POINT TO DISK FILE
7010  !
7020  CREATE BDAT File_specifier$,6,400
7030  ASSIGN @File TO File_specifier$;FORMAT OFF
7040  !
7050  !   ENTER FILE INTO COM AREA
7060  !
7070  OUTPUT @File,1;Bias_voltage,Sample_id$,Temp,Dop_type$,Pnts
7080  OUTPUT @File,2;Meas_fr(*)
7090  OUTPUT @File,3;Conduct_m(*)
7100  OUTPUT @File,4;Capac_m(*)
7110  OUTPUT @File,5;Conduct_p(*)
7120  OUTPUT @File,6;Capac_p(*)
7130  !
7140  !   EXIT LOAD SUBROUTINE
7150  !
7160  SUBEND!
```

```
7170  !
7180  !
7190  SUB Fit
7200  !*************************************************************************
7210  !    SUBROUTINE FIT CALCULATES PARALLEL CONDUCTANCE AND CAPACITANCE
7220  !       FOR GIVEN VALUES OF AREA,OX_CAP,SIGMA_S,DIT,TAU_N
7230  !*************************************************************************
7240  !
7250  !  DECLARATIONS
7260  !
7270  OPTION BASE 1
7280  !
7290  COM /Cond_freq_data/ Bias_voltage,Sample_id$[10],Temp,Dop_typ$[1]
7300  COM /Cond_freq_data/ Pnts,Meas_fr(50),Conduct_m(50),Capac_m(50)
7310  COM /Cond_freq_data/ Conduct_p(50),Capac_p(50)
7320  COM /Cond_freq_data/ Cond_p_fit(100),Cap_p_fit(100),Freq_fit(100)
7321  COM /Cond_freq_data/ Ox_cap,Sigma_s,Dit,Tau_n,Area,Cal_points,Cd
7340  !
7370  REAL F(101),F4(101)
7380  INTEGER J,L,M,N
7390  REAL Constant,Eata,D_eata,I,Integral
7400  !
7410  !   GET THE FITTING PARAMETERS FROM THE USER
7420  !
7430  INPUT "ENTER OXIDE CAPACITANCE IN PF",Ox_cap
7440  Ox_cap=Ox_cap*1.E-12
7450  INPUT "ENTER SIGMA_S",Sigma_s
7460  INPUT "ENTER DIT in Cm^-2ev^-1",Dit
7470  INPUT "AREA IN Cm^2",Area
7480  INPUT "ENTER TAU_N IN SEC ",Tau_n
7481  INPUT "ENTER DEPLETION CAPACITANCE",Cd
7600  !
7610  !  SET UP FREQ LOOP
7620  J=1
7630  Cal_points=0
7640  FOR I=0 TO 5 STEP .1
7650      !
7660      Freq_fit(J)=10^I
7670      D_eata=.4
7680      !
7690      FOR L=1 TO 51
7700         Eata=-10+(L-1)*D_eata
7710         F1=EXP(-Eata^2/(2*Sigma_s^2))
7720         F2=EXP(-Eata)*LOG(1+4*PI^2*Freq_fit(J)^2*Tau_n^2*EXP(2*Eata))
7721         F3=EXP(-Eata)*ATN(2*PI*Freq_fit(J)*Tau_n*EXP(Eata))
7730         F(L)=F1*F2
7731         F4(L)=F1*F3
7740      NEXT L
7750      !
7760      CALL Integration(51,D_eata,F(*),Integral)
7761      CALL Integration(51,D_eata,F4(*),Integral1)
7770      Constant=Area*1.6E-19*Dit/(2*Ox_cap*2*PI*Freq_fit(J)*Tau_n*SQR(2*PI*Si
gma_s^2))
7780      Cond_p_fit(J)=Constant*Integral

7790      Cal_points=Cal_points+1
7880      J=J+1
7890      NEXT I
7900      SUBEND!
```

```
7910      !
7920      SUB Integration(INTEGER N,REAL Step_size,F(*),Integral)
7930      !*************************************************************************
7940      !   SUBROUTINE INTEGRATION TAKES N# OF POINTS (N ODD) AND USES
7950      !         SIMPSON'S RULE TO INTEGRATE ARRAY F(*)
7960      !*************************************************************************
7970      !
7980      ! Declarations
7990      OPTION BASE 1
8000      INTEGER I
8010      Integral=0
8020      !
8030      FOR I=1 TO N-2 STEP 2
8040          Integral=Integral+(Step_size/3)*(F(I)+4*F(I+1)+F(I+2))
8050      NEXT I
8060      !
8070      SUBEND!
```

```
8080      SUB Cap_graph
8090      !*************************************************************************
8100      ! SUBROUTINE CAP_GRAPH MAKES THE CAPAC VS. w CURVES FOR THE DATA
8110      !*************************************************************************
8120      !
8130      !   DECLARATIONS
8140      !
8150      OPTION BASE 1
8160      !
8170      COM /Cond_freq_data/ Bias_votage,Sample_id$[10],Temp,Dop_type$[1]
8180      COM /Cond_freq_data/ Pnts,Meas_fr(50),Conduct_m(50),Capac_m(50)
8190      COM /Cond_freq_data/ Conduct_p(50),Capac_p(50)
8200      COM /Cond_freq_data/ Cond_p_fit(100),Cap_p_fit(100),Freq_fit(100)
8201  COM /Cond_freq_data/ Ox_cap,Sigma_s,Dit,Tau_n,Area,Cal_points,Cd
8220      !
8230      COM /Cap_graphics/ INTEGER Axes_drawn_cap,Cap_screen(12480)
8231      COM /Cap_graphics/ REAL Left,Right,Top,Bottom
8240      !
8250      !   RE-INITIALIZE GRAPHICS DISPLAY
8260      DEG
8270      CLIP OFF
8280      PIVOT 0
8290      PEN 1
8300      LINE TYPE 1,5
8310      LORG 1
8320      CSIZE 5,.6
8330      LDIR 0
8340      MOVE 0,0
8350      !
8360      IF Axes_drawn_cap THEN
8361         GLOAD Cap_screen(*)
8362         WINDOW Left,Right,Bottom,Top
8370         INPUT "SHOULD DATA BE PLOTTED ON SAME CAP AXES(YES OR NO)?",Answer$
8380      END IF
8390      !
8400      !   SET FLAG TO SEE WHETHER TO DRAW AXES
8410      !
8420      Init=((Axes_drawn_cap AND Answer$="NO") OR NOT Axes_drawn_cap)
8430      IF Init THEN
8440         GCLEAR
8450         Axes_drawn_cap=1
8460         !
8470         !   DEFINE GRAPHIC LIMITS
8480         !
8490         Bottom=-.5
8500         Top=2.5
8510         Left=-.1*6
8520         Right=1.1*6
8530         !
8531      END IF
8532      !
8533      !
8543      WINDOW Left,Right,Bottom,Top
8553      !
8554      IF Init THEN
8555         !
8560         !  TITLE GRAPH
8570         !
8580         CSIZE 6,.6
8590         MOVE (Left+Right)/2,Top
```

```
8600          LORG 6
8610          LABEL "Cp/Cox VS. w CURVE"
8620          !
8630          !   LABEL AXES
8640          !
8650          LDIR 90
8660          CSIZE 4,.6
8670          MOVE Left,(Bottom+Top)/2
8680          LORG 6
8690          LABEL "Cp/Cox"
8700          !
8710          LDIR 0
8720          MOVE (Left+Right)/2,Bottom
8730          LORG 4
8740          LABEL "FREQUENCY"
8750          !
8760          CSIZE 3,.6
8770          LORG 8
8780          FOR I=0 TO 2 STEP .5
8790             MOVE 0,I
8800             LABEL I
8810          NEXT I
8820          !
8830          LORG 6
8840          FOR I=0 TO 6
8850             MOVE I,0
8860             LABEL 10^I
8870          NEXT I
8880          !
8890          !   DRAW AXES
8900          !
8910          CLIP 0,6,0,2
8920          AXES 1,1,0,0
8930          !
8940       END IF
8950       !
8960       INPUT "DO YOU WANT TO PLOT FIT (YES OR NO):",Plot_fit$
8970       !
8980       IF Plot_fit$="YES" THEN
8990          !
9000          !  PLOT FIT
9010          !
9020          PENUP
9030          FOR I=1 TO Cal_points
9040             PLOT LGT(Freq_fit(I)),Cap_p_fit(I)
9050          NEXT I
9060       ELSE
9070          !
9080          !  PLOT DATA
9090          !
9100          PENUP
9101          LORG 5
9102          CSIZE 3,.6
9110          FOR I=1 TO Pnts
9120             MOVE LGT(Meas_fr(I)),Capac_p(I)
9121             LABEL "*"
9130          NEXT I
9140       END IF
9141       !
9142       !   STORE THE CAPACITANCE GRAPHICS SCREEN
```

```
9143        !
9144        GSTORE Cap_screen(*)
9150        !
9160        !   EXIT SUBROUTINE
9170        !
9180    SUBEND!
```

```
9190  SUB Store_fit
9200  !*************************************************************************
9210  !   SUBROUTINE STORE STORES DATA FROM INTL FILE TO DISK
9220  !*************************************************************************
9230  !
9240  OPTION BASE 1
9250  !
9260  COM /Cond_freq_data/ Bias_voltage,Sample_id$[10],Temp,Dop_typ$[1]
9270  COM /Cond_freq_data/ Pnts,Meas_fr(50),Conduct_m(50),Capac_m(50)
9280  COM /Cond_freq_data/ Conduct_p(50),Capac_p(50)
9290  COM /Cond_freq_data/ Cond_p_fit(100),Cap_p_fit(100),Freq_fit(100)
9291  COM /Cond_freq_data/ Ox_cap,Sigma_s,Dit,Tau_n,Area,Cal_points,Cd
9310  !
9320  DIM Answer$[3],File_name$[10],File_specifier$[30]
9330  !
9340  !  DEFINE PROGRAM STATE TO USER
9350  !
9360  OUTPUT 1;"****************** STORAGE OF FIT FILE ONTO DISK ***************
      ***************"
9370  !
9380  !  GET FILE NAME
9390  !
9400  INPUT "ENTER FILE NAME TO BE CREATED:",File_name$
9410  INPUT "SPECIFY LEFT OR RIGHT HAND DRIVE(L OR R):",Answer$
9420  IF Answer$="L" THEN
9430     File_specifier$=File_name$&":INTERNAL,4,1"
9440  ELSE
9450     File_specifier$=File_name$&":INTERNAL,4,0"
9460  END IF
9470  !
9480  !   POINT TO DISK FILE
9490  !
9500  CREATE BDAT File_specifier$,4,800
9510  ASSIGN @File TO File_specifier$;FORMAT OFF
9520  !
9530  !   ENTER FILE INTO COM AREA
9540  !
9545  OUTPUT @File,1;Tau_n,Ox_cap,Sigma_s,Area,Temp,Cal_points,Cd
9560  OUTPUT @File,2;Freq_fit(*)
9570  OUTPUT @File,3;Cond_p_fit(*)
9580  OUTPUT @File,4;Cap_p_fit(*)
9610  !
9620  !   EXIT LOAD SUBROUTINE
9630  !
9640  SUBEND!
```

```
14060 SUB View_cond
14070 !
14071 OPTION BASE 1
14072 !
14080 COM /Graphics/ INTEGER Axes_drawn,Cond_screen(12480)
14090 COM /Graphics/ REAL Top,Bottom,Left,Right
14100 !
14110 GLOAD Cond_screen(*)
14120 SUBEND
14130 SUB View_cap
14140 !
14141 OPTION BASE 1
14142 !
14150 COM /Cap_graphics/ INTEGER Axes_drawn_cap,Cap_screen(12480)
14160 COM /Cap_graphics/ REAL Top,Bottom,Left,Right
14170 !
14180 GLOAD Cap_screen(*)
14190 SUBEND!
```

```
14200 SUB Load_fit
14210 !*****************************************************************************
14220 !   SUBROUTINE LOAD RETRIEVES DATA FROM DISK FILE
14230 !*****************************************************************************
14240 !
14250 OPTION BASE 1
14260 !
14270 COM /Cond_freq_data/ Bias_voltage,Sample_id$[10],Temp,Dop_typ$[1]
14280 COM /Cond_freq_data/ Pnts,Meas_fr(50),Conduct_m(50),Capac_m(50)
14290 COM /Cond_freq_data/ Conduct_p(50),Capac_p(50)
14300 COM /Cond_freq_data/ Cond_p_fit(100),Cap_p_fit(100),Freq_fit(100)
14301 COM /Cond_freq_data/ Ox_cap,Sigma_s,Dit,Tau_n,Area,Cal_points,Cd
14320 !
14330 DIM Answer$[3],File_name$[10],File_specifier$[30]
14340 !

14350 !  DEFINE PROGRAM STATE TO USER
14360 !
14370 OUTPUT 1;"****************** LOAD OF FIT FILE FROM DISK ******************
************"
14380 !
14390 !  GET FILE
14400 !
14410 INPUT "ENTER FILE NAME:",File_name$
14420 INPUT "SPECIFY LEFT OR RIGHT HAND DRIVE(L OR R):",Answer$
14430 IF Answer$="L" THEN
14440    File_specifier$=File_name$&":INTERNAL,4,1"
14450 ELSE
14460    File_specifier$=File_name$&":INTERNAL,4,0"
14470 END IF
14480 !
14490 !   POINT TO DISK FILE
14500 !
14510 ASSIGN @File TO File_specifier$;FORMAT OFF
14520 !
14530 !   ENTER FILE INTO COM AREA
14540 !
14550 ENTER @File,1;Tau_n,Ox_cap,Sigma_s,Area,Temp,Cal_points,Cd
14560 ENTER @File,2;Freq_fit(*)
14570 ENTER @File,3;Cond_p_fit(*)
14580 ENTER @File,4;Cap_p_fit(*)
14610 !
14620 !   EXIT LOAD SUBROUTINE
14630 !
14640 SUBEND!
```

REFERENCES

1. Hofstein, S.R., ED-14, No. 11, (Nov. 1967)

2. Hofstein, S.R. and Warfield, G., Solid State Electronics, 8, 321-341, (1965)

3. Schroder, D.K. and Nathanson, H.C., Scientific Paper, 69-1f6-carts-p2, (April 1969)

4. Sah, C.T., Noyce, R.N. and Schockley, W., Proc IRE, 45, 1228, (1957)

5. Grove, A.S., Deal, B.E., Snow, E.H., and Sah, C.T., Solid State Elec., 8, 145-163, (1965)

6. Zerbst, M., Z. Angew. Phys., 22, 30-33, (1966)

7. Lehovec, K. and Slobodskoy, A., Solid State Elec., 7, 59, (1964)

8. Zaininger, K.H. and Warfield, G., Limitations of the MOS Capacitance Method for the Determination of Semiconductor Surface Properties, IEEE Trans. Elec. Devices, ED-12, 179, (1965)

9. Nicollian, E.H. and Goetzburger A., The Si-SiO_2 Interface Electrical Properties as Determined by the Metal-Insulator-Silicon Conductance Technique, Bell Sys. Tech. Journ., 56, 6, p. 1055

10. Nicollian, E.H. and Brews, J.R., The Physics and Technology of MOS Devices, (1982), John Wiley & Sons

11. Castagne, R. and Vapaille, A., Surface Science, 28, 157, (1971)

12. F.C. Blaha, I.A. Mack, and M.H.White, Characterization of the MIS system with C-V and C-t Measurements, Sol. St. Tech. Lab. Mem., SSTL-70-1

VITA

F. Matthew Rhodes was born on November 7, 1957 in Bellefonte, PA to F. Robert Rhodes and Caroline T. Rhodes. He attended Pennsylvania State University from September 1975 to June 1979 where he recieved a Bachelor of Science, Physics with High Distinction. After undergraduate school, he worked for two years at Pennsylvania Power & Light Company as a Scientific Applications Programmer. In January 1982 he returned to Lehigh University to work on a Master's of Science Degree.