University of Alberta

Experimental apparatus for measurement

of generalized permeability coefficients

using steady and unsteady state,

cocurrent and countercurrent flow methods

by

Samuel Chang

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Science

in

Petroleum Engineering

Department of Mining, Metallurgy, and Petroleum Engineering

Edmonton, Alberta

Fall 1996

ISBN 0-612-18244-4

Psalm 23

Dedication

For anyone who is anxious to find a deeper answer

Ecclesiastes 1 : 2
Ecclesiastes 12:13-14

Abstract

Several authors have developed analytically extended formulations of Darcy's law for immiscible, two-phase flow. These results, which can be described by a matrix of generalized permeability coefficients, have led many researchers to investigate various experimental methods for quantifying the matrix.

While some of the researchers have achieved success in quantifying the matrix using various types of experiments, none of them have concluded theoretically or experimentally that the measured matrix is indeed the fundamental building block needed to describe immiscible, two-phase flow or that, by proper utilization of the matrix, one can construct traditional permeability curves needed for any type of flow which may arise.

To arrive at such a conclusion, one needs to test the non-uniqueness of the measured matrix with objective statistical methods. Statistical methods, however, can be used only on adequate and representative data collected from independent experiments. Furthermore, experimental apparatus and related data preparation procedures have to be constructed in a manner such that the limits of uncertainty

f·· the measured matrix will be small, to ensure that unique conclusions can be drawn.

This study focuses on the design and construction of the apparatus and procedures required to measure, within an acceptable limit of uncertainty, the matrix of generalized permeability coefficients so that unique conclusions can be drawn from future statistical analyses.

Acknowledgments

The completion of this study reflects the grace and the mercy of God. Nevertheless, foundations of the work were laid by many people and different organizations. The work is not completed without giving due credits.

Fish and Wildlife - Law Enforcement Division of **Alberta** which provided the weight measuring device.

Imperial Oil Limited of **Canada** which provided the oil samples.

Natural Sciences and Engineering Research Council of **Canada** which made available the entire funding for the research.

Messrs. **Cameron Bentsen, Douglas Booth, John Cruroski, William Diachuk, Jacques Gibeau**, **Robert Smith** and **Robert Stefaniuk** who provided the technical assistance.

Ms. Auton, Ms. Evans and **Mrs. Whiting** who made life a lot more pleasant with their daily smile.

Professor Wayne H. Griffin who made numerous suggestions and provided the source code of VARSIM.

Dr. Wayne R. Tinga who made numerous suggestions, made available some of the equipment and provided a critique on the manuscript.

Dr. Syed M. Farouq Ali who provided a critique on the manuscript.

Very special thanks and gratitude to **Dr. Ramon G. Bentsen**, who provided the entire theoretical development and methodology presented in this study, under whose guidance this work was carried out and provided a critique on the manuscript. His gray hair multiplied exponentially for each question that I raised, although his heart stayed healthy for each daring act that I attempted.

My **parents**, **sister** and my wife, **Claire,** who shared with me moments of frustration, agony and stress, and who allowed me to wander into the twilight zone of theoretical reasoning when I was most needed for doing housework.

Samuel Chang

Table of contents

List of tables

List of Figures

List of nomenclature

Roman Letters

A	Cross-sectional area of packed core (cm^2)
A_c	Area under capillary pressure curve (psi)
A_{ls}	Absorbency of microwave radiation by miscellaneous factors. In the context of the experiments conducted for this study, they are oil, sand core holder, air space. $(dimensionless)$
A_{sys}	Total absorbency of the system $(dimensionless)$
A_w	Absorbency of water $(dimensionless)$
B	Multiplication factor used for microwave radiation absorbtion measurement technique $(dimensionless)$
BV	Bulk volume of core $(c.c.)$
c	Speed of light in vacuum $(2.99793 \cdot 10^{10}\ cm/sec.)$
$\overline{c}$	Concentration of absorber, in the context of the experiments conducted in this study, the concentration of the absorbtive substance in the path of the microwave beam is equal to $\phi \cdot S_w$
d_1	Distance of screw #1 inserted into the double stub tuner (cm)
d_2	Distance of screw #2 inserted into the double stub tuner (cm)
D	Instantaneous electric displacement
D_{max}	Maximum instantaneous electric displacement
$DV1$	Dead volume of end cap #1 $(c.c.)$

$DV2$	Dead volume of end cap #2 $(c.c.)$
$DW1$	Weight of assembled and packed core holder when dry $(gram)$
E	Instantaneous electrical field strength
$E_{\max}$	Maximum instantaneous electric field strength
$EW1$	Weight of assembled core holder when empty $(gram)$
$E(z)$	Expected value of function z
f	Frequency of microwave radiation (Hz)
f_i	Fractional flow of phase i obtained from cocurrent unsteady-state flow experiment; $i = 1,2$ $(dimensionless)$
f_i^0	Fractional flow of phase i obtained from cocurrent steady-state flow experiment; $i = 1,2$ $(dimensionless)$
f_{1s_1}	Functional equation as a function of time t for any saturation S_1
$f_{1t^{\otimes}}$	Fractional flow curve as a function of saturation S_1 for a specific time $t^{\otimes}$ $(dimensionless)$
$f_{2t^{\otimes}}$	Fractional flow curve as a function of saturation S_2 for a specific time $t^{\otimes}$ $(dimensionless)$
$f_1(S_w,t)$	Fractional flow function of phase 1 $(dimensionless)$
$FW1$	Weight of assembled core holder when filled with water $(gram)$
$FWF1$	Final water fraction in end cap number 2 $(fraction)$

$FWF2$	Final water fraction in end cap number 1 $(fraction)$
g	Gravity acceleration (32.1850 ft / sec.)
I_o	Power of microwave radiation used to energize the core holder, measured at the transmitting antenna $(milliwatt)$
I_{sr}	Dimensionless velocity (instability number) for rectangular system $(dimensionless)$
I_x	Power of microwave radiation after the beam has traveled a distance Lx in the medium, measured at receiving antenna on the opposite side of the core holder $(milliwatt)$
$IWF1$	Initial water fraction in end cap number 1 $(fraction)$
$IWF2$	Initial water fraction in end cap number 2 $(fraction)$
k	Extinction coefficient $(dimensionless)$
k_{1r}	Effective permeability to phase 1 at residual saturation to phase 1
k_{2i}	Effective permeability to phase 2 at irreducible saturation to phase 2
k_{effi}	Effective permeability to phase i ; $i = 1,2$ $(dimensionless)$
K	Total absorption coefficient (cm^{-1}) $= 2 \cdot \omega \cdot k / c$
L_x	Distance traveled by the microwave radiation in the absorbtive medium; thickness of the porous medium (cm)
L_y	Distance in the direction of flow, length of core (cm)

L_z Width of the core (cm)

M End point mobility ratio $(dimensionless)$

$$= \frac{k_{1r} \cdot \mu_2}{\mu_1 \cdot k_{2r}}$$

n Refractive index

N_c Capillary number

N_g Gravity number $(dimensionless)$

N_p Total volume of oil produced $(c.c.)$

P Pressure (psi)

P_c Macroscopic static capillary pressure (psi)

P_i Pressure for phase i ; $i = 1,2$ (psi)

PV Pore volume of core $(c.c.)$

$\frac{\partial P_i}{\partial L_y}$ Pressure gradient in the direction of flow of phase i obtained from cocurrent unsteady-state flow experiment; $i = 1,2$ $(psig/cm)$

$\frac{\partial P_i^0}{\partial L_y}$ Pressure gradient in the direction of flow of phase i obtained from cocurrent steady-state flow experiment; $i = 1,2$ $(psig/cm)$

$\frac{\partial P_i^*}{\partial L_y}$ Pressure gradient in the direction of flow of phase i obtained from countercurrent steady-state flow experiment; $i = 1,2$ $(psig/cm)$

q	Total volumetric flow rate $(c.c./hr.)$
q_i	Volumetric flow rate of phase i ; $i = 1,2$ $(c.c./hr.)$
Q_1	Portion of wetting-phase fluid that is inside the core between the initial saturation S_{1i} and S_1 (ft^3)
Q_{1s_1}	Portion of wetting-phase fluid that is inside the core between the initial saturation S_{1i} and a specific S_1 as a function of t (ft^3)
Q_{1t_i}	Portion of wetting-phase fluid that is inside the core between the initial saturation S_{1i} and S_1 at a particular time t_i (ft^3)
Q_{1t}	Portion of wetting-phase fluid that is inside the core between the initial saturation S_{1i} and S_1 at any time t (ft^3)
$Q_{1t^\otimes}$	Portion of wetting-phase fluid that is inside the core between the initial saturation S_{1i} and S_1 at a specific time $t^\otimes$ of interest (ft^3)
R_{12}	Function relating the pressure gradient in phase 1 to that in phase 2 $(dimensionless)$
S_i	Saturation of phase i ; $i = 1,2$ $(fraction)$
S_{1m}	Maximum saturation of phase 1 $(fraction)$
$S_1^\otimes$	Saturation of phase 1 at a specific time $t^\otimes$ of interest $(fraction)$
S_{1i}	Irreducible saturation to phase 1 ; Initial saturation to phase 1 $(fraction)$
S_{2r}	Residual saturation to phase 2 $(fraction)$

t	Time $(sec.)$; Total experimental time $(sec.)$; Injection time $(sec.)$
t_i	Any particular time $(sec.)$
$t^{\otimes}$	A specific time of interest $(sec.)$
t_{∞}	Injection time when incremental production of water is infinitesimal $(sec.)$
T	Ambient room temperature $(deg.Celsius)$
TVP	Total volume of fluid produced $(c.c.)$
υ	Total injection fluids velocity $(cm/sec.)$
υ_i	Darcy velocity to phase i ; $i = 1,2$ $(cm/sec.)$
υ_i^0	Darcy velocity to phase i as obtained from cocurrent, steady-state flow; $i = 1,2$ $(cm/sec.)$
x	Distance of saturation S_1 traveled (cm)
$x(S_1,t)$	Distance of saturation S_1 measured from the inlet end of the core as a function of time t (cm)
$x^{\otimes}$	Distance of a specific saturation $S_1^{\otimes}$ measured from the inlet end of the core at a specific time $t^{\otimes}$ of interest (cm)
x_{t_i}	Position of saturation S_1 at any particular time t_i (cm)
$x_{1t^{\otimes}}$	Position of saturation S_1 at a specific time $t^{\otimes}$ of interest (cm)
w_{cf}	Width of capillary fringe (cm)

W — Total weight of produced fluid $(gram)$

$WW1$ — Weight of assembled and packed core holder when 100 % saturated with water $(gram)$

Greek Letters

α — Relaxation time spread

δ — Phase angle lag between E and D

Δt — Time error introduced by microwave radiation scanning of core $(sec.)$

Δx — Correction of distance traveled by saturation (cm)

ε — $= \frac{D}{E}$ $(dimensionless)$

ε^* — Complex relative dielectric constant $(dimensionless)$

ε' — Relative dielectric constant $(dimensionless)$

ε'' — Relative dielectric loss factor $(dimensionless)$

ε_∞ — Infinite frequency dielectric constant $(dimensionless)$

ε_s — Static dielectric constant $(dimensionless)$

λ_{ij} — $= \frac{k_{ij}}{\mu_j}$, generalized mobility of phase i ; $i, j = 1,2$

λ_i — $= \frac{k_{effi}}{\mu_i}$, effective mobility ratio of phase i ; $i = 1,2$

λ_i^0	$=\frac{k_{effi}}{\mu_i}$, effective mobility ratio of phase i obtained from cocurrent steady-state flow experiment; $i = 1,2$
λ_i^*	$=\frac{k_{effi}}{\mu_i}$, effective mobility ratio of phase i obtained from countercurrent steady-state flow experiment; $i = 1,2$
λ^{-1}	Inverse mobility ratio $(atm.sec./ft^2)$
λ	Wavelength of the microwave radiation (cm)
μ_i	Viscosity of phase i ; $i = 1,2$ (cp)
$\mu_k(z)$	k^{th} central moment of function z
ρ_i	Density of phase i ; $i = 1,2$ $(gram/c.c.)$
σ_{eb}	Bulk frontal tension $= A_c \cdot \phi \cdot (1 - S_{1i} - S_{2r}) \cdot w_{cf}$
τ	Relaxation time $(sec.)$
ϕ	Porosity $(fraction)$
ω	Angular frequency $(sec.^{-1})$ $= 2 \cdot \pi \cdot f$

I. Introduction

I.A. Development of fluid flow theory

Darcy's law (Darcy, 1856) is valid only for single-phase flow. For polyphasic flow, it is usual to extend empirically Darcy's law by introducing a permeability reduction factor (Wyckoff and Botset, 1936), traditionally termed relative permeability, to account for the fact that permeability, when more than one phase is flowing, is saturation dependent.

As pointed out by Rose (Rose, 1972), implicit in this approach is the assumption that, when two immiscible fluids flow through a porous medium, each of the fluids comprises a new porous medium for the other. Furthermore, this approach assumes that boundary conditions at fluid/fluid and fluid/solid interfaces should act in the same way, thus implicitly imbedding in the formulations of relative permeability theory the simplifying but unproved postulate that coupling effects between the fluid phases play no important role in modifying the polyphasic flow phenomena, and that the flux of each phase is proportional to only one driving force, the potential gradient acting across the phase. Thus, the possibility exists that viscous coupling effects between the two phases flowing through a porous medium may affect the flux of each phase. Consequently, a more realistic assumption can be made; that is, the flux of each phase depends linearly on both of the driving forces acting on the phases.

However, a more rigorous understanding of the dynamics of polyphasic flow, in terms of pore scale phenomena, could not be considered at this early stage because the monophasic form of Darcy's law was only understood as an empirical relationship. A number of researchers (Hubbert, 1956; Hall, 1956; Scheidegger, 1974) subsequently speculated that Darcy's law was illustrating the average behavior of the well-understood flow dynamics at the pore scale, the Navier-Stokes equation, and the effects of the complex boundary between the fluid and solid medium. The main obstacle, in these analyses, to a clear formulation of the problem was how to relate the average effect of the Navier-Stokes differential equation at the pore scale to the Darcian description that expressed the problem as a differential equation of average quantities.

A breakthrough came when a spatial averaging theorem was presented independently by Anderson and Jackson (Anderson and Jackson, 1967), Slattery (Slattery, 1967) and Whitaker (Whitaker, 1967) in 1967. This mathematical result expresses the volume average of the spatial derivative of a microscopic (pore-scale) quantity with the derivative of the volume-averaged quantity plus a surface integral over the interfaces between the fluid and solid. As a result, the conventional form of Darcy's equation was constructed from a theoretical standpoint (Slattery, 1969; Neuman, 1977; Lehner, 1979; Whitaker, 1986a).

Further to the theoretical derivation of Darcy's equation, the procedure of volume-averaging was generalized to polyphasic flow (Slattery, 1970; de la Cruz and Spanos, 1983; Whitaker, 1986b; Torres, 1987; Whitaker, 1987) and the polyphasic form of Darcy's law was developed theoretically. Their results show a system of two equations that involves four parameters, generalized permeabilities, that are related to viscous drag phenomena. Subsequently, Kalaydjian (Kalaydjian, 1987; Kalaydjian, 1990) developed an extended formulation of Darcy's law on the basis of Stokes' equations. This formulation leads, starting from mass and momentum-balance equations through results borrowed from the thermodynamics of irreversible processes, to a matrix of relative permeabilities. The nondiagonal coefficients of this matrix are due to the viscous coupling exerted between the fluid phases, while the diagonal coefficients of this matrix represent the contribution of both the fluid phases to the total flow, as if they were alone. Rose (Rose, 1990; Rose, 1993) presented analyses that support this model, based on the earlier analytical work by Yuster (Yuster, 1951).

In recent years, the focus has shifted to the extended formulations of Darcy's law. Rose (Rose, 1988a) found, through a theoretical development, that the equations of Buckley and Leverett can just as well be employed to describe the more generalized oil recovery processes that are controlled by the complex coupling phenomena related to the reciprocal drag exerted interstitially by the contiguous immiscible fluid streams across the local fluid/fluid interfaces of contact. Furthermore, Rose (Rose, 1988b) proposed that two types of experiments involving two measurements apiece are needed at each fluid saturation condition to provide the necessary and sufficient information by which unsteady as well as steady-state transport processes can be established and characterized. Bourbiaux and Kalaydjian (Bourbiaux and Kalaydjian, 1990) conducted experimental work to investigate the generalized permeability coefficients and concluded that the new formulation of Darcy's law that uses a matrix of mobilities agrees with experimental results. Furthermore, they found that the relative permeabilities that enabled a good prediction of the countercurrent oil recovery rates are about 30% less than the conventional cocurrent relative permeabilities at a given water saturation level. Subsequently, work by Bentsen and Manai (Bentsen and Manai, 1991; Bentsen and Manai, 1993) further substantiated these conclusions and revealed, in addition, that the nondiagonal terms appear to be unequal, in contradiction to the results obtained for simple analogues of flow through porous media (Kalaydjian, 1990). Utilizing an analytical procedure outlined by Mannseth (Mannseth, 1991), Bentsen (Bentsen, 1994) conducted further investigations and concluded that the nondiagonal terms cannot be equal to one another, based on the experimental work conducted previously (Bentsen and Manai, 1991). More recently, Liang and Lohrenz (Liang and Lohrenz, 1994) investigated further the measurement of transport coefficients and proposed the use of a combination of steady-state and unsteady-state experiments, thus overcoming once again the difficulties associated with measuring the gravitational effect, and the difference in the vertical and horizontal velocities, which limited the ability of the method proposed by Rose (Rose, 1988b).

I.B. Published experimental methods

Rose (Rose, 1988b) proposed using a combination of a horizontal flow experiment and a vertical flow experiment to obtain the necessary parameters for the calculation of the generalized permeability coefficients. Although Rose did not perform any experiments, a probable route to arrive at the generalized permeability coefficients was outlined. Subsequently, Bourbiaux and Kalaydjian (Bourbiaux and Kalaydjian, 1990; Kalaydjian, 1990) used a vertical core sample under various boundary conditions to generate cocurrent and countercurrent flows. One set of the experiments used cocurrent flow with no capillary pressure gradient and negligible gravity effect, while the other set employed pure countercurrent flow to investigate further the theoretical model that Kalaydjian (Kalaydjian, 1990) developed earlier. This is the first experimental verification of the new theoretical model. Another experimental success was achieved by Bentsen and Manai (Bentsen and Manai, 1991; Bentsen and Manai, 1993) by using a combination of a steady-state, cocurrent flow experiment and a steady-state, countercurrent flow experiment. They eliminated the difficulties of measuring the gravitational effect and the horizontal and vertical flow rate difference in the methods proposed by Rose (Rose, 1988b). More recently, Liang and Lohrenz (Liang and Lohrenz, 1993) proposed using a combination of steady- and unsteady-state displacement experiments to obtain the necessary parameters for calculations of the generalized permeability coefficients.

II. Statement of the problem and objectives

II.A. Motivation of the current study

While several researchers (Bourbiaux and Kalaydjian, 1990; Bentsen and Manai, 1991; Bentsen and Manai, 1992) have achieved some success in quantifying the matrix of generalized permeability coefficients using various types of experiment, none have concluded theoretically or experimentally that the measured matrix of coefficients is indeed the fundamental building block needed to describe a given two-phase system or that, by proper utilization of the matrix of coefficients, one can construct the traditional permeability curves needed for the description of the system under any flow conditions.

Moreover, the experiments needed to determine the matrix of generalized permeability coefficients are tedious and difficult to carry out accurately, and the experiments carried out to date at the University of Alberta have revealed that the apparatus currently employed to measure the coefficients is far from satisfactory. In particular, there are problems with the statistical uncertainty of the measurements of saturation and with the utility of the current data acquisition system (Sarma and Bentsen, 1989). Thus, even if the measured values of the experimental variables randomly deviate only slightly from those that would be indicated by unflawed instrumentation and methodologies, these fluctuations are sometimes large enough to render useless otherwise straightforward calculations based on the data (Rose, 1989).

However, a survey of the literature and the available apparatus revealed that improvements can be made to techniques (Levine, 1954; Parsons, 1975; Bentsen and Saeedi, 1981; Sarma and Bentsen, 1989) and the apparatus (Bentsen and Saeedi, 1981) developed earlier, such that the curves of the generalized permeability coefficients can be constructed, with reduced statistical uncertainties on any measured parameters.

II.B. Constraints on the problem

To draw a valid conclusion on whether the matrix of generalized permeability coefficients can be viewed as the fundamental building block for any fluid flow model, one needs to test the non-uniqueness of the matrix of coefficients with statistical methods. To further define the problem, the standard method of Analysis of Variance (ANOVA) is chosen as the tool to be used for any future analyses.

The method of ANOVA requires that at least two independent sets of the generalized permeability coefficients, each set being arrived at by statistically independent experiments, to be available. Furthermore, the data that constitute the sample space used for the analyses have to be both adequate and representative, if unique conclusions are to be drawn.

Since the new formulation of the two-phase flow equations involves a matrix of four generalized permeability coefficients, a pair of experiments is required to determine the coefficients. To meet the requirements of ANOVA, two pairs of experiments are required to generate the two sets of statistically independent coefficients. To further define the problem, the two pairs of experiments are selected to be steady-state cocurrent and steady-state countercurrent flow experiments, and steady-state cocurrent and unsteady-state cocurrent flow experiments.

In order for the data collected to be adequate for the current study, data for the full saturation range must be available. If the data collected is to be representative for the current study, the statistical uncertainty of the data collected has to be kept within acceptable limits. To further define the problem, it is proposed that the experimental apparatus and procedures be designed so that data for the full saturation range can be collected within acceptable limits of experimental uncertainty. Furthermore, it is proposed that the limits of uncertainty for any measured parameters are to be determined using the Transformation of Moment technique.

Furthermore, in order to limit the scope of fabrication required for the current study, it is proposed that components currently available at the University of Alberta, with any necessary modifications, be used to construct the experimental apparatus required for the collection of data.

Finally, data collected by different people should be consistent. Consistency in data collection can be achieved using a computer program to control the experimental process and to collect the data. It is proposed that a computer program in modular format be written, with sufficient flexibility to allow users to modify the experimental process whenever it is necessary.

II.C. Objectives of the current study

It is the purpose of this study to design a set of valid and coherent experimental methodologies and procedures, along with the necessary experimental apparatus, for collecting the necessary data under the constraints set forth above. It is also the purpose of this study to demonstrate that the limits of statistical uncertainty of any parameters measured by using the designed procedures and apparatus have been improved over the existing generation of apparatus and procedures, and in particular are kept within acceptable limits. Finally, it is the purpose of this study to construct and to fabricate the necessary apparatus, along with the computer program, for the above purpose so that future experimental investigations can be conducted.

II.D. Overview of the existing roadblocks

In order to use the equipment available at the University of Alberta to construct the necessary apparatus under the guidelines of the current study, the shortcomings of the equipment have to be analyzed. The following is a summary of the findings.

II.D.1. Microwave power source malfunction

The existing klystron oscillator had only one thousand hours of useful operating life. The oscillator broke down. Moreover, because the oscillator was originally built for the K band, a frequency band that was developed mainly for military purposes, an equivalent replacement could not be found on the open market.

II.D.2. High attenuation level on microwave power

To utilize the microwave power measurement technique for the in-situ measurement of the saturation level of the core under study, adequate attenuated microwave power must be detectable under the least favorable condition; that is, at one hundred percent water saturation level in the core (Parsons, 1975) in order to construct the required saturation calibration curve. It was observed that, when using the existing experimental apparatus, and with the core fully saturated with water, the attenuated microwave power was below that which could be detected by the apparatus. This observation, however, is not consistent with the existing experimental results (Islam and Bentsen, 1986; Sarma and Bentsen, 1989; Bentsen and Manai, 1991) in that microwave power could be detected when the core was fully saturated with water. After extensive investigations were conducted, it was found that this observation is consistent with the results obtained by Bentsen and Saeedi (Bentsen and Saeedi, 1981), all the physical evidence available and is internally consistent with theoretical calculations on microwave attenuation characteristics. No explanation can be offered for this contradiction.

II.D.3. High equipment noise level on power meter

The specifications of one of the two power meters to be used in this study (Hewlett-Packard 436A manual) stipulated that it could detect microwave power as low as one microwatt. However, after all the equipment had been coupled together, it was found that a realistic expectation of the lower bound of power detection had to be raised to avoid spurious equipment noise and intermittent locking up of the system. The problem was acknowledged by the manufacturer of the power meter but no solution could be offered. This problem affected the statistical uncertainty, the resolution and the range of the saturation levels measured.

II.D.4. Long sensor saturation time

Errors were introduced into the dynamic saturation profiles of the core taken along its length, when using the existing apparatus. The errors, although unavoidable, can be minimized, either by increasing the number of energizing and detecting units or by limiting the time allowed for each stationary measurement of the saturation level taken along the length of the core. The minimum time allowed, however, has to be long enough to allow power sensors to attain the proper sensor saturation level.

It was found that below one hundred microwatts, the sensor saturation time required was one second and longer for each measurement taken (Hewlett-Packard 436A manual). The cumulative time, however, amounted to one hundred seconds or more for a complete scanning of the core along its length. Because it was observed that the system noise was higher than expected, a tradeoff has to be made between proper utilization of the equipment and conformity to the experimental purposes.

II.D.5. High irregular reflection of microwave power from core holder

It was observed that the when the core was fully saturated with water, the impedance mismatch between the generator and the load caused a high percentage of reflected microwave power that could not be quantified and recorded by the existing apparatus. Moreover, the reflection was not consistent along the length of the core, possibly due to the local heterogeneities in the fabricating material, Fiberglas, as reflection and diffraction combined with the coherent output of the microwave generator creates standing waves. This violated the fundamental assumption required for using the microwave power measurement technique; that is, a known and constant microwave power should be used to energize the microwave power absorbing material under study.

II.D.6. Equipment damage

The directional couplers for both the transmitting and receiving antennas were found to be bent. Moreover, one of the couplers was found to have burnt out, thus reducing the directivity of the directional coupler.

One of the power sensors was found to have burnt out and was unable to detect any microwave power.

One of the two positive displacement fluid injection pumps was found to provide a variable flow rate when operated. The flow rate followed no particular pattern of behavior. It was determined that the displacement cylinders were trapping air when operated. The other pump developed the same symptom of injecting fluid with a variable rate. It was determined, however, to have a slipping gear.

One of the two horn antennas was found to be damaged, with microwave power leaking in an uncontrolled manner from the side of the antenna.

One of the two injection caps for the core holder was found to have cracks, with fluid leaking uncontrollable from the side.

II.D.7. Equipment interfacing problem

The existing apparatus requires the interfacing of five electronic instruments from four vendors. To install them all, it was found that both hardware and software capabilities were exceeded on the available computing system. Moreover, it was found that one of the limiting-switches for the safety shutdown of the traveling core prohibited proper operation of the computing system when connected. Proper equipment has to be constructed and appropriate interfacing methods have to be selected.

II.D.8. Pressure drifting

It was observed that the pressure profiles measured with water-wet and oil-wet pressure transducers drifted when the measurements were taken over long periods of time under dynamic fluid conditions. This phenomenon has the potential of causing erroneous inferences regarding wettability to be drawn as well as affecting the accuracy of the permeabilities and dynamic capillary pressures measured for the core under study.

II.D.9. Wettability changing

It was observed that the non-wetting fluid used, LAGO, which contained both paraffinic and aromatic compounds, deposited an unknown substance on the core after an extended period of flow. This has the potential of changing the wettability of the core; this violates the required assumption that the wettability of the core be invariant with time (Jennings, 1957).

II.D.10. High statistical uncertainty with material balance calculations

It was observed that the material balance calculations to determine the saturation level of the core incurred high statistical uncertainty, mainly because of the design of the volumetric measurement apparatus used. The magnitude of the uncertainty grew with each additional experimental step taken to attain the irreducible oil saturation level. A trade off has to be made between the number of experiments required to acquire adequate data for the analyses and the maximum cumulative uncertainty allowed in the material balance calculations.

II.D.11. Limitation of Ruska pumps

The existing configurations of the Ruska injection pumps allowed only a stepwise increase in injection rate. Moreover, the pumps could not provide an appropriate range of injection rates required for the current study.

II.D.12. Limitation of AC advancing motor

It was observed that the AC voltage advancing motor installed on the traveling track could not provide a synchronized movement with the microwave power recording process. This introduced error into the saturation profile measurement, particularly when the core was under dynamic fluid flow conditions.

II.D.13. Limitation of pressure transducer calibration accuracy

The existing apparatus used for the calibration of pressure transducers could not provide an adequate resolution of the reference pressure for the current study. Moreover, it could not be used to calibrate all the pressure transducers at the same time. This shortcoming introduced additional statistical uncertainties into permeability and dynamic capillary pressure calculations due to slight differences that exist between the calibrated transducers.

II.D.14. Selective wetting capillary barriers

It was observed that the selective wetting capillary barriers used for enabling a phase-sensitive pressure measurement of the injected fluids introduced a time lag in the pressure measurements between individual pressure transducers under dynamic conditions. This shortcoming introduced additional errors into the pressure profiles measured for each phase.

II.D.15. Hydrostatic head difference

It was observed that the existing configuration of the core holder introduced a hydrostatic head difference between the measured pressure of each phase, because of the vertical configuration of the transducers. The hydrostatic head, although it could be eliminated during the analyses of the data, could be best eliminated by designing a core with a horizontal configuration for the pressure transducers.

II.D.16. Ambient temperature effect

It was observed that the power sensors used, thermistor mount, could be affected by ambient temperature. This introduced statistical uncertainty into the measured microwave power that could not be eliminated by numerical methods.

It was also observed that the ambient temperature could affect the accuracy of the permeability measurement to an extent that could affect the required measurement accuracy for the current study.

II.D.17. Antenna alignment

It was observed that the measurement of the microwave power was sensitive to the proper alignment of the transmitting and receiving antennas to the extent that precision of the order of one-tenth of a centimeter in any parallel movements and one degree of angle in any inclinations was required.

III. Theory and techniques

The focus of this study is centered on the notion of establishing a set of coherent experimental methodologies such that further experimental work can be carried out in the future to investigate the validity of viewing the generalized permeability coefficients as the fundamental building blocks for any two-phase transport processes. The methodologies, furthermore, are dictated by the two-phase transport theory, the technique used to obtain certain experimental parameters, and the statistical uncertainties of the parameters to be measured by using such methodologies and the apparatus to be constructed. The following is a summary of the pertinent theories and techniques to be used as building blocks for the current study.

III.A. Two-phase transport theory (Kalaydjian, 1987; Kalaydjian, 1990; Bentsen, 1994)

While there are many different theoretical developments of two-phase transport theory, each with its respective merit, the extended formulation of Darcy's law for two-phase flow developed by Kalaydjian is the starting point of investigations for the current study.

III.A.1. Generalized permeability coefficients

On the basis of Navier-Stokes equations written at the pore scale and results derived from the theory of irreversible thermodynamic processes, an extended formulation of Darcy's law for two-phase flow was developed by Kalaydjian (Kalaydjian, 1987; Kalaydjian, 1990),

$$\begin{bmatrix} \vec{\upsilon}_1 \\ \vec{\upsilon}_2 \end{bmatrix} = -\begin{bmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{21} & \lambda_{22} \end{bmatrix} \cdot \begin{bmatrix} gr\vec{a}d(P_1) - \rho_1 \cdot \vec{g} \\ gr\vec{a}d(P_2) - \rho_2 \cdot \vec{g} \end{bmatrix},$$

where the crossterms, λ_{ij}, $i \neq j$, express the influence of the pressure gradient in one phase on the velocity of the other phase.

This extended formulation revealed, at the macroscopic scale, a potential effect of viscous coupling between the two flowing phases on the two-phase transport processes. Furthermore, it led to a matrix of four mobilities, termed generalized permeability coefficients, that can only be resolved by experimental investigations.

In idealized porous media, it has been shown that two of these mobilities, the nondiagonal mobilities that represent the viscous coupling exerted between fluid phases, are equal (Kalaydjian, 1990). However, in real porous media, it has been demonstrated (Bentsen, 1994) that such is not the case by using an approach suggested by Mannseth (Mannseth, 1991) on the experimental data measured by Bentsen and Manai (Bentsen and Manai, 1991; Bentsen and Manai, 1993).

For the case of cocurrent flow, the displacement of one given liquid phase contributes positively to the flow of the other phase because the pressure gradients are oriented in the same direction; that is, viscous coupling has a positive driving effect on cocurrent two-phase flow. On the other hand, the pressure gradients of the two phases have opposite signs in the case of countercurrent flow, and the displacement of one phase counteracts the flow of the other phase; that is, viscous coupling retards countercurrent flow.

III.A.2. Conformity of the fluid flow theory with the experimental apparatus

The critical assumption (Kalaydjian, 1987; Kalaydjian, 1990) made in the extended formulation of Darcy's law is that the porous medium is strongly water wet and that a water film covers the solid surfaces preventing any interaction between the non-wetting and solid phases, in addition to the usual assumptions of incompressible and immiscible fluids being used and that the porous medium is homogeneous and isotropic. To meet these conditions, pure Ottawa sand is used for core packing and a stable and non-reacting oil (Sarma and Bentsen, 1989), LAGO, is used as the non-wetting phase. Furthermore, density and viscosity variations of both injecting fluids were measured by a commercial laboratory in accordance with the methods outlined by the American Society for Testing and Materials, and the ambient temperature was monitored in the laboratory to allow for any temperature correction to be made on measured parameters, in real time.

III.B. Microwave power measurement technique (Parsons, 1975)

The static and dynamic saturation profile measuring method required for this study uses the non-invasive microwave power measurement technique originally described by Parsons in 1975 (Parsons, 1975). This technique is based upon the selective absorption of the microwave power by water according to a modified Beer-Lambert-Bouguer absorption law, allowing the in-situ water saturation level in a porous medium to be determined.

Microwaves are electromagnetic radiation. The absorption of energy as any electromagnetic radiation travels through some absorbing materials can be described with either dielectric or optical phenomena. The materials that the current study are concerned with are non-magnetic and are related only to the electric field-medium interactions. The following is a summary of some pertinent concepts that dictate the designs of the experimental apparatus and procedures.

III.B.1. Complex dielectric constant

When a non-magnetic material is placed in an alternating electric field, the electric charge displacement of the material will oscillate with the electric field frequency, but the two vectors, the electric field vector and the displacement vector, may differ by some phase angle, δ. The instantaneous field strength and electric displacement are described, respectively, by

$$E = E_{max} \cdot cos(2 \cdot \pi \cdot f \cdot t)$$

and

$$D = D_{max} \cdot cos(2 \cdot \pi \cdot f \cdot t - \delta).$$

If $\delta = 0$, so that the vectors E and D are in phase, the relative dielectric constant is defined by $\varepsilon = D/E$. When $\delta \neq 0$, so that the vectors E and D are not colinear, the relative dielectric constant is considered in two parts. That is,

$$\varepsilon' = \frac{D_{max} \cdot cos\delta}{E_{max}}$$

and

$$\varepsilon'' = \frac{D_{max} \cdot sin\delta}{E_{max}},$$

where ε' represents the normal relative dielectric constant, or the proportionality constant between E_{max} and D_{max} when they are in phase with the field and ε" represents the relative loss factor, or the proportionality constant between E_{max} and D_{max} when they are ninety degrees out of phase.

The relative dielectric characteristics of a homogeneous lossy medium (a medium whereby when a plane wave travels through it, some of its energy is absorbed and is turned into heat, with the consequence that the plane wave is attenuated and its amplitude decreases with distance traveled in the medium) can then be expressed in terms of a complex relative dielectric constant, ε *, defined by

$$\varepsilon^* = \varepsilon' - \varepsilon'' \cdot i,$$

where $i = \sqrt{-1}$ and ε' and ε" are both functions of the material, the environment and the field frequency under consideration.

III.B.2. Lambert and Bouguer law for a homogeneous, single-component system

When a parallel beam of monochromatic electromagnetic radiation passes rectilinearly through a homogeneous medium that does absorb some of the energy, Maxwell's equations can be solved for the transmitted power. Neglecting any reflection, this can be expressed as

$$I_x = I_o \cdot e^{(-K \cdot L_x)}, \quad \text{(M-1)}$$

where

$$K = \frac{2 \cdot \omega \cdot k}{c}$$

and

$$\omega = 2 \cdot \pi \cdot f .$$

That is, the change in the intensity of the radiation transmitted through the absorbing medium is related exponentially to the thickness of the absorbing medium and a constant that depends on the material and the wavelength of the radiation. The exponential power relation is merely a derivation from first principles of an experimentally determined spectroscopic law attributed to Lambert and Bouguer (Mellon, 1950).

III.B.3. Beer's law for a homogeneous, multi-component system

For homogeneous, multi-component mixtures, to a first approximation, Beer proposed that each molecule contributes to the attenuation of the electromagnetic wave as if the other molecules were not present. When the absorptive ability of one component far overshadows the others with a much larger extinction coefficient, the total absorption in a sample changes exponentially with the concentration of the component in the path. This is expressed in mathematical format as

$$log\left(\frac{I_x}{I_o}\right) \propto -\bar{c} . \tag{M-2}$$

Strictly speaking, this relation is a limiting case for low concentrations, and deviations are often observed due to solvent-absorber interactions, absorber-absorber inter-actions at high concentrations and changes in the index of refraction with concentration (Mellon, 1950) In short, the Beer law is not a general relation and must be validated for each particular application.

III.B.4. Adaptation of the Beer-Lambert law to the microwave power measurement technique

Parsons (Parsons, 1975) has proposed using the combined Beer-Lambert-Bouguer law to obtain a relation for determining water saturation level in porous media by the microwave power measurement technique. Thus, Equation (M-1) can be modified, in light of Equation (M-2), as

$$I_x = I_o \cdot e^{(-K \cdot \bar{c} \cdot L_x)} . \tag{M-3}$$

That is, the absorption coefficient, K, for the absorber in a homogeneous and electromagnetically transparent solvent, has to be multiplied by the molar concentration, $\bar{c}$, of the absorber to account for the fact that not all the materials in the path traversed by the microwave radiation absorb the radiation.

It is worthwhile to mention here that the concentration, $\bar{c}$, is analogous to the water saturation level in that an elemental volume is being traversed by a plane, monochromatic beam of radiation. The remaining substances, such as the oil, the Ottawa sand and the Plexiglas used in the study, are almost transparent to the microwave radiation at 28 GHz. Thus,

$$\bar{c} = \text{concentration of absorber}$$

$$= \frac{\text{moles of absorber}}{\text{total moles of mixture}}$$

$$= \frac{\text{moles of water molecules}}{\text{total moles of mixture}}$$

which is being approximated by

$$\approx \frac{\text{volume of water molecules}}{\text{total volume of mixture}}$$

$$\approx \frac{\text{pore volume} \cdot \text{water volume}}{\text{bulk volume} \cdot \text{total liquid volume}}$$

$$\approx \phi \cdot S_W .$$

Thus Equation (M-1) can be re-written as

$$I_x = I_o \cdot e^{\left(\frac{-4 \cdot \pi \cdot f \cdot k \cdot \bar{c} \cdot L_x}{c}\right)}, \tag{M-4}$$

where $\bar{c} = \phi \cdot S_W$. (M-5)

In summary, when the absorptive ability of one component far overshadows the others with a much larger extinction coefficient, then the total absorption in a sample is a direct function of the number of those molecules in the radiation path and the above equation would be adequate to describe the relation of the microwave power level and water saturation level in the system. The absorbency, A_w, caused by the water may then be expressed as

$$A_w = log\frac{I_x}{I_o} \quad \text{(M-6)}$$

Combining Equation (M-4) with Equation (M-6), one can demonstrate that

$$A_w = \frac{-4 \cdot \pi \cdot f \cdot k \cdot \bar{c} \cdot L_x}{2.303 \cdot c} \quad \text{(M-7)}$$

In view of Equation (M-5), one can demonstrate further that

$$A_w = B \cdot S_w, \quad \text{(M-8)}$$

where

$$B = \frac{-4 \cdot \pi \cdot f \cdot L_x \cdot k \cdot \phi}{2.303 \cdot c} \quad \text{(M-9)}$$

III.B.5. Further approximation of the Beer-Lambert-Bouguer law

In reality, Plexiglas, oil, sand and moisture in the air space between the antennas do absorb some of the microwave energy. If more than one component in the mixture absorbs the microwave power according to the Beer-Lambert-Bouguer law, the absorbencies are additive. As there will be reflection, refraction, diffraction, standing wave interference within the system and loss of radiation into the air space, the over-all absorbency of the system may be considered in two parts: water and everything else. That is,

$$A_{sys} = A_{ls} + A_w \text{ or, in view of Equation (M-8), } A_{sys} = A_{ls} + B \cdot S_w$$

and the Beer-Lambert-Bouguer law may be further modified as

$$log\frac{I_x}{I_o} = A_{sys} \text{ or } log\frac{I_x}{I_o} = A_{ls} + B \cdot S_w. \quad \text{(M-10)}$$

To use the above equation for the estimation of water saturation level, the parameters A_{1s} and B have to be determined experimentally by scanning the system in question at two levels of water saturation and solving for the parameters. Once the parameters A_{1s} and B are determined, the water saturation level at the corresponding point of the core holder can be estimated using the following relationship

$$S_w = \frac{\left[\log\left(\frac{I_x}{I_o}\right) - A_{1s}\right]}{B}. \qquad \text{(M-11)}$$

III.B.6. Extinction coefficient

To use Equation (M-11) for water saturation estimation, knowledge of the absorption coefficient k is not required. However, a numerical evaluation of the coefficient is required to design a core holder with the appropriate thickness that allows adequate microwave power to pass through at the maximum water saturation level. The coefficient is related to the relative dielectric constant and the relative loss factor as follows (Hasted, 1973),

$$\varepsilon' = n^2 - k^2 \qquad \text{(M-12)}$$

and

$$\varepsilon'' = 2 \cdot n \cdot k \qquad \text{(M-13)}$$

where n is the refractive index.

Solving Equations (M-12) and (M-13) simultaneously to eliminate n, one obtains

$$k = \left\{\left(\frac{1}{2}\right) \cdot \left[-\varepsilon' + \left(\varepsilon'^2 + \varepsilon''^2\right)^{\frac{1}{2}}\right]\right\}^{\frac{1}{2}}. \qquad \text{(M-14)}$$

III.B.7. Cole-Cole equation

To calculate k, values of both ε' and ε'' have to be available for the absorbing material at the field frequency and temperature. Using the Cole-Cole equation (Cole and Cole, 1941), the real and imaginary part of the complex relative dielectric constant can be described by the following equations, using a spread factor, α, for relaxation time (Hasted, 1973).

$$\varepsilon' = \varepsilon_\infty + \frac{(\varepsilon_s - \varepsilon_\infty) \cdot \left[1 + (\omega \cdot \tau)^{1-\alpha} \cdot sin\left(\frac{\alpha \cdot \pi}{2}\right)\right]}{1 + 2 \cdot (\omega \cdot \tau)^{1-\alpha} \cdot sin\left(\frac{\alpha \cdot \pi}{2}\right) + (\omega \cdot \tau)^{2(1-\alpha)}} \qquad \text{(M-15)}$$

and

$$\varepsilon'' = \frac{(\varepsilon_s - \varepsilon_\infty) \cdot (\omega \cdot \tau)^{1-\alpha} \cdot cos\left(\frac{\alpha \cdot \pi}{2}\right)}{1 + 2 \cdot (\omega \cdot \tau)^{1-\alpha} \cdot sin\left(\frac{\alpha \cdot \pi}{2}\right) + (\omega \cdot \tau)^{2(1-\alpha)}}, \qquad \text{(M-16)}$$

where the parameters ε_s, ε_∞ and τ are obtained from experimental measurements.

III.B.8. Conformity of the microwave power measurement technique with experimental apparatus and procedures

To use the above technique to estimate water saturation level in porous media, the following assumptions have to be made.

a. Utilization of the Beer-Lambert-Bouguer law for saturation determination by the microwave power measurement technique is appropriate. In particular, the approximation of the concentration of absorber by the volume fraction $\phi \cdot S_w$ is valid.

b. The absorbency of microwave energy by each component in a multi-component system can be approximated by linear addition. In particular, the phenomena of reflection, refraction, diffraction and standing wave interference can be lump summed linearly into the parameter A_{ls}. Moreover, it is supposed that A_{ls} is constant; that is, it applies to all measurements taken at a given cross-section of the core.

c. The oil saturation level has no significant bearing on the parameter A_{Is}. In particular, changes in oil saturation as the water saturation changes during the course of dynamic flow experiments would not change significantly the value of A_{Is}, which is assumed to be a constant.

Several researchers (Parsons, 1975; Bentsen and Saeedi, 1981; Brost and Davis, 1981; Davis, 1983; O'Steen, 1985; O'Steen and Holm, 1988) performed experiments and found these assumptions to be valid, and the measured water saturation level was found to agree with saturation level obtained by material balance calculation within acceptable limits of uncertainty.

III.C. Transformation of moment technique (Hahn and Shapiro, 1967)

When designing and subsequently utilizing an experimental apparatus and the associated procedures for measuring the generalized permeability coefficients, it is desirable to have an unbiased estimate of the expected coefficients as well as to have an estimate of the statistical spread of the coefficients as a result of using the apparatus and the procedures. Moreover, it is desirable to be able to analyze the relative importance of each component in the apparatus and each step of the procedures in a logical and systematic manner.

In short, the question of whether the apparatus and the procedures are capable of measuring the coefficients within acceptable limits of uncertainty have to be answered, and a quantitative means of expressing the variability of the coefficients due to the statistical randomness of equipment performance and imperfection in the measuring procedure is desired. Moreover, one would like to know which components are the major causes of variability of the result. The transformation of moment technique developed by Hahn and Shapiro (Hahn and Shapiro, 1967) provides the answers to all of the above questions. The following is a summary of the mathematical development of the transformation of moment technique.

III.C.1. Mathematical model

Assume that the relationship between the system function z and the continuous component random variables $x_1, x_2, x_3, \ldots x_n$ is given by the function $z = h(x_1, x_2, x_3, \ldots x_n)$ and let the variables be uncorrelated. The problem then, in mathematical terms, is simply the determination of an estimate of $E(z)$ (expected value) and $\mu_k(z)$ (k^{th} central moment) based on:

a. data on the variables from which estimates of $E(x_i)$ and $\mu_k(x_i)$ can be obtained.

b. knowledge of the relation between the variables and the system function, $h(x_1, x_2, x_3, \ldots x_n)$.

III.C.2. Basic concepts and definitions

Since x_i, the i th component of the variable under consideration, is a continuous random variable and is a function defined on a sample space, its cumulative distribution function, $F(x_{ij})$, following standard statistical terminology, is then defined as $F(x_{ij}) = \Pr(x_i \leq x_{ij})$, where x_{ij} is a particular value of x_i in its sample space.

By definition a probability density function $f(x_i)$ is equal to $\lim_{\Delta x_i \to 0} \frac{\Pr(x_{ij} \leq x_i \leq x_{ij} + \Delta x\ )}{\Delta x_i}$. The probability density function, $f(x_i)$, considering the above definition for $F(x_{ij})$, can then be redefined as $f(x_i) = \frac{d}{dx}[F(x_i)]$.

By definition, the expected value of the variable, x_i, which is also known as the mean of the variable, with the probability density function, $f(x_i)$, is

$$E(x_i) = \int_{-\infty}^{\infty} x_i \cdot f(x_i) \cdot dx_i \,. \tag{S-1}$$

Thus one can demonstrate, in light of Equation (S-1), that

$$E(constant) = constant, \tag{S-2}$$

$$E(constant \cdot x_i) = constant \cdot E(x_i), \tag{S-3}$$

and

$$E\left(\sum_{i=1}^{n}(x_i)\right) = \sum_{i=1}^{n} E(x_i). \tag{S-4}$$

If $h(x_i)$ is a function of x_i, one can also demonstrate that

$$E\left(\sum_{i=1}^{n} h(x_i)\right) = \sum_{i=1}^{n} E[h(x_i)]. \tag{S-5}$$

By definition, the k th central moment of a variable x_i is

$$\mu_k(x_i) = E[x_i - E(x_i)]^k \text{ or } \mu_k(x_i) = \int_{-\infty}^{\infty} (x_i - E(x_i))^k \cdot f(x_i) \cdot dx_i .$$

It follows, then, that the first central moment is equal to zero. The second central moment, after simplifications, becomes

$$\mu_2(x_i) = E(x_i^2) - (E(x_i))^2 \qquad \text{(S-6)}$$

where $\mu_2(x_i)$ is a measure of dispersion of the variable that is known as the variance of the variable. The third and fourth central moments, by analogy, are respectively equal to

$$\mu_3(x_i) = E(x_i^3) - 3 \cdot E(x_i) \cdot E(x_i^2) + 2 \cdot (E(x_i))^3$$

and

$$\mu_4(x_i) = E(x_i^4) - 4E(x_i) \cdot E(x_i^3) + 6(E(x_i))^2 \cdot E(x_i^2) - 3(E(x_i))^4 .$$

They are related to the symmetry and the peakedness of the distribution of the variable.

By definition, two variables are independent if and only if their joint probability density function is the product of their probability density functions; that is, the continuous random variables x_i and x_j are independent if and only if

$$f(x_i, x_j) = f(x_i) \cdot f(x_j), \text{ where } x_i \neq x_j .$$

Coupled with the definition of the expected values of continuous random variables with $h(x_i, x_j)$ as a function of the continuous random variables x_i and x_j, that is, with $E[h(x_i, x_j)] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(x_i, x_j) \cdot f(x_i, x_j) \cdot dx_i dx_j$, one can demonstrate, for the independent continuous random variables, x_i and x_j, that

$$E(x_i \cdot x_j) = E(x_i) \cdot E(x_j)$$

and that

$$E\left\{\left[x_i - E(x_i)\right]^r \cdot \left[x_j - E(x_j)\right]^s\right\} = E\left[x_i - E(x_i)\right]^r \cdot E\left[x_j - E(x_j)\right]^s$$

(S-7)

III.C.3. Calculation of the mean of the system function

If one expands the system function $h(x_1, x_2, x_3, \ldots x_n)$ about $[E(x_1), E(x_2), E(x_3), \ldots E(x_n)]$, the point at which each of the variables takes on its mean value, by a multi-variable Taylor series, one can demonstrate, after retaining only terms up to second order, that

$$h(x_1, x_2, x_3, \ldots, x_n) =$$

$$h\left[E(x_1), E(x_2), E(x_3), \ldots, E(x_n)\right] + \sum_{i=1}^{n}\left\{\frac{\partial \bar{h}}{\partial x_i} \cdot \left[x_i - E(x_i)\right]\right\} +$$

$$\left\{\frac{1}{2}\sum_{i=1}^{n}\frac{\partial^2 \bar{h}}{\partial x_i^2}\left[x_i - E(x_i)\right]^2\right\} + \left\{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\left[x_i - E(x_i)\right]\left[x_j - E(x_j)\right]\frac{\partial^2 \bar{h}}{\partial x_i \cdot \partial x_j}\right\}$$

(S-8)

where $\frac{\partial^2 \bar{h}}{\partial x_i^2}$ denotes $\frac{\partial^2 h}{\partial x_i^2}$ evaluated at $E(x_i)$; that is, with the $E(x_i)$ substituted for the x_i.

Taking the expected values of both sides of Equation (S-8), in light of Equation (S-5), one can demonstrate that

$$E\left[h(x_1, x_2, x_3, \ldots, x_n)\right] = E\left\{h\left[E(x_1), E(x_2), E(x_3), \ldots, E(x_n)\right]\right\} +$$

$$E\left\{\sum_{i=1}^{n}\frac{\partial \bar{h}}{\partial x_i} \cdot \left[x_i - E(x_i)\right]\right\} + E\left\{\frac{1}{2} \cdot \sum_{i=1}^{n}\frac{\partial^2 \bar{h}}{\partial x_i^2} \cdot \left[x_i - E(x_i)\right]^2\right\} +$$

$$E\left\{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\left[x_i - E(x_i)\right]\cdot\left[x_j - E(x_j)\right]\cdot\frac{\partial^2 \bar{h}}{\partial x_i \cdot \partial x_j}\right\}. \tag{S-9}$$

From Equation (S-2), one can demonstrate that

$$E\{h[E(x_1), E(x_2), \ldots E(x_n)]\} = h[E(x_1), E(x_2), \ldots E(x_n)]. \tag{S-10}$$

From Equation (S-3), one can demonstrate that

$$E\left\{\sum_{i=1}^{n}\frac{\partial \bar{h}}{\partial x_i}\cdot\left[x_i - E(x_i)\right]\right\} = \sum_{i=1}^{n} E\left\{\frac{\partial \bar{h}}{\partial x_i}\cdot\left[x_i - E(x_i)\right]\right\}$$

$$= \sum_{i=1}^{n}\frac{\partial \bar{h}}{\partial x_i}\cdot\left\{E\left[x_i - E(x_i)\right]\right\}$$

$$= 0. \tag{S-11}$$

From Equation (S-6), one can demonstrate that

$$E\left\{\frac{1}{2}\cdot\sum_{i=1}^{n}\frac{\partial^2 \bar{h}}{\partial x_i^2}\cdot\left[x_i - E(x_i)\right]^2\right\} = \frac{1}{2}\cdot\sum_{i=1}^{n}\frac{\partial^2 \bar{h}}{\partial x_i^2}\cdot E\left[x_i - E(x_i)\right]^2$$

$$= \frac{1}{2}\cdot\sum_{i=1}^{n}\frac{\partial^2 \bar{h}}{\partial x_i^2}\cdot \mu_2(x_i). \tag{S-12}$$

From Equation (S-7), one can demonstrate for independent variables that

$$E\left\{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\left[x_i - E(x_i)\right]\cdot\left[x_j - E(x_j)\right]\cdot\frac{\partial^2 \bar{h}}{\partial x_i \cdot \partial x_j}\right\}$$

$$= \left\{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} E\left[x_i - E(x_i)\right]\cdot E\left[x_j - E(x_j)\right]\cdot\frac{\partial^2 \bar{h}}{\partial x_i \cdot \partial x_j}\right\}$$

$$= 0. \tag{S-13}$$

Thus, substituting Equations (S-10), (S-11), (S-12) and (S-13) into Equation (S-9), one can demonstrate that

$$E(z) = h[E(x_1), E(x_2), E(x_3), \ldots E(x_n)] + \frac{1}{2} \cdot \sum_{i=1}^{n} \frac{\partial^2 \bar{h}}{\partial x_i^2} \cdot \mu_2(x_i),$$

(S-14)

which is the expected value of the system function $h(x_1, x_2, x_3, \ldots x_n)$. This is the second order approximation of the first central moment. The second term in the above equation is known as the total bias for the function. A large standard deviation of a component variable, depending on the function, may cause the total bias term to be large. That is, setting $E(z) = h[E(x_1), E(x_2), E(x_3), \ldots E(x_n)]$ provides an exact result only when all second and higher order partial derivatives are zero or when the variances of each of the variables are zero.

III.C.4. Calculation of the variance of the system function

In light of Equation (S-6), one can demonstrate that

$$\mu_2[h(x_1, x_2, x_3, \ldots, x_n)] = E\{[h(x_1, x_2, x_3, \ldots, x_n)]^2\} - \{E[h(x_1, x_2, x_3, \ldots, x_n)]\}^2. \quad \text{(S-15)}$$

Taking the square of Equation (S-8), after taking the expected values on a term-by-term basis and retaining the terms up to the third order, one can demonstrate that

$$E\{[h(x_1, x_2, x_3, \ldots, x_n)]^2\} =$$

$$\{h[E(x_1), E(x_2), E(x_3), \ldots, E(x_n)]\}^2 + \sum_{i=1}^{n} \left(\frac{\partial \bar{h}}{\partial x_i}\right)^2 \cdot E[x_i - E(x_i)]^2 +$$

$$h[E(x_1), E(x_2), E(x_3), \ldots, E(x_n)] \cdot$$

$$\sum_{i=1}^{n} \frac{\partial^2 \overline{h}}{\partial x_i^2} \cdot E[x_i - E(x_i)]^2 + \sum_{i=1}^{n} \left(\frac{\partial \overline{h}}{\partial x_i} \right) \cdot \left(\frac{\partial^2 \overline{h}}{\partial x_i^2} \right) \cdot E[x_i - E(x_i)]^3 \qquad \text{(S-16)}$$

Taking the square of Equation (S-9), after retaining only the terms up to the third order, one can demonstrate that

$$\{E[h(x_1, x_2, x_3, \ldots, x_n)]\}^2$$

$$= \left\{ h[E(x_1), E(x_2), E(x_3), \ldots, E(x_n)] + \frac{1}{2} \cdot \sum_{i=1}^{n} \frac{\partial^2 \overline{h}}{\partial x_i^2} \cdot \mu_2(x_i) \right\}^2$$

$$= \{h[E(x_1), E(x_2), E(x_3), \ldots, E(x_n)]\}^2 +$$

$$h[E(x_1), E(x_2), E(x_3), \ldots, E(x_n)] \cdot \sum_{i=1}^{n} \left(\frac{\partial^2 \overline{h}}{\partial x_i^2} \right) \cdot E[x_i - E(x_i)]^2 \qquad \text{(S-17)}$$

Substituting Equation (S-16) and Equation (S-17) into Equation (S-15), one can demonstrate that

$$\mu_2[h(x_1, x_2, x_3, \ldots, x_n)] = \sum_{i=1}^{n} \left(\frac{\partial \overline{h}}{\partial x_i} \right)^2 \cdot E[x_i - E(x_i)]^2 +$$

$$\sum_{i=1}^{n} \left(\frac{\partial \overline{h}}{\partial x_i} \right) \cdot \left(\frac{\partial^2 \overline{h}}{\partial x_i^2} \right) \cdot E[x_i - E(x_i)]^3$$

$$\mu_2(z) = \sum_{i=1}^{n} \left(\frac{\partial \overline{h}}{\partial x_i} \right)^2 \cdot \mu_2(x_i) + \sum_{i=1}^{n} \left(\frac{\partial \overline{h}}{\partial x_i} \right) \cdot \left(\frac{\partial^2 \overline{h}}{\partial x_i^2} \right) \cdot \mu_3(x_i), \qquad \text{(S-18)}$$

the variance of the system function in terms of its component variables. This is the second order approximation of the second central moment. For the purpose of this study, uncertainty is defined as three times of the square root of variance.

III.C.5. Conformity of the transformation of moment technique with the experimental apparatus and procedures

In practice only the expected values and the variances of the variables that the current experimental apparatus is dealing with are known. Their complete and exact probabilistic distributions, however, are not quantified. The transformation of moment technique described above, by virtue of the Taylor series expansion, allows the moments of dependent variable $h(x_1, x_2, x_3, \ldots x_n)$ to be estimated directly from the moments of the independent variables $x_1, x_2, x_3, \ldots x_n$. The moments for the independent variables, however, can be either moments as calculated from probability density functions or, alternatively sample moments as estimated from experimental observations of the current study.

III.C.6. Evaluation of the expected value and the variance of a system function

III.C.6.a. Analytical method

It is worthwhile to mention here that the expression of expected value and variance of a multi variable system function, $z = h(x_1, x_2, x_3, \ldots x_n)$, in analytical terms, are

$$E(z) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} h(x_1, x_2, x_3, \ldots x_n) \cdot f(x_1, x_2, x_3, \ldots x_n) \cdot dx_1 \cdot dx_2 \cdot dx_3 \ldots dx_n,$$

and

$$\mu_2(z) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} \left(h(x_1, x_2, x_3, \ldots x_n) - E(z)\right)^2 \cdot$$

$$f(x_1, x_2, x_3, \ldots x_n) \cdot dx_1 \cdot dx_2 \cdot dx_3 \ldots dx_n$$

By using a symbolic logic mathematical package, the value of $E(z)$ and $\mu_2(z)$ can be evaluated.

III.C.6.b. VARSIM (Griffin and Hamilton, 1990)

A program, VARSIM (Griffin and Hamilton, 1990), that uses the transformation of moment technique, was used for the current study to facilitate the calculation of the expected value and the variance of a system function. The program is capable of numerically evaluating the bias, expected value and variance of a multi-variable function in one single pass of calculations.

Note that the program only approximates the variance of a function by the following equation,

$$\mu_2(z) = \sum_{i=1}^{n} \left(\frac{\partial \bar{h}}{\partial x_i} \right)^2 \cdot \mu_2(x_i),$$

which is an approximation to Equation (S-18).

III.D. Experimental model (Kalaydjian 1990; Sarma and Bentsen, 1989; Bentsen and Manai, 1991; Bentsen and Manai, 1993; Liang and Lohrenz, 1994; Bentsen, 1996)

Using generalized permeability coefficients, fluid flow equations for a two-phase fluid system can be described as (Kalaydjian, 1990),

$$\upsilon_1 = -\lambda_{11} \cdot \frac{\partial P_1}{\partial L_y} - \lambda_{12} \cdot \frac{\partial P_2}{\partial L_y} \tag{F-1}$$

$$\upsilon_2 = -\lambda_{21} \cdot \frac{\partial P_1}{\partial L_y} - \lambda_{22} \cdot \frac{\partial P_2}{\partial L_y}. \tag{F-2}$$

As the governing equations contain four unknowns, the generalized permeability coefficients, a pair of two-phase flow experiments has to be carried out to quantify them. To be able to verify that the measured generalized permeability coefficients are indeed the fundamental parameters for describing the two-phase flow system, one needs to carry out a minimum of two pairs of experiments, each pair being able to resolve the generalized permeability coefficients independently. One way to achieve this goal is by using a steady-state cocurrent flow experiment coupled with a steady-state countercurrent flow experiment while the other pair involves a steady-state cocurrent flow experiment coupled with an unsteady-state cocurrent flow experiment. The experimental methodology is then dictated by how the generalized permeability coefficients are being resolved into experimentally measurable parameters from these three types of experiments, and the coefficients that can be obtained from these two pairs of experiments can then be analyzed by using the ANOVA technique in the future. The following is a summary of how the generalized permeability coefficients are being resolved into experimentally measurable parameters.

III.D.1. Fundamental equations

Combining Equations (F-1) and (F-2) proposed by Kalaydjian with the conventional transport Equations (F-3) and (F-4) for two-phase flow (Wyckoff and Botset, 1936) and the recent inferences (F-5) and (F-6) drawn by Bentsen and Manai (Bentsen and Manai, 1993), the fundamental equations of any two-phase system can be described completely using the following results:

$$\upsilon_1 = -\lambda_{11} \cdot \frac{\partial P_1}{\partial L_y} - \lambda_{12} \cdot \frac{\partial P_2}{\partial L_y} \qquad \text{(F-1)}$$

$$\upsilon_2 = -\lambda_{21} \cdot \frac{\partial P_1}{\partial L_y} - \lambda_{22} \cdot \frac{\partial P_2}{\partial L_y} \qquad \text{(F-2)}$$

$$\upsilon_1 = -\lambda_1 \cdot \frac{\partial P_1}{\partial L_y} \qquad \text{(F-3)}$$

$$\upsilon_2 = -\lambda_2 \cdot \frac{\partial P_2}{\partial L_y} \qquad \text{(F-4)}$$

$$R_{12} = \frac{\partial P_1^0}{\partial L_y} / \frac{\partial P_2^0}{\partial L_y} \qquad \text{(F-5)}$$

$$R_{12} = -\frac{\partial P_1^*}{\partial L_y} / \frac{\partial P_2^*}{\partial L_y} \qquad \text{(F-6)}$$

III.D.1.a. Steady-state cocurrent and steady-state countercurrent flow experiments

By writing Equations (F-1), (F-2), (F-3) and (F-4) for steady-state countercurrent flow and utilizing Equation (F-6), one can demonstrate that

$$\lambda_1^* = \frac{\lambda_{11} \cdot R_{12} - \lambda_{12}}{R_{12}} \qquad \text{(F-7)}$$

and

$$\lambda_2^* = -\lambda_{21} \cdot R_{12} + \lambda_{22}, \qquad \text{(F-8)}$$

the two defining equations for steady-state countercurrent flow that relate the conventional mobilities λ_1^* and λ_2^* to Kalaydjian's generalized permeability coefficients.

By writing Equations (F-1) and (F-2) for steady-state cocurrent flow, and using Equations (F-7) and (F-8), one can obtain the generalized permeability coefficients with parameters that can be obtained by either steady-state cocurrent flow or steady-state countercurrent flow experiments, namely:

$$\lambda_{11} = \frac{\upsilon \cdot f_1^0 - \lambda_1^* \cdot R_{12} \cdot \frac{\partial P_2^0}{\partial L_y}}{-R_{12} \cdot \frac{\partial P_2^0}{\partial L_y} - \frac{\partial P_1^0}{\partial L_y}}, \tag{F-9}$$

$$\lambda_{12} = \frac{\upsilon \cdot f_1^0 \cdot R_{12} + \lambda_1^* \cdot R_{12} \cdot \frac{\partial P_1^0}{\partial L_y}}{-R_{12} \cdot \frac{\partial P_2^0}{\partial L_y} - \frac{\partial P_1^0}{\partial L_y}}, \tag{F-10}$$

$$\lambda_{21} = \frac{\upsilon \cdot f_2^0 + \lambda_2^* \cdot \frac{\partial P_2^0}{\partial L_y}}{-R_{12} \cdot \frac{\partial P_2^0}{\partial L_y} - \frac{\partial P_1^0}{\partial L_y}} \tag{F-11}$$

and

$$\lambda_{22} = \frac{\upsilon \cdot f_2^0 \cdot R_{12} - \lambda_2^* \cdot \frac{\partial P_1^0}{\partial L_y}}{-R_{12} \cdot \frac{\partial P_2^0}{\partial L_y} - \frac{\partial P_1^0}{\partial L_y}}. \tag{F-12}$$

III.D.1.b. Unsteady-state cocurrent and steady-state cocurrent flow experiments

By writing Equations (F-1), (F-2), (F-3) and (F-4) for the steady-state cocurrent flow, and utilizing Equation (F-5), one can demonstrate that

$$\lambda_1^0 = \frac{\lambda_{11} \cdot R_{12} + \lambda_{12}}{R_{12}} \tag{F-13}$$

and that

$$\lambda_2^0 = \lambda_{21} \cdot R_{12} + \lambda_{22}, \tag{F-14}$$

the two defining equations for steady-state cocurrent flow that relate the conventional mobilities λ_1^0 and λ_2^0 to Kalaydjian's generalized permeability coefficients.

By writing Equations (F-1) and (F-2) for unsteady-state cocurrent flow, and using Equations (F-13) and (F-14), one can obtain the generalized permeability coefficients with parameters that can be obtained by either unsteady-state cocurrent flow or steady-state cocurrent flow experiments, namely:

$$\lambda_{11} = \frac{\upsilon \cdot f_1 + \lambda_1^0 \cdot R_{12} \cdot \frac{\partial P_2}{\partial L_y}}{R_{12} \cdot \frac{\partial P_2}{\partial L_y} - \frac{\partial P_1}{\partial L_y}}, \tag{F-15}$$

$$\lambda_{12} = \frac{-\upsilon \cdot f_1 \cdot R_{12} - \lambda_1^0 \cdot R_{12} \cdot \frac{\partial P_1}{\partial L_y}}{R_{12} \cdot \frac{\partial P_2}{\partial L_y} - \frac{\partial P_1}{\partial L_y}}, \tag{F-16}$$

$$\lambda_{21} = \frac{\upsilon \cdot f_2 + \lambda_2^0 \cdot \frac{\partial P_2}{\partial L_y}}{R_{12} \cdot \frac{\partial P_2}{\partial L_y} - \frac{\partial P_1}{\partial L_y}} \tag{F-17}$$

and

$$\lambda_{22} = \frac{-\upsilon \cdot f_2 \cdot R_{12} - \lambda_2^0 \cdot \dfrac{\partial P_1}{\partial L_y}}{R_{12} \cdot \dfrac{\partial P_2}{\partial L_y} - \dfrac{\partial P_1}{\partial L_y}}. \tag{F-18}$$

III.D.2. Improved methodology for unsteady-state experiment

Equations (F-9), (F-10), (F-11) and (F-12) and Equations (F-15), (F-16), (F-17) and (F-18) enable predictions of the generalized permeability coefficients at one saturation point from two independent pairs of experiments. To construct the curves of the generalized permeability coefficients versus saturation level, however, one needs to have a complete data set for each coefficient, with the data points spread over the entire saturation range.

Conventional steady-state cocurrent flow experiments and steady-state countercurrent flow experiments are sufficient to provide the necessary measured parameters for calculation of the coefficients at any saturation level, provided that the experiments are conducted to achieve the required levels of saturation with appropriate injection rates. However, to obtain the measured parameters for the full saturation range from the unsteady state experiments, one cannot rely on the conventional unsteady state method, because of the rapid changes in saturation level that take place in the immediate vicinity of a displacement front. This prevents the estimation of relative permeabilities over the entire saturation range of interest.

This limitation can be removed, if data from an unstabilized displacement may be used; that is, inject the fluids at such a rate that a displacement front does not form. However, at such injection rates the displacement would be unstabilized and, as a consequence, additional problems would arise. That is, because the flow is unstabilized, the usual external-drive equations used to estimate the saturation, S_w, and the reciprocal mobility λ^{-1}, cannot be used. Moreover, use of the standard (Eulerian) methods for estimating the fraction of water or oil flowing at a particular location cannot be used either. Sarma and Bentsen (Sarma and Bentsen, 1989) developed an improved unsteady-state method that allows the utilization of stable and unstabilized displacement data for measurements of permeabilities over the entire saturation range of interest. Their results can be borrowed for this study.

III.D.3. Implementation of experimental model

III.D.3.a. Steady-state cocurrent flow and steady-state countercurrent flow

III.D.3.a.i. Steady-state cocurrent flow experiments

By performing a series of experiments, each at a successively higher wetting-phase, steady-state saturation level by changing the ratio of the non-wetting-phase injection rate to the wetting-phase injection rate, one can obtain the measured parameters υ_1^0, υ_2^0, R_{12}, $\partial P_1^0/\partial L_y$ and $\partial P_2^0/\partial L_y$ versus saturation level from each experiment.

III.D.3.a.ii. Steady-state countercurrent flow experiments

By performing a series of experiments, each at a successively higher wetting-phase, steady-state saturation level by changing the ratio of the non-wetting-phase injection rate to the wetting-phase injection rate, one can obtain the parameters λ_1^* and λ_2^* versus saturation level from each experiment. Substituting the parameters υ_1^0, υ_2^0, R_{12}, $\partial P_1^0/\partial L_y$, $\partial P_2^0/\partial L_y$, λ_1^* and λ_2^* into Equations (F-9), (F-10), (F-11) and (F-12), one can calculate the generalized permeability coefficients at a specific saturation level. Curves for the coefficients can then be constructed by evaluating Equations (F-9), (F-10), (F-11) and (F-12) at different saturation points from all the experiments.

III.D.3.b. Steady-state cocurrent flow and improved unsteady-state cocurrent flow experiments

III.D.3.b.i. Steady-state cocurrent flow experiments

By performing a series of experiments, each at a successively higher wetting-phase, steady-state saturation level by changing the ratio of the non-wetting-phase injection rate to the wetting-phase injection rate, one can obtain the measured parameters λ_1^0, λ_2^0 and R_{12} versus saturation level from each experiment.

III.D.3.b.ii. Improved unsteady-state cocurrent flow experiment

Using the microwave power measurement apparatus, one can obtain the saturation profiles of the core under dynamic conditions at any time. If the experiment is conducted in such a way that a displacement front does not form in the core (that is, an unstabilized displacement) and if there are no viscous fingers propagating anywhere in the core (that is, a stable displacement), then the parametric equation for describing the saturation profile versus distance would simply be a monotonically decreasing polynomial function starting from the injection end. If one were to curve fit the data, one would obtain for a particular time t_i, a parametric equation,

$$x_{t_i} = f(S_1) \tag{F-18}$$

between the two levels of saturation S_{1i} and S_{1m}.

Now, $Q_{1t_i} = A \cdot \phi \cdot \int_{S_{1i}}^{S_1} x_{t_i} \cdot dS_1$ (F-19)

is the portion of wetting-phase fluid that is inside the core between the initial saturation levels S_{1i} and S_1 at the particular time t_i, if the assumptions underlying the Buckley-Leverett equation are met and if $x(S_1, t)$ is a single-valued function of S_1 with $f_1(S_w, t)$ a well-behaved function.

Repeating the process at a number of different instantaneous times before and after the particular time, t_i, one would have a collection of Equations (F-19) for Q_{1t} as a function of S_1 for various times t_i in the experimental time frame up to time t where $t >> t_i$.

The $f_1 = \frac{1}{q} \cdot \frac{dQ_1}{dt}$ required for the study is for a specific time $t^{\otimes}$, and cannot be obtained by simply differentiating the above obtained Q_{1t} equations with respect to time t and then evaluating at time $t^{\otimes}$, as the Q_{1t} equations are themselves functions of S_1.

Evaluating the different Q_{1t} equations at different saturation points, one can construct a table of Q_1 versus time t for any specific saturation level S_1. By using a least-squares fit method on the data, one can obtain the parametric equations as a function of time, t, for any specific saturation level S_1,

$$Q_{1S_1} = f(t) \qquad \text{(F-20)}$$

By taking the derivative of the above Q_{1S_1} equations with respect to time t, one can obtain a number of functional equations, f_{1S_1}, as a function of time t for any saturation level S_1. If one evaluates the functions f_{1S_1} obtained for a specific time $t^{\otimes}$ of interest at different saturation levels S_1, one can construct the fractional flow curve $f_{1t^{\otimes}}$ versus S_1 at the specific time $t^{\otimes}$. As $f_1 + f_2 = 1$ for any saturation level and time, one can also evaluate $f_{2t^{\otimes}}$ for the corresponding time $t^{\otimes}$.

From Equation (F-18), the position $x^{\otimes}$ for a specific saturation level $S_1^{\otimes}$ at the corresponding time $t^{\otimes}$ needed for the $f_{1t^{\otimes}}$ curve construction can be obtained. From the experimental data, one can obtain also the pressure profiles of both phases at any time and at any location. Fitting parametric equations respectively to both pressure profiles at the same specific time $t^{\otimes}$, one can obtain, after differentiating with respect to location x, functions to relate the pressure gradient of both phases to any locations along the core for the same specific time $t^{\otimes}$. Evaluating the functions obtained at the specific location $x^{\otimes}$, one can obtain, for a specific time $t^{\otimes}$ and saturation level $S_1^{\otimes}$, the pressure gradients $\partial P_1/\partial x$ and $\partial P_2/\partial x$ needed for the improved unsteady-state cocurrent flow method.

Substituting the parameters λ_1^0, λ_2^0 and R_{12}, into Equations (F-15), (F-16), (F-17) and (F-18), along with the parameters $f_{1t^{\otimes}}, f_{2t^{\otimes}}, \partial P_1/\partial x$ and $\partial P_2/\partial x$ evaluated at the same saturation level, one can calculate the generalized permeability coefficients at the specific saturation level. Curves for the coefficients can then be constructed by evaluating Equations (F-15), (F-16), (F-17) and (F-18) at different saturation points.

III.D.3.b.iii. Time domain migrations of saturation profiles for unsteady-state experiments

The assumption is made when using the microwave power measurement technique that the saturation profile is an instantaneous snapshot of the core holder. Such is not the case; that is, an interval of time is required to completely scan the core along its length. A correction procedure is required to migrate all the measured saturation profiles, which span over a time interval of approximately 35 seconds, to an instantaneous time reference, so that Equation (F-18) be free of error.

Because, if a displacement is unstabilized, the fraction of water flowing at a particular location along the core is a function of both of the time and the saturation level, the integrated form of the Buckley-Leverett equation cannot be used directly. That is, equation $x = \frac{q}{A \cdot \phi} \cdot \int_0^t \frac{\partial f_1}{\partial S_1} \cdot dt$ is a function of both the saturation level S_1 and the time t. However, an approximation can be obtained if the time interval required for a single scan is small enough such that the derivative $\frac{\partial f_1}{\partial S_1}$ can be assumed to be constant over the time interval, in which case $\Delta x = \frac{q}{A \cdot \phi} \cdot \frac{\partial f_1}{\partial S_1} \cdot \Delta t$. The time shift procedure then is simply a matter of evaluating the first time around the f_{1t^*} curve without any time correction, then using the curve constructed and a known Δt for the corresponding position in the core to estimate the correction Δx required for the particular location. The saturation profile can then be corrected by using the approximation in an iterative manner to successively refine the profile until the limits of uncertainty are kept within acceptable limits.

III.D.3.b.iv. Smoothing of saturation profiles for unsteady-state experiments

When appropriate injection rates have been chosen, the saturation profile of the core under dynamic conditions can be approximated by a monotonic function

Because of local heterogeneities in porosity and permeability, the measured saturation profiles have to be smoothed to remove any perturbations. That is, Equation (F-18), even after proper time migration, cannot be used directly. To smooth a saturation profile, a parametric equation has to be used to fit the Q_1 versus saturation data, for the specific time $t^{\circledast}$. Then by virtue of

$x_{1t*} = \frac{1}{A \cdot \phi} \cdot \frac{dQ_{1t*}}{dS_1}$, the distance traveled by a number of different saturation points can be determined.

III.D.3.c. Conformity of unsteady-state experiments with the one-dimensional displacement model

The improved unsteady-state method is based on one-dimensional theory. As a consequence, the displacements used to measure various parameters must be conducted in systems that are linear and homogeneous. Moreover, the pressure and saturation level must be uniform at each cross-section along the length of the core. (That is, no viscous fingers propagating anywhere in the system) Finally, since an improved unsteady-state method is to be used (Sarma and Bentsen, 1989), a Lagrangian formulation of the problem must be permissible. Thus the saturation level must decrease monotonically along the length of the core.

To ensure the conformity of unsteady-state experiments with the one-dimensional displacement model, tests can be conducted. Furthermore, the inlet-end and outlet-end effect have to be avoided to maintain the quality of data collected. The following is a summary of tests that can be conducted.

III.D.3.c.i. Test of instability

An instability number (Bentsen, 1985) is used to test whether the displacement is unstable. For the horizontal rectangular shaped system used in this study, the instability number is defined by

$$I_{sr} = \frac{\mu_1 \upsilon (M - 1 - N_g)}{k_{1r} \cdot \sigma_{eb}} \frac{L_x^2 L_z^2}{(L_x^2 + L_z^2)} Cr(M),$$

where

$$Cr(M) = \frac{4 \cdot \left(M^{\frac{5}{3}} + 1 \right)}{(M+1) \cdot \left(M^{\frac{1}{3}} + 1 \right)^2}$$

and

$$\sigma_{eh} \quad A_c \cdot \phi \cdot (1 - S_{1r} - S_{2r}) \cdot w_{ef} .$$

The instability number has to be smaller than or equal to π^2, (Sarma and Bentsen, 1987; Bentsen, 1990; Bentsen, 1996) for a displacement to be stable. This result has to be confirmed before any data from unsteady-state experiment can be used.

III.D.3.c.ii. Capillary number

The wetting-phase or the displacing phase has to be injected in such a way that a saturation front does not form. (That is, unstabilized displacement)

To determine whether the displacement is unstabilized, the capillary number is used (Bentsen, 1978). For the same system, the capillary number, defined by $N_c = \dfrac{A_c \cdot K_{1r}}{\upsilon \cdot L_v \cdot \mu_1}$, has to be greater than 0.1. This result has to be confirmed before any data from unsteady-state experiment can be used.

III.D.3.c.iii. Inlet-end effect

If a displacement is unstabilized, the saturation level at the inlet end of the core may not increase to its maximum value by the time breakthrough of the injected fluid is achieved (Islam and Bentsen, 1986). Moreover, the injected fluid may enter the inlet face of the core at a point (or several selected points) rather than over the entire face of the core (Kyte and Rapoport, 1958). As pressure in the oil and water near the inlet-end of the core holder may be disturbed by the inlet-end effect for an unstabilized displacement, any core holder used has to provide an adequate length for the experimental study.

III.D.3.c.iv. Outlet-end effect

The outlet-end effect can compromise also the data used in the improved unsteady-state method. If a displacement is unstabilized, the saturation level of the injected fluid in the immediate vicinity of the outlet end of the core can increase rapidly to a value near its maximum value (Bentsen and Saeedi, 1981), contrary to what is implicitly assumed in the Buckley-Leverett theory. Moreover, such outlet-end effects can introduce perturbations into the phase pressure measured in the immediately vicinity of the outlet face of the core.

III.D.3.d. Regression analysis

To draw a conclusion that the generalized permeability coefficients are indeed the fundamental parameters for describing fluid flow behavior, one needs to demonstrate that the generalized permeability coefficients as measured from the pair of steady-state cocurrent flow and steady-state countercurrent flow experiments are indeed identical to those as measured from the pair of steady-state cocurrent flow and improved unsteady-state cocurrent flow experiments. The standard statistical technique of Analysis of Variance (ANOVA) can be used to test the hypothesis. The implementation of ANOVA to experimental data is beyond the scope the current study and details will not be presented here.

IV. Equipment apparatus and procedures

IV.A. Fundamental measurements

It can be seen, from the previous section, that the matrix of generalized permeability coefficients can be evaluated from experimentally measured parameters. In the context of this study, these parameters are the static and dynamic saturation profiles of the core; the static and dynamic, phase dependent pressure profiles of the core; the injection rates of the fluids; the viscosities of the fluids and the cross-sectional area of the core.

In addition, the determination of the saturation profiles requires knowledge of the input and output power level of the microwave energy; the frequency of the microwave radiation; the extinction coefficient of water, the porosity and thickness of the core under investigation. Furthermore, the measurement of pressure profiles, injection rates and porosity can be further resolved into fundamental measurements of pressure, weight, time and dimensions.

IV.B. Modifications of the experimental apparatus

Since the statistical uncertainty of any measurement can propagate and affect the limits of uncertainty of any parameters derived, the statistical uncertainty has to be kept within acceptable limits. The available equipment has to be modified and equipment with improved performances has to be constructed with a view to limiting the statistical uncertainties associated with the performances of the equipment.

IV.B.1. Microwave-power measurement apparatus

IV.B.1.a. Apparatus

The intended apparatus for the microwave power measurement technique is presented in Figure IV-1. However, before the intended apparatus can be used, the purposes of its components have to evaluated and proper components have to be selected.

IV.B.1.b. Components

IV.B.1.b.i. Oscillator

Instead of staying with the existing type of microwave oscillator, a reflex klystron oscillator that broke down and became obsolete, a decision was made to use a Gunn diode-type oscillator as the generating source of the microwave power. The oscillator has a stated power output of 250 milliwatts at ambient temperature. Moreover, the spectrum width of the oscillator is only 250 MHz with a factory fixed operating frequency of 28 GHz. This allows for the majority of the hardware used in the existing experimental apparatus to be used in the current study and provides a more stable and purer source of microwave power for conducting experiments.

IV.B.1.b.ii. Isolator

A 20 dB ferrite isolator with a 0.2 dB insertion loss was put in line after the output port of the Gunn oscillator. This permits maximum transmitted power to leave from the oscillator but prevents any reflected power from entering the oscillator. Consequently, the oscillator sees a matched load, and interfering effects such as variation in output power and pulling of frequency due to any variation in the impedance of the core are minimized (Collin, 1966).

IV.B.1.b.iii. Low-pass filter

A low-pass filter, with a broad stop band designed to suppress the second and the third harmonics of the pass-band frequencies, was put in line after the output port of the isolator. Regardless of the mode of propagation, harmonics are attenuated at least 35 dB in the stop band. This further improves the spectral purity of the forward traveling microwave.

IV.B.1.b.iv. Directional coupler

The forward directional coupler was changed from the 10 dB type used previously to a 20 dB type. The coupler was subsequently repaired by the manufacturer. Furthermore, all couplers were re-certified for their reliability and performance according to the standards set forth by the National Institute of Standards and Testing (NIST). This allows for more microwave power to pass through the main waveguide arm and still allows representative sampling of the forward power flow by the power sensor and power meter that are connected to the auxiliary arm of the coupler, thus extending the possible range of saturation measurements while still maintaining the overall limits of uncertainty of the measurement of the microwave power.

IV.B.1.b.v. Attenuator

An attenuator was put in line before the input port of the power sensor. This allows for a controlled percentage of the microwave power to be detected by the power sensor, thus preventing any damaging effects that excessive power levels may have on the power sensor. It is used in the apparatus for insertion loss measurement of the microwave components as well as measurement of the conversion factor for the power meters.

IV.B.1.b.vi. Thermistor mounts

The thermistor mounts, after 15 years of usage, were repaired by the manufacturer. Furthermore, they were re-certified for their reliability and performance according to the standards set forth by the NIST. This provides a quantifiable knowledge of their limits of uncertainty for any measurements made.

IV.B.1.b.vii. Slide-screw tuner

A slide-screw tuner was put in line before the input port of the transmitting antenna, allowing for matching of the receiving load to the characteristic impedance of the transmission line. This device reduces the unrecorded amount of reflected microwave power by virtue of canceling the existing reflection in the transmission line between the transmitting antenna and the receiving load, thus improving the limits of uncertainty of the saturation measurement. An additional home-made impedance tuner was fabricated and installed on the receiving end of the apparatus for a similar purpose.

IV.B.1.b.viii. Thermal insulators

A thermal insulator was installed onto each of the power sensors to prevent any fluctuation of measurements due to any changes in the ambient temperature. This permits measurement of the microwave power that is free of drift with temperature, thus improving the limits of uncertainty of the saturation measurement.

IV.B.1.b.ix. Waveguides and waveguide bends

All unnecessary waveguides and waveguide bends were eliminated to reduce the insertion loss in the transmission line, thus making available more microwave power to the transmitting antenna for energizing the core. This allows the range of saturation measurement to be extended.

IV.B.1.b.x. Reflected microwave power monitoring

An additional power sensor and an additional power meter were installed to monitor the reflected microwave power. During an actual experiment, they are to be used in conjunction with the slide screw tuner to minimize the overall reflected power along the length of the core at each saturation level achieved during the steady-state experiment. This allows for more accurate measurement of the saturation level and extends the range of the saturation level measurement, as more microwave power is directed towards the core. This technique, however, cannot be used for the unsteady state experiments because the load, water in the core in this case, is not a constant along the length of the core for each experiment but varies with time as saturation changes over time.

IV.B.1.b.xi. X-Y-Z antenna movement control

A caliper that allows for fine adjustments of the microwave antennas in X-Y-Z directions along with tilt was installed. This allows the range of saturation measurement to be extended as microwave power captured by the receiving antenna is sensitive to proper alignment and has to be above the minimum detection level of the power sensor connected on the receiving end of the apparatus.

IV.B.1.b.xii. Power meters

The power meters used for the current study were sent out to the manufacturer for repair. Furthermore, they were re-certified for their reliability and performance according to the standards set forth by the NIST. This provides a quantifiable knowledge of their limits of uncertainty for any measurements made.

IV.B.1.c. Estimation of the extinction coefficient

To verify the absorption characteristics of the existing core holder and to design a core holder with improved performance, the extinction coefficients for water have to be evaluated. By performing a least-squares fit on the static dielectric constants of water versus the ambient temperature data (Malmberg and Maryott, 1956), a parametric equation was obtained (Hasted, 1973),

$$\varepsilon_s = 87.740 - 0.40008 \cdot T + 9.398 \cdot 10^{-4} \cdot T^2$$

$$-1.410 \cdot 10^{-6} \cdot T^3 ,$$

where T is the temperature of water in degrees Celsius. By coupling the parametric equation with Equations (M-12), (M-13) and (M-14), along with the published data on τ, α and ε_∞ (Hasted, 1973), the extinction coefficients of water at different temperatures with the given frequency of the microwave power were calculated. The data are presented in Table IV-1 and Figure IV-2.

IV.B.1.d. Selection of the required spacing for the double-stub tuner

To provide for a means of matching the impedance between the core and the receiving end of the microwave power measurement apparatus, a double-stub tuner was designed and built.

The double-stub tuner has two adjustable-length, shorted stubs with lengths d_1 and d_2, and a fixed length section of waveguides between them that must be less than half of the wavelength of the microwave power. For the microwave power measurement apparatus, the load of the core was not known before the design time, as the effects of the variation of the wetting-phase saturation upon the load could not be quantified and analyzed by using an analytical method. Hence, a distance of $\frac{3 \cdot \lambda}{8} + \frac{\lambda}{11}$ between the stubs was chosen to allow for the widest possible range of matching of the load for any slight variation in the load or the frequency (Laverghetta, 1976). The design of the double stub tuner is presented in Figure IV-3 and the data is presented in Table IV-2.

IV.B.1.e. Evaluation of the power measurement conversion factors

Because of the directional couplers used, the measured microwave power must be multiplied by a conversion factor to convert it to the proper magnitude. Using the microwave apparatus shown in Figures IV-4 and IV-5, the conversion factor for the power meter HP432C was evaluated. For the same reason, the conversion factor for the power meter HP432A was evaluated using the apparatus shown in Figures IV-6 and IV-7. To evaluate the conversion factor of the power meter at the receiving end of the apparatus, the insertion loss of the double-stub tuner had to be measured. By using the apparatus shown in Figures IV-8 and IV-9, the insertion loss of the double-stub tuner was evaluated and the conversion factor for the power meter HP436A was calculated. The data are presented in Table IV-2.

IV.B.1.f. Evaluation of the available microwave power at the transmitting antenna

By using the microwave apparatus shown in Figures IV-10 and IV-11, the insertion loss of the slide-screw tuner in the microwave circuit with the screw withdrawn was evaluated. Based on the microwave apparatus shown in Figure IV-12, the microwave power level available at the input port of the transmitting antenna was estimated by using the insertion loss evaluated and by assuming that there would be no reflected power coming from the direction of the core. The data is presented in Table IV-2.

IV.B.1.g. Estimation of the noise level of the power meter HP436A

Using the apparatus shown in Figure IV-7, with a successively decreasing setting on the attenuator, the noise level of the power meter HP436A was determined. The data are presented in Table IV-2. To prevent any erroneous measurements from being recorded in the range where the noise could be interpreted as measured power, and to allow for faster capturing of data with an appropriate sensor saturation time, an arbitrary requirement of 100 micro-watts arriving at the receiving antenna was imposed at the 100 % water saturation point for designing a new core holder.

IV.B.1.h. Selection of the appropriate measurement time required for proper data capturing

Based on the information presented in the operating manual of the power meter HP436A, a minimum time required at each cross-section along the core was selected, to allow for proper capturing of the microwave power by the receiving power sensor, even at the lowest expected power level when the core is fully saturated with water. The time selected based on the optimal power meter performance, 0.1 seconds for each cross-section, however, has the potential of causing an erroneous saturation profile to be measured and a numerical method has to be used to correct for any potential deviation of the measured profile from the true profile.

IV.B.1.i. Selection of the optimal motor advancement speed and the gear-box ratio

To further minimize any errors from being introduced into the saturation profiles measured under dynamic conditions, the time required for taking a complete scan of the core has to be kept to a minimum. Because the minimum time required for proper capturing of the microwave power at each cross-section of the core cannot be shortened further, there exists only the possibility of advancing the core across the antennas at the highest speed possible when measurements are not being taken.

Based on the limitations of both an internal buffer that translates computer generated pulses into steps for the stepper motor to advance, and the highest frequency that a computer can generate pulses without loading down other operations performed simultaneously by the computer, the maximum frequency to be used by the computer to generate core advancing pulses can be calculated. The data is presented in Table IV-2. Using the calculated frequency to generate pulses, a complete scan of the core can be made in under 35 seconds.

IV.B.1.j. Estimation of the required core thickness

The primary objective of having a non-metallic core holder is not only to have a housing for the unconsolidated sand-pack, but also to allow for a means to measure the fluid saturation at any point in the core holder at any time. The core holder, however, has to allow for enough microwave power to pass through even at the maximum attenuation level of the microwave absorbing fluid. From Equation (M-9) and (M-10), one can see that the only restriction that can be relaxed, under the dynamic saturation range, is the thickness of the core. The thickness of the core, furthermore, is dictated by the microwave power available at the transmitting antenna as well as the lowest detection level imposed on the receiving power sensor.

IV.B.1.j.i. Theoretical verification of the attenuation characteristics of the existing core holder

The existing core holder, with an internal thickness of 1.18 cm for the sand pack, was suspected of having excessive attenuation of the microwave power at the 100 % saturation level. By using Equation (M-7), along with the input values from Table IV-3, expected values of the microwave power arriving at the receiving antenna were evaluated. The data are presented in Table IV-4. Note that with 100 % water saturation, the attenuated power level drops off to 0.084 micro-watts, below the published power sensor specifications, thus partially confirming the suspicion mentioned above.

IV.B.1.j.ii. Experimental verification of the attenuation characteristics of the existing core holder

To confirm the theoretical evaluation, measurements were made using the existing core holder. The data, presented in Table IV-5, were plotted in conjunction with the theoretical data from Table IV-4. The shift in the trend of the experimental data from the trend of the theoretical data in Figure IV-13 simply confirms that the absorbency constant, A_{ls}, was not being considered in the theoretical evaluation. Note that, even at a 1 micro-watt received power level (corresponds to an approximated value of -5.18 for the power ratio $Log\frac{I_x}{I_o}$), the manufacturer specified capability of the power sensor and the power meter combination, the maximum water saturation detectable by the apparatus is still being capped at approximately 75 % of the total saturation range on the actual response curve in Figure IV-13. Using the arbitrary requirement of 100 micro-watts (corresponds to an approximated value of -3.18 for the power ratio $Log\frac{I_x}{I_o}$) to be detected by the receiving antenna, the maximum saturation in the core detectable by the apparatus drops off to approximately 40 % on the actual response curve in Figure IV-13. Both the theoretical and experimental data confirm that the existing core holder can not meet the requirements for the current study and an improved core holder is required.

IV.B.1.j.iii. Theoretical consideration of the attenuation characteristics of the improved core holder

To design a new core holder for the current study, the maximum thickness of the sand pack has to be estimated analytically, to allow for a minimum detectable microwave power by the receiving antenna at 100 % water saturation. Using data from Table IV-6, along with the arbitrarily set lowest power level of 100 microwatts, the theoretical thickness of the core was evaluated to be 0.3952 cm by using Equation (M-9) and Equation (M-10). The actual core, however, was designed with a thickness of 0.3000 cm, giving a 25 % safety margin for any unexpected attenuation of microwave power by the configuration of the microwave power measurement apparatus. The dimensions of the improved core holder are presented in Table IV-7 and the projected attenuation characteristics of the improved core holder are presented in Figures IV-14 and IV-15. Note that, with the steeper attenuation characteristics curve resulting from requiring a minimum of 100 microwatts of power to be detected and applying the safety factor in the core design, the new core holder is expected to have less resolution in differentiating the fluid saturation at any level.

IV.B.1.j.iv. Statistical uncertainty analysis of the attenuation characteristics of the new core holder

Based on the above analyses, a decision was made to construct a Plexiglas core holder with adequate strength to sustain a minimum 20 psig internal pressure and still provide a microwave power passage through a fully saturated core with a thickness of 0.3000 cm.

Because all the input values, except the speed of light, do have a range of uncertainty in their measured values, the received attenuated microwave power also has a range of uncertainty.

Based on the input values as presented in Table IV-8, the range of uncertainty of the attenuated power was evaluated at each saturation point by using the Transformation of Moment program, VARSIM. The data are presented in Table IV-9, Figures IV-16 and IV-17. Note that at any given core saturation value, the variation in the microwave attenuation is caused mainly by incident power variation. The range of uncertainty presented in Figure IV-17, however, represents the maximum uncertainty at a fixed cross-section of the core holder due to the experimental apparatus only. Local variations in the thickness, the porosity or the saturation level of the core under study are not taken into account.

IV.B.2. Pressure-transducer calibration apparatus

An individually measured pressure by a pressure transducer calibrated with the existing calibration cell had a maximum relative uncertainty[1] of 5.5 %, mainly due to an approximately 0.2 psig offset that exists between each of the calibrated transducers. The data are presented in Tables IV-10 and IV-11 and Figures IV-18 and IV-19. The uncertainty in the pressure profiles measured for the complete core is presented in Figure IV-20. A calibration cell with a slanted manometer and with two pressure equalization blocks was constructed, allowing for the simultaneous calibration of all transducers with their respective wetting fluids. An absolute pressure transducer was installed to correct for any changes in the atmospheric pressure during the process of calibration and the calibration procedures were modified as well.

The improved calibration apparatus allows one to eliminate any offsets that might exist between the pressure transducers and to maintain their respective wettabilities after the calibration process. The apparatus is presented in Figure IV-21. The improved calibration procedure involves a 10 step procedure in which the pressure increases. A calibrated and known pressure is applied to the transducers and the responses of the transducers are recorded. This process is repeated 10 times to increase the calibration pressure from 0.0 psig to 10.0 psig. A least-squares fit can then be performed on the data, using the applied calibration pressure as the abscissa and the transducer response as the ordinate. Assuming that the calibration pressure has no inherent uncertainty, the slope and the intercept

---

[1] Relative uncertainty = Uncertainty / Expected value * 100 %

of the least-squares fit can then be calculated. The slope and the intercept for each of the transducers are then converted to the required formats in terms of gain and offset, for converting the transducer responses in millivolts into psig.

The calibration apparatus is capable of providing a 0.00628 psig per millimeter mercury rise in the manometer. When combined with the 10 step calibration procedures, the maximum relative uncertainty for the interpolated pressure within the range of interest is reduced to approximately 1.6 % from 2.7 % between all the transducers. The data are presented in Tables IV-12 and IV-13 and Figures IV-22 and IV-23. The level of uncertainty of the pressure profile measured for the core holder is presented in Figure IV-24.

IV.B.3. Elimination of the hydrostatic difference

Since the existing core holder required all the pressure transducers to be mounted in such a way that one is situated above the other in the vertical plane, a simple calibration cell was constructed to measure the in-situ hydrostatic difference between the transducers. After all the transducers were mounted onto the core holder, the calibration cell was connected to the core holder and a constant pressure was applied to the core holder. This allowed for the correction of all the transducer responses to the same reference datum. A calculation showed that approximately 0.2165 psig of static discrepancy was eliminated from the measured pressure for each transducer. Based on this, the new core holder was designed to allow for all the transducers to be mounted at the same elevation, thus eliminating the use of the calibration cell to correct for any hydrostatic difference between the transducer responses. The apparatus is presented in Figure IV-25.

IV.B.4. Dead-volume measurement apparatus

The ends of the core holder are sealed by two end caps. Both of the end caps have a fixed quantity of dead volume that can hold fluid. To ensure that the proper material balance calculations can be carried out on the experimental data, the dead volume has to be quantified properly. A simple volumetric measurement cell was constructed. The apparatus is presented in Figure IV-26.

IV.B.5. Traveling-track apparatus

To control precisely the advancement of the core with respect to the microwave antennas, a decision was made to change the motorized servo from the existing continuous rotating motor to a stepwise rotating motor. The improved apparatus allows advancing of the core at a precisely controlled speed as well as stopping at any specific point of the core in front of the microwave antennas for measurement taking. The apparatus is presented in Figure IV-27. A tape measure was mounted on the trolley and a laser pointer was mounted onto the transmitting antenna to allow for verifying the position of the core.

To allow for any accidental over travel of the trolley, a wax-type device is used for transferring the movement of the chain to the trolley. Any over travel of the trolley can be stopped by breaking the wax seal automatically at the stopping blocks located at the extreme ends of the track, thus terminating any transfer of motion from the chain drive to the trolley.

Because, during an actual experiment, vibrations from the movement of the trolley can be substantial, all the equipment is tied down using guy wires and turn-buckles.

IV.B.6. Vacuum-drawing apparatus

To saturate completely the core holder initially with water and to improve the wettabilities of the selective wetting capillary barriers, a vacuum-drawing apparatus was constructed. The apparatus is presented in Figure IV-28. The capillary barriers can be immersed in their respective wetting fluids inside a container and a vacuum is drawn on the container to ensure no air is trapped inside the capillary barriers. The capillary barriers can then be mounted into the inlet taps of the pressure transducers to maintain a proper capillary contact between the transducers and the unconsolidated sand pack.

IV.B.7. Fluid-injection apparatus

In order that the measured parameters obtained from a series of steady-state experiments are more equally spread over the entire saturation range, a decision was made to modify the injection pumps to allow for variable injection rates. The pumps were then repaired to eliminate any leakage in the linkages. They were subsequently calibrated using an electronic balance and timer. The outlet ends of the pumps were fitted with pulsation dampeners to ensure a smoother output of the fluid flows. The apparatus is presented in Figure IV-29.

IV.B.8. Configuration of the improved core holder

The improved core holder was designed specifically to allow for a minimum of 100 cm in length for in-situ saturation investigation without any inlet- or outlet-end effects. Moreover, the improved core holder allows all the transducers to be mounted at the same reference datum with equal spacing. The configuration is presented in Figure IV-30.

IV.B.9. Injection fluids

After the existing core holder with an unconsolidated sand pack inside was subjected to an extended period of flow of both water and oil, it was observed that the pressure readings between the water-wet transducers and the oil-wet transducers drifted. Further examination revealed that the sand grains in the packed core had developed a black coloration, and the oil was found to contain paraffinic and aromatic compounds. A survey of the literature revealed that these two compounds were suspected of having the potential of changing the wetting characteristics of a rock-fluid system. (Jennings, 1957). No alternative injection fluids have been found yet for the current study. A pure hydrocarbon in the alkene series with high molecular weight is suggested to be used in the future.

In order that the viscosity and density of the oil used can be corrected for any temperature effects, the oil was sent to a commercial laboratory for density and viscosity measurement. The data are presented in Table IV-14. The viscosity and density of the water to be used, however, were obtained from correlation tables and the data are presented in Table IV-14. Furthermore, in order to maintain the consistency of the water to be used, an ionic filter was installed to eliminate any ionic particles

IV.B.10. Injection caps

Injection caps are used not only to seal the ends of the core holder, but also to bleed off of any air entrapped in the injection caps. Furthermore, the injection caps designed for the improved core holder have reduced dead volume to allow for more accurate material balance calculations to be carried out for saturation evaluation. The design is presented in Figure IV-31.

IV.B.11. Pressure transducers

All pressure transducers are equipped with either an 8.0 psig or a 12.5 psig diaphragm. Because the experimental methods used to obtain the generalized permeability coefficients require the utilization of both the static and dynamic pressure profiles of the fluids measured at various cross-sections along the length of the core, a small range of static uncertainty and a good dynamic response are required for the pressure measurement. A decision was made to change all the selective wetting capillary barriers fabricated previously in-house to fritted discs and Teflon discs with commercial qualities. They were mounted onto the fluid intake ports of the pressure transducers. Fluid intake ports were enlarged with recesses to facilitate a better dynamic response between the capillary barriers and the pressure transducers. The apparatus is presented in Figure IV-32. To use the pressure transducers in any experiments, it is necessary to flush and fill them with their respective wetting fluids prior to the experiments.

IV.B.12. Weight-measurement apparatus

The existing weight-measurement apparatus is presented in Figure IV-33 and the equations are presented in Appendix A-1. However, the analyses as presented in Tables IV-15 and IV-16 and Figures IV-34, IV-35 and IV-36 show that the level of uncertainty in the calculated saturation levels is too high. The uncertainty is caused by errors in measuring the volume of the produced fluids (the configuration as presented in Figure IV-33 allows a +/- 5.0 c.c. in accuracy for each volumetric measurement), as well as the number of experimental steps required to attain the residual oil saturation. The errors in volumetric measurement of the produced fluids also cause a high uncertainty in the initial water saturation calculated. To facilitate more accurate material balance calculations, a weight measurement method of the produced fluids is used, whereby the weights of the produced fluids are measured by two electronic balances connected to the exit ends of the core holder. Furthermore, the balances are enclosed in draft shields and put on the top of vibration shields, allowing for real-time weight recording of the produced fluids through RS232 interfaces connected to a computer. The improved apparatus is presented in Figure IV-37 and the equations used are presented in Appendix A-2 The analyses presented in Tables IV-17 and IV-18 and Figures IV-38, IV-39 and IV-40 show that with the improved material balance method, the level of uncertainty in the saturation measurements is reduced substantially over the entire saturation range of interest.

IV.B.13. Hardware configuration for data collection

To collect all the necessary data systematically and repeatedly, all equipment was interfaced to a Windows based PC through interface cards. In addition, only shielded cables are used for connecting all the equipment to prevent any cross talk. A functional chart is presented in Figure IV-41.

IV.B.14. Software configuration for data collection

Data collection software was written in a modular format for the current study. The software has the flexibility to allow for any new types of experiments to be conducted with the experimental apparatus by properly combining the modules, and yet maintains the rigidity required so that the same experimental procedures can be followed and the same format of results can be obtained from different users. A functional chart is presented in Figure IV-42 and the logic codes are presented in Appendix B. A working copy of the programs is included with the thesis and can be read by Visual Basic®[2] version 3.0 or higher.

IV.B.15. Tubing connections

As air has a much lower extinction coefficient than water, any trapped air in the system could cause erroneous interpretation of the water saturation in the core. A decision was made to change all the connections to quick connects to reduce any dead volume of air entrapped during the interchanging of connections.

IV.B.16. Core packing technique

It was found that using a hammer to induce a sudden acceleration from the bottom of the vertically hanging core holder had a better result than using a mechanical vibrator to further consolidate the loose sand grains. As the internal thickness of the improved core holder is substantially thinner than the existing core holder, it is harder to pack the core into a homogeneous porous medium, the improved packing technique should be used and the improved core holder was designed to handle shocks induced by such hammering. This will provide a uniformly packed core with consistent porosity along the length, and thus reduce the statistical uncertainty of received microwave power.

---

[2] Visual Basic® is a registered trademark of Microsoft Corporation.

IV.C. Improved experimental procedure

IV.C.1. Steady-state cocurrent flow experiments

a. Measure the dead volume of preparation caps. The volumes are designated as dead volume $DV1$ and dead volume $DV2$.

b. Wet sieve the 70 - 140 mesh Ottawa sand to 80 - 120 mesh size three times to ensure a proper size distribution and consistency.

c. Measure the weight of the empty core holder along with all the necessary dummy plugs, both preparation caps, screws and O-rings. The weight is designated as empty weight $EW1$.

d. Measure the weight of the water filled core holder along with all the necessary dummy plugs, both preparation caps, dead volume water, screws and O - rings. The weight is designated as filled weight $FW1$.

e. Wet pack core

- **i.** Suspend the empty core holder vertically with the bottom preparation cap and the dummy plugs on, leaving off the top preparation cap.
- **ii.** Fill the core holder with a constant height of filtered water.
- **iii.** Fill the core holder with the sieved sand.
- **iv.** Maintain a constant height of filtered water above the top of the sand while repeating the filling procedure.
- **v.** Put on the top preparation cap.
- **vi.** Vibrate the core holder for 24 hours, then hammer the core holder for another 2 hours to consolidate all loose sand grains.

f. Weigh the packed core holder. The weight is designated as weight $CW1$.

g. Suspend the core holder vertically and test for leakage with 25.0 psig air from the top of the core holder with leak detection compound.

h. Air dry the packed core with compressed air from the top of the core holder for 48 hours.

i. Weigh the dry packed core with the dummy plugs and the preparation caps. The weight is designated as dry weight $DW1$.

j. Pull a vacuum on the packed core for 48 hours.

k. Scan the core holder at zero water saturation with the microwave setup, store the data for future S_1 calibration curve construction at $S_1 = 0.0$.

l. Saturate the packed core holder with filtered water by imbibition from the bottom up, ensuring that there is no trapped air in the system during the process of making connections; regulate the water imbibition rate into the core holder using a very slow rate, approximately 10 cc/hour.

m. Weigh the packed core with dummy plugs, both preparation caps and 100 % saturated sand; the weight is designated as $WW1$ and should be at least equal to $CW1$ measured previously. This step is for diagnostic purposes only.

n. If necessary, flush the core with water at the highest possible rate combined from two pumps to ensure 100 % water saturation of core.

o. Calculate the pore volume of packed core using the material balance equations presented in Appendix A-2.

p. Change all the dummy plugs to the transducers, put the water wet transducers on one side and the oil wet transducers on the other side of the core holder.

i. During the installation of the water-wet transducers, let the water circulate into the core by injecting from the Ruska pump.

ii. During the installation of the oil-wet transducers, stop the circulation of the water and add additional oil into the pressure tap to ensure good contact of the Teflon disc and the core body.

q. Scan the packed core at 100 % water saturation; store the data for future S_1 calibration curve construction at $S_1 = 1.0$.

r. Set up the Heise gauge to determine if any offsets exist between the transducers and eliminate them if necessary by adjusting the offset in the transducer calibration coefficients.

s. Calculate the absolute permeability of the packed core with the filtered water, using Darcy's law. Repeat several times at different injection rates to obtain a statistical mean of the permeability.

t. Let the core sit for 24 hours with 100 % water saturation to render it water wet.

u. Inject oil at a rate higher than that required to obtain S_{1i}, record the injection time, t, oil injection rate, q_2, and weight produced at the exit end of the core, W; calculate the initial water saturation, S_{1i}, using the material balance equation presented in Appendix A-2. Scan at S_{1i}, store the data for future S_1 calibration ($S_1 = S_{1i}$).

v. Calculate the effective oil permeability of the packed core at S_{1i}, using Darcy's law. Repeat several times at different injection rates to obtain a statistical mean of the effective permeability.

w. Change to the injection caps, ensuring that no air is trapped during the preparation process.

x. Inject both water and oil in a fixed ratio from the same end of the core holder until steady-state is achieved; record the saturation profile and the pressure profile. Repeat for 10 different steady-state saturation stages. Record the injection rates used, time of injection and the weight of fluids produced at the exit end of the core holder. Use the material balance equation presented in Appendix A-2 to calculate the saturation level.

y. Flush the core to S_{1m} with the highest possible water rate.

z. Scan the core at S_{1m}, store the data for future calibration curve construction at $S_1 = S_{1m}$.

IV.C.2. Steady-state countercurrent flow experiments

Repeat all the procedures except for step x from the above procedures, inject the water and the oil into opposite ends of the core holder.

IV.C.3. Unsteady-state cocurrent flow experiment

Repeat all the procedures except for step x from the above procedures, injecting the water and the oil at a rate that would produce a stable and unstabilized displacement The injection rates have to be determined in a trial and error manner be observing the saturation profiles produced in order to obtain the most appropriate rates that can minimize any model variations.

Table IV-1: Extin... temperature for water at 28.0 GHz

T_K	τ	α	ε_∞	ε_s	ε'	ε''	k
20.0	9.30E-12[1]	0.0130[2]	4.2300[3]	80.103040	25.363813	33.246516	2.868196
21.0	9.09E-12	0.0129	4.2270	79.739714	25.932275	33.418022	2.860701
22.0	8.88E-12	0.0128	4.2240	79.378090	26.524095	33.584141	2.852282
23.0	8.67E-12	0.0127	4.2210	79.018159	27.140279	33.743836	2.842866
24.0	8.46E-12	0.0126	4.2180	78.659913	27.781859	33.895957	2.832376
25.0	8.25E-12	0.0125	4.2150	78.303344	28.449882	34.039225	2.820726
26.0	8.04E-12	0.0124	4.2120	77.948443	29.145408	34.172220	2.807828
27.0	7.83E-12	0.0123	4.2090	77.595201	29.869501	34.293373	2.793583
28.0	7.62E-12	0.0122	4.2060	77.243611	30.623219	34.400947	2.777890
29.0	7.41E-12	0.0121	4.2030	76.893663	31.407605	34.493025	2.760636
30.0	7.20E-12[4]	0.0120[5]	4.2000[6]	76.545350	32.223670	34.567494	2.741704

Excessive significant digits are used for the purpose of carrying out numerical evaluations only.

[1] Data from Hasted, 1973.
[2] Data from Hasted, 1973.
[3] Data from Hasted, 1973.
[4] Data from Hasted, 1973.
[5] Data from Hasted, 1973.
[6] Data from Hasted, 1973.

Table IV-2: Design parameters used for the microwave power measurement apparatus

Parameters:	Expected value
432A conversion factor	10.05081433
432C conversion factor	98.10121137
436A conversion factor	1.01269811
Slide screw tuner insertion loss	0.0900 dB
Double stub tuner insertion loss	0.0548 dB
Power available at the transmitting antenna	149.95 milliwatts
436A noise level	40.00 microwatts
Stepping motor conversion factor	367 pulses/ cm core travel
436A response time at 1 micro-watt	10.0 seconds[1]
436A response time at 10 micro-watts	1.0 second[2]
436A response time at 100 micro-watt	0.1 second[3]
Double stub tuner spacing	2.01 cm

Excessive significant digits are used for the purpose of carrying out numerical evaluations only.

[1] Data from HP436A operating manual.
[2] Data from HP436A operating manual.
[3] Data from HP436A operating manual.

Table IV-3: Input parameters used for the evaluation of the microwave power attenuation characteristics of the existing core holder

Parameters:	Expected value
$f(Hz)$	$2.80 \cdot 10^{10}$
L_x (cm)	1.18
k	2.820726
ϕ $(fraction)$	0.3691
c $(cm / sec.)$	$2.9979 \cdot 10^{10}$
I_o (mW)	153.0900

Excessive significant digits are used for the purpose of carrying out numerical evaluations only.

Table IV-4: Theoretical evaluation of microwave power attenuation characteristics of the existing core holder

S_w (fraction)	I_o assumed (mW)	I_x calculated (mW)	$Log\left(\frac{I_x}{I_o}\right)$ calculated (dimensionless)
0.0	153.0900	153.09000000	0.00000000
0.1	153.0900	36.21204108	-0.62609382
0.2	153.0900	8.56562754	-1.25218764
0.3	153.0900	2.02612095	-1.87828146
0.4	153.0900	0.47926040	-2.50437527
0.5	153.0900	0.11336467	-3.13046909
0.6	153.0900	0.02681538	-3.75656291
0.7	153.0900	0.00634293	-4.38265673
0.8	153.0900	0.00150036	-5.00875055
0.9	153.0900	0.00035490	-5.63484437
1.0	153.0900	8.395E-05	-6.26093819

Excessive significant digits are used for the purpose of carrying out numerical evaluations only.

Table IV-5: Experimental measurements of microwave power attenuation characteristics of the existing core holder

File name	S_w measured (fraction)	I_o measured (mW)	I_x measured (mW)	$Log\left(\frac{I_x}{I_o}\right)$ calculated (dimensionless)
SCAN6.DAT	0.0000	153.0900	12.5000	-1.08803681
12SCAN1.DAT	0.1200	163.8200	4.0545	-1.60642962
SSCO4.DAT	0.1485	164.7270	1.4718	-2.04891600
34SCAN2.DAT	0.3400	160.0000	0.2064	-2.88941029

Excessive significant digits are used for the purpose of carrying out numerical evaluations only.

Table IV-6: Input parameters used for designing the thickness of the improved core holder

Parameters:	**Expected value**
$f\,(Hz)$	$2.80 \cdot 10^{10}$
k	2.820726
ϕ $(fraction)$	0.3691
$c\,(cm/sec.)$	$2.9979 \cdot 10^{10}$
$I_o\,(mW)$	153.0900
I_x @ $S_w = 0\,(mW)$	12.5000
I_x @ $S_w = 1\,(mW)$	0.1000

Excessive significant digits are used for the purpose of carrying out numerical evaluations only.

Table IV-7: Dimensions of the improved core holder

Internal width - L_x (cm)	Internal height L_z (cm)	Internal length L_y (cm)	Internal cross-sectional area $A\ (cm^2)$
0.30	5.00	140.00	1.50

Table IV-8: Input parameters used for the evaluation of the microwave power attenuation characteristics of the improved core holder

Parameters:	Expected value	Uncertainty
$f\,(Hz)$	$2.80 \cdot 10^{10}$	$2.50 \cdot 10^{8}$
$h\,(cm)$	0.30	0.00
k	2.820726	0.024548
$\phi\ (fraction)$	0.3691	0.0000
$c\,(cm/sec.)$	$2.9979 \cdot 10^{10}$	0.0000
$I_o\,(mW)$	153.0900	15.3090
$S_w\ (fraction)$	*Various*	0.0

Excessive significant digits are used for the purpose of carrying out numerical evaluations only.

Table IV-9: Theoretical evaluation of microwave power attenuation level, lower and upper bounds of the power ratio calculated, and level of relative uncertainty in the power ratio calculated for the improved core holder

S_w (dimensionless)	I_x calculated (mW)	Uncertainty of I_x calculated (mW)	$Log\left(\frac{I_x}{I_o}\right)$[1] calculated (dimensionless)	Upper bound of $Log\left(\frac{I_x}{I_o}\right)$[2] calculated (dimensionless)	Lower bound of $Log\left(\frac{I_x}{I_o}\right)$[3] calculated (dimensionless)	+/- relative uncertainty in $Log\left(\frac{I_x}{I_o}\right)$ calculated (%)
0.0	12.5000	1.2500001	-1.08803678	-1.00088660	-1.17518695	8.0099
0.1	8.6644	0.86710	-1.24720945	-1.16003006	-1.33439285	6.9903
0.2	6.0057	0.60310	-1.40638213	-1.31906664	-1.49373639	6.2099
0.3	4.1629	0.40330	-1.56555481	-1.[illegible]7963702	-1.65119469	5.4793
0.4	2.8855	0.29700	-1.72472749	-1.63642134	-1.81329082	5.1275
0.5	2.0001	0.20510	-1.88390017	-1.79572314	-1.97229742	4.6865
0.6	1.3864	0.14380	-2.04307284	-1.95444282	-2.13200608	4.3456
0.7	0.9610	0.10090	-2.20224552	-2.11311236	-2.29178235	4.0569
0.8	0.6661	0.07197	-2.36141820	-2.27109208	-2.45245765	3.8402
0.9	0.4617	0.04986	-2.52059088	-2.43028927	-2.61161530	3.5967
1.0	0.3200	0.03515	-2.67976355	-2.58877564	-2.77172562	3.4134

Excessive significant digits are used for the purpose of carrying out numerical evaluations only.

[1] I_o = nominal I_o; I_x = nominal I_x
[2] I_o = 90 % of nominal I_o; I_x = nominal I_x + Uncertainty
[3] I_o = 110 % of nominal I_o; I_x = nominal I_x - Uncertainty

Table IV-10: Evaluation of the conversion factors for transducer response - 2-point calibration procedure

Parameters	Expected value	Uncertainty
Calibrated pressure 1 (psig)	1.0000	0.01934
Calibrated pressure 2 (psig)	10.0000	0.01934
Transducer response 1 (millivolt)	5.0000	0.0500
Transducer response 2 (millivolt)	40.0000	0.4000
Slope (millivolt/psig)	3.88888889	0.046323430
Intercept (millivolt)	1.11111111	0.071635244
Gain (psig/millivolt)	0.25714285	0.30630185E-2
Offset (psig)	-0.28571425	0.18741034E-1

Excessive significant digits are used for the purpose of carrying out numerical evaluations only.

Table IV-11: Evaluation of the lower and upper bounds, and relative uncertainty in the interpolated pressure with transducers calibrated using 2-point calibration procedure

Transducer response (millivolt)	Interpolated pressure (psig)	Lower bound of interpolated pressure (psig)	Upper bound of interpolated pressure (psig)	+/- relative uncertainty in interpolated pressure (%)
5.0	0.99999998	0.97259372	1.02740588	2.7406
10.0	2.28571380	2.24153954	2.32988807	1.9326
15.0	3.57142780	3.50857913	3.63427647	1.7598
20.0	4.85714180	4.77499568	4.93928792	1.6912
25.0	6.14285580	6.04113205	6.24457955	1.6560
30.0	7.42856980	7.30715206	7.54998754	1.6345
35.0	8.71428380	8.57308177	8.85548583	1.6204
40.0	10.00000000	9.83893462	10.16106538	1.6107

Excessive significant digits are used for the purpose of carrying out numerical evaluations only.

Table IV-12: Evaluation of the conversion factors for transducer response - 10 -point calibration procedure

Parameters	Expected value	Uncertainty
Calibrated pressure 1 (psig)	1.0000	0.000628
Calibrated pressure 2 (psig)	10.0000	0.000628
Transducer response 1 (millivolt)	5.0000	0.0500
Transducer response 2 (millivolt)	40.0000	0.4000
Slope (millivolt/psig)	3.7619933	0.023099409
Intercept (millivolt)	-1.2311427	0.078049307
Gain (psig/millivolt)	0.26581655	0.16322990E-2
Offset (psig)	0.32725818	0.20849792E-1

Excessive significant digits are used for the purpose of carrying out numerical evaluations only.

Table IV-13: Evaluation of the lower and upper bounds, and relative uncertainty in the interpolated pressure with transducers calibrated using 10-point calibration procedure

Transducer response (millivolt)	Interpolated pressure (psig)	Lower bound of interpolated pressure (psig)	Upper bound of interpolated pressure (psig)	+/- relative uncertainty in interpolated pressure (%)
5.0	1.65634070	1.63030076	1.68238064	1.5721
10.0	2.98542320	2.94790940	3.02293700	1.2566
15.0	4.31450570	4.26327165	4.36573976	1.1875
20.0	5.64358820	5.57779723	5.70937917	1.1658
25.0	6.97267070	6.89193219	7.05340921	1.1579
30.0	8.30175320	8.20585982	8.39764658	1.1551
35.0	9.63083570	9.51966363	9.74200777	1.1543
40.0	10.95991800	10.83344365	11.08639235	1.1540

Excessive significant digits are used for the purpose of carrying out numerical evaluations only.

Table IV-14: Fluid properties

Wetting-phase	Temperature (degree Celsius)	Density (gram/c.c.)	Dynamic viscosity (cp.)
Water[1]	20.0	0.998203	1.0020
	25.0	0.997044	0.8904
	30.0	0.995646	0.7975
Non-wetting-phase	**Temperature (degree Celsius)**	**Density (gram/c.c.)**	**Dynamic viscosity (cp.)**
LAGO[2]	20.0	0.848400	6.1400
	25.0	0.852000	5.3390
	30.0	0.855600	4.6690

Excessive significant digits are used for the purpose of carrying out numerical evaluations only.

[1] Data from CRC Handbook of Chemistry and Physics, 53rd edition (1972), The Chemical Rubber Company.
[2] Data from Core Laboratory of Canada; report number 52136-96-0621.

Table IV-15: Evaluation of the pore volume, porosity and initial water saturation calculated by conventional material balance method

Measured Parameters	**Expected value**	**Uncertainty**
Empty core holder weight (*gram*)	6304.90	1.00
Water filled core holder weight (*gram*)	6961.10	2.00
Dry core weight (*gram*)	7356.00	1.00
Wet core weight (*gram*)	7600.70	2.00
Water density (*gram/c.c.*)	0.997044	0.000261
Dead volume #1 (*c.c.*)	2.00	0.01
Dead volume #2 (*c.c.*)	2.00	0.01
Total fluid produced (*c.c.*)	1208.00	5.00
Total oil produced (*c.c.*)	1000.00	5.00
Fraction of oil in end cap #2 (*fraction*)	0.50	0.20
Calculated parameters	**Expected value**	**Uncertainty**
Pore volume calculated (*c.c.*)	241.42548	2.2436794
Porosity calculated (*fraction*)	0.36907001	0.36584312E-2
Initial water saturation (*fraction*)	0.15087669	0.30367497E-1

Excessive significant digits are used for the purpose of carrying out numerical evaluations only.

Table IV-16: Evaluation of the uncertainty, lower and upper bounds in water saturation calculated by conventional material balance method

Experimental steps	Saturation calculated (*fraction*)	Uncertainty of saturation calculated (*fraction*)	Lower bound of saturation calculated (*fraction*)	Upper bound of saturation calculated (*fraction*)
Swi determination	0.1509	0.0304	0.1205	0.1812
1	0.2544	0.0368	0.2176	0.2912
2	0.3580	0.0422	0.3157	0.4002
3	0.4615	0.0471	0.4145	0.5086
4	0.5651	0.0514	0.5136	0.6165
5[1]	0.66[illegible]	0.0561	0.6125	0.7248
6[2]	0.7[illegible]	0.0607	0.7115	0.8329
7[3]	0.87[illegible]	0.0653	0.8104	0.9411
8[4]	0.9793	0.0699	0.9094	1.0492

Excessive significant digits are used for the purpose of carrying out numerical evaluations only.

[1] Linearly extrapolated data.
[2] Linearly extrapolated data.
[3] Linearly extrapolated data.
[4] Linearly extrapolated data.

Table IV-17: Evaluation of the pore volume, porosity and initial water saturation calculated by improved material balance method

Measured Parameters	Expected value	Uncertainty
Water density (*gram/c.c.*)	0.997044	0.0000261
Oil density (*gram/c.c.*)	0.852000	0.0000720
Total fluid produced (*gram*)	907.06	0.01
Oil injection rate (*c.c./hr.*)	25.0000	0.0025
Total injection time (*hr.*)	50.0000	0.0000
Cross section area (cm^2)	1.500	0.015
Porosity (*fraction*)	0.36907001	0.36584312E-2
Length of core (*cm*)	140.00	0.01
Dead volume #1 (*c.c.*)	2.00	0.01
Dead volume #2 (*c.c.*)	2.00	0.01
Initial fraction of water in end cap #1 (*fraction*)	1.00	0.00
Final fraction of water in end cap #1 (*fraction*)	0.00	0.20
Initial fraction of water in end cap #2 (*fraction*)	1.00	0.00
Final fraction of water in end cap #2 (*fraction*)	0.50	0.20
Calculated parameters	**Expected value**	**Uncertainty**
Initial water saturation (*fraction*)	0.15088319	0.19918023E-2

Excessive significant digits are used for the purpose of carrying out numerical evaluations only.

Table IV-18: Evaluation of the uncertainty, lower and upper bounds in water saturation calculated by improved material balance method

Experimental steps	Saturation calculated (*fraction*)	Uncertainty of saturation calculated (*fraction*)	Lower bound of saturation calculated (*fraction*)	Upper bound of saturation calculated (*fraction*)
Swi determination	0.1509	0.1992E-2	0.1489	0.1529
1	0.2019	0.6549E-2	0.1953	0.2084
2	0.2528	0.9060E-2	0.2437	0.2618
3	0.3022	0.1102E-1	0.2912	0.3132
4	0.3549	0.1639E-1	0.3385	0.3713
5	0.4000	0.1773E-1	0.3823	0.4178
6[1]	0.4500	0.2183E-1	0.4282	0.4718
7[2]	0.5000	0.2531E-1	0.4747	0.5253
8[3]	0.5500	0.2878E-1	0.5212	0.5788
9[4]	0.6000	0.3226E-1	0.5677	0.6323
10[5]	0.6500	0.3573E-1	0.6143	0.6857
11[6]	0.7000	0.3921E-1	0.6608	0.7392
12[7]	0.7500	0.4269E-1	0.7073	0.7927
13[8]	0.8000	0.4616E-1	0.7538	0.8462

Excessive significant digits are used for the purpose of carrying out numerical evaluations only.

[1] Linearly extrapolated data.
[2] Linearly extrapolated data.
[3] Linearly extrapolated data.
[4] Linearly extrapolated data.
[5] Linearly extrapolated data.
[6] Linearly extrapolated data.
[7] Linearly extrapolated data.
[8] Linearly extrapolated data.

Figure IV-1: Microwave power measurement apparatus

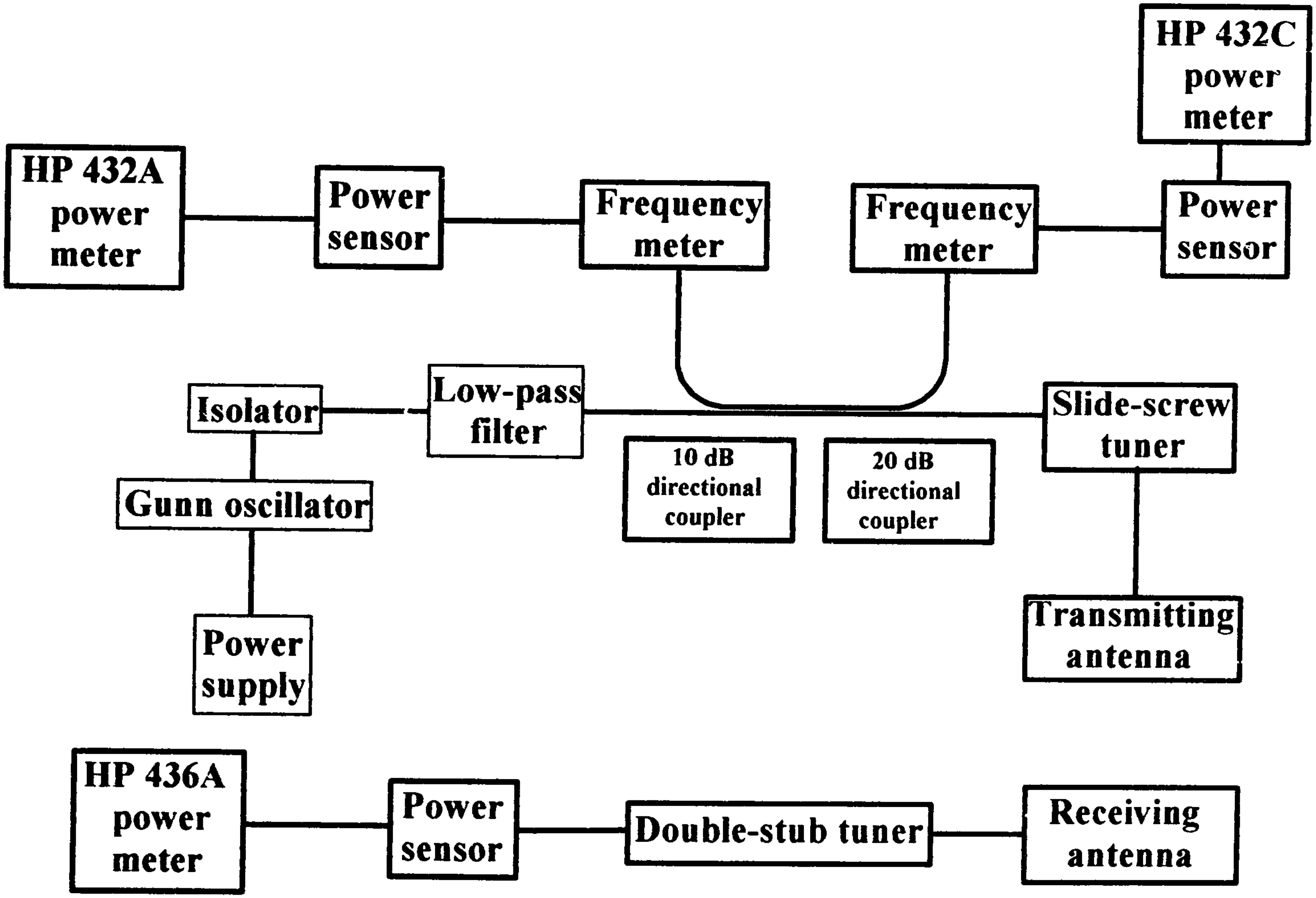


Figure IV-2: Extinction coefficients versus temperature for water at 28.0 GHz

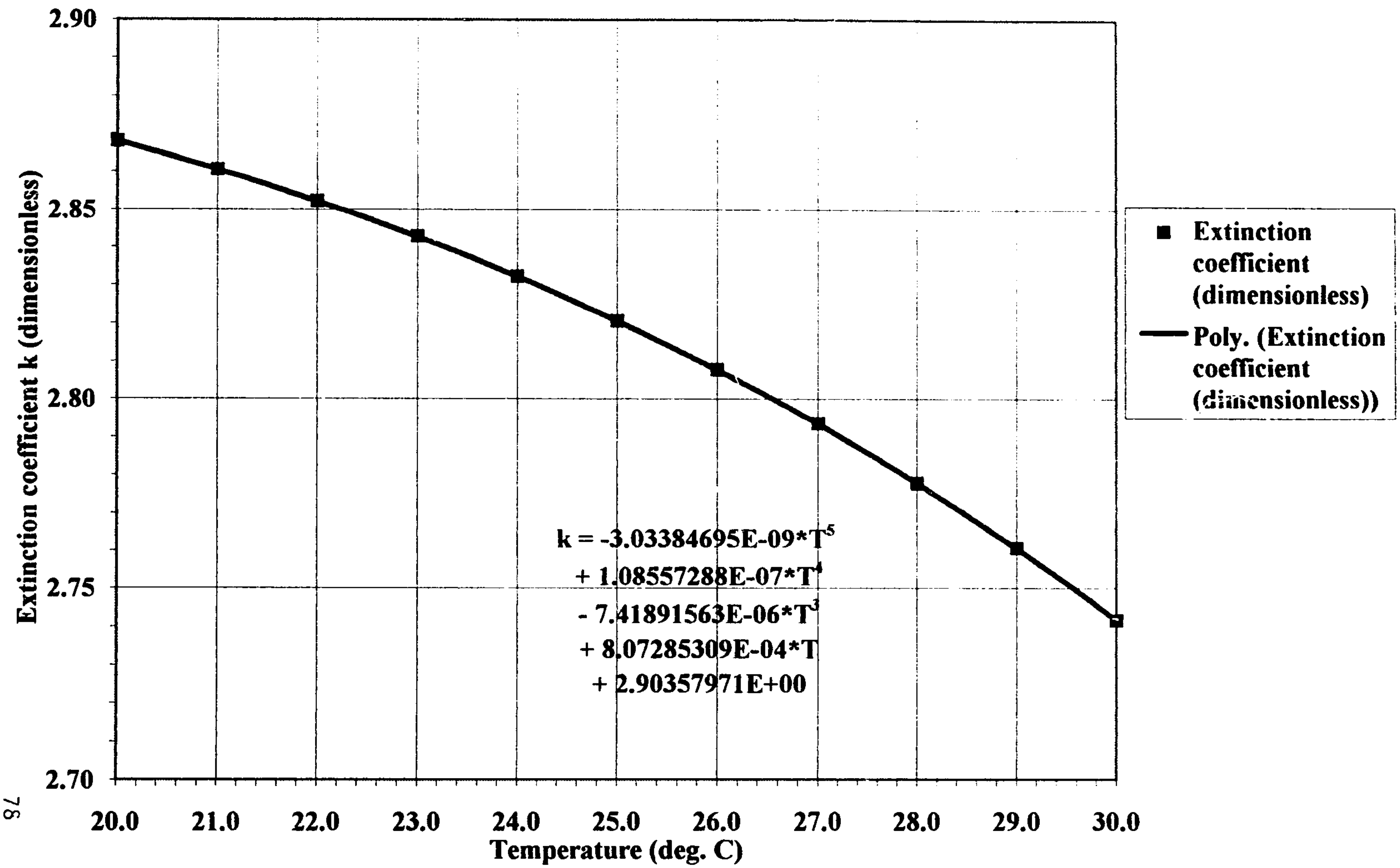


Figure IV-3: Double-stub tuner design

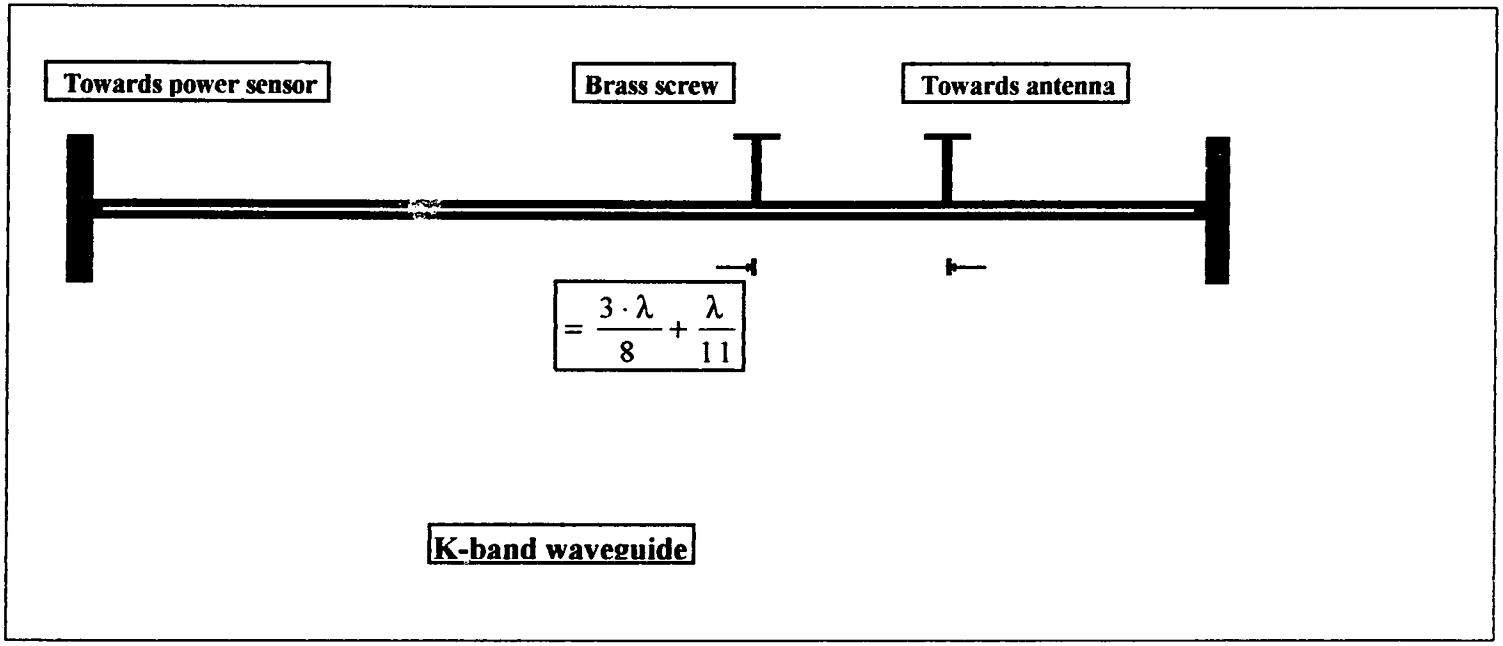


Figure IV-4: Configuration #1 - HP432C conversion coefficient measurement

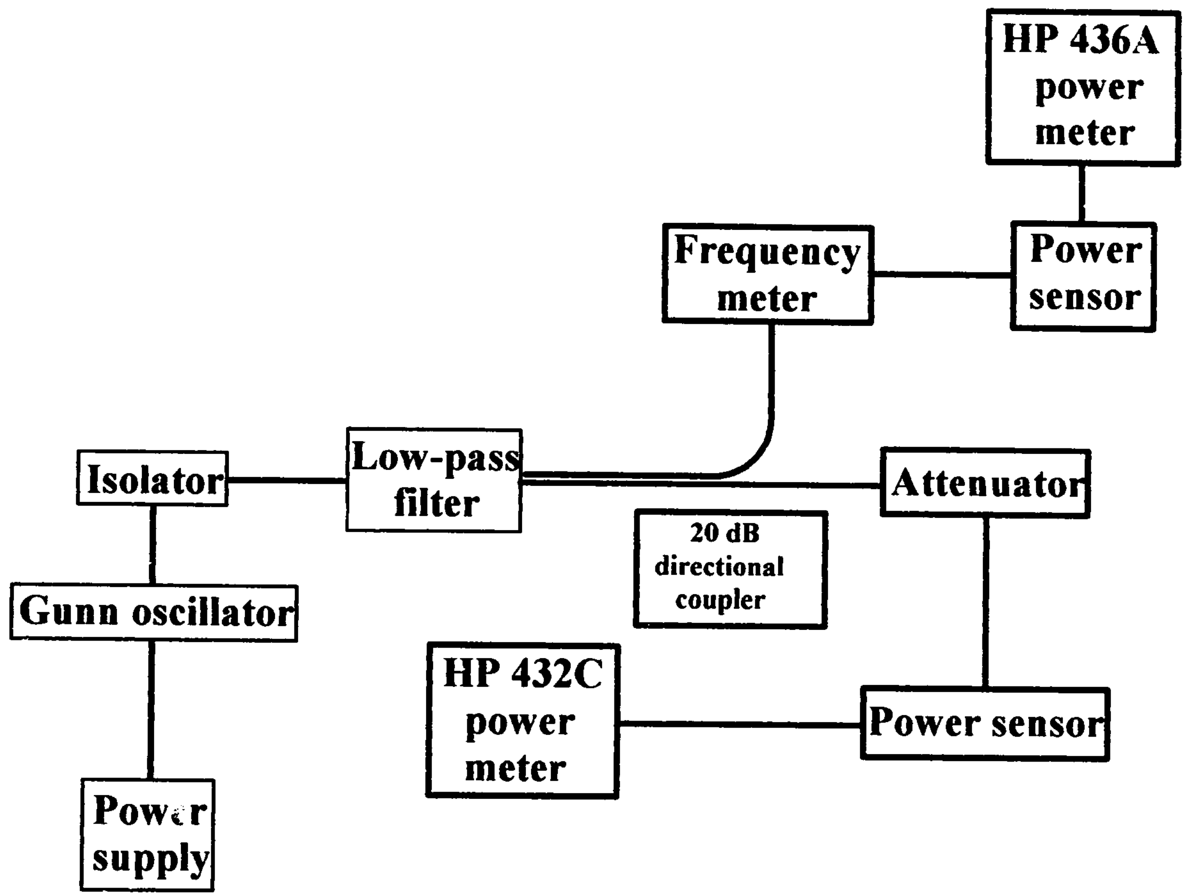


Figure IV-5: Configuration #2 - HP432C conversion coefficient measurement

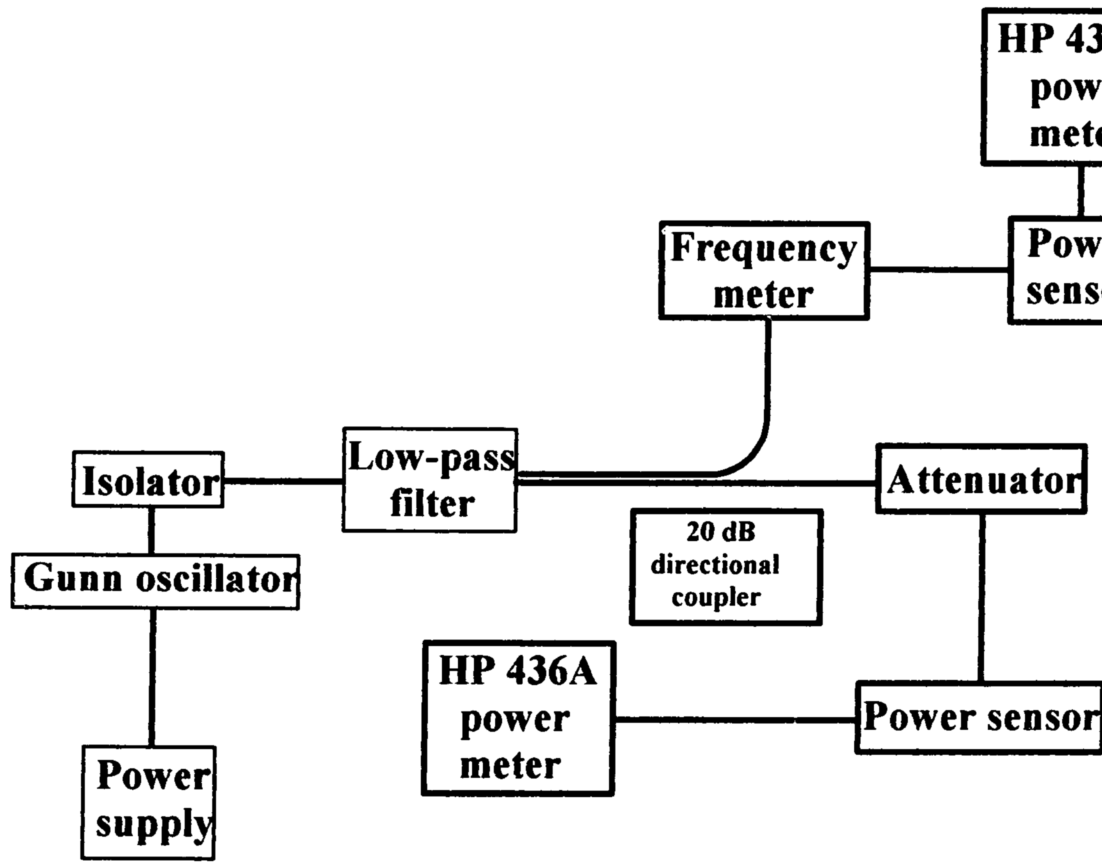


Figure IV-6: Configuration #1 - HP432A conversion coefficient measurement, HP436A noise level determination

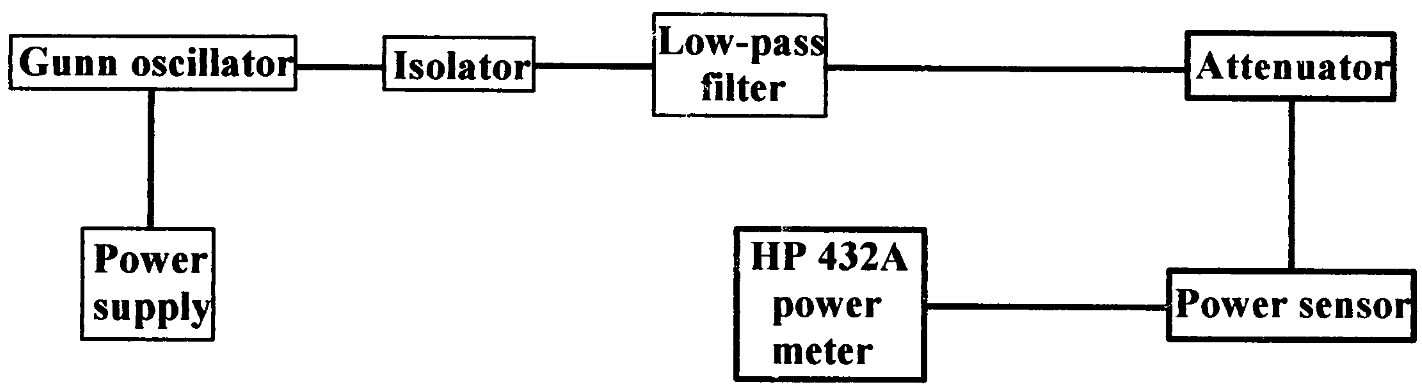


Figure IV-7: Configuration #2 - HP432A conversion coefficient measurement, HP436A noise level determination

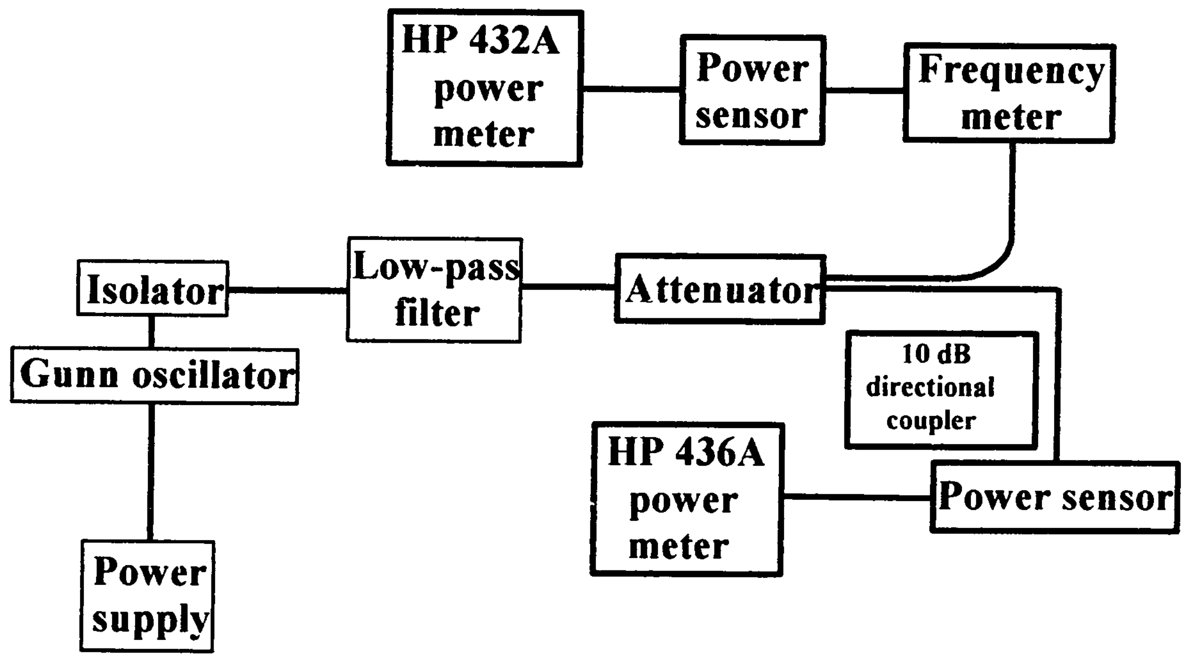


Figure IV-8: Configuration #1 - Double-stub tuner insertion loss measurement

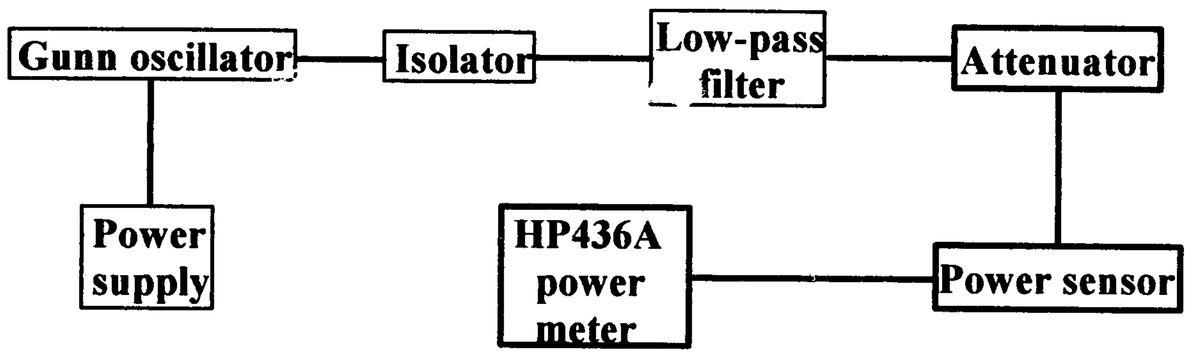


Figure IV-9: Configuration #2 - Double-stub tuner insertion loss measurement

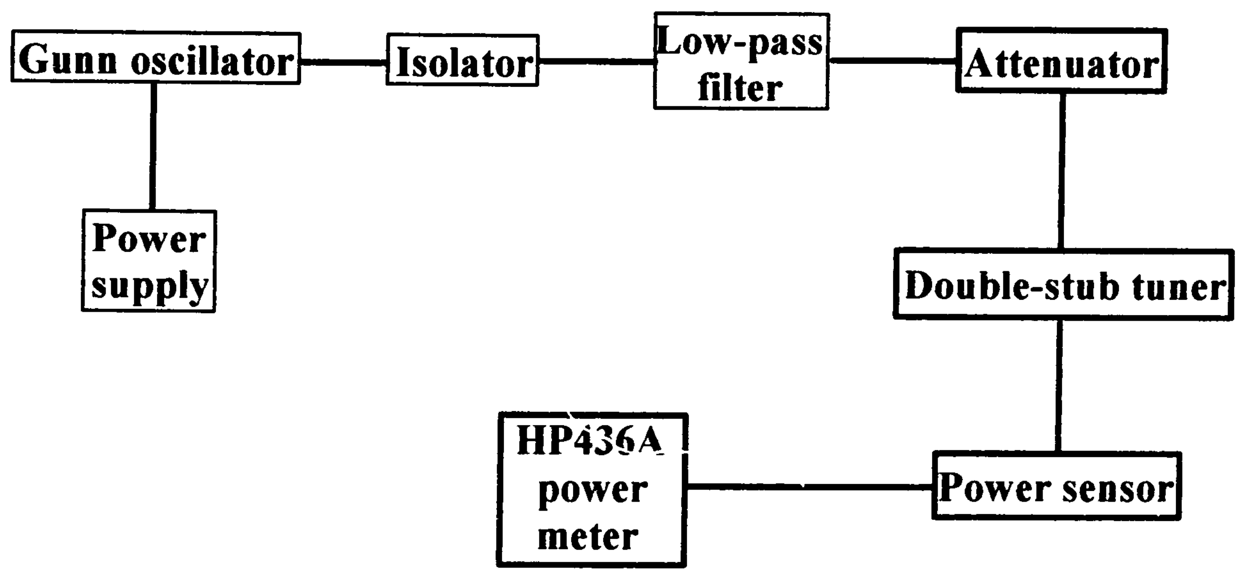


Figure IV-10: Configuration #1 - Slide-screw tuner insertion loss measurement

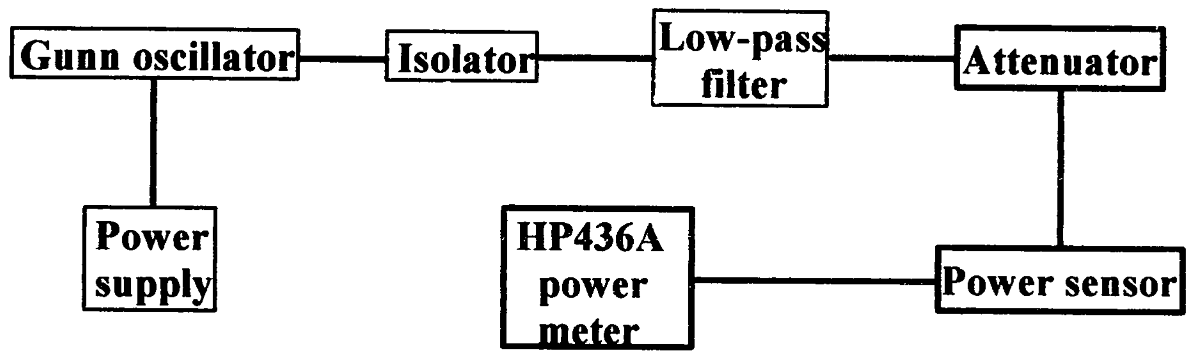


Figure IV-11: Configuration #2 - Slide-screw tuner insertion loss measurement

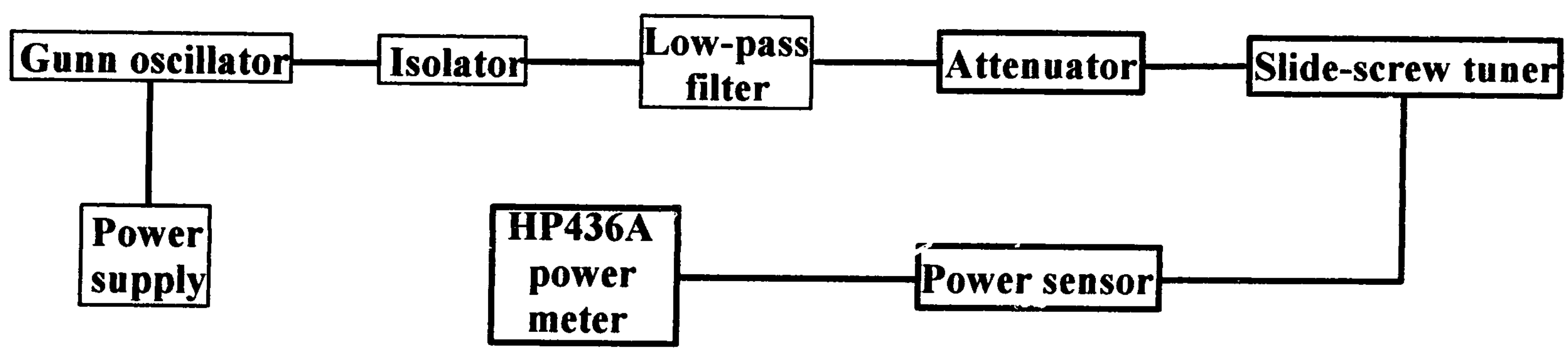


Figure IV-12: Configuration - Evaluation of the available power level at the transmitting antenna

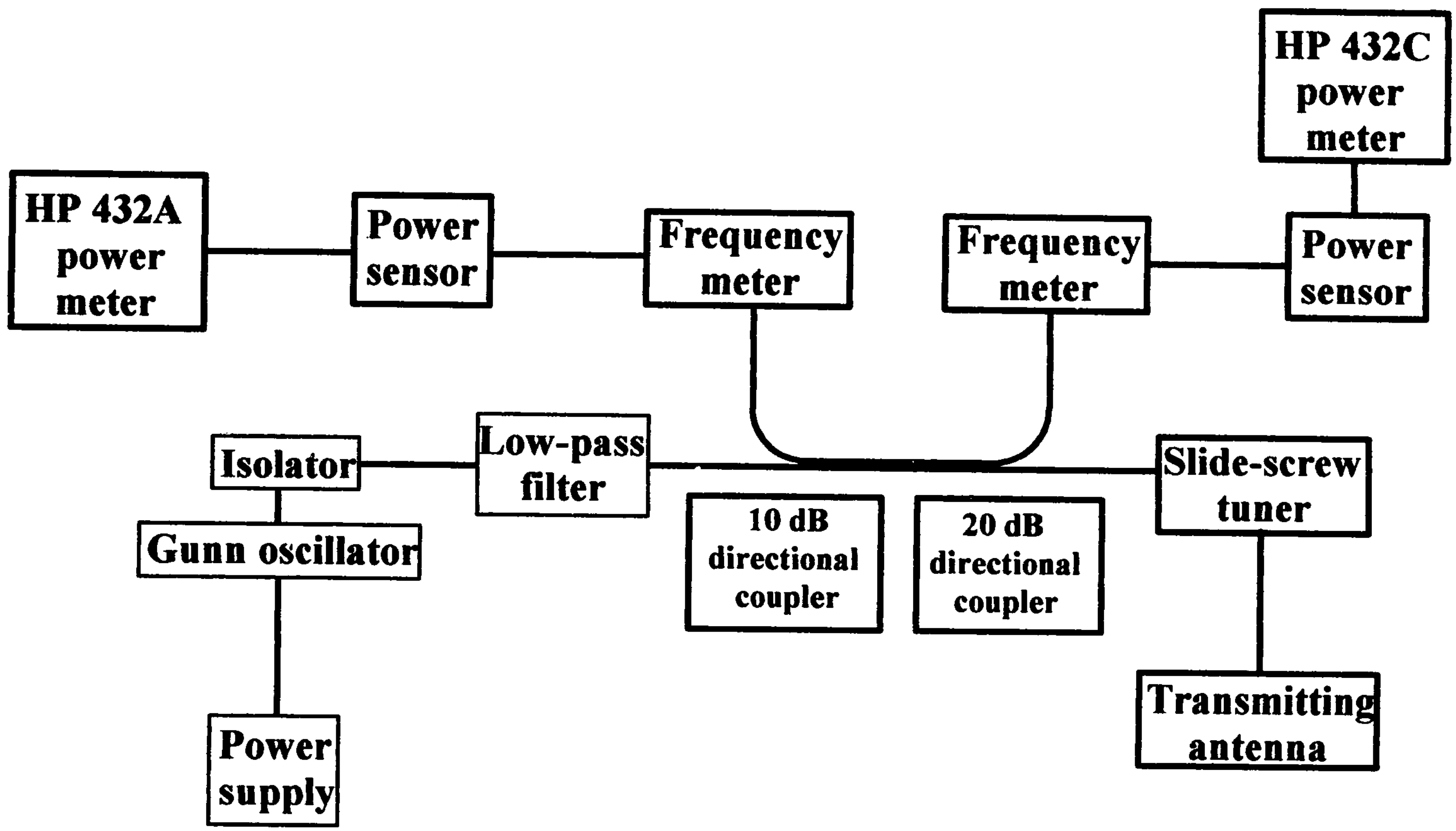


Figure IV-13: Microwave power attenuation characteristics of the existing core holder - Theoretical and experimental data

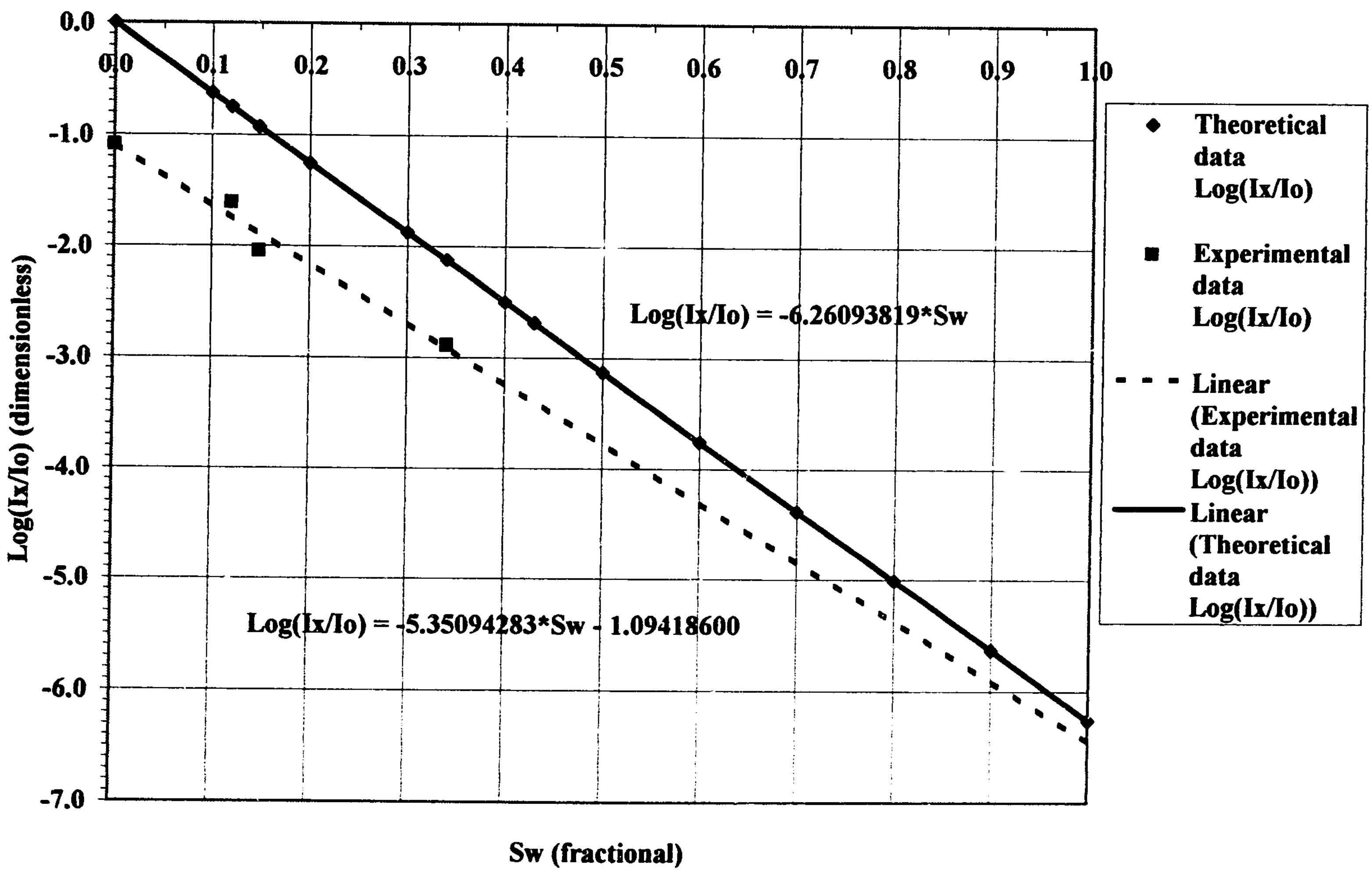


Figure IV-14: Theoretical microwave power attenuation characteristics without safety margin of the improved core holder

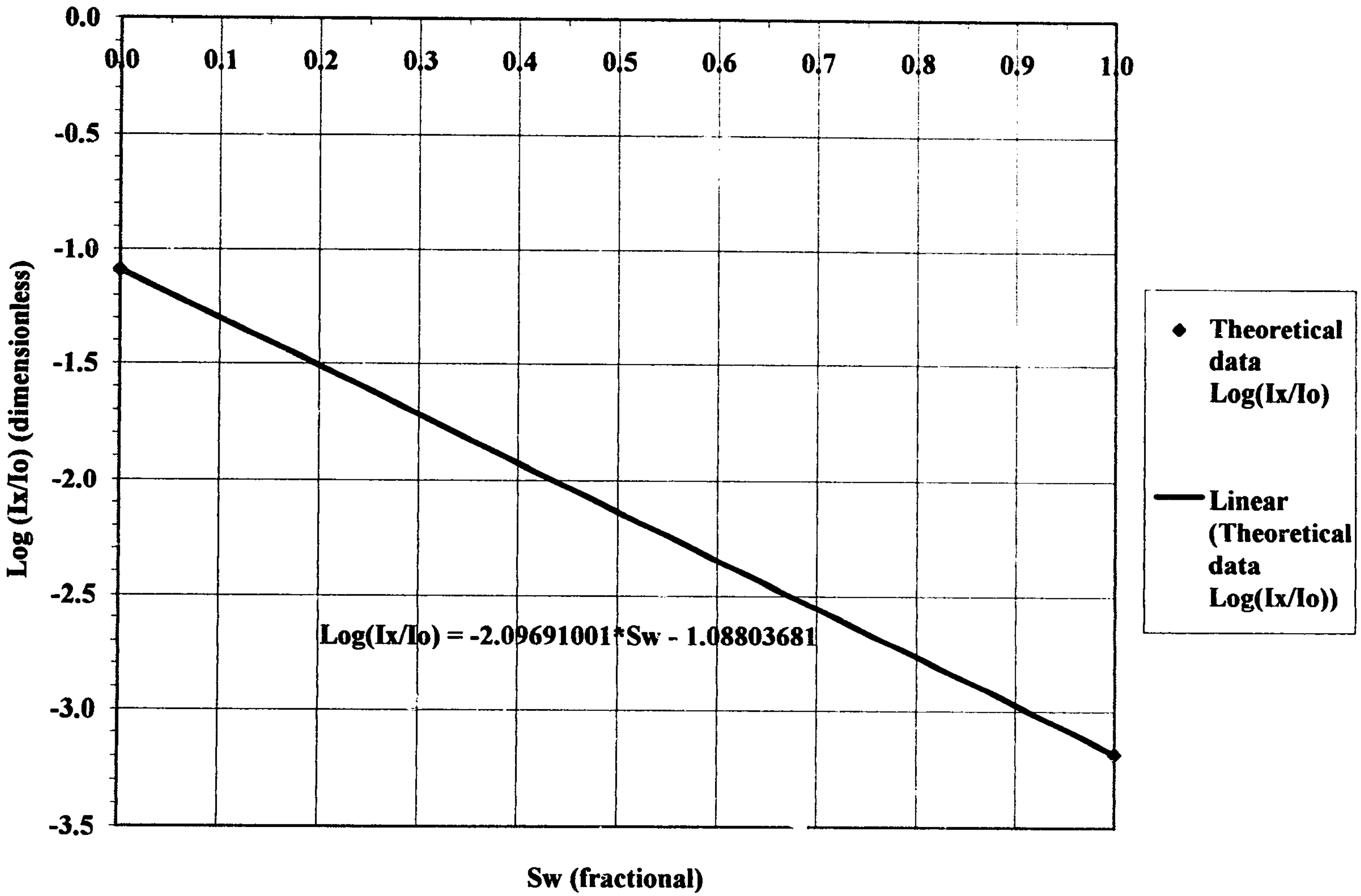


Figure IV-15: Theoretical microwave power attenuation characteristics with safety margin of the improved core holder

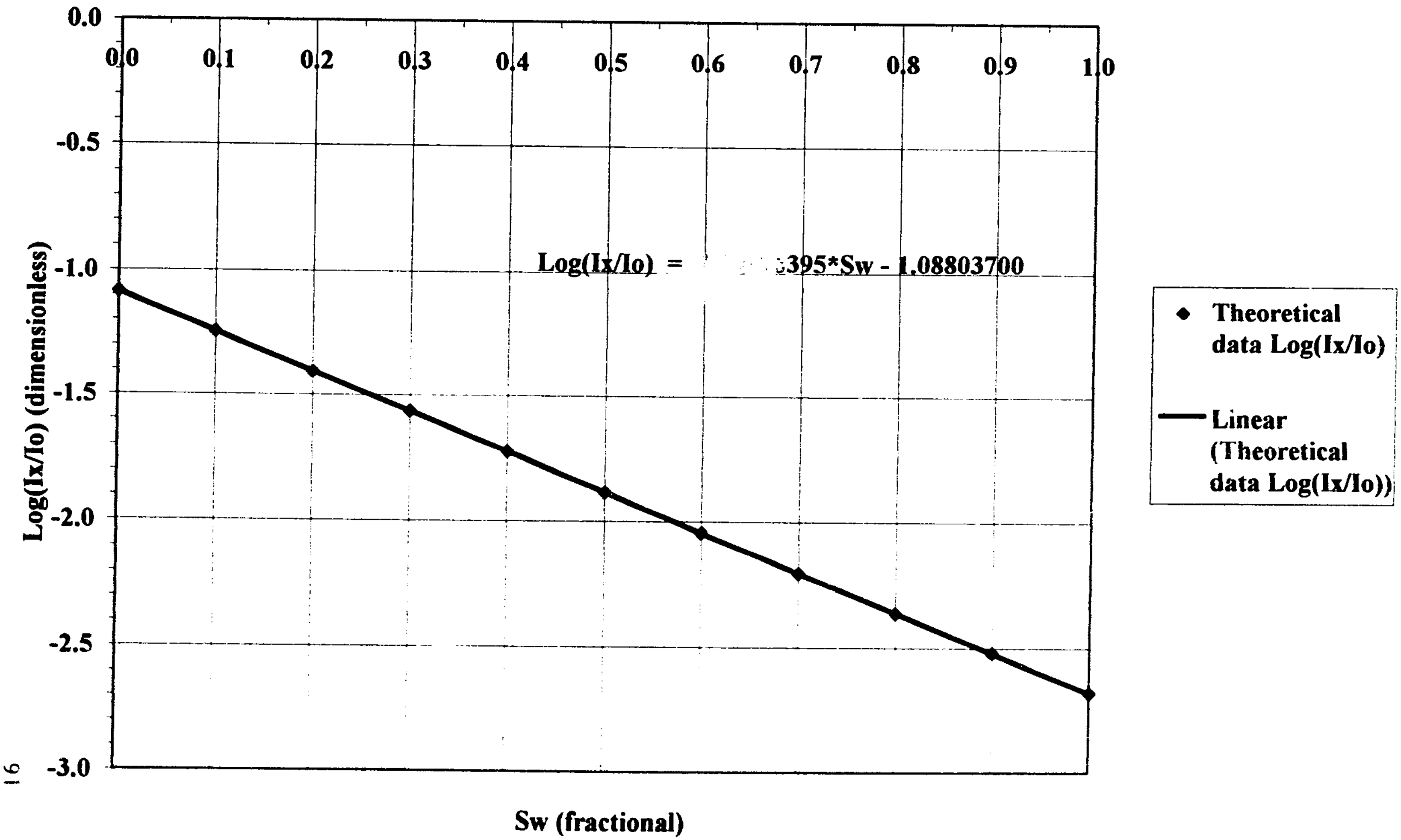


Figure IV-16: Upper and lower error bounds in Log(Ix/Io) measured - Improved core holder

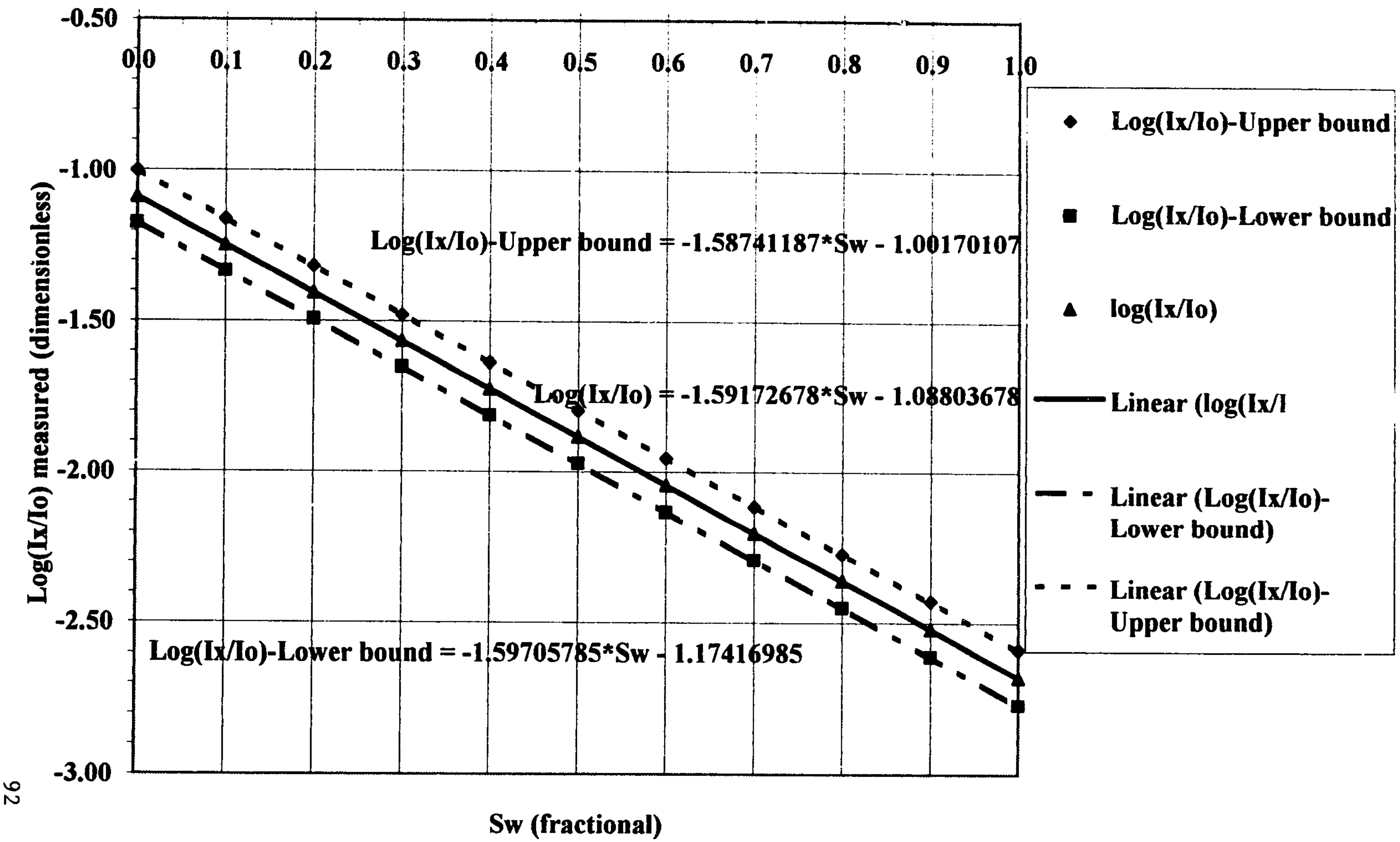


Figure IV-17: Relative uncertainty in Log(Ix/Io) measured - Improved core holder

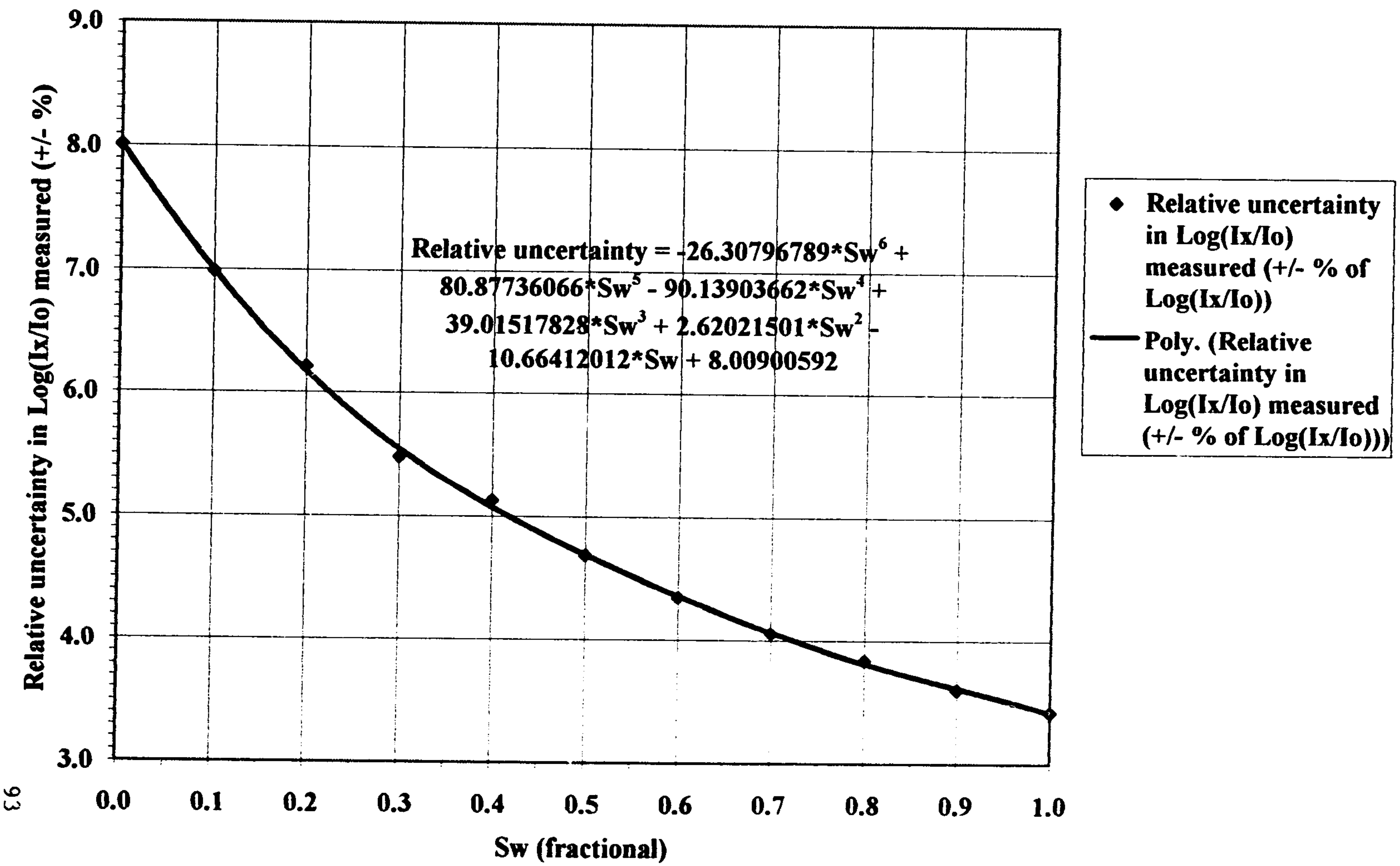


Figure IV-18: Upper and lower error bounds in individual pressure interpolated - 2-point calibration procedures

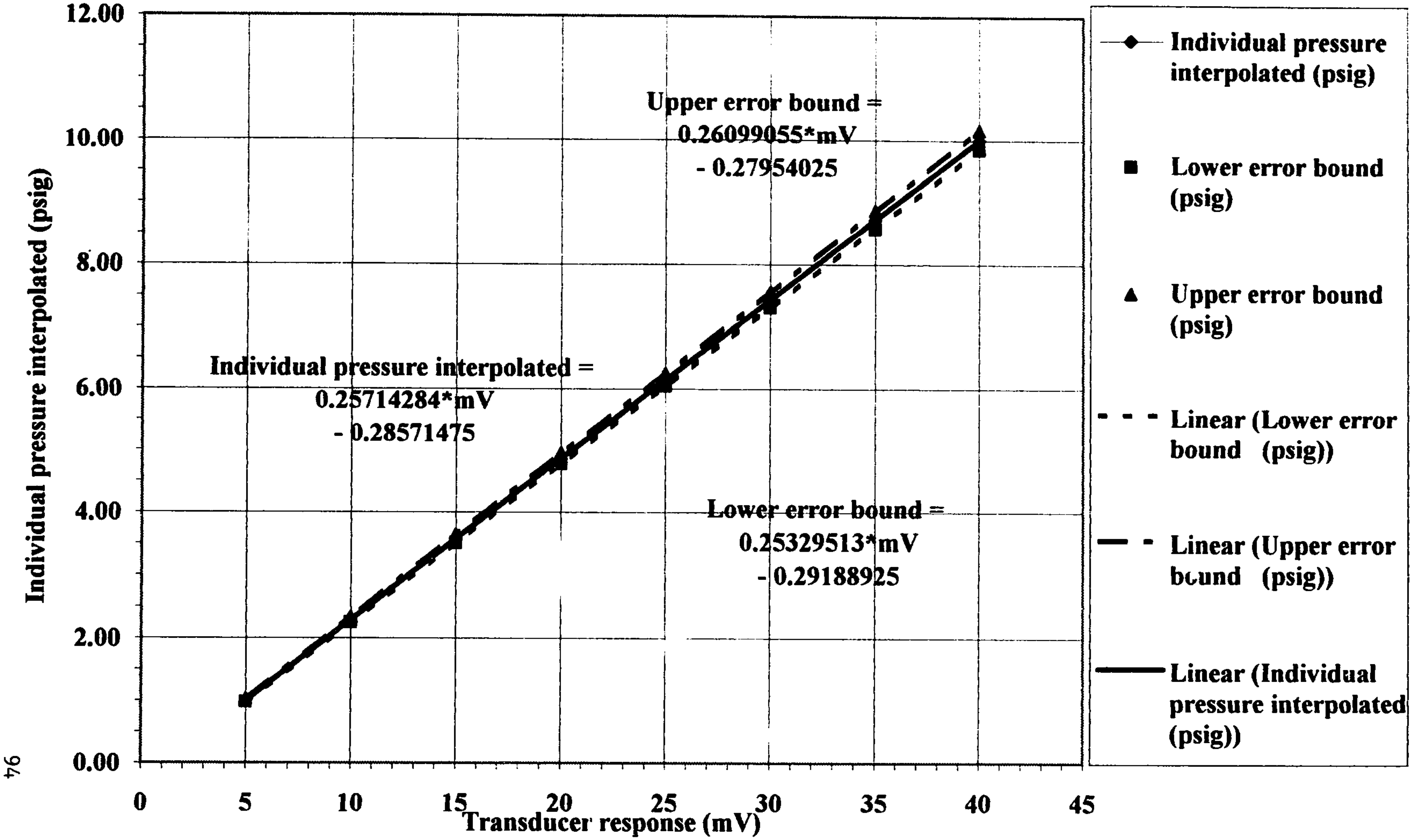


Figure IV-19: Relative uncertainty in individual pressure interpolated - 2-point calibration procedures

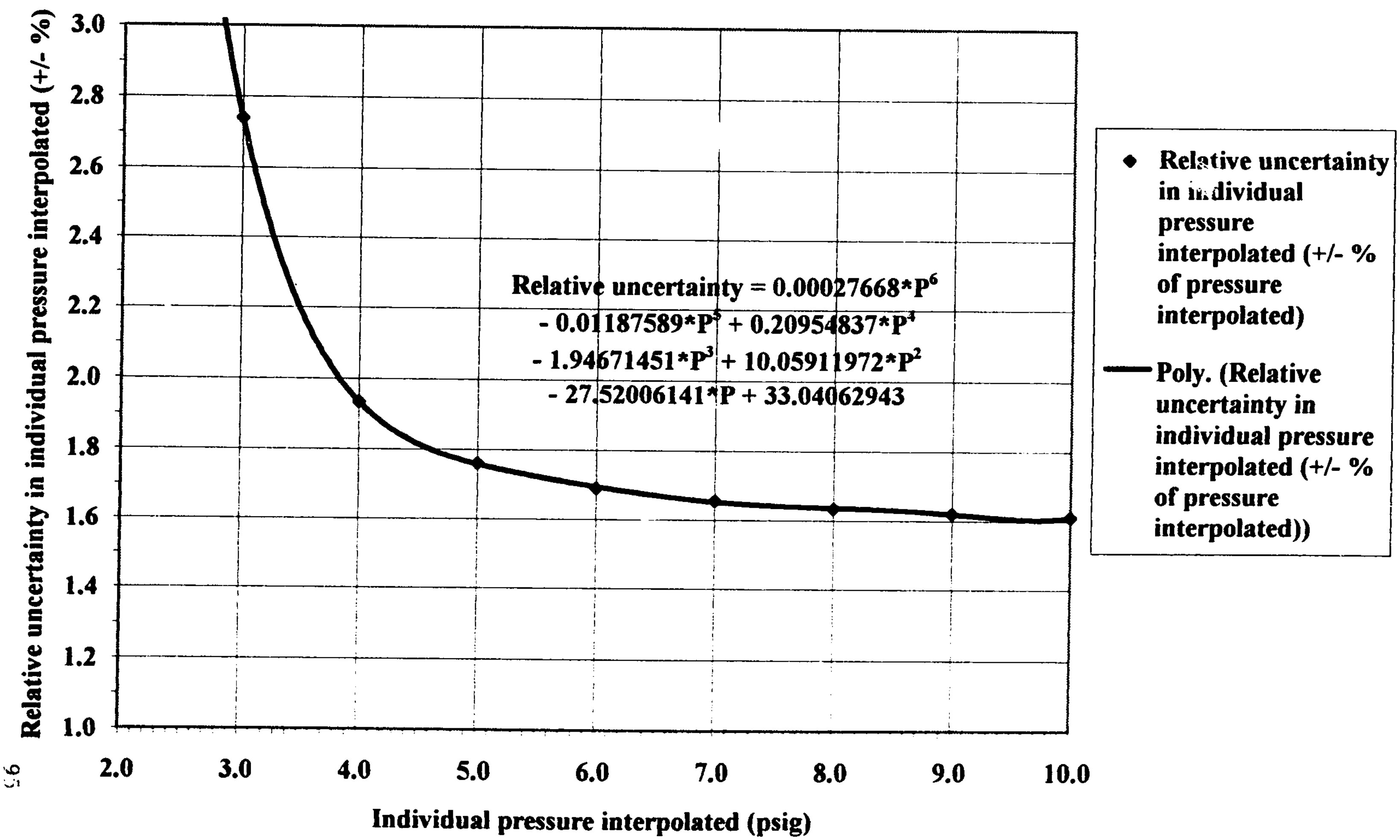


Figure IV-20: Upper and lower error bounds in pressure profile measured - 2-point calibration procedures

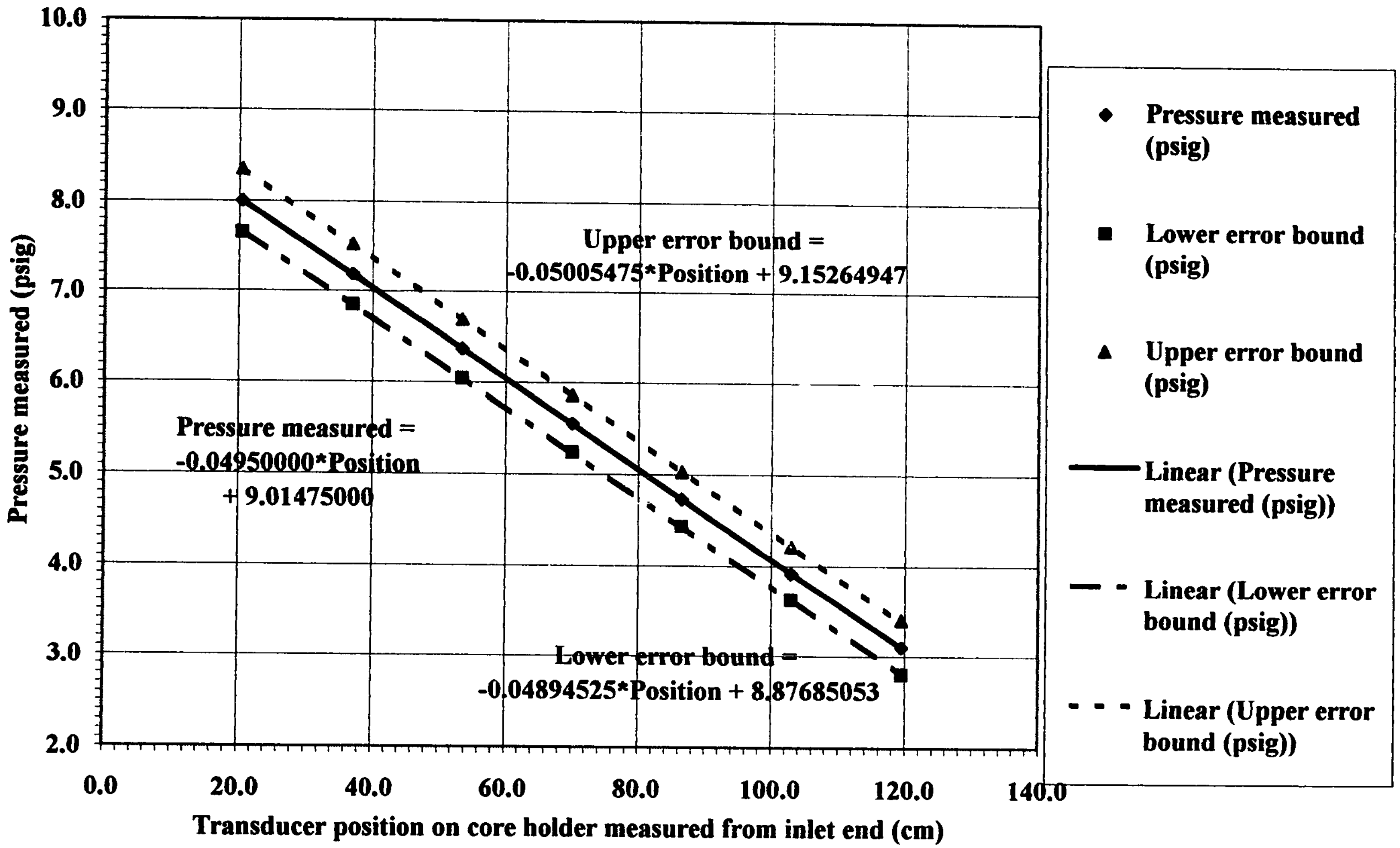


Figure IV-21: Pressure-transducer calibration apparatus

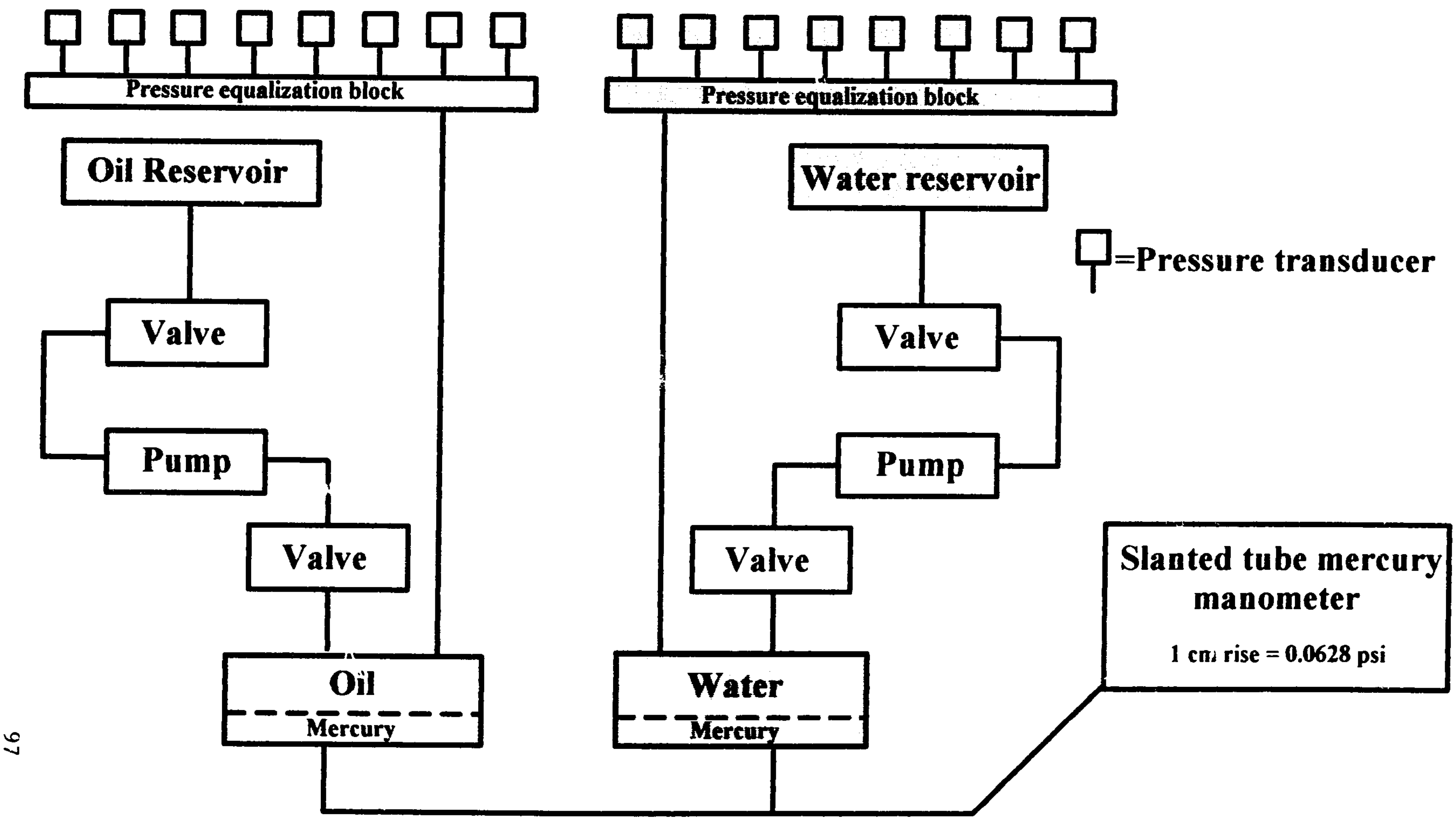


Figure IV-22: Upper and lower error bounds in individual pressure interpolated - 10-point calibration procedures

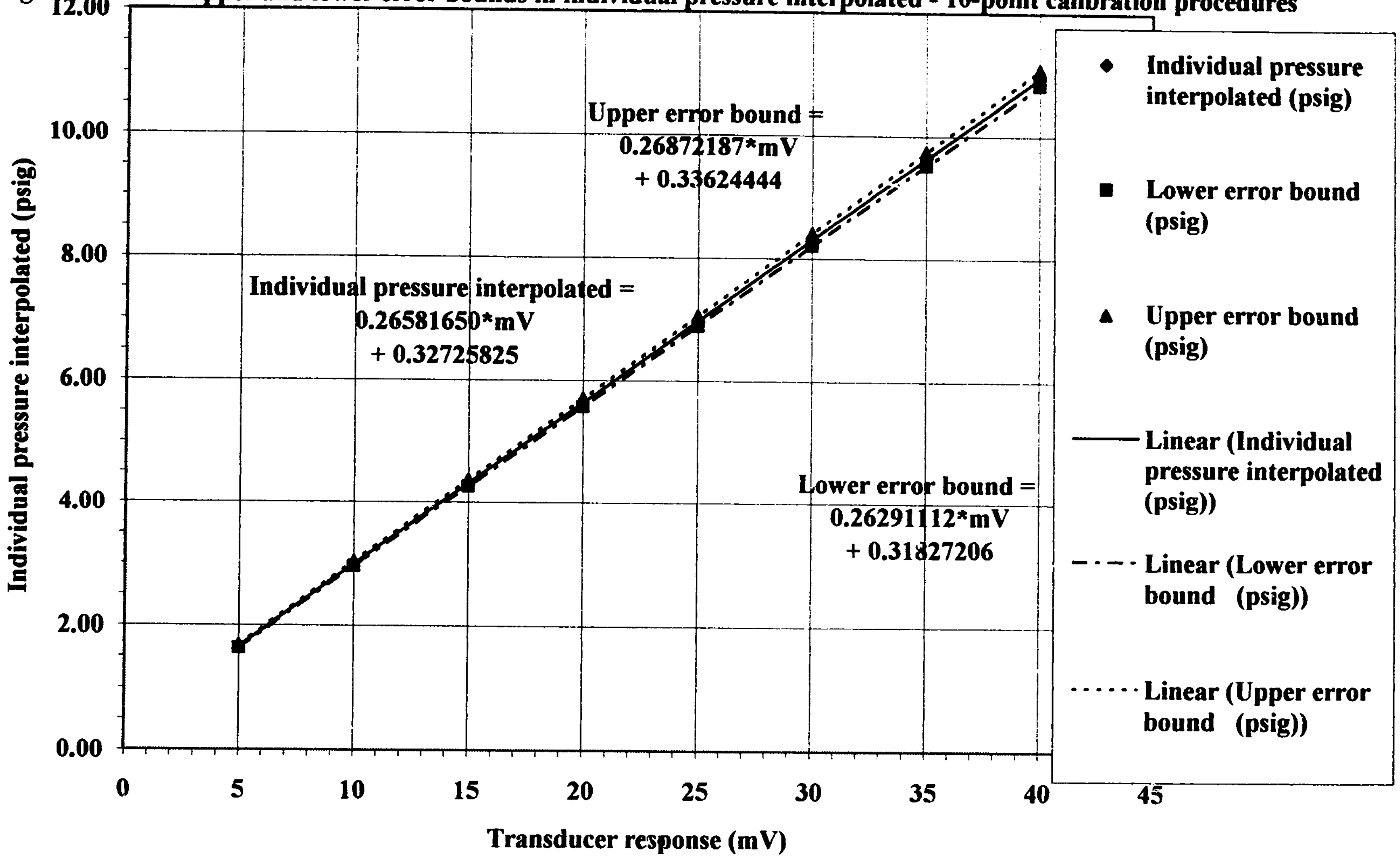


Figure IV-23: Relative uncertainty in individual pressure interpolated - 10-point calibration procedures

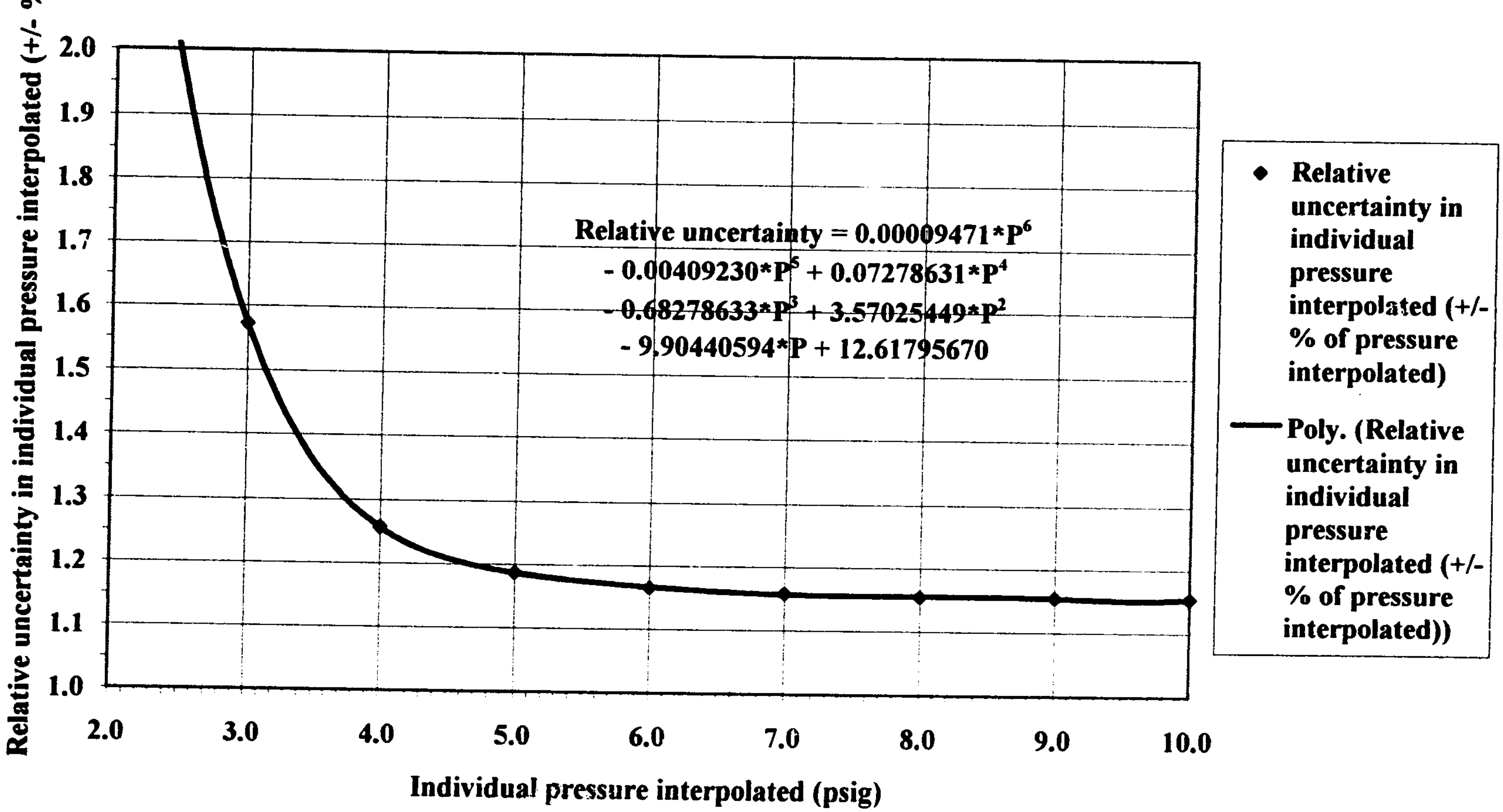


Figure IV-24: Upper and lower error bounds in pressure profile measured - 10-point calibration procedures

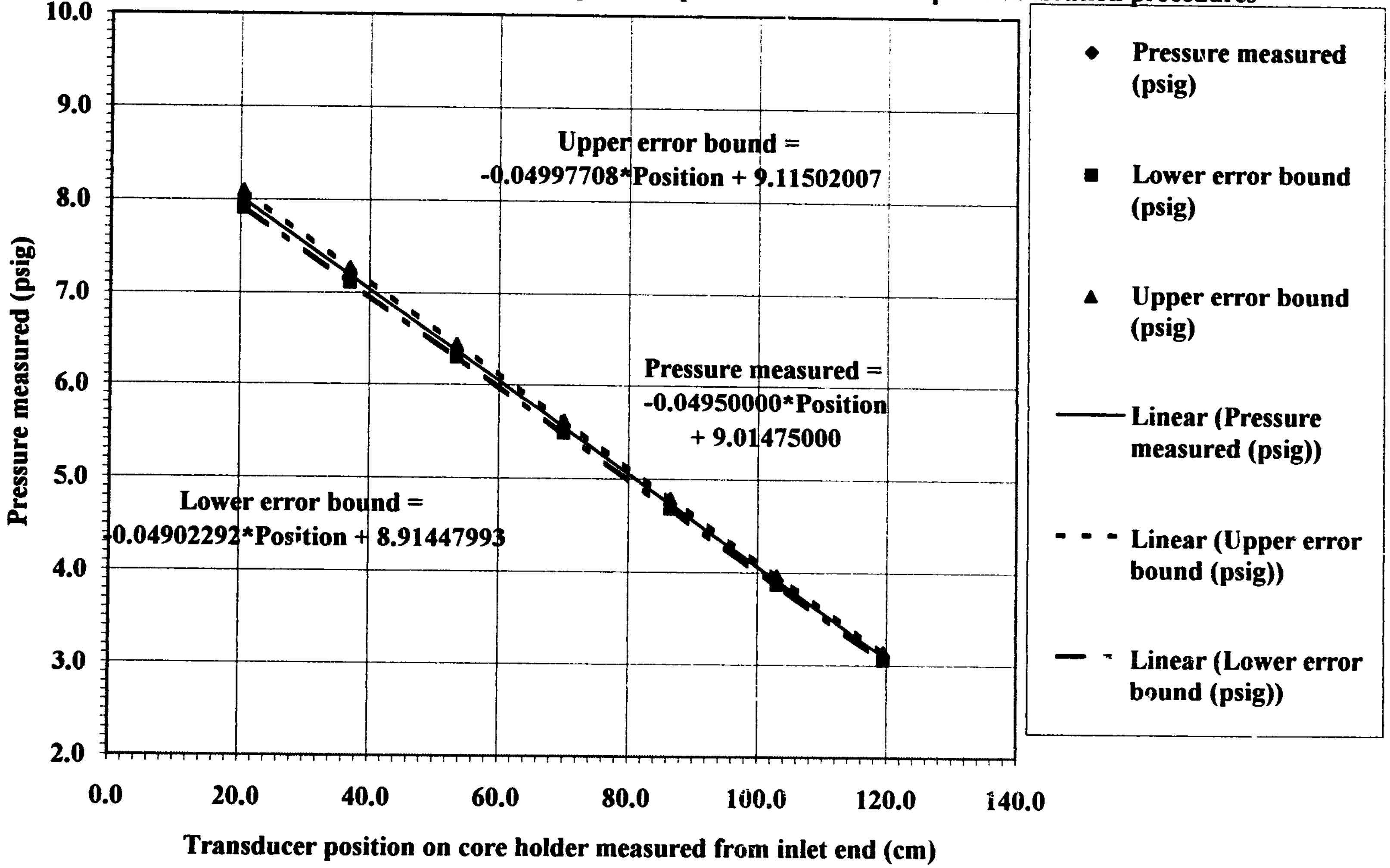


Figure IV-25: Hydrostatic-difference elimination apparatus

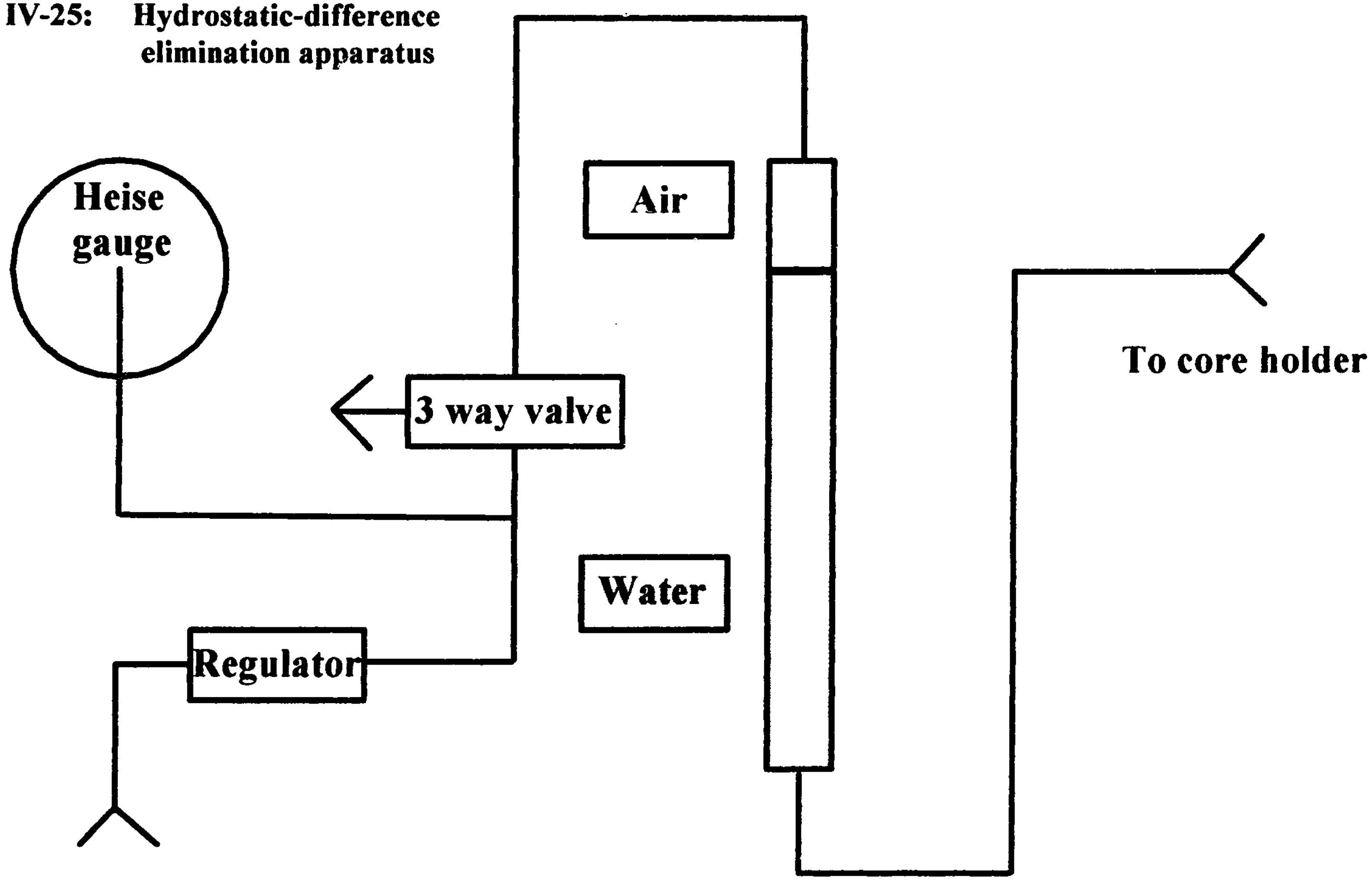


Figure IV-26: Dead-volume measurement apparatus

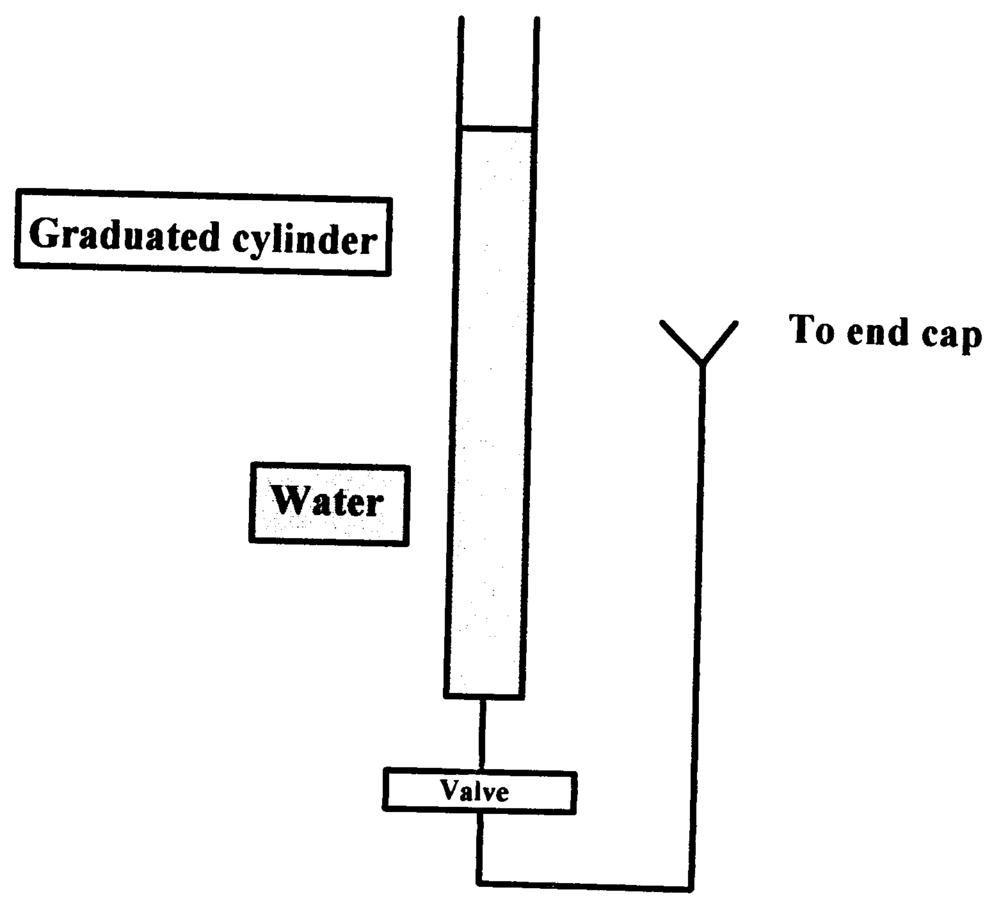


Figure IV-27: Traveling-track apparatus

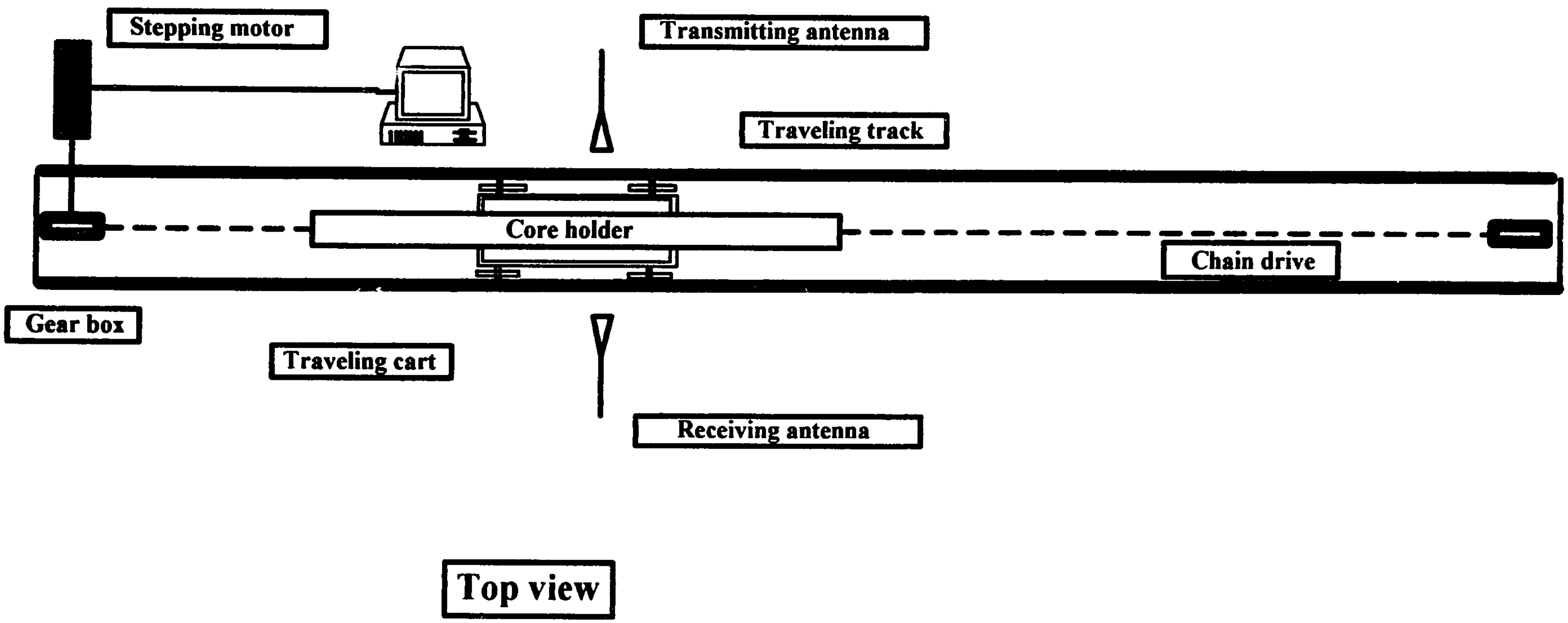


Figure IV-28: Vacuum-drawing apparatus

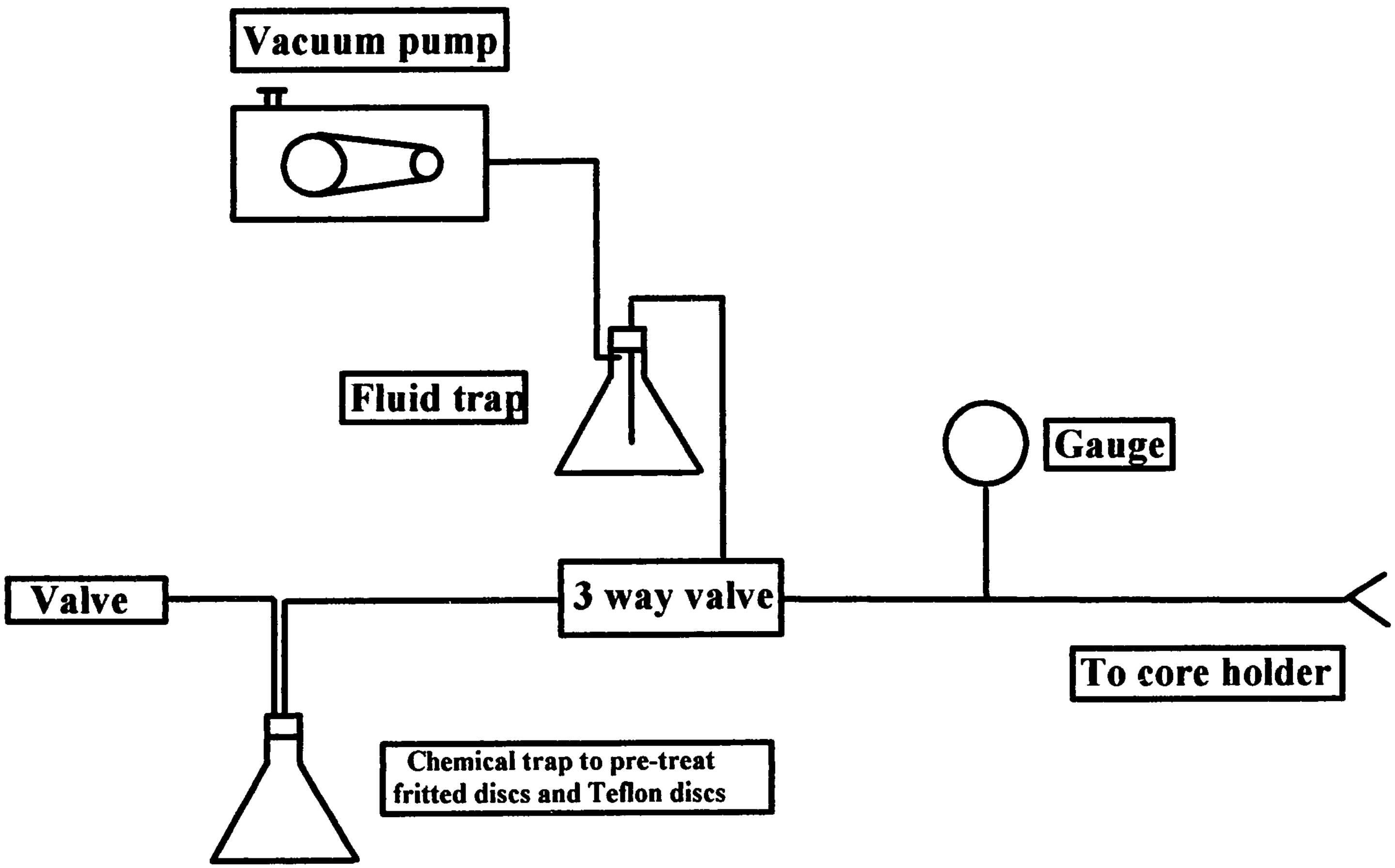


Figure IV-29: Fluid-injection apparatus

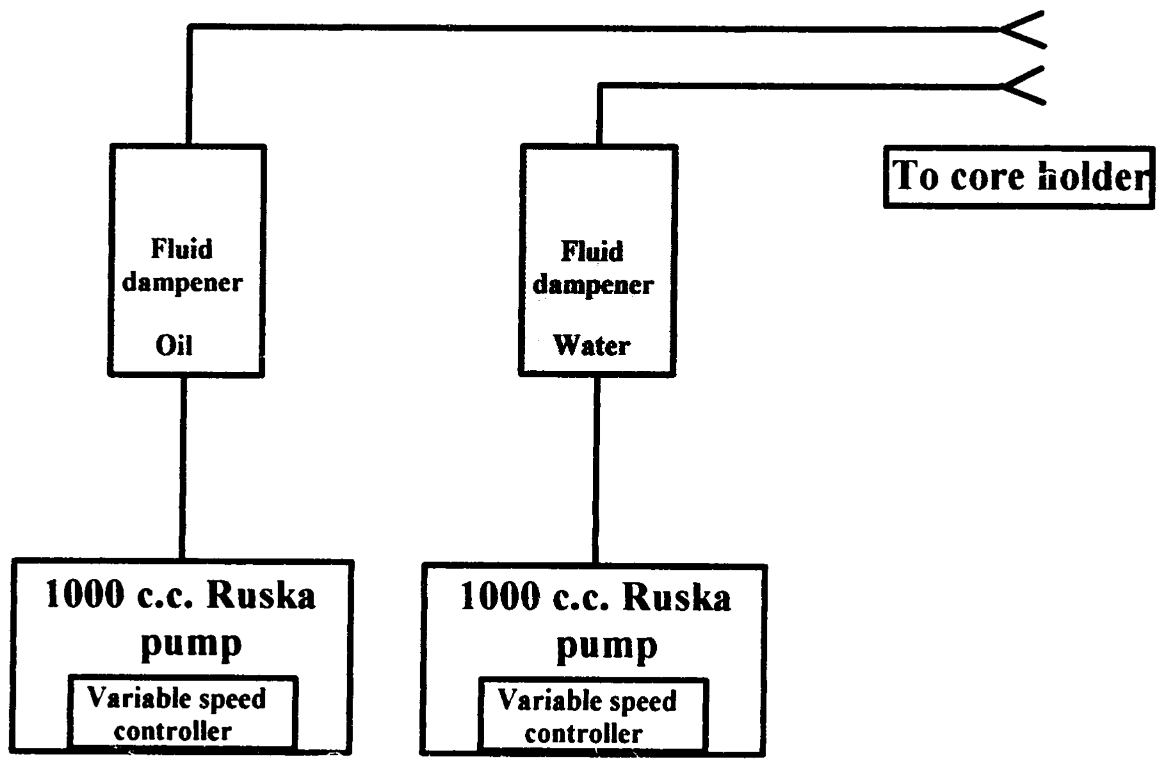


Figure IV-30: Core-holder setup

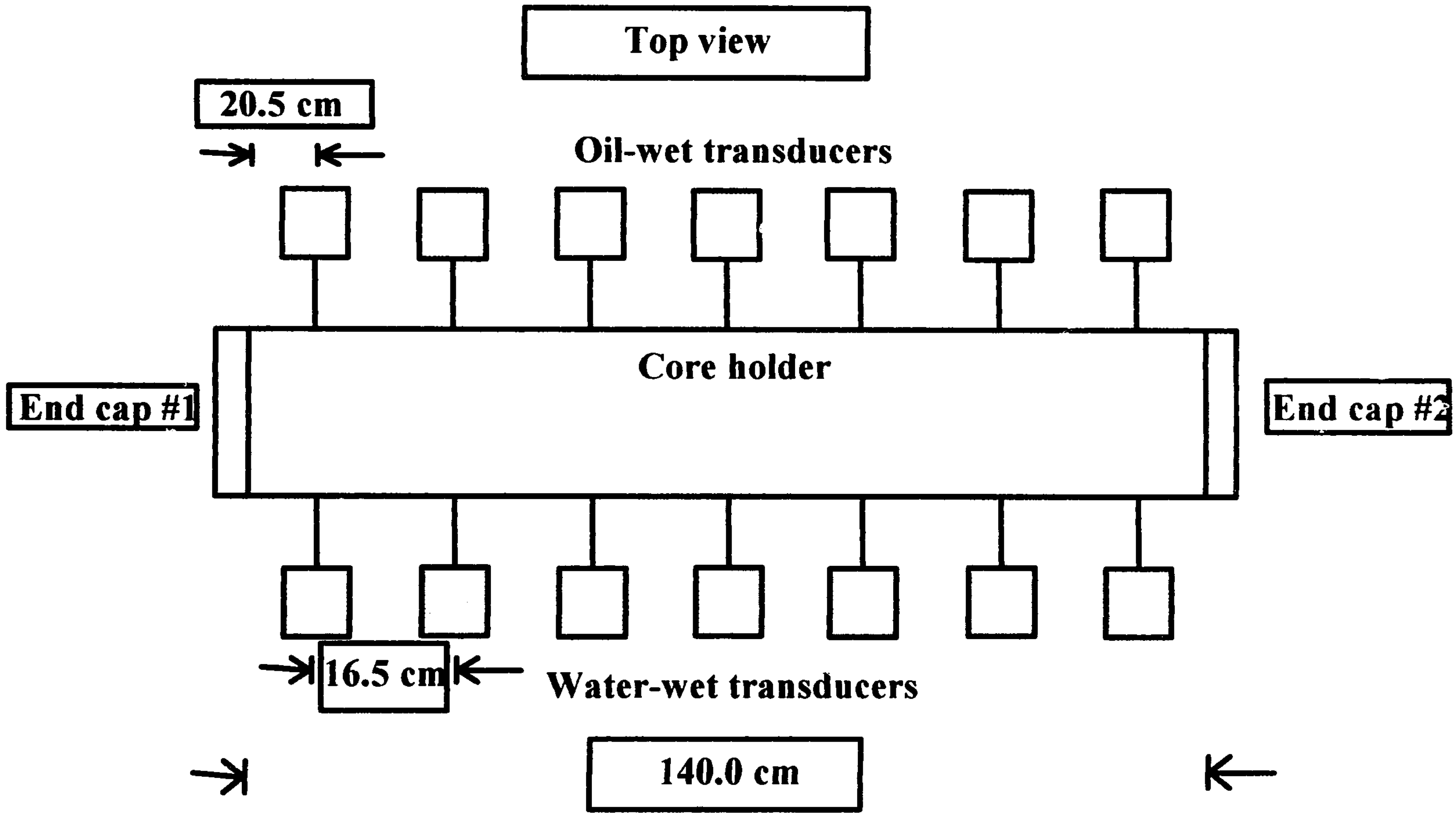


Figure IV-31: Injection-cap apparatus

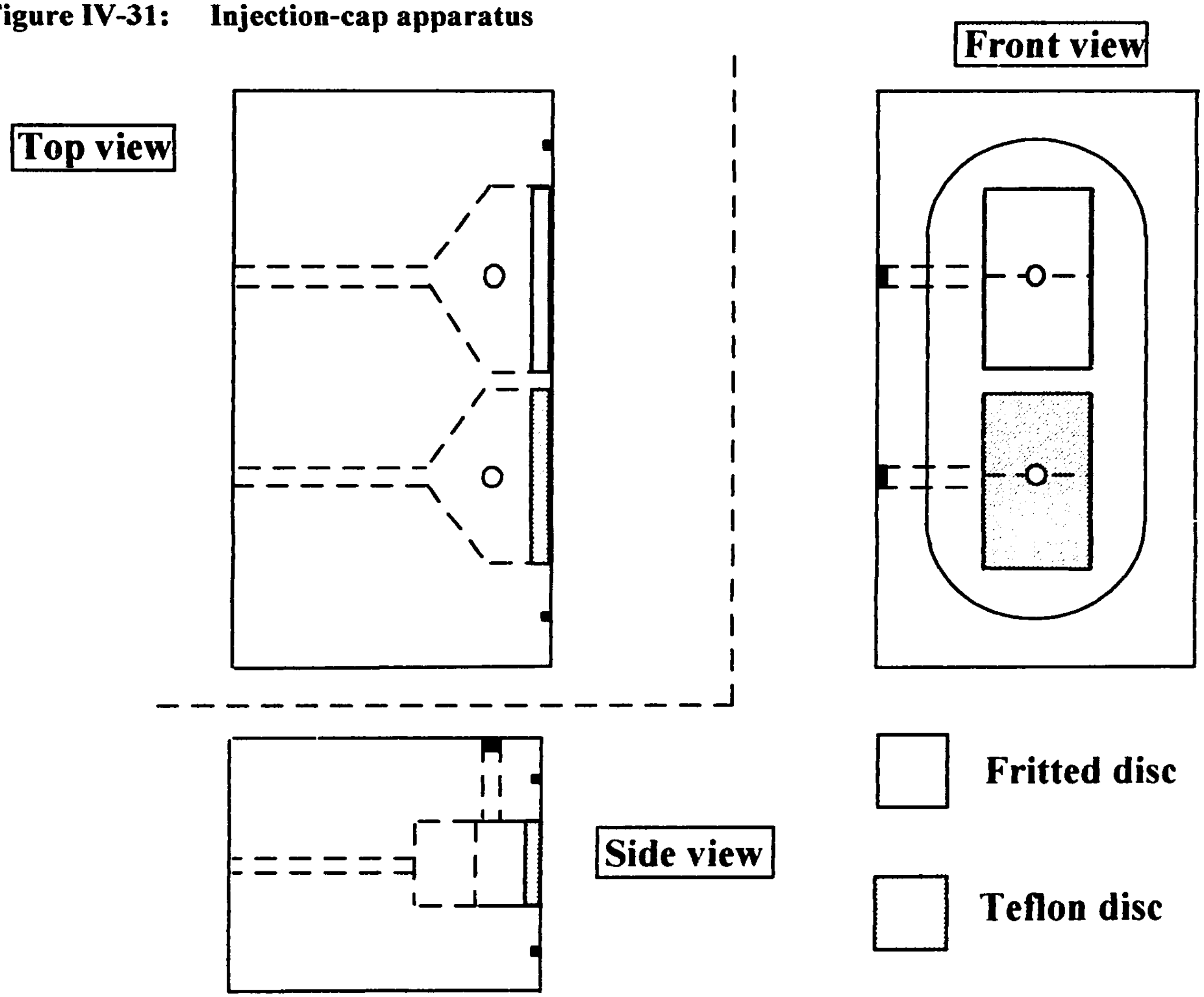


Figure IV-32: Pressure-transducer apparatus

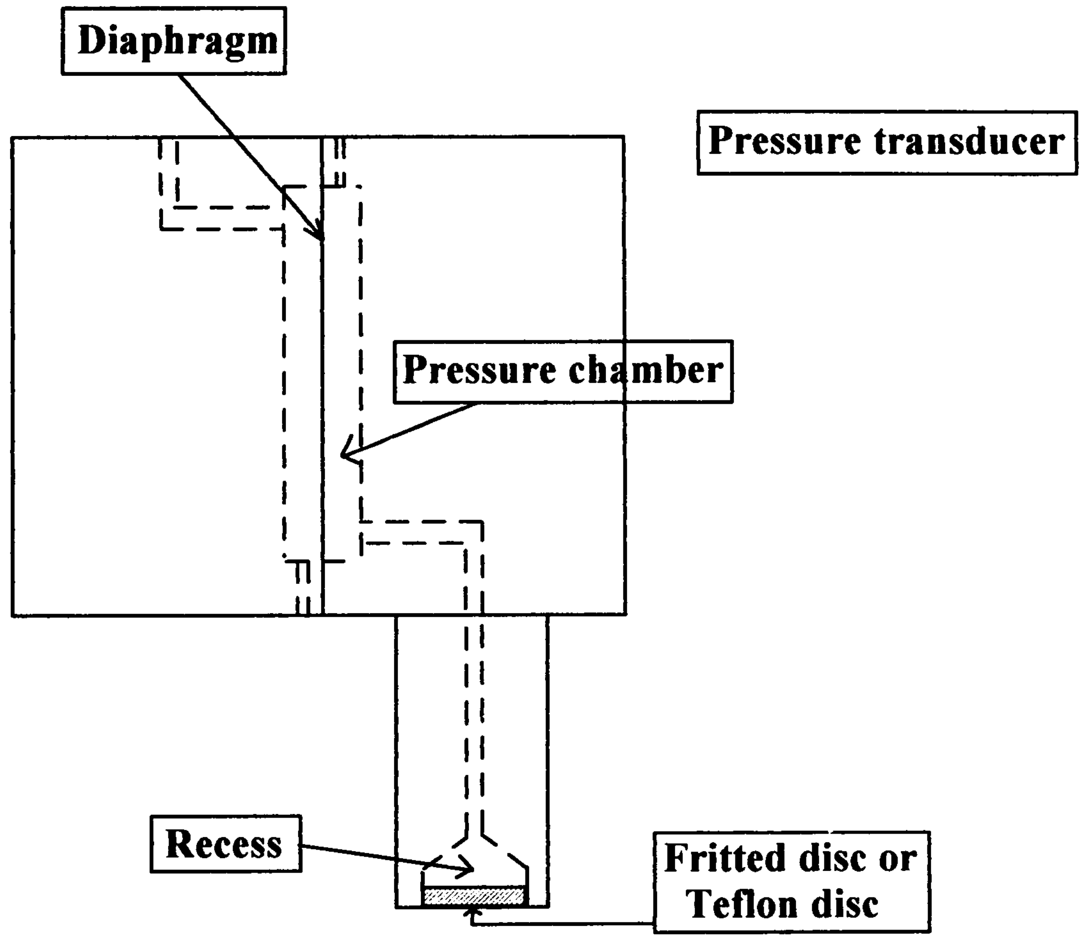


Figure IV-33: Previous weight-measurement apparatus

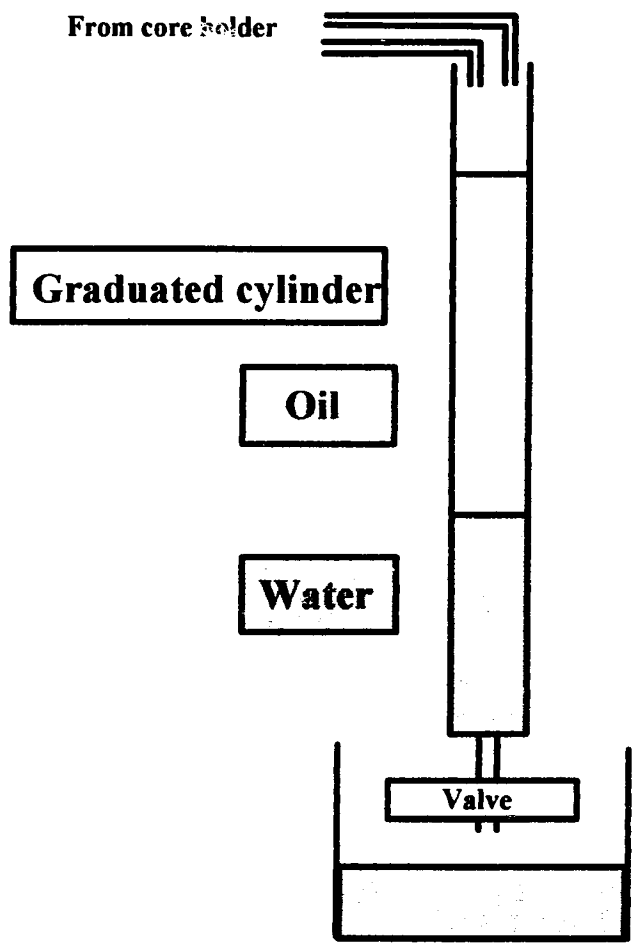


Figure IV-34: Uncertainty in saturation calculated - Conventional material balance method

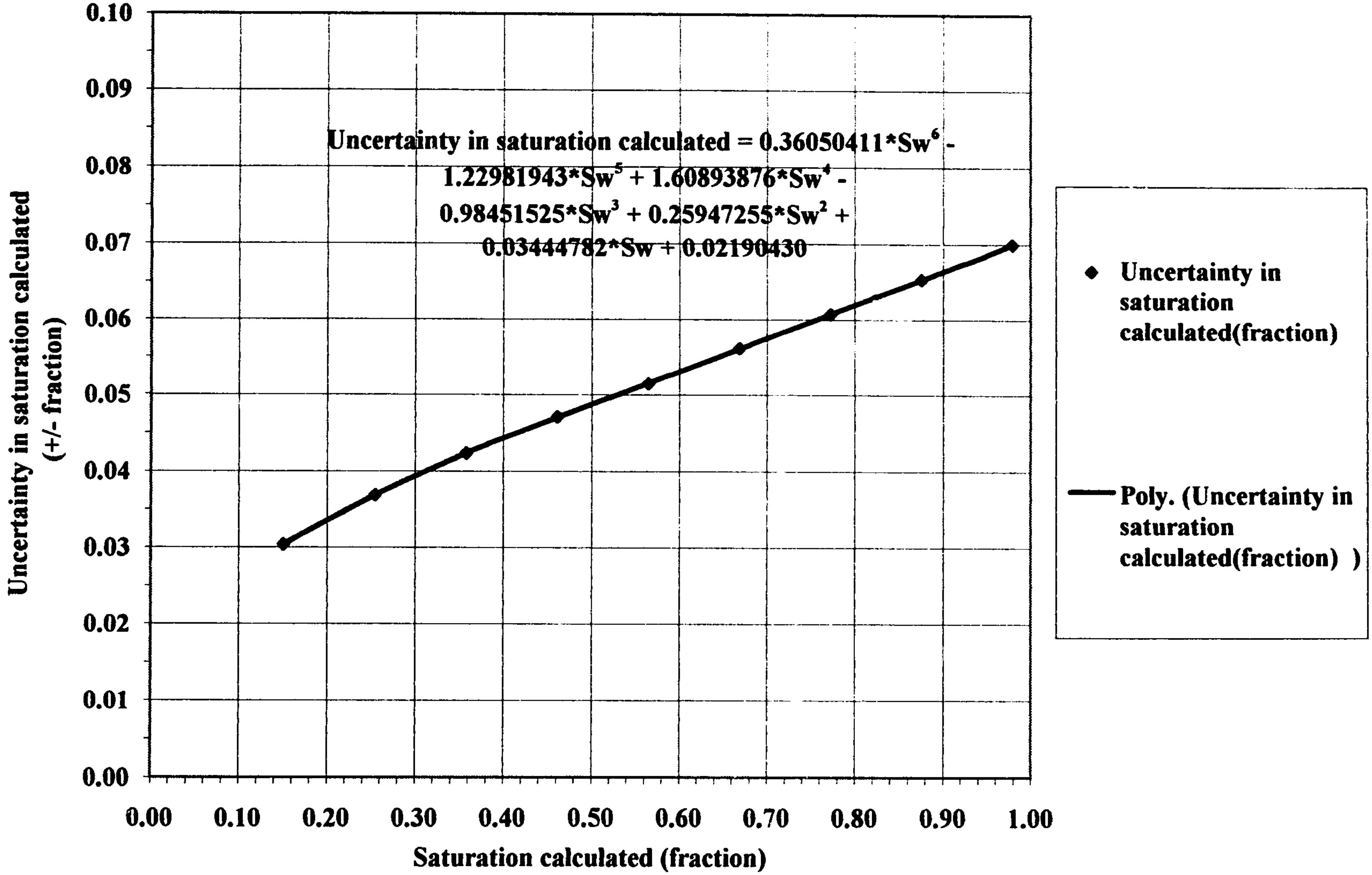


Figure IV-35: Relative uncertainty in saturation calculated - Conventional material balance method

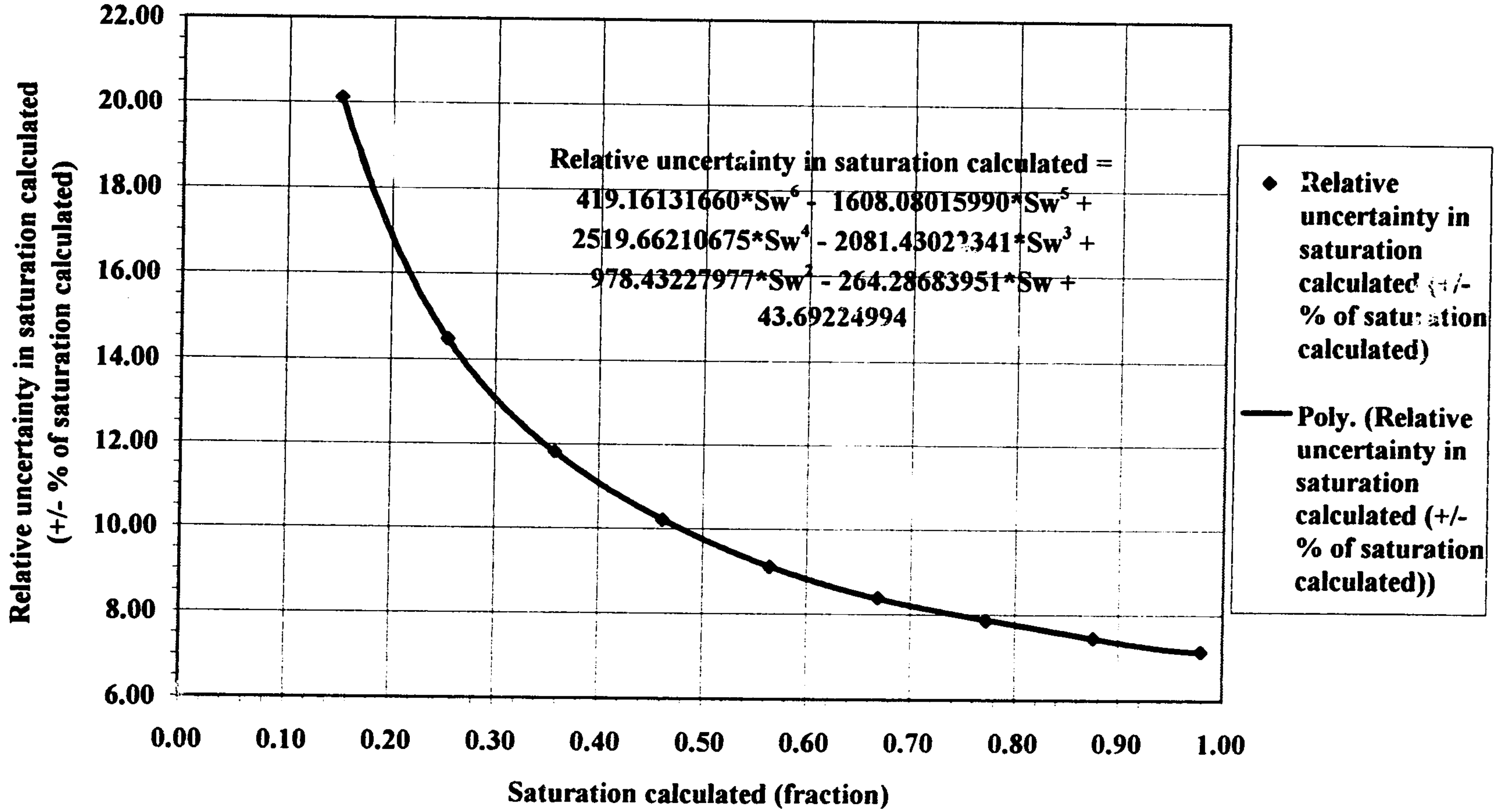


Figure IV-36: Upper and lower error bounds in saturation calculated - Conventional material

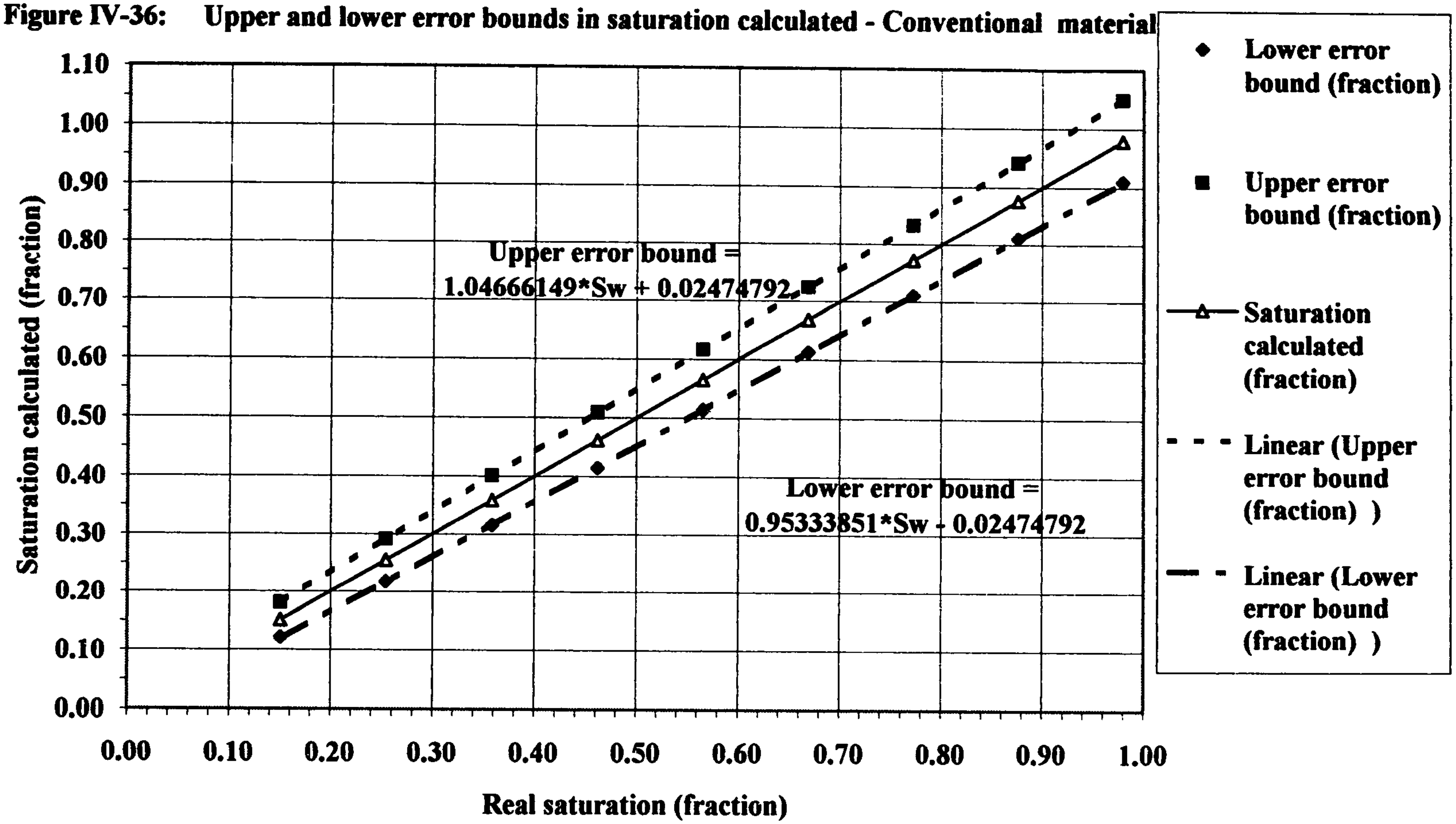


Figure IV-37: Improved weight-measurement apparatus

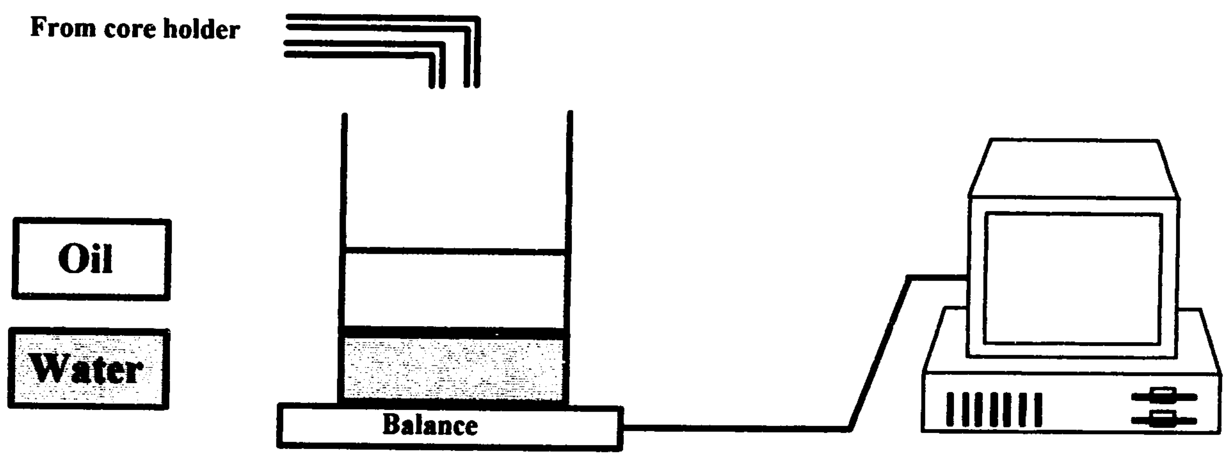


Figure IV-38: Uncertainty in saturation calculated - Improved material balance method

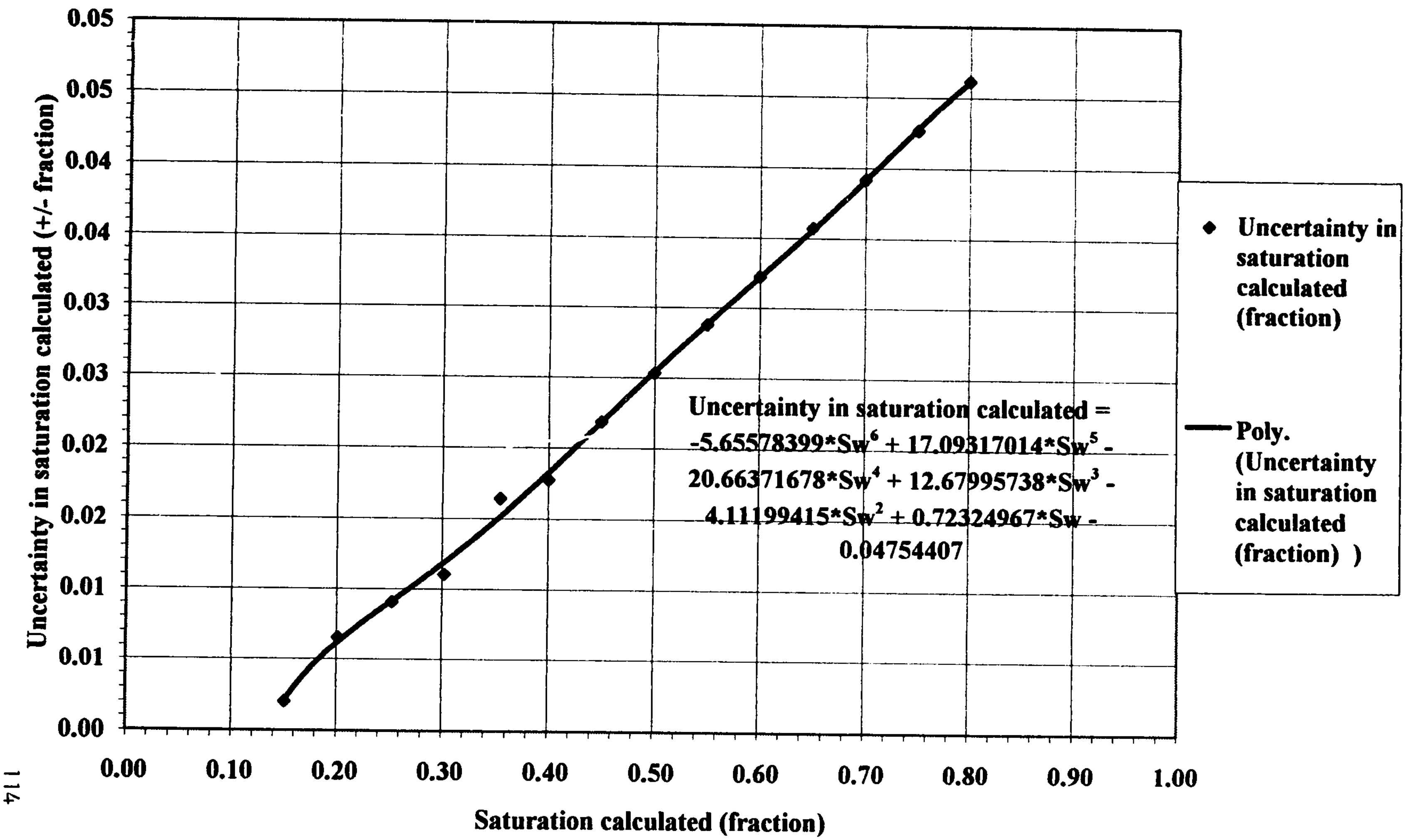


Figure IV-39: Relative uncertainty in saturation calculated - Improved material balance method

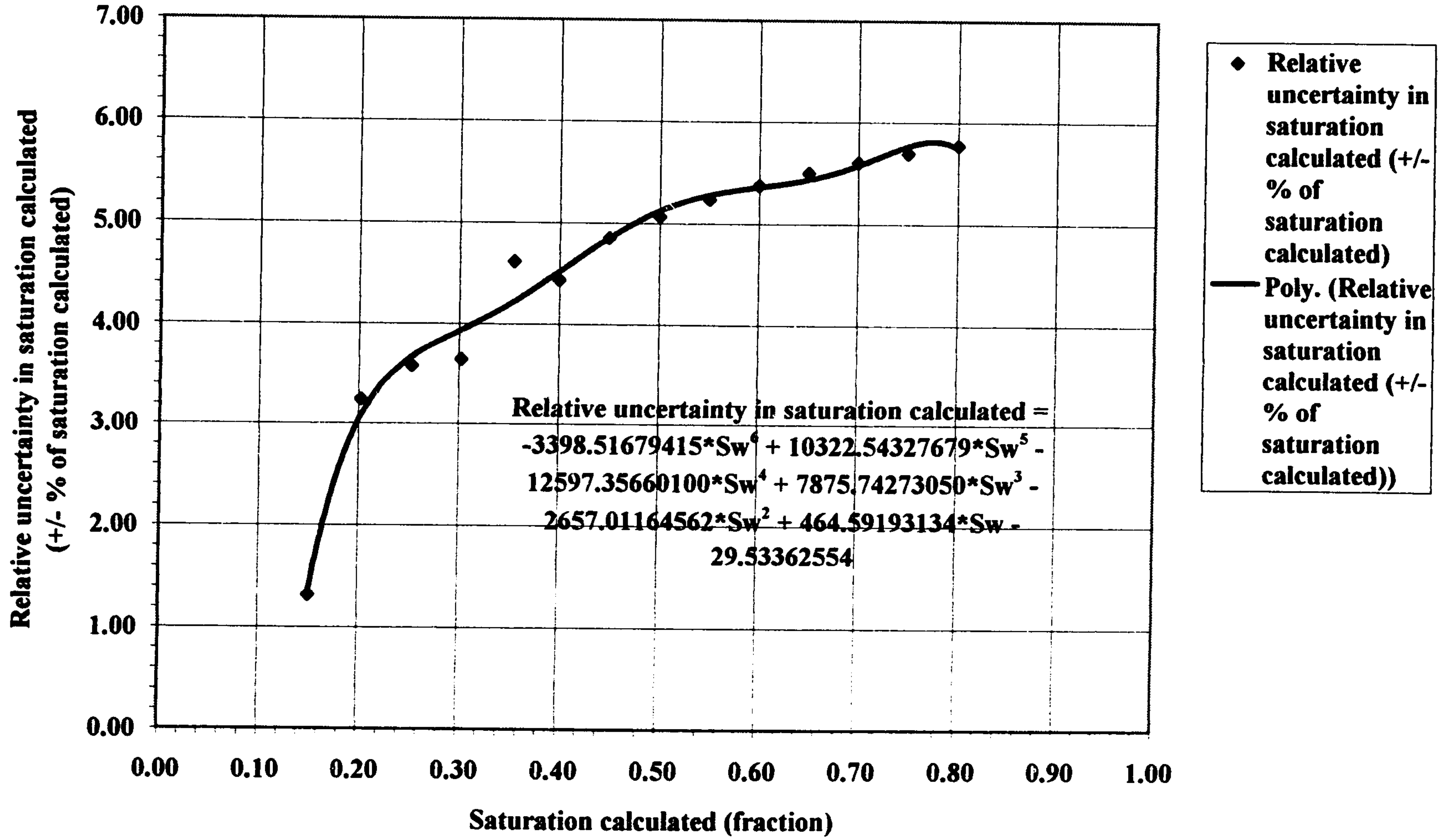


Figure IV-40: Upper and lower error bounds in saturation calculated - Improved material balance method

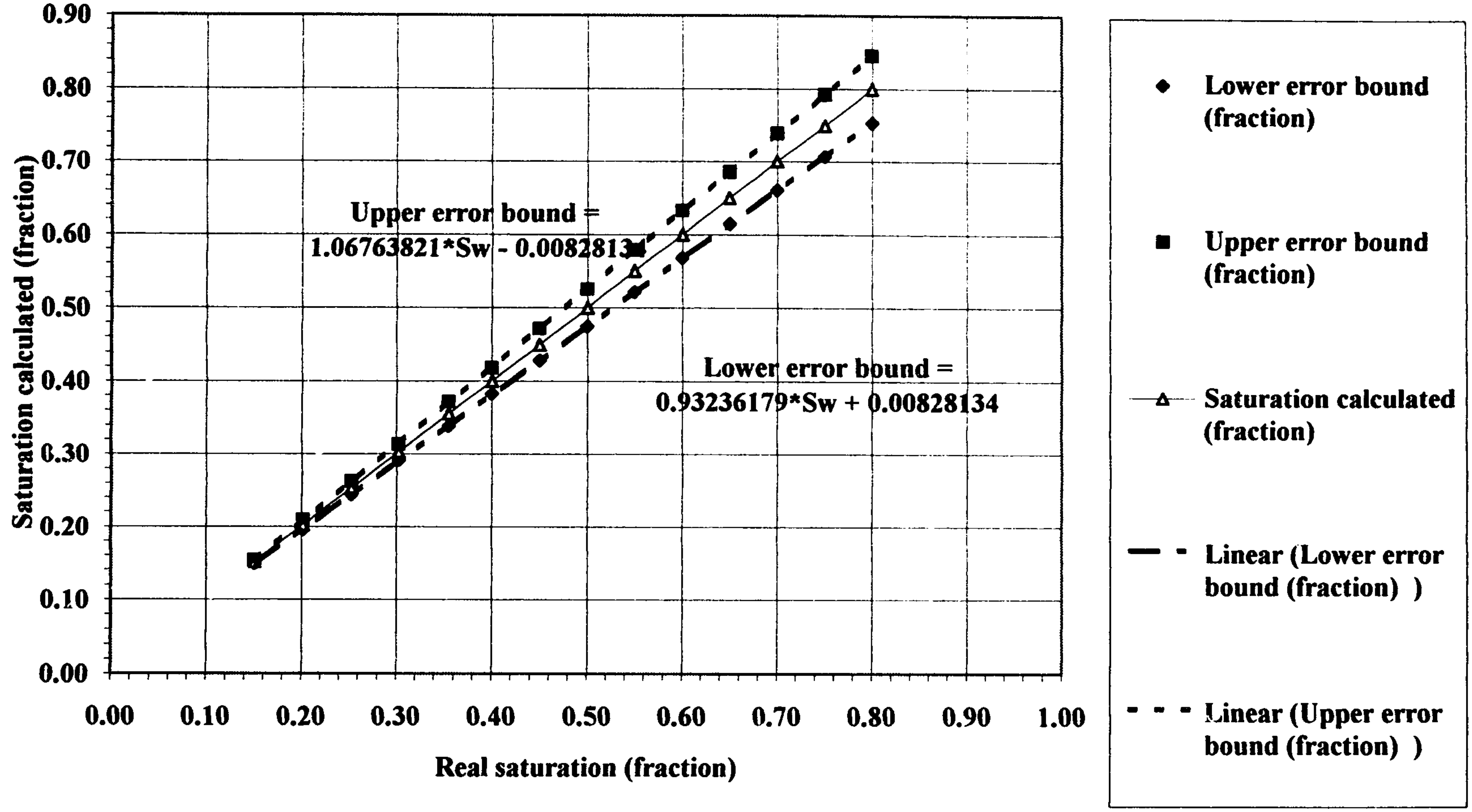


Figure IV-41: Hardware configuration for data collection

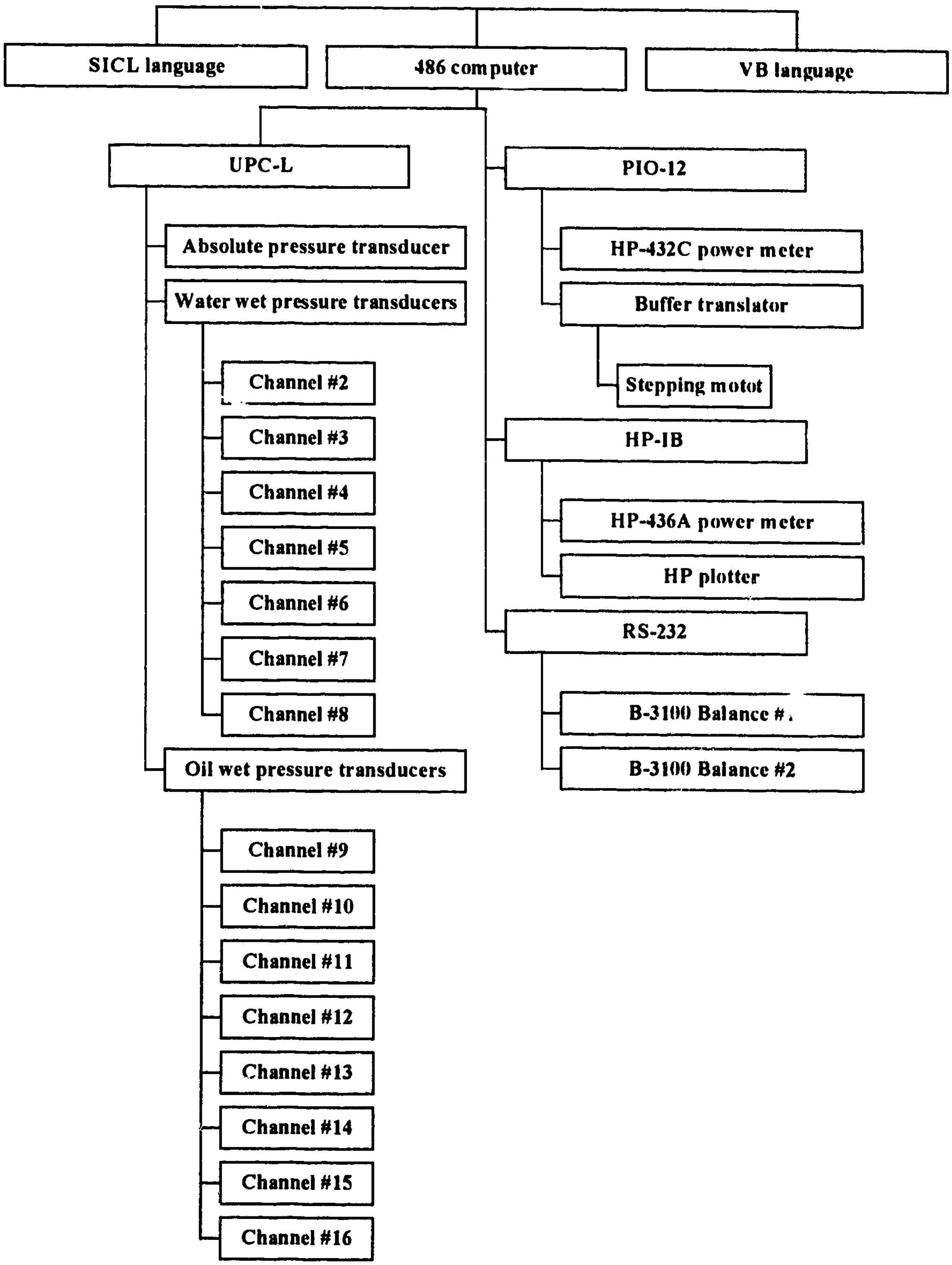
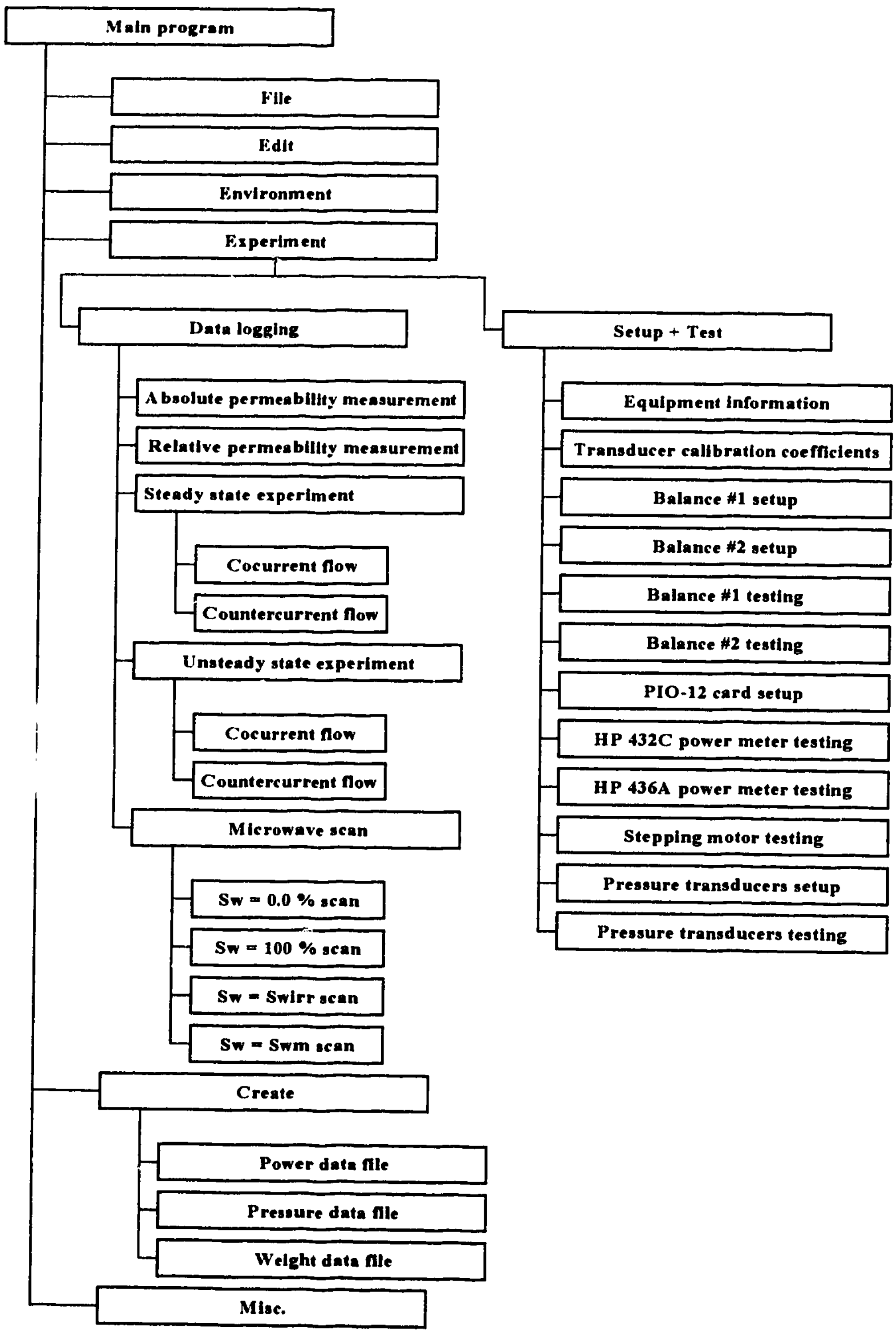


Figure IV-42: Software configuration for data collection



V. Statistical analysis

V.A. Permeability measurement

V.A.1. Phase dependent pressure-gradient determination

For the steady-state cocurrent or countercurrent flow experiments, the pressure profiles of each phase are measured using seven transducers spread evenly over the entire investigation length of the core holder. The recorded pressures are expected to vary linearly as the pressure drops from the injection end to the exit end for each phase.

To obtain the pressure gradient of each phase, a least-squares fit method is used, with the position of the transducer as the abscissa and the measured pressure as the ordinate. The slope thus calculated from the least-squares fit method is the pressure gradient of the phase. A comparison of uncertainty of the pressure gradient calculated between the conventional method (with transducers calibrated by 2-point procedure) and the improved method (with transducers calibrated by 10-point procedure) is presented in Table V-1. Note that the relative uncertainty in the pressure calculated is reduced to 1.7 % from 7.4 %, as a result of the improved calibration procedure.

V.A.2. Inlet capillary-pressure determination

The inlet capillary pressure of the core is obtained from the intercept of the least-squares fitted line obtained above, extrapolated to the position of the inlet point of the core holder. For a cocurrent-flow experiment, the inlet points are at the same coordinate. For steady-state, countercurrent flow, however, the inlet points are at the opposite ends of the core holder. A comparison of the uncertainty of the inlet capillary pressure extrapolated between the conventional method and the improved method is presented in Table V-2. Note that the relative uncertainty in the inlet capillary pressure calculated is reduced to 0.8 % from 3.3 %, as a result of the improved calibration procedure.

V.A.3. Permeability determination

By using the conventional fluid flow equations for the two-phase flow model, the effective permeabilities of each phase can be calculated. A comparison of the permeability measured between the conventional experimental method and the improved experimental method is presented in Table V-3. Note that with the previous 3.0 % relative uncertainty in the viscosity of the oil, LAGO, (corresponding to approximately one degree Celsius variation in ambient temperature) and transducers calibrated with the 2-point procedure, the relative uncertainty in the calculated permeability is approximately 3.2 %. However, using the improved 0.3 % relative uncertainty in the viscosity of the oil, achieved mainly by having a more stable ambient temperature, along with transducers calibrated with 10-point procedure, the relative uncertainty in the calculated permeability is only approximately 1.0 %.

V.B. Saturation level measurement

V.B.1. Steady-state saturation level determination

An inspection of Equation (M-11) revealed the fact that although the parameter B could be evaluated from a theoretical standpoint, the parameter A_{ls} has to be evaluated by using experimental data. Thus the theoretical response curve presented in Figure V-16 cannot be used directly for saturation level evaluation as the parameter A_{ls} was evaluated using the existing core holder. Moreover, the parameter A_{ls} is likely to change along the length of the core mainly because of slight variations in local heterogeneities of the core; thus, an empirical calibration curve has to be constructed for each cross-section (100 in this case) along the length of the improved core holder.

By using the results of the microwave power measurement from two steady-state experiments, one to be conducted at irreducible saturation level and the other to be conducted at residual oil saturation level, coupled with the results from two microwave power measurements of the core, one to be conducted at zero saturation level and the other to be conducted at 100 % saturation level, one can use a least-squares method to obtain the empirical relationship between the saturation level of the core at a specific cross-section and the microwave power level. By repeating the process for 100 cross-sections that are spread evenly over the entire length of the core, the saturation level of the core at any cross-section can then be interpolated by using the corresponding empirical equation constructed. The analyses of the range of uncertainty of the saturation level interpolated at each cross-section from using such an equation, due to the uncertainties in various parameters, are presented in Table V-4 and Figures V-1 and V-2. The average core saturation level resulting from a steady-state experiment can be obtained from a least-squares fit of all the saturation level measurements at each cross-section.

The above conventional method of using four-point calibration procedures to arrive at the empirical equations (Parsons, 1975) creates an unacceptable level of uncertainty for the saturation level interpolated for the complete length of the core. This is due to a combination of the high uncertainty in the initial water saturation level and residual oil saturation level arrived at by the conventional material balance method, the artificially imposed lowest microwave power detection limit (100 mW in this case), as well as the various uncertainties in the measured parameters, in particular, the high uncertainty in the measured microwave power at any water saturation level. The analyses are presented in Figures V-3 and V-4. Even when the empirical equations are constructed with the improved material balance method, the uncertainty of the interpolated water saturation level for each cross-section as well as for the entire length of the core from using the above method is still high when compared with the small uncertainties achieved with the other parameters for the current study. The analyses are presented in Table V-5 and Figures V-5, V-6, V-7 and V-8. To further reduce the range of uncertainty of the saturation level interpolated, a modification of the method used to construct the empirical equations can be made by using instead two empirical points with one at 0 % saturation level and the other at 100 % saturation level. The analyses for each cross-section and for the entire length of the core are presented in Table V-6 and Figures V-9, V-10, V-11 and V-12.

To measure the steady-state saturation level of the core, either the improved material balance method or the improved microwave power measurement technique can be used. The improved material balance method, however, cannot be used for the unsteady-state saturation level measurement. Furthermore, it can be seen from the analysis that the uncertainty of saturation level interpolated by either method is of the same order; thus, in order to be consistent in the methodology used, the improved microwave power measurement technique has to be used as the primary means of measuring saturation levels attained by steady-state experiments and the improved material balance method is used as a diagnostic check only.

V.B.2. Unsteady-state saturation level determination

Once the empirical equation needed for each cross-section has been determined from the above procedure, the dynamic saturation profile of the core can be obtained by scanning the complete length of the core for microwave power and then estimating the saturation level at each cross-section by using the empirical equations obtained by the improved microwave power measurement technique.

Table V-1: Comparison of the uncertainty of the pressure gradient calculated

Input variable	Expected value	Uncertainty (Transducers calibrated with 2-point procedure)	Uncertainty (Transducers calibrated with 10-point procedure)
Transducer #1 position (cm)	20.50	0.10	0.10
Transducer #2 position (cm)	37.00	0.10	0.10
Transducer #3 position (cm)	53.50	0.10	0.10
Transducer #4 position (cm)	70.00	0.10	0.10
Transducer #5 position (cm)	86.50	0.10	0.10
Transducer #6 position (cm)	103.00	0.10	0.10
Transducer #7 position (cm)	119.50	0.10	0.10
Transducer #1 reading (psig)	8.0000	0.347455	0.092503
Transducer #2 reading (psig)	7.1833	0.334882	0.083062
Transducer #3 reading (psig)	6.3665	0.323236	0.073996
Transducer #4 reading (psig)	5.5498	0.311960	0.065164
Transducer #5 reading (psig)	4.7330	0.300905	0.056634
Transducer #6 reading (psig)	3.9163	0.293429	0.049706
Transducer #7 reading (psig)	3.0995	0.297102	0.047058
Calculated parameter	**Expected value**	**Uncertainty**	**Uncertainty**
Pressure gradient (psig/cm)	-0.0495	0.366826E-2	0.821943E-3

Excessive significant digits are used for the purpose of carrying out numerical evaluations only.

Table V-2: Comparison of the uncertainty of the inlet capillary pressure calculated

Input variable	Expected value	Uncertainty (Transducers calibrated with 2-point procedure)	Uncertainty (Transducers calibrated with 10-point procedure)
Transducer #1 position (cm)	20.50	0.10	0.10
Transducer #2 position (cm)	37.00	0.10	0.10
Transducer #3 position (cm)	53.50	0.10	0.10
Transducer #4 position (cm)	70.00	0.10	0.10
Transducer #5 position (cm)	86.50	0.10	0.10
Transducer #6 position (cm)	103.00	0.10	0.10
Transducer #7 position (cm)	119.50	0.10	0.10
Transducer #1 reading (psig)	8.0000	0.347455	0.092503
Transducer #2 reading (psig)	7.1833	0.334882	0.083062
Transducer #3 reading (psig)	6.3665	0.323236	0.073996
Transducer #4 reading (psig)	5.5498	0.311960	0.065164
Transducer #5 reading (psig)	4.7330	0.300905	0.056634
Transducer #6 reading (psig)	3.9163	0.293429	0.049706
Transducer #7 reading (psig)	3.0995	0.297102	0.047058
Calculated parameter	**Expected value**	**Uncertainty**	**Uncertainty**
Inlet capillary pressure (psig)	9.0000	0.294417	0.725247E-1

Excessive significant digits are used for the purpose of carrying out numerical evaluation only.

Table V-3: Comparison of the uncertainty of the permeability calculated

	Conventional method		Improved method	
Input Parameters	**Expected value**	**Uncertainty**	**Expected value**	**Uncertainty**
$\mu\ (cp)$	5.3390	0.1602	5.3390	0.01602
$q\ (cc/hr)$	45.0000	0.0045	45.0000	0.0045
$P_{ou}\ (psig)$	2.0000	0.0320[1]	2.0000	0.0230
$P_{in}\ (psig)$	9.0000	0.1440[2]	9.0000	0.1035
L_{you}	140.00	0.10	140.00	0.10
L_{yin}	0.00	0.10	0.00	0.10
A	1.50	0.015	1.50	0.015
Calculated parameter	**Expected value**	**Uncertainty**	**Expected value**	**Uncertainty**
K	13.076902	0.41361985	13.076902	0.13653494

Excessive significant digits are used for the purpose of carrying out numerical evaluation only.

[1] 0.2165 psig possible offset between transducers not included.
[2] 0.2165 psig possible offset between transducers not included.

Table V-4: Construction of empirical equation using the conventional microwave power measurement technique and the conventional material balance method

Experiments	Expected value			Uncertainty		
	I_x (milliwatt)	I_o (milli-Watt)	S_w (fraction)	I_x (milliwatt)	I_o (milli-Watt)	S_w (fraction)
Dry core scan	12.500001	153.0900	0.0000	1.2500001	15.3090	0.00000000
Irreducible water saturation scan	7.1904416	153.0900	0.1509	0.7207513	15.3090	0.30367497E-1
Irreducible oil saturation scan	0.66608342	153.0900	0.8000	0.71969485E-1	15.3090	0.61964704E-1
Wet core scan	0.32002453	153.0900	1.0000	0.35151535E-1	15.3090	0.00000000
B calculated	-1.59172680			0.23268369		
Als calculated	-1.08803680			0.25697757		

Excessive significant digits are used for the purpose of carrying out numerical evaluation only.

Table V-5: Construction of empirical equation using the conventional microwave power measurement technique and the modified material balance method

Experiments	Expected value			Uncertainty		
	I_x (milliwatt)	I_o (milli-Watt)	S_w (fraction)	I_x (milliwatt)	I_o (milli-Watt)	S_w (fraction)
Dry core scan	12.500001	153.0900	0.0000	1.2500001	15.3090	0.00000000
Irreducible water saturation scan	7.1904416	153.0900	0.1509	0.7207428	15.3090	0.19918023E-2
Irreducible oil saturation scan	0.66608342	153.0900	0.8000	0.71969485E-1	15.3090	0.46161310E-1
Wet core scan	0.32002453	153.0900	1.0000	0.35151535E-1	15.3090	0.00000000
B calculated	-1.59172190			0.15833601		
Als calculated	-1.08804180			0.14312652		

Excessive significant digits are used for the purpose of carrying out numerical evaluation only.

Table V-6: Construction of empirical equation using the modified microwave power measurement technique and the modified material balance method

Experiments	Expected value			Uncertainty		
	I_x (milli-Watt)	I_o (milli-Watt)	S_w (fraction)	I_x (milli-Watt)	I_o (milli-Watt)	S_w (fraction)
Dry core scan	12.500001	153.0900	0.0000	1.2500001	15.3090	0.00000000
Wet core scan	0.32002453	153.0900	1.0000	0.35151535E-1	15.3090	0.00000000
B calculated	-1.5917268			0.088183294		
Als calculated	-1.0880368			0.126036000		

Excessive significant digits are used for the purpose of carrying out numerical evaluation only.

Figure V-1: Upper and lower error bounds of saturation interpolated at each cross-section - Conventional microwave power measurement technique with conventional material balance method

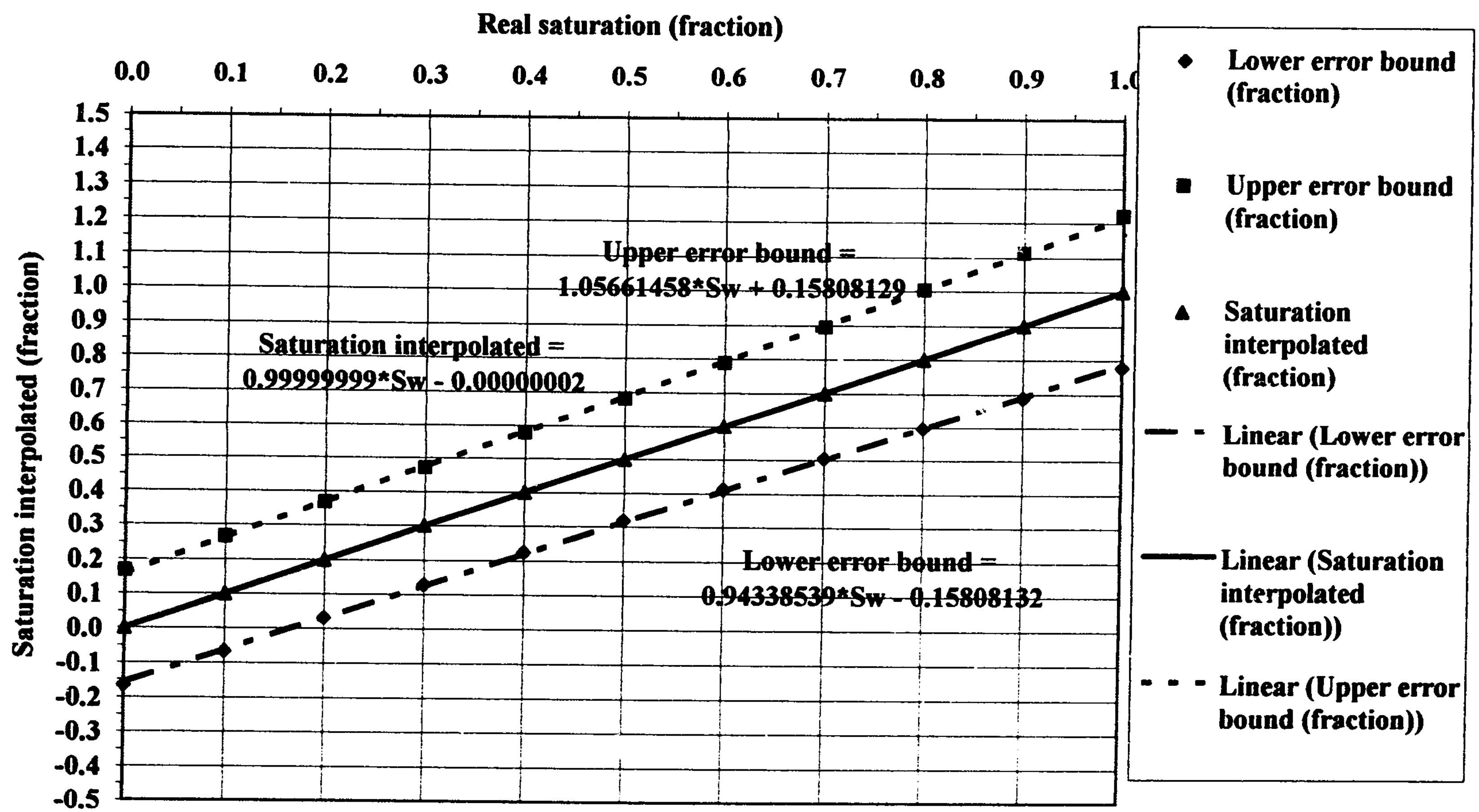


Figure V-2: Relative uncertainty in saturation interpolated at each cross-section - Conventional microwave power measurement technique with conventional material balance method

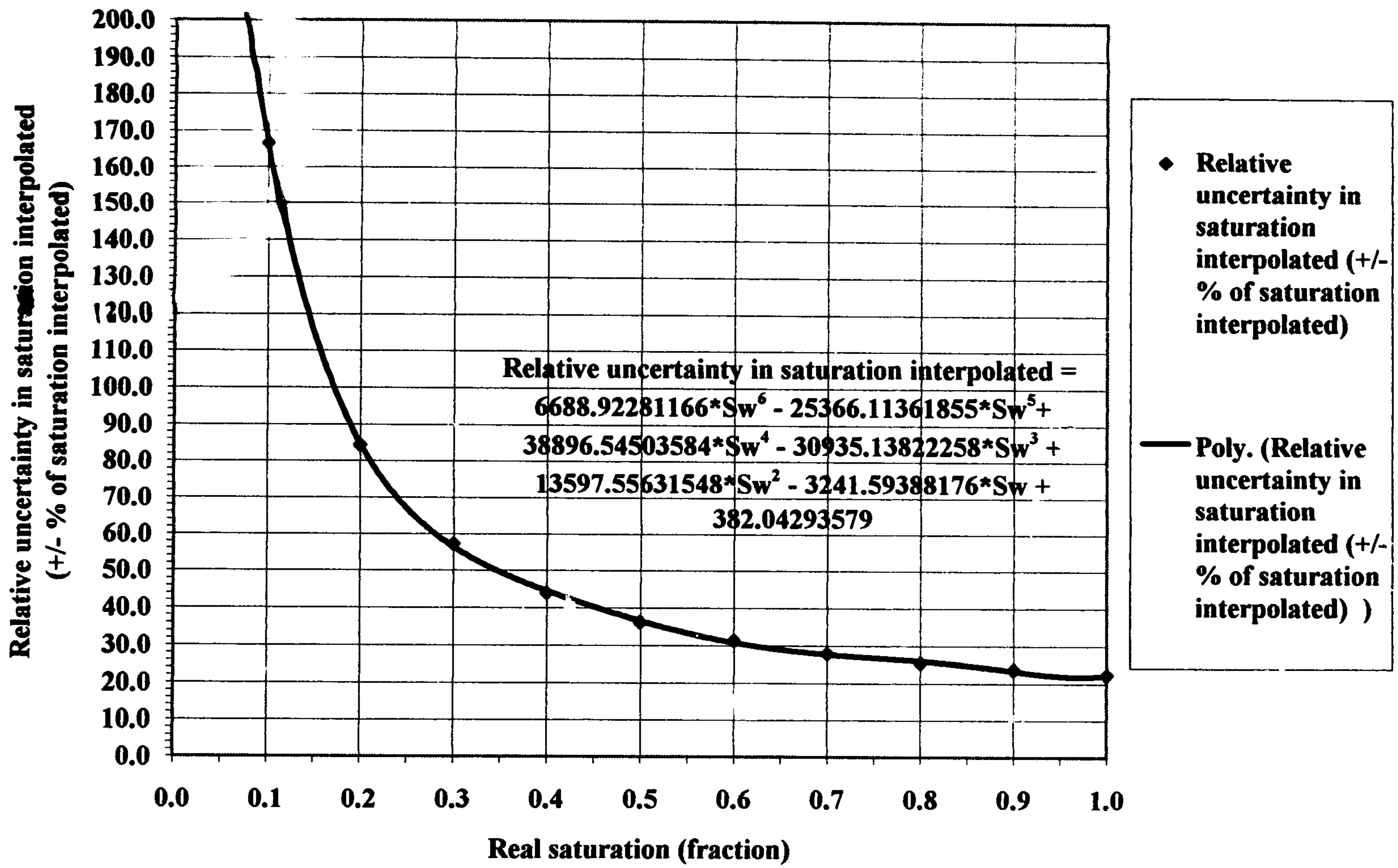


Figure V-3: Upper and lower error bounds in saturation interpolated for the complete core - Conventional microwave power measurement technique with conventional material balance method

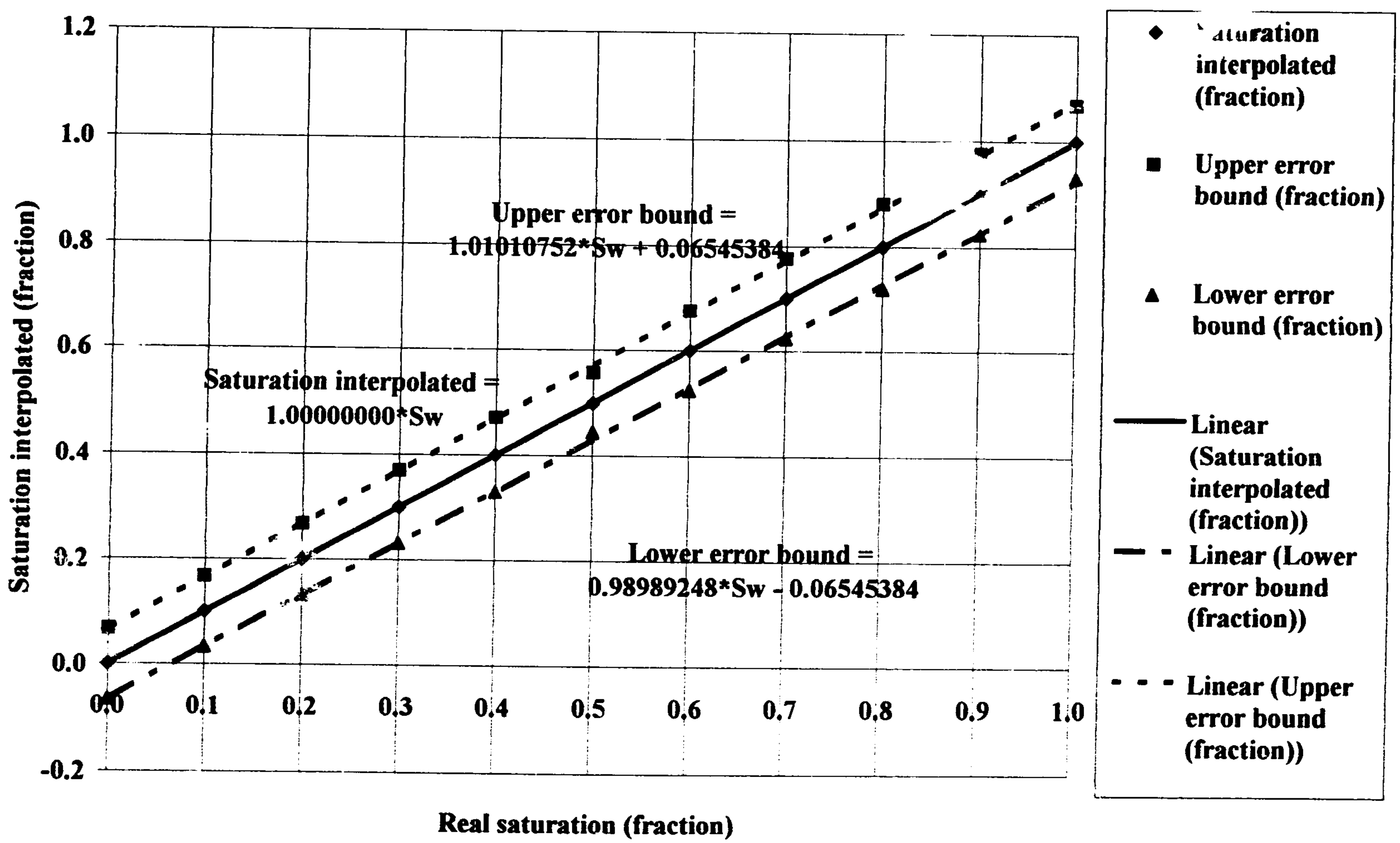


Figure V-4: Relative uncertainty in saturation interpolated for the complete core - Conventional microwave power measurement technique with conventional material balance method

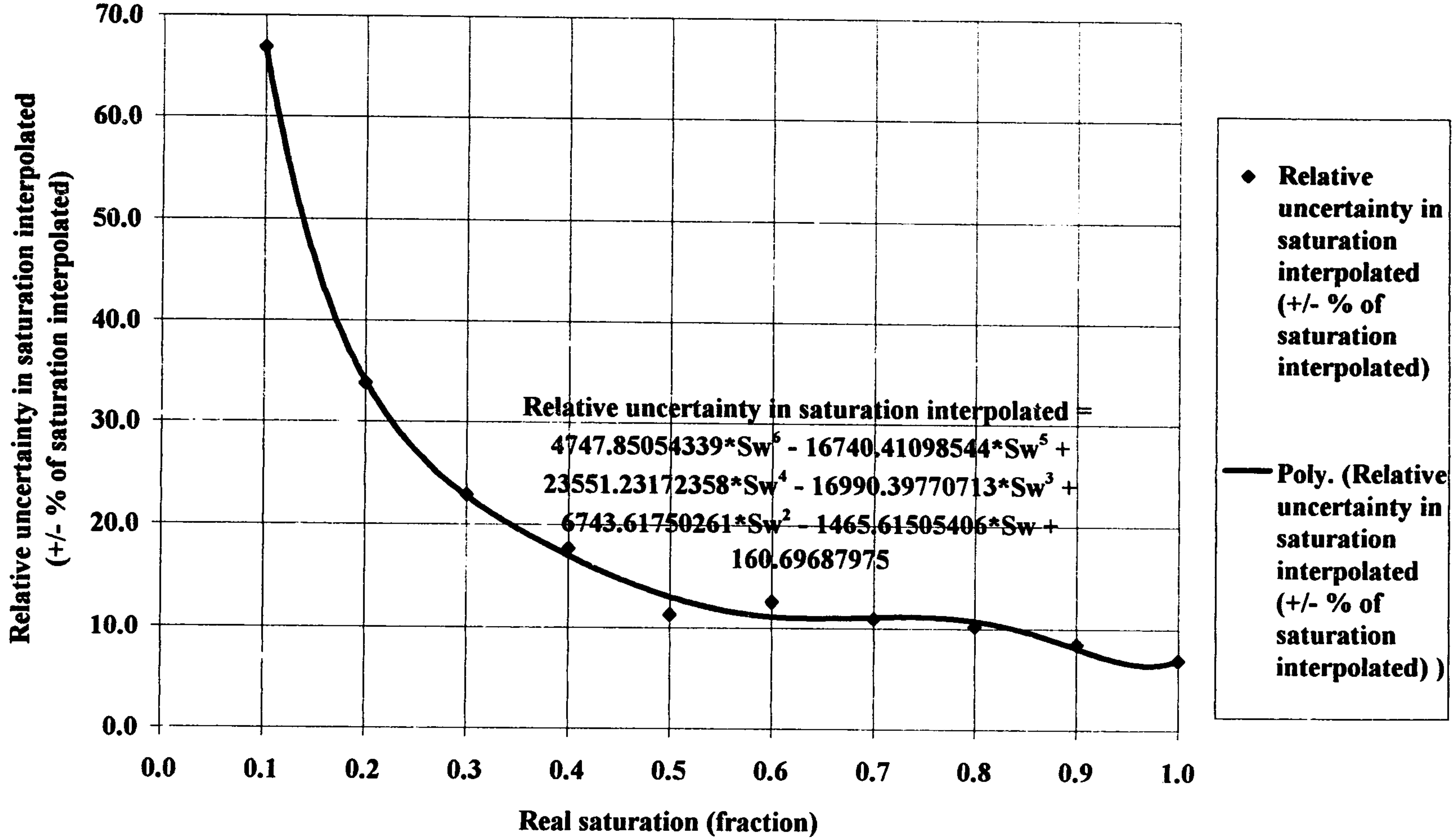


Figure V-5: Upper and lower error bound of saturation interpolated at each cross-section - Conventional microwave power measurement technique with improved material balance method

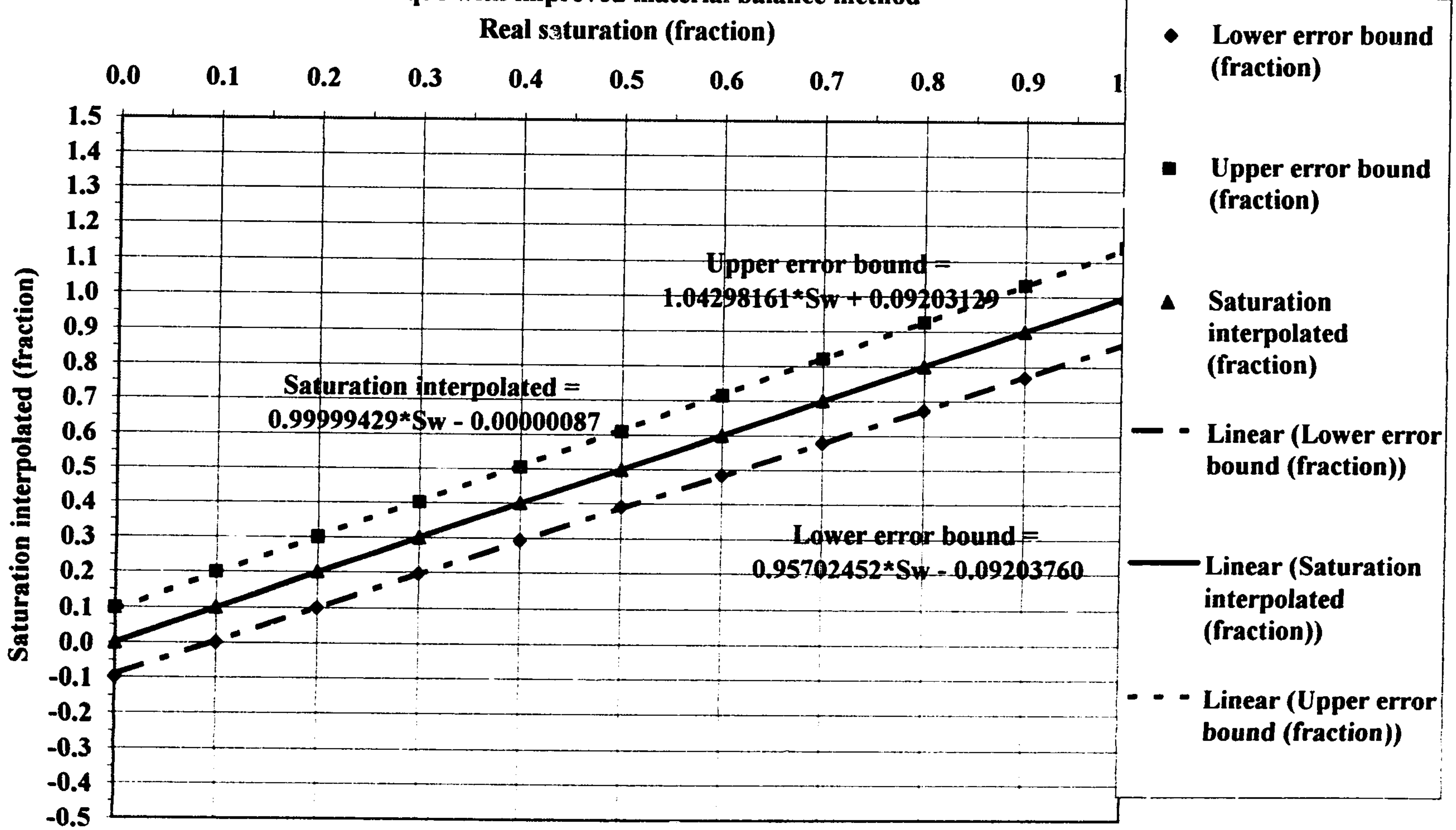


Figure V-6: Relative uncertainty in saturation interpolated at each cross-section - Conventional microwave power measurement technique with improved material balance method

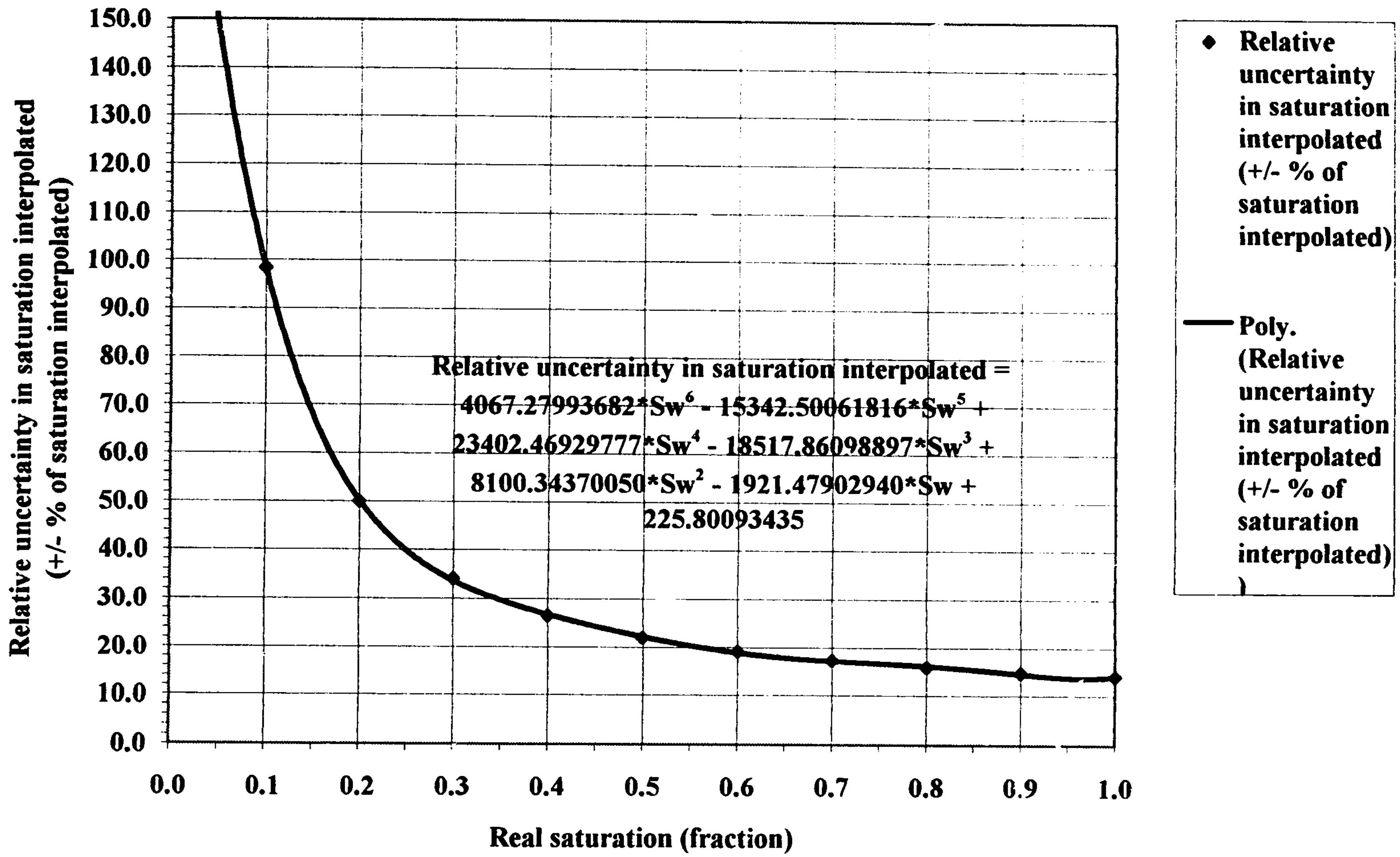


Figure V-7: Upper and lower error bound of saturation interpolated for the complete core - Conventional microwave power measurement technique with improved material balance method

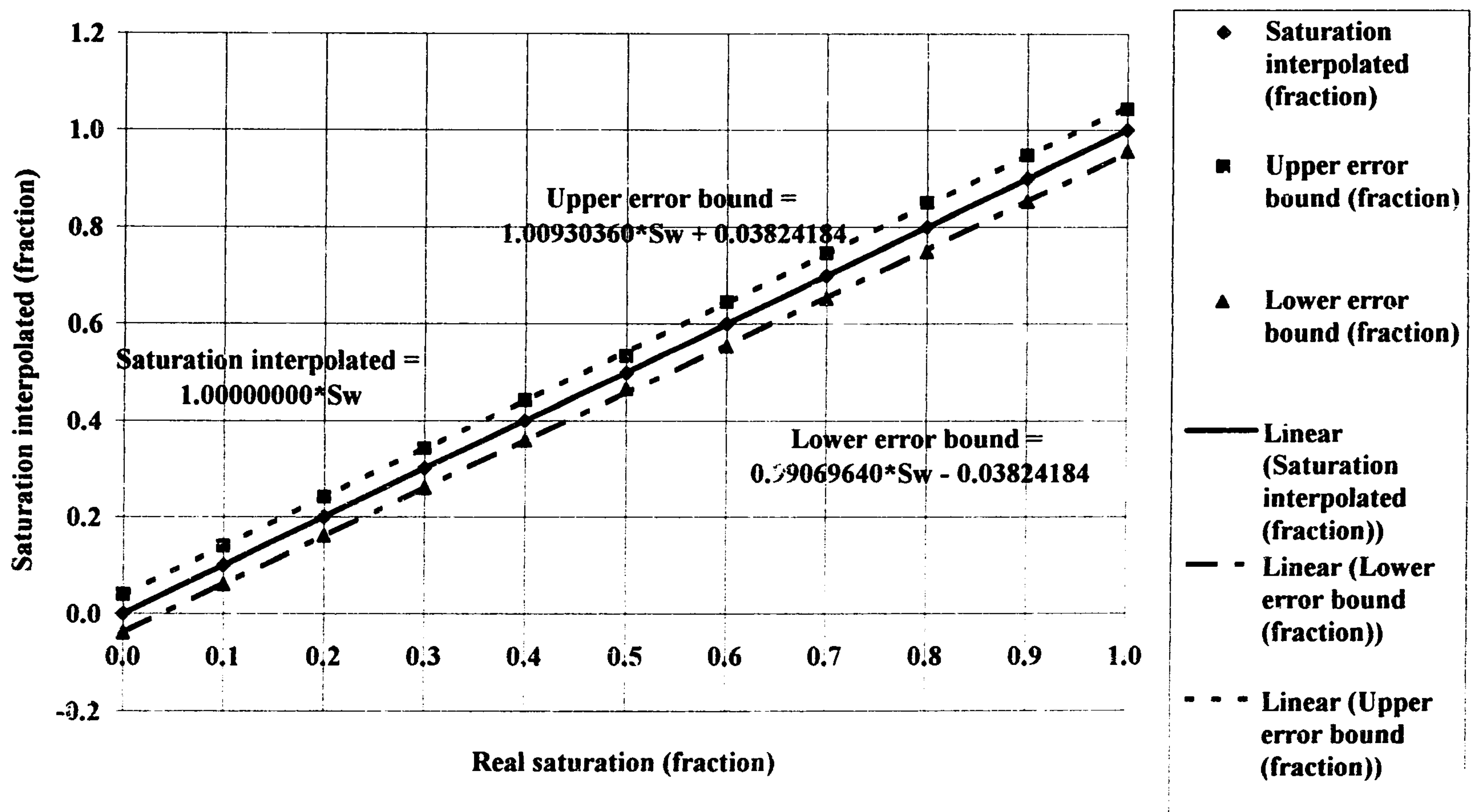


Figure V-8: Relative uncertainty in saturation interpolated for the complete core - Conventional microwave power measurement technique with improved material balance method

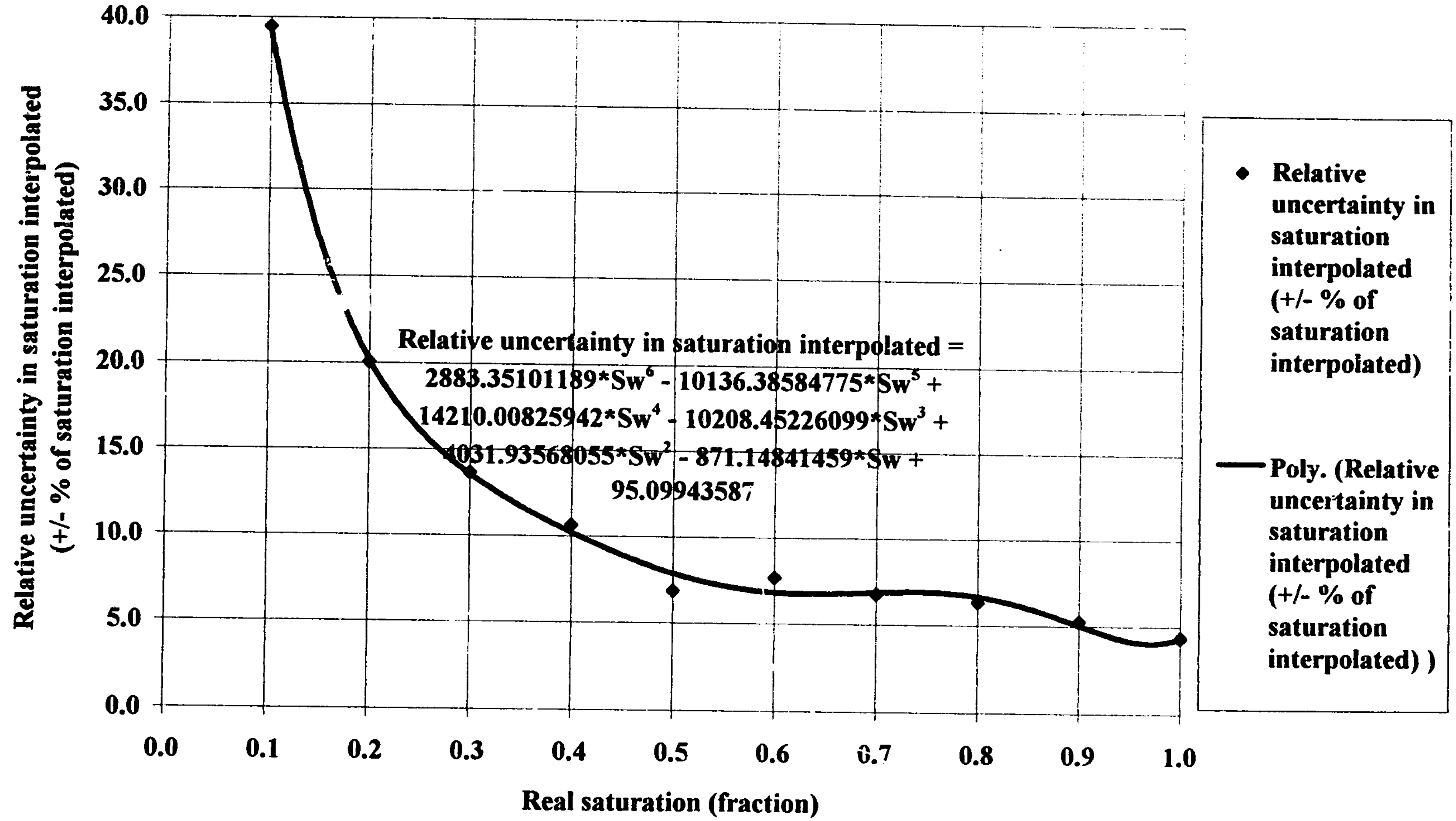


Figure V-9: Upper and lower error bounds in saturation interpolated at each cross-section - Improved microwave power measurement technique

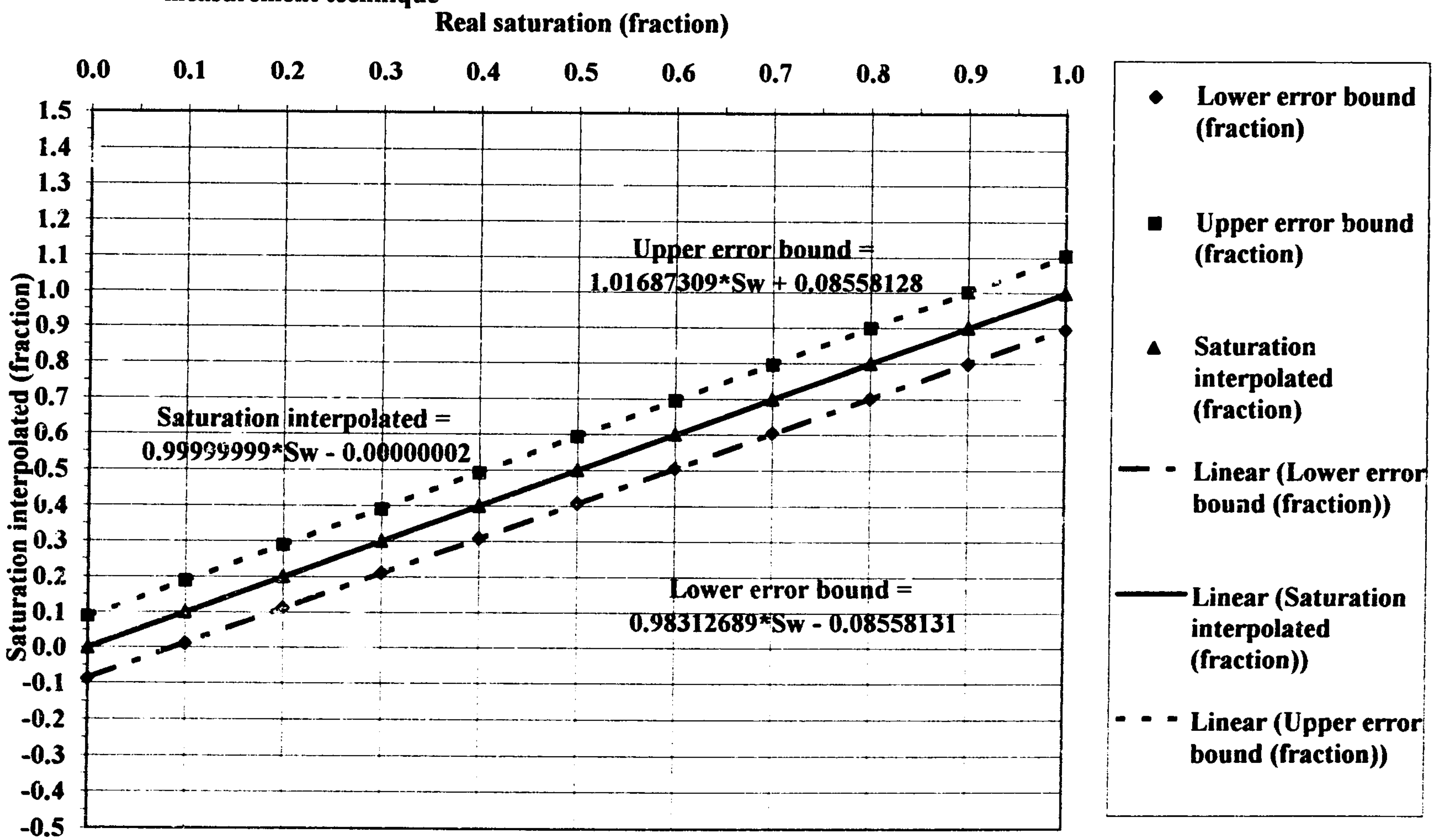


Figure V-10: Relative uncertainty in saturation interpolated at each cross-section - Improved microwave power measurement technique

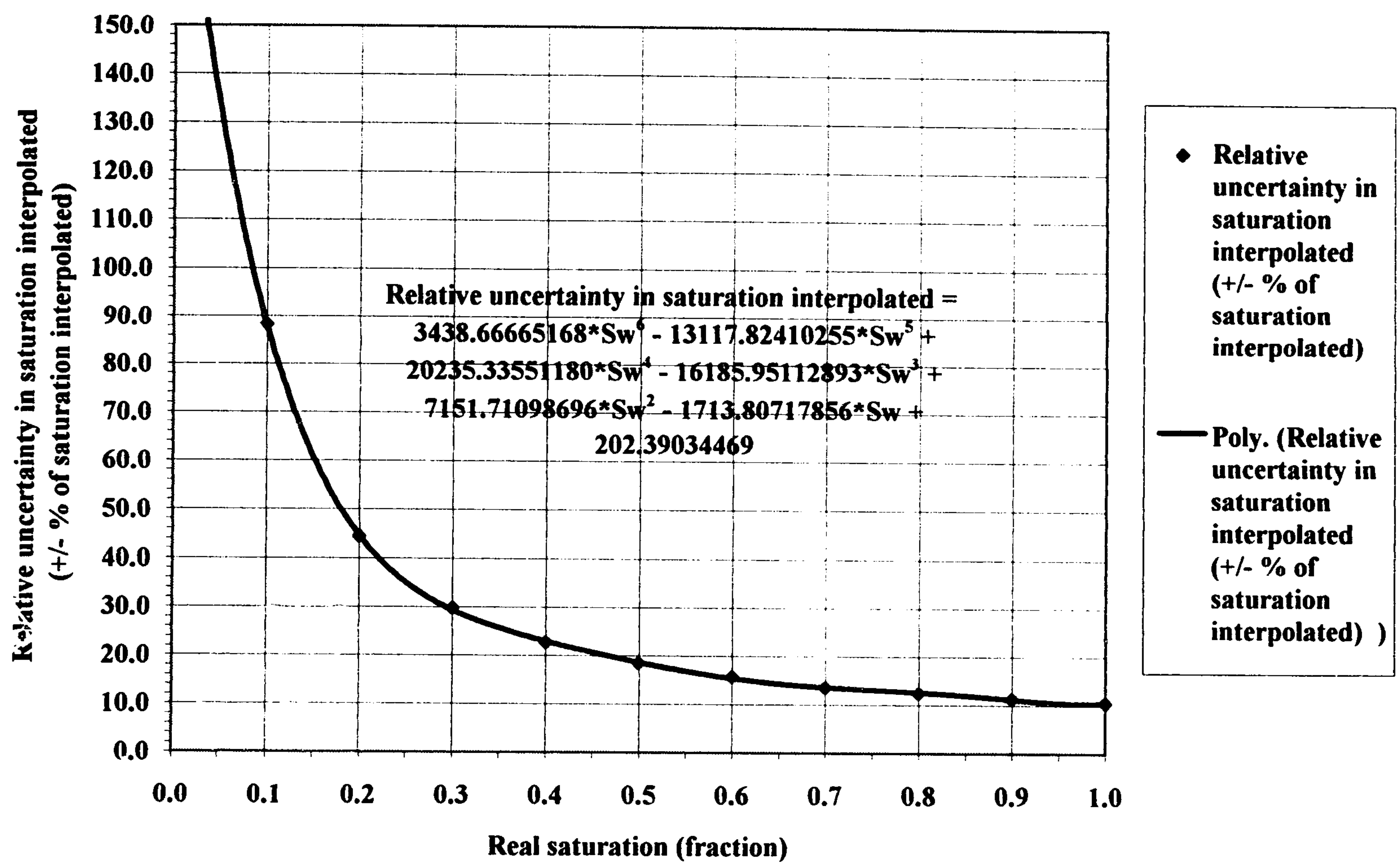


Figure V-11: Upper and lower error bound in saturation saturation for the complete core - Improved microwave power measurement technique

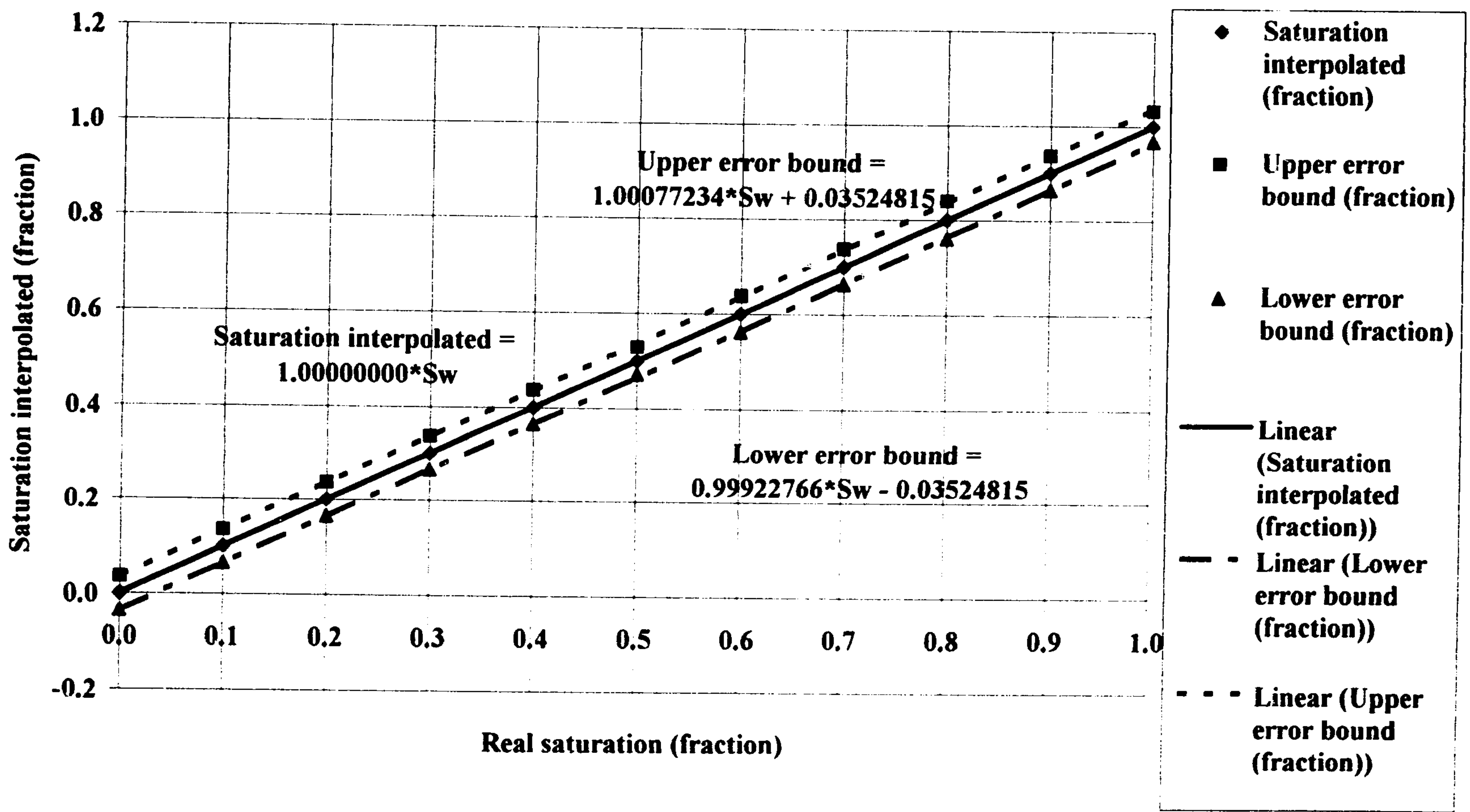
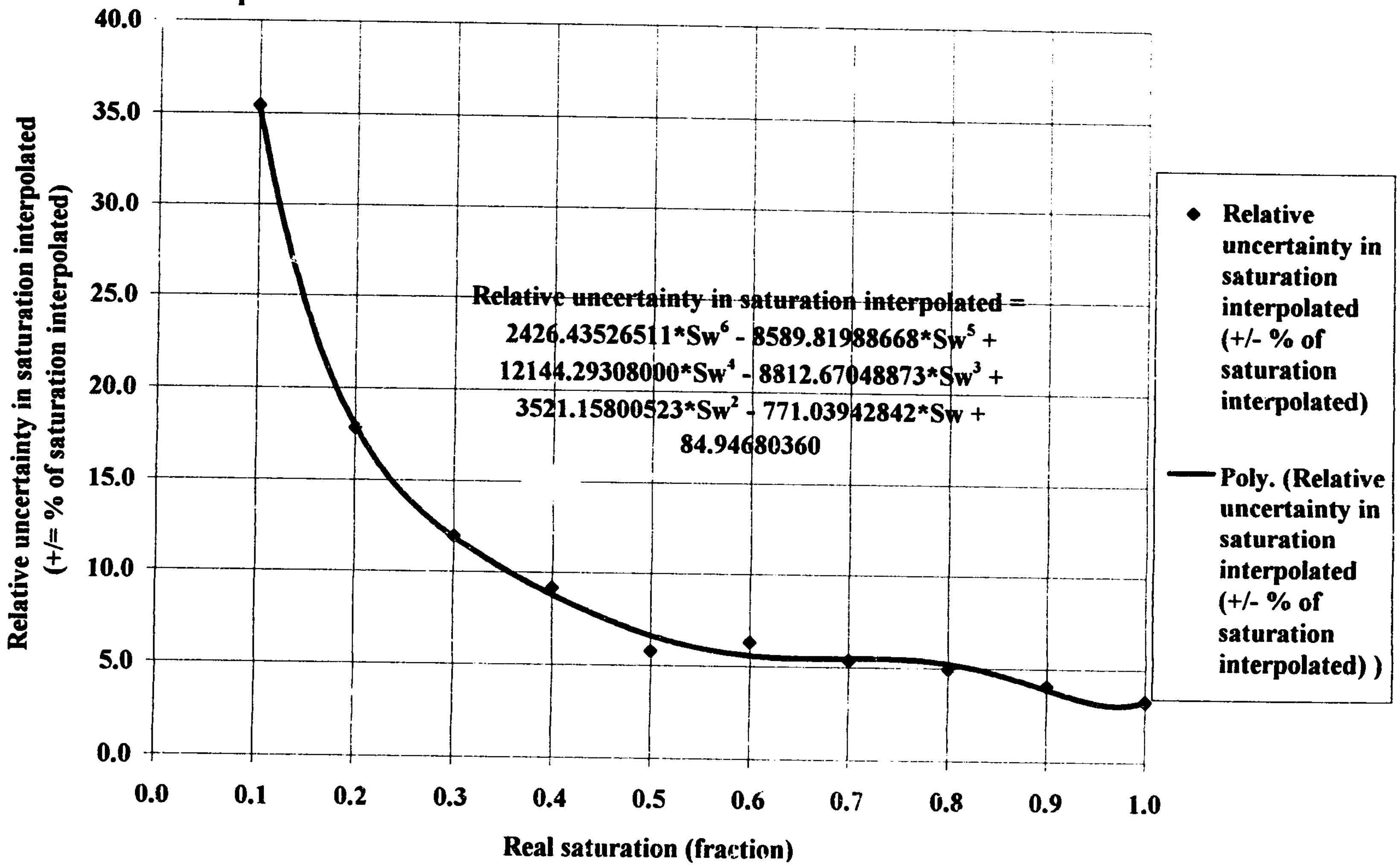


Figure V-12: Relative uncertainty in saturation interpolated for the complete core - Improved microwave power measurement technique



VI. Concluding remarks

In order to arrive at a statistically valid conclusion as to whether the matrix of four generalized permeability coefficients can be viewed as the fundamental building block for any two-phase transport processes, two sets of data must be available, each being arrived at by independent pairs of experiments. The two pairs of experiments, on the other hand, have to be designed based on theoretically valid concepts. Based on published experimental methods and measurement techniques, an experimental methodology has been outlined in this study. Note that one of the three types of experiments is common to the two pairs of experiments designed. Strictly speaking, this violates the fundamental assumption of the ANOVA method that data have to be statistically independent. It is recommended to construct one additional type of experiment so that the two pairs of experiments can provide mutually exclusive data.

On the other hand, the experimental apparatus required to provide the fundamental parameters has to be capable of providing the necessary parameters within an acceptable range of uncertainty. Utilizing equipment available at the University of Alberta, an experimental apparatus was designed and constructed. Furthermore, based on statistical analyses, the apparatus is expected to provide all the necessary parameters within an acceptable range of uncertainty, with the exception of the water saturation level of the core under study. This result is presented in Table VI-1.

Further improvement on the range of uncertainty for water saturation level arrived at by the microwave power measurement method, however, can be made when the required microwave power of 100 mW at 100 % water saturation level can be lowered further without further increasing the required sensor saturation time. Moreover, if measurements can be made on the reflected microwave power simultaneously with other power measurements, additional improvement of the range of uncertainty can be expected for saturation measurement.

Table VI-1 Summary of measurement uncertainties

Parameters	Approximate limits of relative uncertainty	Reference
Calibrated pressure obtained from each pressure transducer	+/- 1.2 % to 1.6 % for any measurement within the range of interest (1.5 psig to 11.0 psig)	Tables IV-13
Phase-dependent pressure gradient in the core attained by each experiment	1.7 %	Table V-1
Inlet capillary pressure attained by each experiment	0.8 %	Table V-2
Absolute permeability	1.0 %	Table V-3
Initial water saturation for the core[1]	1.3 %	Tables IV-17
Water saturation for the entire core attained by each experiment[2]	1.0 % to 6.0 % for any measurements within the range of interest (Sw = 0.15 to Sw = 0.80)	Figures IV-39
Water saturation at each cross section of the core attained by each experiment[3]	10.0 % to 60.0 % for any measurements within the range of interest (Sw = 0.15 to Sw = 0.80)	Figures V-10
Water saturation of the entire core attained by each experiment[4]	5.0 % to 25.0 % for any measurements within the range of interest (Sw = 0.15 to Sw = 0.80)	Figures V-12

[1] Obtained by performing material balance calculations.
[2] Obtained by performing material balance calculations.
[3] Obtained by using microwave power measurement technique.
[4] Obtained by using microwave power measurement technique.

VII. Suggestions for future studies

VII.A. Theoretical models and experimental studies

1. Conduct experiments to investigate the matrix of generalized permeability coefficients and the validity of assuming the generalized permeability coefficients to be the fundamental building blocks for any two-phase transport process.

2. Further investigate the limits of uncertainty for the generalized permeability coefficients calculated based on the uncertainties in the experimentally measured parameters. It is recommended that any future sensitivity analysis be carried out with analytical equations rather using VARSIM, in order to achieve a better stability of the reported data.

3. Develop and investigate a 2-D fluid flow model based on the extended formulation of Darcy's law.

4. Expand the current study to a different mobility ratio on a linear model.

5. Develop and investigate a 3-phase linear model.

6. Develop a reservoir simulator using newly found understanding of two-phase fluid flow equations.

7. Consider using a scheme of injecting both water and oil into the core holder until steady state has been achieved, then shut in the water injection and keep the oil injection going. This would provide one more type of experiment, thus making the methodology presented in this study comply fully with the requirements of ANOVA.

8. Investigate the contradiction between the finding of the current study and the previous studies on the maximum saturation level that can be detected by the microwave power measurement technique.

VII.B. Equipment

1. Consider further boosting the power available for microwave power measurement by purchasing a different oscillator or by using a combination of two oscillators and a power combiner. This is expected to have a substantial impact on the accuracy of the saturation levels measured, under both steady-state and unsteady state flow.

2. Record the reflected microwave power and perform corrections for saturation level estimation. The current software and microwave apparatus have provisions for incorporating one more power meter. This would provide a more accurate measurement of the microwave power actually energizing the core under study and improve the accuracy of the saturation level measurement.

3. Continue to look for a better selective wetting capillary barrier for the pressure taps. Consider using a different wetting medium for proper oil-water separation at the exit -end and inlet-end of the core holder.

4. Consider using a pure hydrocarbon in the alkene series with high molecular weight as the non-wetting phase injection fluid (Jennings, 1957). However, some researchers (Bourbiaux and Kalaydjian, 1990) have used a paraffinic refined oil, Soltrol 130™, with a viscosity of 1.5 cp. and a density of 0.76 g/c.c. at 20 degree Celsius as the wetting phase fluid. An adequate stock of LCGO and Ethylzio 16H Polyalphaolefins in addition to LAGO has been accumulated in the laboratory and their viscosity and density data are available for future usage.

5. On the basis of error analysis, it is suggested that the future laboratory work be carried out in a temperature monitored and controlled environment, to facilitate a more accurate material balance calculation as well as for permeability determination. The current hardware and software

configurations for the computer have provisions for incorporating a temperature sensor.

6. Current hardware and software configurations for the computer have provisions for incorporating the Ruska injection pumps for automatic start/stop. This would provide a fully automatic system for research purposes.

VII.C. Data analysis methods

Construct an iterative algorithm to correct for the deviations introduced into the saturation profiles because of the long sweep time of a complete microwave scan.

References

Anderson, T.B. and **Jackson, R.**: "A Fluid Mechanical Description of Fluidized Beds," Ind. Eng. Chem. Fund. (November 1967), Vol. 6, No. 4, 527-539.

Bentsen, R.G.: "Conditions Under which the Capillary Term May be Neglected," Journal of Canadian Petroleum Technology (October-December 1978), Vol. 17, No. 4, 25-30.

Bentsen, R.G.: "A New Approach to Instability Theory in Porous Media," SPE J. (October 1985), Vol. 25, No. 5, 765-779.

Bentsen, R.G.: "On the Use of the Theory of Gibbs to Estimate the "Frontal Tension" for Analysis of Stability of Fluid Flow in Porous Media," AOSTRA Journal of Research (1990), Vol. 6, No. 3, 155-170.

Bentsen, R.G.: "An Investigation into Whether the Nondiagonal Mobility Coefficients which Arise in Coupled, Two Phase Flow are Equal" Transport in Porous Media (January 1994), Vol. [illegible], No. 1, 23-32.

Bentsen, R.G.: "Impact of Model Error on the Measurement of Flow Properties Needed to Describe Flow Through Porous Media," Paper in preparation (1996).

Bentsen, R.G. and **Manai, A.A.**: "Measurement of Cocurrent and Countercurrent Relative Permeability Curves Using the Steady-State Method," AOSTRA Journal of Research (Summer 1991), Vol. 7, No. 3, 169-181.

Bentsen, R.G. and **Manai, A.A.**: "On the Use of Conventional Cocurrent and Countercurrent Effective Permeabilities to Estimate the Four Generalized Permeabilities Coefficients which Arise in Coupled, Two-Phase Flow," Transport in Porous Media (1993), Vol. 11, 243-262.

Bentsen, R. G. and **Saeedi, J.**: "Liquid-Liquid Immiscible Displacement in Unconsolidated Porous Media," Journal of Canadian Petroleum Technology (January-March 1981), Vol. 20, No. 1, 93-103.

Bourbiaux, B.J. and **Kalaydjian, F.J.**: "Experimental Study of Cocurrent and Countercurrent Flows in Natural Porous Media," Society of Petroleum Engineers Reservoir Engineering (August 1990), Vol. 5, No. 3, 361-368.

Brost, D.F. and **Davis, L.A.**: "Determination of Oil Saturation Distributions in Field Cores by Microwave Spectroscopy," Society of Petroleum Engineers 10110 presented at the SPE-AIME 56th Annual Technical Conference and Exhibition, San Antonio, Texas (October 5-7, 1981).

Cole, K.S. and **Cole, R.H.**: "Dispersion and Absorption in Dielectrics," Journal of Chem. Phys. (1941), Vol. 9, 341-351.

Collin, R.E.: Foundations for Microwave Engineering, McGraw-Hill Book Company, New York (1966).

Darcy, H.: "Les fontaines Publiques de la Ville de Dijon," Victor Dalmont, Paris, (1856).

Davis Jr., L.A.: " Computer-Controlled Measurement of Laboratory Areal Flood Saturation Distributions," SPE 12037 (October 1983).

de la Cruz, V. and **Spanos, T.J.T.**: "Mobilization of Oil Ganglia," AIChE J. (1983), Vol. 29, No. 5, 854-858.

Griffin, W.H. and **Hamilton, R.W.**: "VARSIM: Interactive Sensitivity Analysis and Simulation Using Variance Components," ACM, 1990.

Hahn, G.J. and **Shapiro, S.S.**: Statistical Models in Engineering, John Wiley and Sons, Inc., New York (1967).

Hall, W.A.: "An Analytic Derivation of the Darcy Equation," EOS Trans., AGU (1956), Vol. 37, No. 185.

Hasted, J.B.: Aqueous Dielectrics, Chapman and Hall Ltd., London (1973).

Hubbert, M.K.: "Darcy's Law and the Field Equations of the Flow of Underground Fluids," Bulletin de l'Association Internationale d'Hydrologie Scientifique (1957), No. 5, 24-59.

Islam, M.R. and **Bentsen, R.G.**: "A Dynamic Method for Measuring Relative Permeability," Journal of Canadian Petroleum Technology (January-February 1986), Vol. 25, No. 1, 39-50.

Jennings Jr., H.Y.: "Surface Properties of Natural and Synthetic Porous Media," Producers Monthly (March 1957), Vol. 21, No. 5, 20-24.

Kalaydjian, F.J.: "A Macroscopic Description of Multiphase Flow in Porous Media Involving Spacetime Evolution of Fluid/Fluid Interface," Transport in Porous Media (December 1987), Vol. 2, No. 6, 537-552.

Kalaydjian, F.J.: "Origin and Quantification of Coupling between Relative Permeabilities for Two-Phase Flows in Porous Media," Transport in Porous Media (June 1990), Vol. 5, No. 3, 215-229.

Kyte, J.R. and **Rapoport, L.A.**: "Linear Waterflood Behavior and End Effects in Water-Wet Porous Media," Transaction, AIME (1958), Vol. 213, 423-426.

Laverghetta, T.S.: Microwave Measurements and Techniques, Artech House, Inc., Massachusetts (1976).

Lehner, F.K.: "A Derivation of the Field Equations for Slow Viscous Flow Through a Porous Medium," Ind. Eng. Chem. Fund. (1979) Vol. 18, No. 1, 41-45.

Levine, J.S.: "Displacement Experiments in a Consolidated Porous System," Petroleum Transactions, AIME (1954), Vol. 201, 57-66.

Liang, Q. and **Lohrenz, J.**: "Dynamic Methods of Measuring Coupling Coefficients of Transport Equations of Two-Phase Flow in Porous Media," Transport in Porous Media (April 1994), Vol. 15, No. 1, 71-79.

Malmberg, C.G. and **Maryott, A.A.**: "Dielectric Constant of Water from 0^0 to 100^0 C," J. Res. Nat. Bur. Standards (January 1956), Vol. 56, No. 1, 1-8.

Mannseth, T.: "Commentary on 'Origin and Quantification of Coupling Between Relative Permeabilities for Two-Phase Flows in Porous Media' By F. Kalaydjian," Transport in Porous Media (August 1991), Vol. 6, No. 4, 469-471.

Mellon, M. G.: Analytical Absorption Spectroscopy, John Wiley and Sons, Inc., New York (1950).

Muskat, M.: The Flow of Homogeneous Fluids Through Porous Media, International Human Resources Development Corporation, Boston (1982).

Neuman, S.P.: "Theoretical Derivation of Darcy's Law," Acta Mechanica (1977), Vol. 25, No. 3-4, 153-170.

O'Steen, B.L.: " Time-Dependent Flow Behavior During Immiscible Displacement in Porous Media," SPE 14369 (September 1985).

O'Steen, B.L. and **Holm, L.W.**: "Saturation Distributions During Displacements Generated by the Simultaneous Injection of Oil and Water," SPE 15198 (November 1988).

Parsons, R.W.: "Microwave Attenuation - A New Tool for Monitoring Saturations in Laboratory Flooding Experiments," SPE J. (August 1975), Vol. 15, No. 4, 302-309.

Rose, W.: "Second Thoughts on Darcy's Law," Iran. Pet. Inst. Bull (1972), Vol. 48, 25-30.

Rose, W.: "Attaching New Meanings to the Equations of Buckley and Leverett," Journal of Petroleum Science and Engineering (August 1988a), Vol. 1, 223-228.

Rose, W.: "Measuring Transport Coefficients Necessary for the Description of Coupled Two-Phase Flow of Immiscible Fluids in Porous Media," Transport in Porous Media (April 1988b), Vol. 3, No. 2, 163-171.

Rose, W.: "Data Interpretation Problems to be Expected in the Study of Coupled Fluid Flow in Porous Media," Transport in Porous Media (April 1989), Vol. 4, No. 2, 185-198.

Rose, W.: "Coupling Coefficients for Two-Phase Flow in Pore Spaces of Simple Geometry," Transport in Porous Media (February 1990), Vol. 5, No. 1, 97-102.

Rose, W.: "Letter to the Editor: Coupling Coefficients for Two-Phase Flow in Pore Spaces of Simple Geometry," Transport in Porous Media (1993), Vol. 10, 293-296.

Sarma, H.K. and **Bentsen, R.G.**: "An Experimental Verification of a Modified Instability Theory for Immiscible Displacements in Porous Media," Journal of Canadian Petroleum Technology (July-August 1987), Vol. 26, No. 4, 88-99.

Sarma, H.K. and **Bentsen, R.G.**: "A New Method for Estimating Relative Permeabilities from Unstabilized Displacement Data," Journal of Canadian Petroleum Technology (July-August 1989), Vol. 28, No. 4, 118-128.

Scheidegger, A.E.: The Physics of Flow Through Porous Media, University of Toronto Press, Toronto, Ontario, (1974).

Slattery, J.C.: "Flow of Viscoelastic Fluids Through Porous Media," AIChE J. (November 1967), Vol. 13, No. 6, 1066-1071.

Slattery, J.C.: "Single-Phase Flow Through Porous Media," AIChE J. (November 1969), Vol. 15, No. 6, 866-872.

Slattery, J.C.: "Two-Phase Flow Through Porous Media," AIChE J. (May 1970), Vol. 16, No. 3, 345-352.

Torres, F.E.: "Closure of the Governing Equations for Immiscible, Two-Phase Flow: A Research Comment," Transport in Porous Media (August 1987), Vol. 2, No. 4, 383-393.

Whitaker, S.: "Diffusion and Dispersion in Porous Media," AIChE J. (May 1967), Vol. 13, No. 3, 420-427.

Whitaker, S.: "Flow in Porous Media I: A Theoretical Derivation of Darcy's Law," Transport in Porous Media (1986a), Vol. 1, No. 1, 3-25.

Whitaker, S.: "Flow in Porous Media II: The Governing Equations for Immiscible Two-Phase Flow," Transport in Porous Media (1986b), Vol. 1, No. 2, 105-125.

Whitaker, S.: "Comments on Closure of the Governing Equations for Immiscible, Two-Phase Flow: A research comment By F.E. Torres," Transport in Porous Media (August 1987), Vol. 2, No. 4, 395.

Wyckoff, R.D. and **Botset, H.G.**: "Flow of Gas-Liquid Mixtures Through Unconsolidated Sands," Physics, (September 1936), Vol. 7, 325-345.

Yuster, S.T.: "Theoretical Considerations of Multiphase Flow in Idealized Capillary Systems," Proc. Third World Petroleum Congress, Section II, E.J. Brill, The Hague (1951), 437-445.

HP436A Power Meter Operating and Service Manual, Hewlett-Packard, July 1984.

Appendix A- 1 Equations used for the conventional material balance calculation

Pore volume :

$$PV = \frac{WW1 - DW1}{\rho_l} - (DV1 + DV2) \tag{A1-1}$$

Bulk volume :

$$BV = \frac{FW1 - EW1}{\rho_l} - (DV1 + DV2) \tag{A1-2}$$

Porosity :

$$\phi = \frac{PV}{BV} \tag{A1-3}$$

Initial water saturation :

$$S_{li} = 1 - \frac{\left[TVP - N_p - DV1 - DV2 \cdot (1 - FWF2)\right]}{PV} \tag{A1-4}$$

Water saturation at end of each experiment :

$$S_l = \frac{S_{li} \cdot PV + \left[q_l \cdot t_\infty - (TVP - N_p)\right]}{PV} + \tag{A1-5}$$

$$\frac{DV1 \cdot (IWF1 - FWF1)}{PV} + \frac{DV2 \cdot (IWF2 - FWF2)}{PV}$$

Appendix A- 2 Equations used for the improved material balance calculation

Pore volume :

$$PV = \frac{(WW1 - DW1)}{\rho_l} - (DV1 + DV2) \tag{A2-1}$$

Bulk volume :

$$BV = \frac{(FW1 - EW1)}{\rho_l} - (DV1 + DV2) \tag{A2-2}$$

Porosity :

$$\phi = \frac{PV}{BV} \tag{A2-3}$$

Initial water saturation :

$$S_{1i} = 1 - \left(\frac{W - \rho_2 \cdot q_2 \cdot t_\infty}{\rho_1 - \rho_2} \right) \cdot \frac{1}{PV} + \frac{\left[DV1 \cdot (1 - FWF1) + DV2 \cdot (1 - FWF2) \right]}{PV} \tag{A2-4}$$

Water saturation at end of each experiment :

$$S_1 = S_{1i} + \frac{1}{PV} \frac{\left[(\rho_2 \cdot q_1) \cdot t_\infty + (\rho_2 \cdot q_2) \cdot t_\infty - W \right]}{(\rho_1 - \rho_2)} + \frac{\left[DV1 \cdot (IWF1 - FWF1) + DV2 \cdot (IWF2 - FWF2) + q_l \cdot t_\infty \right]}{PV}$$

(A2-5)

Appendix B - 1 TRANSCAL.CTL

Pressure transducer calibration numbeis control file for TRANCALI.FRM.
Transducer #1 gain :1.0107
Transducer #2 gain :0.2474
Transducer #3 gain :0.2442
Transducer #4 gain :0.5431
Transducer #5 gain :0.2660
Transducer #6 gain :0.2688
Transducer #7 gain :0.3394
Transducer #8 gain :0.3028
Transducer #9 gain :0.2640
Transducer #10 gain :0.2772
Transducer #11 gain :0.3092
Transducer #12 gain :0.2610
Transducer #13 gain :0.2623
Transducer #14 gain :0.2627
Transducer #15 gain :0.2402
Transducer #16 gain :0.2199
Transducer #1 offset :1.0180
Transducer #2 offset :0.0052
Transducer #3 offset :0.5490
Transducer #4 offset :-1.1857
Transducer #5 offset :0.3219
Transducer #6 offset :-0.5115
Transducer #7 offset :0.2369
Transducer #8 offset :-1.1387
Transducer #9 offset :0.2110
Transducer #10 offset :-0.6665
Transducer #11 offset :-0.2865
Transducer #12 offset :0.4718
Transducer #13 offset :-0.6030
Transducer #14 offset :-0.2104
Transducer #15 offset :0.7780
Transducer #16 offset :-0.3631

Appendix B - 2 SCAN.CTL

Equipment and experiment settings control file for SCAN.FRM.
Frequency meter setting (GHz):28.0
Attenuator setting (dB):0.00
Gunn current (A):1.54
Gunn voltage (V):6.00
Tuner angle (deg.):169
Starting temperature (deg. C):22.3
Number of scans :1
Number of stations :100
Scan interval (sec.):201
Porosity (%):37
Run number :13
Core number :02
Saturation :012

Appendix B - 3 PERM.CTL

Equipment and experiment settings control file for PERM.FRM
Flowing Fluid :Oil
Fluid density @ start temp. (g/cc):0.9400
Fluid viscosity @ start temp. (cp.):4.0000
Starting Temperature (deg. C):23.1
Pump number :2
Pump rate (cc/hr):0.0250
Balance number :2
Porosity (%):37
Recording interval (sec.):1
Number of scans :15
Run number :1
Core number :02

Appendix B - 4 LOGSET.CTL

Equipment and experiment settings control file for LOGSET.FRM.
User :Samuel Chang
File Name :..\data\test4.dat
VR interface address (Hex) :290
Comm2 Balance #1 setup :2400,e,7,1
Comm4 Balance #2 setup :2400,e,7,1
Motor controller address (Hex) :200
432C controller address (Hex) :300
436A controller address (HPIB) :hpib7,13

Appendix B - 5 FLOW.CTL

Equipment and experiment settings control file for FLOW.FRM.
Frequency meter setting (GHz):28.0
Attenuator setting (dB):0.00
Gunn current (A):1.54
Gunn voltage (V):6.00
Tuner angle:169
Water density @ start temp. (g/cc):1.0000
Water viscosity @ start temp. (cp.):1.0000
Oil density @ start temp. (g/cc):0.9400
Oil viscosity @ start temp. (cp.):4.0000
Pump #1 water rate (cc/hr):0.0114
Pump #2 oil rate (cc/hr):0.0082
Starting Temperature (deg. C):24.1
Porosity (%):37
Absolute permeability (md.):1100
Number of stations :100
Number of scans :1
Scan interval (sec.):1200
Run number :2
Core number :2

Appendix B - 6 EQUIP.CTL

Equipment serial numbers control file for EQUIP.FRM.
Pump #1 :UA tag 0219577
Pump #2 :UA tag 0219578
Stepping motor :UA tag 0178521
Stepping motor controller :UA tag 0178520
Balance #1 :UA tag 0216228
Balance #2 :UA tag 0216229
Gunn oscillator :GKA-280 5J12
Power supply :HP KR51302513
Low pass filter :HP 1248
Frequency meter #1 :Microlab EE Dept.
Frequency meter #2 :Microlab MMPE Dept.
Attenuator :HP 1911
Isolator :MIR-28 105
Tuner :Waveline 1083
Directional coupler #1 :HP 1292
Directional coupler #2 :HP 8791
Thermistor sensor #1 :HP 04975
Thermistor sensor #2 :HP 05101
Power sensor :HP 00193
Power meter #1 :UA tag 0113507
Power meter #2 :UA tag 0147603
Power meter #3 :UA tag 0091379
PIO-12 card #1 :PIO-12 91483
PIO-12 card #2 :PIO-12 92753
601-L card :Validyne E1321-1
HP-IB card :HP 3009
Mercury barometer :CPE-250
Heise gauge :UA tag 0102302
Transducer #1 :AP10-42,93034
Transducer #2 :DP15-44,17354
Transducer #3 :DP15-40,65942
Transducer #4 :DP15-40,65945
Transducer #5 :DP15-TL,4533
Transducer #6 :DP15-TL,65183
Transducer #7 :DP15-TL,4538
Transducer #8 :DP15-TL,65184
Transducer #9 :DP15-56,66116
Transducer #10 :DP15-44,23869
Transducer #11 :DP15-TL,1995

Transducer #12 :DP15-56,66113
Transducer #13 :DP15-44,17353
Transducer #14 :DP15-56,66117
Transducer #15 :DP15-TL,65167
Transducer #16 .DP15-40,65940

Appendix B - 7 ABOUT.FRM

```
'********** Software information form **********
'This program is written to display the software information.

   Option Explicit

Sub cmdOK_Click ()

   ' This form is loaded as a modal dialog.  Use the Unload statement
   ' here to unload the form from memory when the user clicks the
   ' OK command button.

   Unload frmAbout

End Sub

Sub Form_Load ()

   ' The form is horizontally and vertically.
   ' centered when loaded.

   Top = Screen.Height / 2 - Height / 2
   Left = Screen.Width / 2 - Width / 2

End Sub
```

Appendix B - 8 BALSET1.FRM

```
'********** Balance # 1 communication setting form **********
' This program is written to take advantage of serial communication
' capability of an RS-232 interface card. The program is written to
' set the communication settings in order to communicate with a
' Mettier Toledo B3002 balance. There are total of four
' ports installed on an IBM-486 machine with high speed communication
' chips.

   Option Explicit

   DefInt A-Z

   ' Temporary configuration settings
   Dim Shared NewPort!
   Dim Shared NewBaud1$
   Dim Shared NewParity1$
   Dim Shared NewData1$
   Dim Shared NewStop1$
   Dim Shared NewShake1

   ' Variables used for form unloading
   Dim Msg As Variant
   Dim Response As Variant

   Dim Cancel As Integer
   Dim Echo As Integer
   Dim OldPort As Integer
   Dim Reopen As Integer
   Dim Port As Integer
   Dim FirstComma As Integer
   Dim SecondComma As Integer
   Dim ThirdComma As Integer
   Dim Baud$
   Dim Parity$
   Dim DBits$
   Dim SBits$
   Dim CommEvent1 As Integer
```

```
    Dim DgDef As Variant
    Dim Title As Variant

Sub Baud12_Click ()

    ' 1200 baud option button
    NewBaud1$ = "1200"

End Sub

Sub Baud24_Click ()

    ' 2400 baud option button
    NewBaud1$ = "2400"

End Sub

Sub Baud3_Click ()

    ' 300 baud option button
    NewBaud1$ = "300"

End Sub

Sub Baud48_Click ()

    ' 4800 baud option button
    NewBaud1$ = "4800"

End Sub

Sub Baud6_Click ()

    ' 600 baud option button
    NewBaud1$ = "600"

End Sub

Sub Baud96_Click ()
```

```
    ' 9600 baud option button
    NewBaud1$ = "9600"

End Sub

Sub BothFlow_Click ()

    ' Both RTS and Xon/Xoff handshaking option button
    NewShake1 = 3

End Sub

Sub CancelButton_Click ()

    Unload frmBalSet1

    frmSetupTest.Command1.Enabled = False
    frmSetupTest.Command3.Enabled = True
    frmSetupTest.Command4.Enabled = True
    frmSetupTest.Command5.Enabled = True

End Sub

Sub ComPort_Click (Index As Integer)

    ' Select new port number using option button index.
    NewPort1 = Index

End Sub

Sub Data7_Click ()

    ' 7 data bits option button
    NewData1$ = "7"

End Sub

Sub Data8_Click ()

    ' 8 data bits option button
```

```
  NewData1$ = "8"

End Sub

Sub EchoOff_Click ()

  ' Echo off option button
  Echo = 0

End Sub

Sub EchoOn_Click ()

  ' Echo on option button
  Echo = True

End Sub

Sub EvenParity_Click ()

  ' Even parity option button
  NewParity1$ = "E"

End Sub

Sub Form_Load ()

  ' The form is horizontally and vertically centered when loaded.
  Top = Screen.Height / 2 - Height / 2
  Left = Screen.Width / 2 - Width / 2

  ' 1. Comm port.
  ' Get current port number from pre-defined MSComm1 control.
  Port = frmMain.MSComm1.CommPort

  ' Set ComPort option button using port number from above, set the value
  ' to true for the specific port.
  frmBalSet1.ComPort(Port).Value = True

  ' Define variable to trap communication error
```

```
CommEvent1 = frmMain.MSComm1.CommEvent

' 2. Baud rate.
' Get current baud from pre-defined MSComm1 control, set the position
' of the first occurrence of 'comma' within the setting string of MSCOMM1.
FirstComma = InStr(frmMain.MSComm1.Settings, ",")

' Return the leftmost 'FirstComma-1' characters of the string
' 'frmMain.MSComm1.Settings'.
Baud$ = Left$(frmMain.MSComm1.Settings, FirstComma - 1)

' Set baud button using baud rate from above, set the value to true
' for the specific baud.
Select Case Val(Baud$)

    Case 300
      frmBalSet1.Baud3.Value = True

    Case 600
      frmBalSet1.Baud6.Value = True

    Case 1200
      frmBalSet1.Baud12.Value = True

    Case 2400
      frmBalSet1.Baud24.Value = True

    Case 4800
      frmBalSet1.Baud48.Value = True

    Case 9600
      frmBalSet1.Baud96.Value = True

End Select

' 3. Parity.
' Get current parity from pre-defined MSComm1 control.
Parity$ = Mid$(frmMain.MSComm1.Settings, FirstComma + 1, 1)

' Set parity option button using parity from above, set the value to true
```

```
' for the specific parity.
Select Case UCase$(Parity$)

    Case "N"
      frmBalSet1.NoParity.Value = True

    Case "E"
      frmBalSet1.EvenParity.Value = True

    Case "O"
      frmBalSet1.OddParity.Value = True

End Select

' 4. Data bit.
' Get data bits from pre-defined MSComm1 control.
SecondComma = FirstComma + 2
DBits$ = Mid$(frmMain.MSComm1.Settings, SecondComma + 1, 1)

' Set data bit button using data bit from above, set the value to true
' for the specific parity.
Select Case Val(DBits$)

    Case 7
      frmBalSet1.Data7.Value = True

    Case 8
      frmBalSet1.Data8.Value = True

End Select

' 5. Stop bit.
' Get stop bits from pre-defined MSComm1 control.
ThirdComma = SecondComma + 2
SBits$ = Mid$(frmMain.MSComm1.Settings, ThirdComma + 1, 1)

' Set stop bit button using stop bit from above, set the value to true
' for the specific stop bit.
Select Case Val(SBits$)
```

```
        Case 1
            frmBalSet1.Stop1.Value = True

        Case 2
            frmBalSet1.Stop2.Value = True

    End Select

    ' 6. Flow control.
    ' Set active flow control option button from pre-defined MSComm1 control.
    Select Case frmMain.MSComm1.Handshaking

        Case 0
            frmBalSet1.NoFlow.Value = True

        Case 1
            frmBalSet1.XonFlow.Value = True

        Case 2
            frmBalSet1.RTSFlow.Value = True

        Case 3
            frmBalSet1.BothFlow.Value = True

    End Select

    ' 7. Echo mode.
    ' Set active echo mode option button from pre-defined MSComm1 control.

    If Echo Then
        frmBalSet1.EchoOn.Value = True
    Else
        frmBalSet1.EchoOff.Value = True
    End If

End Sub

Sub Form_QueryUnload (Cancel As Integer, Unloadmode As Integer)

    ' Close MSComm1 if not closed already.
```

```
    If frmMain.MSComm1.PortOpen = True Then
        frmMain.MSComm1.PortOpen = False
    End If

    ' Put together a message box with all the proper components.
    Title = "Warning"
    Msg = "Do you really want to close the window?"
    DgDef = MB_YESNO + MB_ICONSTOP + MB_DEFBUTTON2

    Cancel = True

    ' Evaluate unload mode and give warning.
    If Unloadmode = 0 Then
        Response = MsgBox(Msg, DgDef, Title)
        If Response = IDYES Then
          Cancel = False
          Unload frmBalTest1
        Else
          Cancel = True
        End If
    Else
          Cancel = False
    End If

    frmSetupTest.Command1.Enabled = False
    frmSetupTest.Command3.Enabled = True
    frmSetupTest.Command4.Enabled = True
    frmSetupTest.Command5.Enabled = True

End Sub

Sub NoFlow_Click ()

    ' No handshaking option button
    NewShake1 = 0

End Sub

Sub NoParity_Click ()
```

```
    ' No parity option button
    NewParity1$ = "N"

End Sub

Sub OddParity_Click ()

    ' Odd parity option button
    NewParity1$ = "O"

End Sub

Sub OkButton_Click ()

    Msg = "Save Data before closing?"
    Response = MsgBox(Msg, 3 + 16, "Save Dialog")

    Select Case Response

        ' Do not save communication settings.
        ' Do not exit from Communication settings screen.
        Case 2
            Cancel = -1
            Msg = "Command has been canceled."
            Exit Sub

        ' Save communication settings to MSComm1.
        ' Exit from communication settings screen.
        Case 6
        ' OK button actions
        On Error Resume Next

        OldPort = frmMain.MSComm1.CommPort
        ' If port num changing, close the old port
        If NewPort1 <> OldPort Then
            If frmMain.MSComm1.PortOpen Then
                frmMain.MSComm1.PortOpen = False
                Reopen = True
            End If
```

```
        ' Set new port number
        frmMain.MSComm1.CommPort = NewPort1
        If Err = 0 Then
            If Reopen Then
                frmMain.MSComm1.PortOpen = True
            End If
        End If

        If Err Then
            MsgBox Error$, 48
            frmMain.MSComm1.CommPort = OldPort
            Exit Sub
        End If
    End If

    frmMain.MSComm1.Settings = NewBaud1$ + "," + NewParity1$ + "," +
    NewData1$ + "," + NewStop1$
    If Err Then
        MsgBox Error$, 48
        Exit Sub
    End If

    frmMain.MSComm1.Handshaking = NewShake1
    If Err Then
        MsgBox Error$, 48
        Exit Sub
    End If

    Msg = "Data saved.'"

    ' Do not save communication settings.
    ' Exit from communication settings screen.
    Case 7
        Msg = "Data not saved."
End Select

' Display message according to action chosen.
MsgBox Msg, 0, "Confirm"

' OK button action
```

```
    Unload frmBalSet1
    frmSetupTest.Command1.Enabled = False
    frmSetupTest.Command3.Enabled = True
    frmSetupTest.Command4.Enabled = True
    frmSetupTest.Command5.Enabled = True

End Sub

Sub RTSFlow_Click ()

    ' RTS handshaking option button
    NewShake1 = 2

End Sub

Sub Stop1_Click ()

    ' 1 stop bit option button
    NewStop1$ = "1"

End Sub

Sub Stop2_Click ()

    ' 2 stop bits option button
    NewStop1$ = "2"

End Sub

Sub XonFlow_Click ()

    ' XON handshaking option button
    NewShake1 = 1

End Sub
```

Appendix B - 9 BALSET2.FRM

```
'********** Balance # 2 communication setting form **********
' This program is written to take advantage of serial communication
' capability of an RS-232 interface card. The program is written to
' set the communication settings in order to communicate with a
' Mettler Toledo B3002 balance. There are total of four
' ports installed on an IBM-486 machine with high speed communication
' chips.

    Option Explicit

    DefInt A-Z

    ' Temporary configuration settings
    Dim Shared NewPort2
    Dim Shared NewBaud2$
    Dim Shared NewParity2$
    Dim Shared NewData2$
    Dim Shared NewStop2$
    Dim Shared NewShake2

    ' Variables used for form unloading
    Dim Msg As Variant
    Dim Response As Variant

    Dim Cancel As Integer
    Dim Echo As Integer
    Dim OldPort As Integer
    Dim Reopen As Integer
    Dim Port As Integer
    Dim FirstComma As Integer
    Dim SecondComma As Integer
    Dim ThirdComma As Integer
    Dim Baud$
    Dim Parity$
    Dim DBits$
    Dim SBits$
    Dim CommEvent2 As Integer

    Dim DgDef As Variant
```

```
   Dim Title As Variant

Sub Baud12_Click ()

   ' 1200 baud option button
   NewBaud2$ = "1200"

End Sub

Sub Baud24_Click ()

   ' 2400 baud option button
   NewBaud2$ = "2400"

End Sub

Sub Baud3_Click ()

   ' 300 baud option button
   NewBaud2$ = "300"

End Sub

Sub Baud48_Click ()

   ' 4800 baud option button
   NewBaud2$ = "4800"

End Sub

Sub Baud6_Click ()

   ' 600 baud option button
   NewBaud2$ = "600"

End Sub

Sub Baud96_Click ()

   ' 9600 baud option button
```

```
    NewBaud2$ = "9600"

End Sub

Sub BothFlow_Click ()

    ' Both RTS and Xon/Xoff handshaking option button
    NewShake2 = 3

End Sub

Sub CancelButton_Click ()

    Unload frmBalSet2

    frmSetupTest.Command1.Enabled = False
    frmSetupTest.Command3.Enabled = True
    frmSetupTest.Command4.Enabled = True
    frmSetupTest.Command5.Enabled = True

End Sub

Sub ComPort_Click (Index As Integer)

    ' Select new port number using option button index.
    NewPort2 = Index

End Sub

Sub Data7_Click ()

    ' 7 data bits option button
    NewData2$ = "7"

End Sub

Sub Data8_Click ()

    ' 8 data bits option button
    NewData2$ = "8"
```

```
End Sub

Sub EchoOff_Click ()

    ' Echo off option button
    Echo = 0

End Sub

Sub EchoOn_Click ()

    ' Echo on option button
    Echo = True

End Sub

Sub EvenParity_Click ()

    ' Even parity option button
    NewParity2$ = "E"

End Sub

Sub Form_Load ()

    ' The form is horizontally and vertically centered when loaded.
    Top = Screen.Height / 2 - Height / 2
    Left = Screen.Width / 2 - Width / 2

    ' 1. Comm port.
    ' Get current port number from  MSComm2 control.
    Port = frmMain.MSComm2.CommPort

    ' Set ComPort option button using port number from above, set the value
    ' to true for the specific port.
    frmBalSet2.ComPort(Port).Value = True

    ' Define variable to trap communication error
    CommEvent2 = frmMain.MSComm2.CommEvent
```

```
' 2. Baud rate.
' Get current baud from  MSComm2 control, set the position
' of the first occurrence of 'comma' within the setting string of MSCOMM2.
FirstComma = InStr(frmMain.MSComm2.Settings, ",")

' Return the leftmost 'FirstComma-1' characters of the string
' 'frmMain.MSComm2.Settings'.
Baud$ = Left$(frmMain.MSComm2.Settings, FirstComma - 1)

' Set baud button using baud rate from above, set the value to true
' for the specific baud.
Select Case Val(Baud$)

    Case 300
      frmBalSet2.Baud3.Value = True

    Case 600
      frmBalSet2.Baud6.Value = True

    Case 1200
      frmBalSet2.Baud12.Value = True

    Case 2400
      frmBalSet2.Baud24.Value = True

    Case 4800
      frmBalSet2.Baud48.Value = True

    Case 9600
      frmBalSet2.Baud96.Value = True

End Select

' 3. Parity.
' Get current parity from pre-defined MSComm2 control.
Parity$ = Mid$(frmMain.MSComm2.Settings, FirstComma + 1, 1)

' Set parity option button using parity from above, set the value to true
 for the specific parity.
```

```
Select Case UCase$(Parity$)

    Case "N"
      frmBalSet2.NoParity.Value = True

    Case "E"
      frmBalSet2.EvenParity.Value = True

    Case "O"
      frmBalSet2.OddParity.Value = True

End Select

' 4. Data bit.
' Get data bits from pre-defined MSComm2 control.
SecondComma = FirstComma + 2
DBits$ = Mid$(frmMain.MSComm2.Settings, SecondComma + 1, 1)

' Set data bit button using data bit from above, set the value to true
' for the specific parity.
Select Case Val(DBits$)

    Case 7
      frmBalSet2.Data7.Value = True

    Case 8
      frmBalSet2.Data8.Value = True

End Select

' 5. Stop bit.
' Get stop bits from pre-defined MSComm2 control.
ThirdComma = SecondComma + 2
SBits$ = Mid$(frmMain.MSComm2.Settings, ThirdComma + 1, 1)

' Set stop bit button using stop bit from above, set the value to true
' for the specific stop bit.
Select Case Val(SBits$)

    Case 1
```

```
            frmBalSet2.Stop1.Value = True

        Case 2
            frmBalSet2.Stop2.Value = True

    End Select

    ' 6. Flow control.
    ' Set active flow control option button from pre-defined MSComm2 control.
    Select Case frmMain.MSComm2.Handshaking

        Case 0
            frmBalSet2.NoFlow.Value = True

        Case 1
            frmBalSet2.XonFlow.Value = True

        Case 2
            frmBalSet2.RTSFlow.Value = True

        Case 3
            frmBalSet2.BothFlow.Value = True

    End Select

    ' 7. Echo mode.
    ' Set active echo mode option button from pre-defined MSComm2 control.

    If Echo Then
        frmBalSet2.EchoOn.Value = True
    Else
        frmBalSet2.EchoOff.Value = True
    End If

End Sub

Sub Form_QueryUnload (Cancel As Integer, Unloadmode As Integer)

    ' Close MSComm2 if not closed already.
    If frmMain.MSComm2.PortOpen = True Then
```

```
        frmMain.MSComm2.PortOpen = False
    End If

    ' Put together a message box with all the proper components.
    Title = "Warning"
    Msg = "Do you really want to close the window?"
    DgDef = MB_YESNO + MB_ICONSTOP + MB_DEFBUTTON2

    Cancel = True

    ' Evaluate unload mode and give warning.
    If Unloadmode = 0 Then
        Response = MsgBox(Msg, DgDef, Title)
        If Response = IDYES Then
          Cancel = False
          Unload frmBalTest2
        Else
          Cancel = True
        End If
    Else
          Cancel = False
    End If

    frmSetupTest.Command1.Enabled = False
    frmSetupTest.Command3.Enabled = True
    frmSetupTest.Command4.Enabled = True
    frmSetupTest.Command5.Enabled = True

End Sub

Sub NoFlow_Click ()

    ' No handshaking option button
    NewShake2 = 0

End Sub

Sub NoParity_Click ()

    ' No parity option button
```

```
    NewParity2$ = "N"

End Sub

Sub OddParity_Click ()

    ' Odd parity option button
    NewParity2$ = "O"

End Sub

Sub OkButton_Click ()

    Msg = "Save Data before closing?"
    Response = MsgBox(Msg, 3 + 16, "Save Dialog")

    Select Case Response

        ' Do not save communication settings.
        ' Do not exit from Communication settings screen.
        Case 2
            Cancel = -1
            Msg = "Command has been canceled."
            Exit Sub

        ' Save communication settings to MSComm2.
        ' Exit from communication settings screen.
        Case 6
        ' OK button actions
        On Error Resume Next

        OldPort = frmMain.MSComm2.CommPort
        ' If port num changing, close the old port
        If NewPort2 <> OldPort Then
            If frmMain.MSComm2.PortOpen Then
                frmMain.MSComm2.PortOpen = False
                Reopen = True
            End If

            ' Set new port number
```

```
        frmMain.MSComm2.CommPort = NewPort2
        If Err = 0 Then
            If Reopen Then
                frmMain.MSComm2.PortOpen = True
            End If
        End If

        If Err Then
            MsgBox Error$, 48
            frmMain.MSComm2.CommPort = OldPort
            Exit Sub
        End If
    End If

    frmMain.MSComm2.Settings = NewBaud2$ + "," + NewParity2$ + "," +
    NewData2$ + "," + NewStop2$
    If Err Then
        MsgBox Error$, 48
        Exit Sub
    End If

    frmMain.MSComm2.Handshaking = NewShake2
    If Err Then
        MsgBox Error$, 48
        Exit Sub
    End If

    Msg = "Data saved."'

    ' Do not save communication settings.
    ' Exit from communication settings screen.
    Case 7
        Msg = "Data not saved."
End Select

' Display message according to action chosen.
MsgBox Msg, 0, "Confirm"

' OK buttons actions
Unload frmBalSet2
```

```
    frmSetupTest.Command1.Enabled = False
    frmSetupTest.Command3.Enabled = True
    frmSetupTest.Command4.Enabled = True
    frmSetupTest.Command5.Enabled = True

End Sub

Sub RTSFlow_Click ()

    ' RTS handshaking option button
    NewShake2 = 2

End Sub

Sub Stop1_Click ()

    ' 1 stop bit option button
    NewStop2$ = "1"

End Sub

Sub Stop2_Click ()

    ' 2 stop bits option button
    NewStop2$ = "2"

End Sub

Sub XonFlow_Click ()

    ' XON handshaking option button
    NewShake2 = 1

End Sub
```

Appendix B - 10 BALTEST1.FRM

```
'********** Balance #1 testing program **********
' This program is written to test the operation of a Mettler Toledo B3002
' balance using RS-232 communication.

    Option Explicit

    Dim Answer1 As Single
    Dim DefVal1 As String
    Dim Msg1 As String
    Dim Title1 As String
    Dim Duration As Single

    Dim Answer2 As Single
    Dim DefVal2 As String
    Dim Msg2 As String
    Dim Title2 As String

    Dim Answer3 As Variant
    Dim DefVal3 As String
    Dim DefVal4$
    Dim Msg3 As String
    Dim Title3 As String

    Dim DgDef As Variant
    Dim Msg As String
    Dim Response As Variant
    Dim Title As Variant

    Dim Instruction$
    Dim Crlfpos As Integer
    Dim Inputdata As String
    Dim OutputCmd As String
```

Sub cmdFinish_Click ()

```
    ' Close MSComm1 if not closed already.
    If frmMain.MSComm1.PortOpen = True Then
        frmMain.MSComm1.PortOpen = False
    End If
```

```
    ' Release frmBalTest1 from memory.
    Unload frmBalTest1

    ' Activate command and option buttons in frmSetupTest.
    frmSetupTest.Command2.Enabled = False
    frmSetupTest.Command3.Enabled = True
    frmSetupTest.Command4.Enabled = True
    frmSetupTest.Command5.Enabled = True

End Sub

Sub cmdSend_Click ()

    ' Open MSComm1 if not opened already.
    If frmMain.MSComm1.PortOpen = False Then
        frmMain.MSComm1.PortOpen = True
    End If

    ' Set up an error handler within this subroutine that will get
    ' called if a communication error occurs.
    On Error GoTo ErrorHandler

    ' Clear the transmitter buffer of MSComm1.
    frmMain.MSComm1.OutBufferCount = 0

    ' Clear the receive buffer of MSComm1.
    frmMain.MSComm1.InBufferCount = 0

    ' Set and return the number of characters the Input property reads from
    ' the receive buffer.
    ' Setting InputLen to 0 causes the communications control to read the
    ' entire contents of the receive buffer when Input is used.
    frmMain.MSComm1.InputLen = 0

    ' Enable the Data Terminal Ready line during communications.  The
    ' Data Terminal Ready signal is sent by computer to device to indicate
    ' that the computer is ready to accept incoming transmission.
    frmMain.MSComm1.DTREnable = True
```

```
' Enable Request To Send line on MSComm1. Request permission
' to transmit data from computer to device.
' Set Request To Send line to high for RTS/CTS hardware handshaking.
frmMain.MSComm1.RTSEnable = True

' Determine whether data can be sent by querying the state of the
' Clear To Send line. The Clear to Send signal is send from the
' device to computer to indicate that transmission can proceed.
If frmMain.MSComm1.CTSHolding = True Then
' Determine the state of the Data Set Ready line. Data Set Ready
' signal is sent by device to computer to indicate that it is
' ready to operate.
    If frmMain.MSComm1.DSRHolding = True Then
      ' Send out command along with CR and LF to device when both
      ' CTSHolding and DSRHolding are true(high).
      ' Then reset Output buffer to zero.
      OutputCmd = Instruction$ & Chr$(13) & Chr$(10)
      frmMain.MSComm1.Output = OutputCmd
    Else
      ' If DSRHolding is not set to high by device, relinquish time
      ' to other processing until DSRHolding is set to high by device.
      ' Then send out command along with CR and LF to device.
      Do
          'DoEvents
      Loop Until frmMain.MSComm1.DSRHolding = True
      frmMain.MSComm1.Output = Instruction$ & Chr$(13) & Chr$(10)
    End If
End If

' Disabled command button Send.
cmdSend.Enabled = False

' Determine the state of the Data Set Ready line. Data Set Readv
' signal is sent by device to computer to indicate that it is
' ready to operate. Relinquish time to other processing until
' device is ready to send data.
Do
    'DoEvents
Loop Until frmMain.MSComm1.DSRHolding = True
```

```
    ' Relinquish time to other processing and wait for response to come
    ' back from device.
    Do
        'DoEvents
    Loop Until frmMain.MSComm1.InBufferCount >= 0

    ' Delay timer for 1 second to handle timing problem.
    Duration! = Timer + 1
    Do Until Timer > Duration!
        'DoEvents
    Loop

    ' Read data out from the buffer.
    Inputdata = frmMain.MSComm1.Input
    ' Check for LF at the end of response.
    If InStr(Inputdata, Chr$(10)) >= 0 Then
        ' Set string pointer to LF character.
        Crlfpos = InStr(Inputdata, Chr$(10))
        ' Strip out LF from response string.
        Inputdata = Left$(Inputdata, Crlfpos - 1)
        ' Set string pointer to CR character.
        Crlfpos = InStr(Inputdata, Chr$(13))
        ' Strip out CR from response string.
        Inputdata = Left(Inputdata, Crlfpos - 1)
        ' Display response.
        txtBININ.Text = Inputdata
    Else
        txtBININ.Text = "Response not valid"
    End If

    ' Enable cmd Send.
    cmdSend.Enabled = True

Exit Sub
```

ErrorHandler:

```
    ' Display the error message in the txtBININ textbox
    txtBININ.Text = "Error : " + Error$
```

```
' Close the comm port if not already closed.
If frmMain.MSComm1.PortOpen = True Then
    frmMain.MSComm1.PortOpen = False
End If

' Enable the button used to initiate I/O.
cmdSend.Enabled = True

Exit Sub
```

End Sub

Sub Form_Load ()

```
' The form is horizontally and vertically centered when loaded.
Top = Screen.Height / 2 - Height / 2
Left = Screen.Width / 2 - Width / 2

' Add items to List1 for selection.
List1.AddItem "Enter a text string to balance."
List1.AddItem "Clear display on balance."
List1.AddItem "Switch balance display to weight mode."
List1.AddItem "Request type of the implemented command set."
List1.AddItem "Request balance identification."
List1.AddItem "Request software version and type."
List1.AddItem "Request serial number of balance."
List1.AddItem "Enable key function with transmission of key code."
List1.AddItem "Enable key function without transmission of key code."
List1.AddItem "Disable key function with transmission of key code."
List1.AddItem "Disable key function without transmission of key code."
List1.AddItem "Send the current net weight value when stable."
List1.AddItem "Send the current net weight value immediately."
List1.AddItem "Send the current net weight values repeatedly."
List1.AddItem "Send the current stable weight value on weight change."
List1.AddItem "Store the next stable weight value as a new tare weight."
List1.AddItem "Set tare weight value when stable."
List1.AddItem "Clear tare weight value."
List1.AddItem "Store the current weight value as a new tare weight."
List1.AddItem "Zero the balance."
List1.AddItem "Reset balance, but without a zero setting being performed."
```

```
End Sub

Sub Form_QueryUnload (Cancel As Integer, Unloadmode As Integer)

    ' Close MSComm1 if not closed already.
    If frmMain.MSComm1.PortOpen = True Then
        frmMain.MSComm1.PortOpen = False
    End If

    ' Put together a message box with all the proper components.
    Title = "Warning"
    Msg = "Do you really want to close the window?"
    DgDef = MB_YESNO + MB_ICONSTOP + MB_DEFBUTTON2

    Cancel = True

    ' Evaluate unload mode and give warning.
    If Unloadmode = 0 Then
        Response = MsgBox(Msg, DgDef, Title)
        If Response = IDYES Then
          Cancel = False
          Unload frmBalTest1
        Else
          Cancel = True
        End If
    Else
          Cancel = False
    End If

    frmSetupTest.Command2.Enabled = False
    frmSetupTest.Command3.Enabled = True
    frmSetupTest.Command4.Enabled = True
    frmSetupTest.Command5.Enabled = True

End Sub

Sub List1_Click ()

    If List1.Text = "Clear display on balance." Then
```

```
    Instruction$ = "D C"
End If

If List1.Text = "Switch balance display to weight mode." Then
    Instruction$ = "D W"
End If

If List1.Text = "Request type of the implemented command set." Then
    Instruction$ = "I1"
End If

If List1.Text = "Request balance identification." Then
    Instruction$ = "I2"
End If

If List1.Text = "Request software version and type." Then
    Instruction$ = "I3"
End If

If List1.Text = "Request serial number of balance." Then
    Instruction$ = "I4"
End If

If List1.Text = "Enable key function with transmission of key code." Then
    Instruction$ = "K EE"
End If

If List1.Text = "Enable key function without transmission of key code." Then
    Instruction$ = "K ED"
End If

If List1.Text = "Disable key function with transmission of key code." Then
    Instruction$ = "K DE"
End If

If List1.Text = "Disable key function without transmission of key code." Then
    Instruction$ = "K DD"
End If

If List1.Text = "Send the current net weight value when stable." Then
```

```
        Instruction$ = "S"
    End If

    If List1.Text = "Send the current net weight value immediately." Then
        Instruction$ = "SI"
    End If

    If List1.Text = "Send the current net weight values repeatedly." Then
        Instruction$ = "SIR"
    End If

    If List1.Text = "Send the current stable weight value on weight change." Then
        ' Set prompt.
        Msg1 = "Enter a preset value in grams greater than 3.0 grams."
        ' Set title.
        Title1 = "Preset value input"
        ' Set default return value.
        DefVal1 = "3.0"
        Do
          ' Get user input.
          Answer1 = Val(InputBox(Msg1, Title1, DefVal1))
        Loop Until Answer1 >= 3# And Answer1 <= 3000#
        ' Display message.
        MsgBox "You entered " & Answer1 & "grams as preset load change."
        Instruction$ = "SR " & Answer1 & " g"
    End If

    If List1.Text = "Store the next stable weight value as a new tare weight." Then
        Instruction$ = "T"
    End If

    If List1.Text = "Set tare weight value when stable." Then
        ' Set prompt.
        Msg2 = "Enter a preset value in grams."
        ' Set title.
        Title2 = "Preset tare value input"
        ' Set default return value.
        DefVal2 = "0.0"
        Do
          ' Get user input.
```

```
        Answer2 = Val(InputBox(Msg2, Title2, DefVal2))
      Loop Until Answer2 >= 0# And Answer2 <= 6000#
      ' Display message.
      MsgBox "You entered " & Answer2 & "grams as preset tare value."
      Instruction$ = "TA " & Answer2 & " g"
  End If

  If List1.Text = "Clear tare weight value." Then
      Instruction$ = "TA C"
  End If

  If List1.Text = "Store the current weight value as a new tare weight." Then
      Instruction$ = "TI"
  End If

  If List1.Text = "Zero the balance." Then
      Instruction$ = "Z"
  End If

   If List1.Text = "Reset balance, but without a zero setting being performed."
Then
      Instruction$ = "@"
  End If

  If List1.Text = "Enter a text string to balance." Then
      ' Set prompt.
      Msg3 = "Enter a text string to be displayed in upper case only and less then
18
      characters."
      ' Set title.
      Title3 = "Message display"
      ' Set default return value.
      DefVal3 = "Bal #1"
      DefVal4$ = UCase$(DefVal3)
      Do
        ' Get user input.
        Answer3 = InputBox(Msg3, Title3, DefVal4$)
      Loop Until Len(Answer3) >= 0 And Len(Answer3) <= 10
      ' Display message.
```

```
        MsgBox "You entered message '" & UCase$(Answer3) & "'to be
displayed."
        Instruction$ = "D T " & Answer3
    End If
```

End Sub

Sub List1_DblClick ()

```
    cmdSend_Click
```

End Sub

Appendix B - 11 BALTEST2.FRM

```
'********** Balance #2 testing program **********
' This program is written to test the operation of a Mettler Toledo B3002
' balance using RS-232 communication.

    Option Explicit

    Dim Answer1 As Single
    Dim DefVal1 As String
    Dim Msg1 As String
    Dim Title1 As String
    Dim Duration!

    Dim Answer2 As Single
    Dim DefVal2 As String
    Dim Msg2 As String
    Dim Title2 As String

    Dim Answer3 As Variant
    Dim DefVal3 As String
    Dim DefVal4$
    Dim Msg3 As String
    Dim Title3 As String

    Dim DgDef As Variant
    Dim Msg As String
    Dim Response As Variant
    Dim Title As Variant

    Dim Instruction$
    Dim Crlfpos As Integer
    Dim Inputdata As String
    Dim OutputCmd As String
```

Sub cmdFinish_Click ()

```
    ' Close MSComm2 if not closed already.
    If frmMain.MSComm2.PortOpen = True Then
        frmMain.MSComm2.PortOpen = False
    End If
```

```
    ' Release frmBalTest2 from memory.
    Unload frmBalTest2

    ' Activate command and option buttons in frmSetupTest.
    frmSetupTest.Command2.Enabled = False
    frmSetupTest.Command3.Enabled = True
    frmSetupTest.Command4.Enabled = True
    frmSetupTest.Command5.Enabled = True

End Sub

Sub cmdSend_Click ()

    ' Open MSComm2 if not opened already.
    If frmMain.MSComm2.PortOpen = False Then
        frmMain.MSComm2.PortOpen = True
    End If

    ' Set up an error handler within this subroutine that will get
    ' called if a communication error occurs.
    On Error GoTo ErrorHandler

    ' Clear the transmitter buffer of MSComm2.
    frmMain.MSComm2.OutBufferCount = 0

    ' Clear the receive buffer of MSComm2.
    frmMain.MSComm2.InBufferCount = 0

    ' Set and return the number of characters the Input property reads from
    ' the receive buffer.
    ' Setting InputLen to 0 causes the communications control to read the
    ' entire contents of the receive buffer when Input is used.
    frmMain.MSComm2.InputLen = 0

    ' Enable the Data Terminal Ready line during communications.  The
    ' Data Terminal Ready signal is sent by computer to device to indicate
    ' that the computer is ready to accept incoming transmission.
    frmMain.MSComm2.DTREnable = True

    ' Enable Request To Send line on MSComm2.  Request permission
```

```
' to transmit data from computer to device.
' Set Request To Send line to high for RTS/CTS hardware handshaking.
frmMain.MSComm2.RTSEnable = True

' Determine whether data can be sent by querying the state of the
' Clear To Send line. The Clear to Send signal is send from the
' device to computer to indicate that transmission can proceed.
If frmMain.MSComm2.CTSHolding = True Then
' Determine the state of the Data Set Ready line. Data Set Ready
' signal is sent by device to computer to indicate that it is
' ready to operate.
    If frmMain.MSComm2.DSRHolding = True Then
      ' Send out command along with CR and LF to device when both
      ' CTSHolding and DSRHolding are true(high).
      OutputCmd = Instruction$ & Chr$(13) & Chr$(10)
      frmMain.MSComm2.Output = OutputCmd
    Else
      ' If DSRHolding is not set to high by device, relinquish time
      ' to other processing until DSRHolding is set to high by device.
      ' Then send out command along with CR and LF to device.
      Do
          'DoEvents
      Loop Until frmMain.MSComm2.DSRHolding = True
      frmMain.MSComm2.Output = Instruction$ & Chr$(13) & Chr$(10)
    End If
End If

' Disabled command button Send.
cmdSend.Enabled = False

' Determine the state of the Data Set Ready line. Data Set Ready
' signal is sent by device to computer to indicate that it is
' ready to operate. Relinquish time to other processing until
' device is ready to send data.
Do
    'DoEvents
Loop Until frmMain.MSComm2.DSRHolding = True

' Relinquish time to other processing and wait for response to come
' back from device.
```

```
Do
    'DoEvents
Loop Until frmMain.MSComm2.InBufferCount >= 0

' Delay timer to handle timing problem.
Duration! = Timer + 1
Do Until Timer > Duration!
    'DoEvents
Loop

' Read data out from the buffer.
Inputdata = frmMain.MSComm2.Input
' Check for LF at the end of response.
If InStr(Inputdata, Chr$(10)) >= 0 Then
    ' Set string pointer to LF character.
    Crlfpos = InStr(Inputdata, Chr$(10))
    ' Strip out LF from response string.
    Inputdata = Left$(Inputdata, Crlfpos - 1)
    ' Set string pointer to CR character.
    Crlfpos = InStr(Inputdata, Chr$(13))
    ' Strip out CR from response string.
    Inputdata = Left(Inputdata, Crlfpos - 1)
    ' Display response.
    txtBININ.Text = Inputdata
Else
    txtBININ.Text = "Response not valid"
End If

' Enable cmd Send.
cmdSend.Enabled = True

Exit Sub
```

ErrorHandler:

```
' Display the error message in the txtBININ textbox
txtBININ.Text = "Error : " + Error$

' Close the comm port if not already closed.
If frmMain.MSComm2.PortOpen = True Then
```

```
        frmMain.MSComm2.PortOpen = False
    End If

    ' Enable the button used to initiate I/O.
    cmdSend.Enabled = True

    Exit Sub

End Sub

Sub Form_Load ()

    ' The form is horizontally and vertically centered when loaded.
    Top = Screen.Height / 2 - Height / 2
    Left = Screen.Width / 2 - Width / 2

    ' Add items to List1 for selection.
    List1.AddItem "Enter a text string to balance."
    List1.AddItem "Clear display on balance."
    List1.AddItem "Switch balance display to weight mode."
    List1.AddItem "Request type of the implemented command set."
    List1.AddItem "Request balance identification."
    List1.AddItem "Request software version and type."
    List1.AddItem "Request serial number of balance."
    List1.AddItem "Enable key function with transmission of key code."
    List1.AddItem "Enable key function without transmission of key code."
    List1.AddItem "Disable key function with transmission of key code."
    List1.AddItem "Disable key function without transmission of key code."
    List1.AddItem "Send the current net weight value when stable."
    List1.AddItem "Send the current net weight value immediately."
    List1.AddItem "Send the current net weight values repeatedly."
    List1.AddItem "Send the current stable weight value on weight change."
    List1.AddItem "Store the next stable weight value as a new tare weight."
    List1.AddItem "Set tare weight value when stable."
    List1.AddItem "Clear tare weight value."
    List1.AddItem "Store the current weight value as a new tare weight."
    List1.AddItem "Zero the balance."
    List1.AddItem "Reset balance, but without a zero setting being performed."

End Sub
```

```
Sub Form_QueryUnload (Cancel As Integer, Unloadmode As Integer)

    ' Close MSComm2 if not closed already.
    If frmMain.MSComm2.PortOpen = True Then
        frmMain.MSComm2.PortOpen = False
    End If

    ' Put together a message box with all the proper components.
    Title = "Warning"
    Msg = "Do you really want to close the window?"
    DgDef = MB_YESNO + MB_ICONSTOP + MB_DEFBUTTON2

    Cancel = True

    ' Evaluate unload mode and give warning.
    If Unloadmode = 0 Then
        Response = MsgBox(Msg, DgDef, Title)
        If Response = IDYES Then
          Cancel = False
          Unload frmBalTest2
        Else
          Cancel = True
        End If
    Else
          Cancel = False
    End If

    frmSetupTest.Command2.Enabled = False
    frmSetupTest.Command3.Enabled = True
    frmSetupTest.Command4.Enabled = True
    frmSetupTest.Command5.Enabled = True

End Sub

Sub List1_Click ()

    If List1.Text = "Clear display on balance." Then
        Instruction$ = "D C"
    End If
```

```
If List1.Text = "Switch balance display to weight mode." Then
    Instruction$ = "D W"
End If

If List1.Text = "Request type of the implemented command set." Then
    Instruction$ = "I1"
End If

If List1.Text = "Request balance identification." Then
    Instruction$ = "I2"
End If

If List1.Text = "Request software version and type." Then
    Instruction$ = "I3"
End If

If List1.Text = "Request serial number of balance." Then
    Instruction$ = "I4"
End If

If List1.Text = "Enable key function with transmission of key code." Then
    Instruction$ = "K EE"
End If

If List1.Text = "Enable key function without transmission of key code." Then
    Instruction$ = "K ED"
End If

If List1.Text = "Disable key function with transmission of key code." Then
    Instruction$ = "K DE"
End If

If List1.Text = "Disable key function without transmission of key code." Then
    Instruction$ = "K DD"
End If

If List1.Text = "Send the current net weight value when stable." Then
    Instruction$ = "S"
End If
```

```
If List1.Text = "Send the current net weight value immediately." Then
    Instruction$ = "SI"
End If

If List1.Text = "Send the current net weight values repeatedly." Then
    Instruction$ = "SIR"
End If

If List1.Text = "Send the current stable weight value on weight change." Then
    ' Set prompt.
    Msg1 = "Enter a preset value in grams greater than 3.0 grams."
    ' Set title.
    Title1 = "Preset value input"
    ' Set default return value.
    DefVal1 = "3.0"
    Do
      ' Get user input.
      Answer1 = Val(InputBox(Msg1, Title1, DefVal1))
    Loop Until Answer1 >= 3# And Answer1 <= 3000#
    ' Display message.
    MsgBox "You entered " & Answer1 & "grams as preset load change."
    Instruction$ = "SR " & Answer1 & " g"
End If

If List1.Text = "Store the next stable weight value as a new tare weight." Then
    Instruction$ = "T"
End If

If List1.Text = "Set tare weight value when stable." Then
    ' Set prompt.
    Msg2 = "Enter a preset value in grams."
    ' Set title.
    Title2 = "Preset tare value input"
    ' Set default return value.
    DefVal2 = "0.0"
    Do
      ' Get user input.
      Answer2 = Val(InputBox(Msg2, Title2, DefVal2))
    Loop Until Answer2 >= 0# And Answer2 <= 6000#
```

```
    ' Display message.
    MsgBox "You entered " & Answer2 & "grams as preset tare value."
    Instruction$ = "TA " & Answer2 & " g"
End If

If List1.Text = "Clear tare weight value." Then
    Instruction$ = "TA C"
End If

If List1.Text = "Store the current weight value as a new tare weight." Then
    Instruction$ = "TI"
End If

If List1.Text = "Zero the balance." Then
    Instruction$ = "Z"
End If

If List1.Text = "Reset balance, but without a zero setting being performed."
Then
    Instruction$ = "@"
End If

If List1.Text = "Enter a text string to balance." Then
    ' Set prompt.
    Msg3 = "Enter a text string to be displayed in upper case only and less then
    18 characters."
    ' Set title.
    Title3 = "Message display"
    ' Set default return value.
    DefVal3 = "Bal #2"
    DefVal4$ = UCase$(DefVal3)
    Do
      ' Get user input.
      Answer3 = InputBox(Msg3, Title3, DefVal4$)
    Loop Until Len(Answer3) >= 0 And Len(Answer3) <= 10
    ' Display message.
    MsgBox "You entered message '" & UCase$(Answer3) & "'to be
    displayed."
    Instruction$ = "D T " & Answer3
End If
```

```
End Sub

Sub List1_DblClick ()
   cmdSend_Click
End Sub
```

Appendix B - 12 CALC.FRM

```
'********** Calculator subprogram **********
' This program is written to perform basic numerical calculation.

Option Explicit

   ' Previously input operand.
   Dim Op1, Op2
   ' Decimal point present yet?
   Dim DecimalFlag As Integer
   ' Number of operands.
   Dim NumOps As Integer
   ' Indicate type of last keypress.
   Dim LastInput
   ' Indicate pending operation.
   Dim OpFlag
   Dim TempReadout
   Dim Title As String
   Dim Msg As String
   Dim DgDef As Integer
   Dim Response As Integer

' Click event procedure for C (cancel) key.
' Reset the display and initializes variables.
'
Sub Cancel_Click ()

   ReadOut = "0."
   Op1 = 0
   Op2 = 0
   Form_Load

End Sub

' Click event procedure for CE (cancel entry) key.
'
Sub CancelEntry_Click ()

   ReadOut = "0."
   DecimalFlag = False
```

```
  LastInput = "CE"

End Sub

' Click event procedure for decimal point (.) key.
' If last keypress was an operator, initialize
' readout to "0." Otherwise, append a decimal
' point to the display.
'
Sub Decimal_Click ()

   If LastInput = "N[illegible]" [illegible]n
        ReadOut = "-[illegible]
   ElseIf LastInput <> "NUMS" Then
        ReadOut = "0."
   End If
   DecimalFlag = True
   LastInput = "NUMS"

End Sub

Sub Form_Load ()

   ' Initialization routine for the form.
   ' Set all variables to initial values.

   DecimalFlag = False
   NumOps = 0
   LastInput = "NONE"
   OpFlag = " "

   ' The form is horizontally and vertically centered when loaded.
   Top = Screen.Height / 2 - Height / 2
   Left = Screen.Width / 2 - Width / 2

End Sub

Sub Form_QueryUnload (Cancel As Integer, Unloadmode As Integer)

   ' Put together a message box with all the proper components.
```

```
    Title = "Warning"
    Msg = "Do you really want to close the window?"
    DgDef = MB_YESNO + MB_ICONSTOP + MB_DEFBUTTON2

    Cancel = True

    ' Evaluate unload mode and give warning.
    If Unloadmode = 0 Then
        Response = MsgBox(Msg, DgDef, Title)
        If Response = IDYES Then
            Cancel = False
            Unload frmCalculator
        Else
            Cancel = True
        End If
    Else
            Cancel = False
    End If

End Sub

' Click event procedure for number keys (0-9).
' Appends new number to the number in the display.
'
Sub Number_Click (Index As Integer)

    If LastInput <> "NUMS" Then
        ReadOut = "."
        DecimalFlag = False
    End If

    If DecimalFlag Then
        ReadOut = ReadOut + Number(Index).Caption
    Else
        ReadOut = Left(ReadOut, InStr(ReadOut, ".") - 1) +
        Number(Index).Caption + "."
    End If

    If LastInput = "NEG" Then ReadOut = "-" & ReadOut
```

```
    LastInput = "NUMS"

End Sub

' Click event procedure for operator keys (+, -, x, /, =).
' If the immediately preceeding keypress was part of a
' number, increment NumOps. If one operand is present,
' set Op1. If two are present, set Op1 equal to the
' result of the operation on Op1 and the current
' Input string, and display the result.

Sub Operator_Click (Index As Integer)

    TempReadout = ReadOut

    If LastInput = "NUMS" Then
        NumOps = NumOps + 1
    End If

    Select Case NumOps
        Case 0
        If Operator(Index).Caption = "-" And LastInput <> "NEG" Then
          ReadOut = "-" & ReadOut
          LastInput = "NEG"
        End If

        Case 1
        Op1 = ReadOut
        If Operator(Index).Caption = "-" And LastInput <> "NUMS" And OpFlag
        <>      "=" Then
          ReadOut = "-"
          LastInput = "NEG"
        End If

        Case 2
        Op2 = TempReadout
        Select Case OpFlag
          Case "+"
              Op1 = Val(Op1) + Val(Op2)
          Case "-"
```

```
            Op1 = Op1 - Op2
        Case "X"
            Op1 = Op1 * Op2
        Case "/"
            If Op2 = 0 Then
              MsgBox "Can't divide by zero", 48, "Calculator"
            Else
              Op1 = Op1 / Op2
            End If
        Case "="
            Op1 = Op2
        Case "%"
            Op1 = Op1 * Op2
        End Select
      ReadOut = Op1
      NumOps = 1
  End Select

  If LastInput <> "NEG" Then
      LastInput = "OPS"
      OpFlag = Operator(Index).Caption
  End If
```

End Sub

```
' Click event procedure for percent key (%).
' Compute and display a percentage of the first operand.
```

Sub Percent_Click ()

```
  ReadOut = ReadOut / 100
  LastInput = "Ops"
  OpFlag = "%"
  NumOps = NumOps + 1
  DecimalFlag = True
```

End Sub

Appendix B - 13 EQUIP.FRM

```
'**********Equipment number form**********
'This form is written to display serial numbers for equipment.

Option Explicit

   Dim Title As Variant
   Dim Msg As Variant
   Dim DgDef As Variant
   Dim Response As Integer

Sub Form_Load ()

   ' The form is horizontally and vertically centered when loaded.
   Top = Screen.Height / 2 - Height / 2
   Left = Screen.Width / 2 - Width / 2

   Frame1.Enabled = True
   Option1.Enabled = True
   Option2.Enabled = True
   Option3.Enabled = True

   ' Call reset to load text boxes from control file.
   Option1.Value = False
   Option3.Value = False
   Option2.Value = True

End Sub

Sub Form_QueryUnload (Cancel As Integer, Unloadmode As Integer)

   ' Put together a message box with all the proper components.
   Title = "Warning"
   Msg = "Do you really want to close the window?"
   DgDef = MB_YESNO + MB_ICONSTOP + MB_DEFBUTTON2

   Cancel = True
   ' Evaluate unload mode and give warning.
   If Unloadmode = 0 Then
      Response = MsgBox(Msg, DgDef, Title)
      If Response = IDYES Then
```

```
        Cancel = False
        Unload frmEquipmentRecord
    Else
        Cancel = True
    End If
Else
        Cancel = False
End If

frmSetUPTest.Command1.Enabled = True
frmSetUPTest.Command2.Enabled = True
frmSetUPTest.Command3.Enabled = True
frmSetUPTest.Command4.Enabled = True
frmSetUPTest.Command5.Enabled = True
```

End Sub

Sub Option1_Click ()

```
Dim EquipFile As Integer
Dim Response As Integer
Dim Msg As String
Dim Title As String

' Option 1 saves the current text box values to
' the control file.

Msg = "Save these settings?"
Title = "Saving Equipment Usage Record"
Response = MsgBox(Msg, 4 + 16 + 256, Title)
If Response = IDNO Then
    Option1.Value = False
    Exit Sub
End If

EquipFile = FreeFile

Open "Equip.ctl" For Output As EquipFile

Print #EquipFile,
```

```
Print #EquipFile, "Equipment serial numbers control file."
Print #EquipFile, "Pump #1 :"; TxtPump1serial.Text
Print #EquipFile, "Pump #2 :"; TxtPump2Serial.Text
Print #EquipFile, "Stepping motor :"; TxtMotorSerial.Text
Print #EquipFile, "Stepping motor controller :"; TxtMotorControllerSerial.Text
Print #EquipFile, "Balance #1 :"; TxtBalance1Serial.Text
Print #EquipFile, "Balance #2 :"; TxtBalance2Serial.Text
Print #EquipFile, "Gunn oscillator :"; TxtGunnOscillator.Text
Print #EquipFile, "Power supply :"; TxtPowerSupply.Text
Print #EquipFile, "Low pass filter :"; TxtFilterSerial.Text
Print #EquipFile, "Frequency meter #1 :"; TxtFreqMeter1Serial.Text
Print #EquipFile, "Frequency meter #2 :"; TxtFreqMeter2Serial.Text
Print #EquipFile, "Attenuator :"; TxtAttenuatorSerial.Text
Print #EquipFile, "Isolator :"; TxtIsolator.Text
Print #EquipFile, "Tuner :"; TxtTuner.Text
Print #EquipFile, "Directional coupler #1 :"; TxtCoupler1Serial.Text
Print #EquipFile, "Directional coupler #2 :"; TxtCoupler2Serial.Text
Print #EquipFile, "Thermistor sensor #1 :"; TxtThermistor1Serial.Text
Print #EquipFile, "Thermistor sensor #2 :"; TxtThermistor2Serial.Text
Print #EquipFile, "Power sensor :"; TxtPowerSensorSerial.Text
Print #EquipFile, "Power meter #1 :"; TxtPowerMeter1Serial.Text
Print #EquipFile, "Power meter #2 :"; TxtPowerMeter2Serial.Text
Print #EquipFile, "Power meter #3 :"; TxtPowerMeter3Serial.Text
Print #EquipFile, "PIO-12 card #1 :"; TxtPIO1Serial.Text
Print #EquipFile, "PIO-12 card #2 :"; TxtPIO2Serial.Text
Print #EquipFile, "601-L card :"; Txt601LSerial.Text
Print #EquipFile, "HP-IB card :"; TxtHPIBSerial.Text
Print #EquipFile, "Mercury barometer :"; TxtBarometerSerial.Text
Print #EquipFile, "Heise gauge :"; TxtHeiseGaugeSerial.Text
Print #EquipFile, "Transducer #1 :"; TxtTransducer1Serial.Text
Print #EquipFile, "Transducer #2 :"; TxtTransducer2Serial.Text
Print #EquipFile, "Transducer #3 :"; TxtTransducer3Serial.Text
Print #EquipFile, "Transducer #4 :"; TxtTransducer4Serial.Text
Print #EquipFile, "Transducer #5 :"; TxtTransducer5Serial.Text
Print #EquipFile, "Transducer #6 :"; TxtTransducer6Serial.Text
Print #EquipFile, "Transducer #7 :"; TxtTransducer7Serial.Text
Print #EquipFile, "Transducer #8 :"; TxtTransducer8Serial.Text
Print #EquipFile, "Transducer #9 :"; TxtTransducer9Serial.Text
Print #EquipFile, "Transducer #10 :"; TxtTransducer10Serial.Text
Print #EquipFile, "Transducer #11 :"; TxtTransducer11Serial.Text
```

```
    Print #EquipFile, "Transducer #12 :"; TxtTransducer12Serial.Text
    Print #EquipFile, "Transducer #13 :"; TxtTransducer13Serial.Text
    Print #EquipFile, "Transducer #14 :"; TxtTransducer14Serial.Text
    Print #EquipFile, "Transducer #15 :"; TxtTransducer15Serial.Text
    Print #EquipFile, "Transducer #16 :"; TxtTransducer16Serial.Text
    Print #EquipFile,

    Close EquipFile

    Unload frmEquipmentRecord

End Sub

Sub Optio[illegible]k ()

    Dim Equip[illegible] As Integer
    Dim Char As Variant
    ReDim TextData(43) As Variant
    Dim I As Integer

    ' Option to loads the values found in the control
    ' file into the text boxes.

    EquipFile = FreeFile

    Open "Equip.ctl" For Input As EquipFile

    For I = 0 To 43
      Do
        Char = Input(1, EquipFile)
      Loop Until Char = Chr(58)
      TextData(I) = ""
      While Char <> Chr(13)
        Char = Input(1, EquipFile)
        TextData(I) = TextData(I) & Char
      Wend
    Next I

    Close EquipFile
```

```
TxtPump1serial.Text = Left(TextData(0), Len(TextData(0)) - 1)
TxtPump2Serial.Text = Left(TextData(1), Len(TextData(1)) - 1)
TxtMotorSerial.Text = Left(TextData(2), Len(TextData(2)) - 1)
TxtMotorControllerSerial.Text = Left(TextData(3), Len(TextData(3)) - 1)
TxtBalance1Serial.Text = Left(TextData(4), Len(TextData(4)) - 1)
TxtBalance2Serial.Text = Left(TextData(5), Len(TextData(5)) - 1)
TxtGunnOscillator.Text = Left(TextData(6), Len(TextData(6)) - 1)
TxtPowerSupply.Text = Left(TextData(7), Len(TextData(7)) - 1)
TxtFilterSerial.Text = Left(TextData(8), Len(TextData(8)) - 1)
TxtFreqMeter1Serial.Text = Left(TextData(9), Len(TextData(9)) - 1)
TxtFreqMeter2Serial.Text = Left(TextData(10), Len(TextData(10)) - 1)
TxtAttenuatorSerial.Text = Left(TextData(11), Len(TextData(11)) - 1)
TxtIsolator.Text = Left(TextData(12), Len(TextData(12)) - 1)
TxtTuner.Text = Left(TextData(13), Len(TextData(13)) - 1)
TxtCoupler1Serial.Text = Left(TextData(14), Len(TextData(14)) - 1)
TxtCoupler2Serial.Text = Left(TextData(15), Len(TextData(15)) - 1)
TxtThermistor1Serial.Text = Left(TextData(16), Len(TextData(16)) - 1)
TxtThermistor2Serial.Text = Left(TextData(17), Len(TextData(17)) - 1)
TxtPowerSensorSerial.Text = Left(TextData(18), Len(TextData(18)) - 1)
TxtPowerMeter1Serial.Text = Left(TextData(19), Len(TextData(19)) - 1)
TxtPowerMeter2Serial.Text = Left(TextData(20), Len(TextData(20)) - 1)
TxtPowerMeter3Serial.Text = Left(TextData(21), Len(TextData(21)) - 1)
TxtPIO1Serial.Text = Left(TextData(22), Len(TextData(22)) - 1)
TxtPIO2Serial.Text = Left(TextData(23), Len(TextData(23)) - 1)
Txt601LSerial.Text = Left(TextData(24), Len(TextData(24)) - 1)
TxtHPIBSerial.Text = Left(TextData(25), Len(TextData(25)) - 1)
TxtBarometerSerial.Text = Left(TextData(26), Len(TextData(26)) - 1)
TxtHeiseGaugeSerial.Text = Left(TextData(27), Len(TextData(27)) - 1)
TxtTransducer1Serial.Text = Left(TextData(28), Len(TextData(28)) - 1)
TxtTransducer2Serial.Text = Left(TextData(29), Len(TextData(29)) - 1)
TxtTransducer3Serial.Text = Left(TextData(30), Len(TextData(30)) - 1)
TxtTransducer4Serial.Text = Left(TextData(31), Len(TextData(31)) - 1)
TxtTransducer5Serial.Text = Left(TextData(32), Len(TextData(32)) - 1)
TxtTransducer6Serial.Text = Left(TextData(33), Len(TextData(33)) - 1)
TxtTransducer7Serial.Text = Left(TextData(34), Len(TextData(34)) - 1)
TxtTransducer8Serial.Text = Left(TextData(35), Len(TextData(35)) - 1)
TxtTransducer9Serial.Text = Left(TextData(36), Len(TextData(36)) - 1)
TxtTransducer10Serial.Text = Left(TextData(37), Len(TextData(37)) - 1)
TxtTransducer11Serial.Text = Left(TextData(38), Len(TextData(38)) - 1)
TxtTransducer12Serial.Text = Left(TextData(39), Len(TextData(39)) - 1)
```

```
    TxtTransducer13Serial.Text = Left(TextData(40), Len(TextData(40)) - 1)
    TxtTransducer14Serial.Text = Left(TextData(41), Len(TextData(41)) - 1)
    TxtTransducer15Serial.Text = Left(TextData(42), Len(TextData(42)) - 1)
    TxtTransducer16Serial.Text = Left(TextData(43), Len(TextData(43)) - 1)

    Option2.Value = False

End Sub

Sub Option3_Click ()

    ' Option 3 resets the text boxes and unloads the form.

    Option2.Value = True

    Unload frmEquipmentRecord

End Sub
```

Appendix B - 14 FLOW.FRM

```
'**********Flowing experiment control form**********
'This form is written to control data collection for steady state and unsteady state
'experiments.

Option Explicit

    Dim Alarmtime As Variant
    Dim Delaytime As Variant
    Dim Starttime As Variant
    Dim EstStoptime As Variant
    Dim Alarmsounded As Integer

    Dim OldBox As Control
    Dim NewBox As Control
    Dim OldText As String

Sub CmdNo_Click ()

    GraphPressure.Visible = False
    GraphPower.Visible = False
    CmdNo.Enabled = False
    CmdYes.Enabled = True

End Sub

Sub CmdYes_Click ()

    GraphPressure.DrawMode = 3
    GraphPower.DrawMode = 3
    GraphPressure.Visible = True
    GraphPower.Visible = True
    CmdYes.Enabled = False
    CmdNo.Enabled = True

End Sub

Sub Form_Load ()

    Alarmsounded = False
    Delaytime = 0
```

```
GraphPressure.Visible = False
GraphPower.Visible = False

' These are used in conjunction with GFocus to mimic
' standard windows text box changing methods while
' retaining certain controls on the text allowed.
Set NewBox = Nothing
Set OldBox = Frame1
OldText = ""

' The form is horizontally and vertically centered when loaded.
Top = Screen.Height / 2 - Height / 2
Left = Screen.Width / 2 - Width / 2

' Call reset to load the text boxes from the control file.
Option1.Value = False
Option2.Value = False
Option4.Value = False
Option5.Value = False
Option3.Value = True
```

End Sub

Sub Form_QueryUnload (Cancel As Integer, Unloadmode As Integer)

```
Dim Title As String
Dim Msg As String
Dim DgDef As Integer
Dim Response As Integer

' Put together a message box with all the proper components.
Title = "Warning"
Msg = "Do you really want to close the window?"
DgDef = MB_YESNO + MB_ICONSTOP + MB_DEFBUTTON2

Cancel = True

' Evaluate unload mode and give warning.
If Unloadmode = 0 Then
```

```
      Response = MsgBox(Msg, DgDef, Title)
      If Response = IDYES Then
         Cancel = False
         Unload frmFlow
      Else
         Cancel = True
      End If
   Else
         Cancel = False
   End If

End Sub

Sub Form_Unload (Cancel As Integer)

   'Call iclear(HPIBID)
   'Call ilocal(HPIBID)
   Call iclose(HPIBID)

  ' Tell SICL to clean up for this task
  Call siclcleanup

End Sub

Sub GraphPower_Click ()

   GraphPower.DrawMode = 5

End Sub

Sub GraphPressure_Click ()

   GraphPressure.DrawMode = 5

End Sub

Sub Option1_Click ()

   Dim I As Integer
   Dim J As Integer
```

```
Dim K As Integer
Dim L As Integer
Dim M As Integer
Dim N As Integer
Dim O As Integer
Dim Scan As Integer
Dim Duration As Double
Dim TimeInterval As Double
Dim DeltaTime As Double
Dim ScanStartTime As Variant
Dim ScanStopTime As Variant
Dim PressureRecordTime As Variant
Dim PressureRecordTimer As Double
Dim TempTime As Long
Dim Timestr As String
Dim WaitTime As Double

Set NewBox = Frame1
Call GFocus(OldBox, NewBox, OldText)
Set OldBox = NewBox
OldText = ""

Station = Int(Val(TxtStation.Text))

If Station <> 0 Then
   Option1.Value = True
Else
   MsgBox "# of stations must be greater than zero!"
   Option4.Value = True
   Exit Sub
End If

IntervalDistance = 100 / Station
Scan = Int(Val(TxtScan.Text))

If Scan <> 0 Then
   Option1.Value = True
Else
   MsgBox "# of scan must be greater than zero!"
   Option4.Value = True
```

```
    Exit Sub
End If

IntervalPulse = DistanceToPulse(IntervalDistance)
TotalPulse = CLng(IntervalPulse * (Station + 0))
TimeInterval = Int(Val(TxtInterval.Text))
Delaytime = CVDate(Delaytime)
Starttime = Time
TxtProgStart.Text = Starttime
Alarmtime = Starttime + Delaytime
TxtRecStart.Text = Alarmtime

If Alarmtime = Starttime Then
    TxtDelay.Text = "00:00:00"
End If

If SaturationTime = .2 Then
    If TxtInterval.Text < 60 Then
        MsgBox "Scan interval must be greater than 59 sec.!"
        Option4.Value = True
        Exit Sub
    End If
End If

If SaturationTime = 1 Then
    If TxtInterval.Text < 200 Then
        MsgBox "Scan interval must be greater than 199 sec.!"
        Option4.Value = True
        Exit Sub
    End If
End If

If SaturationTime = 5 Then
    If TxtInterval.Text < 600 Then
        MsgBox "Scan interval must be greater than 599 sec.!"
        Option4.Value = True
        Exit Sub
    End If
End If
```

```
If SaturationTime = 10 Then
   If TxtInterval.Text < 1200 Then
      MsgBox "Scan interval must be greater than 1199 sec.!"
      Option4.Value = True
      Exit Sub
   End If
End If

Option1.Enabled = False

TempTime = TimeInterval * Scan
Timestr = Str$(CInt(TempTime Mod 60))
TempTime = Int(TempTime / 60)
Timestr = Str$(CInt(TempTime Mod 60)) & ":" & Timestr
TempTime = Int(TempTime / 60)
Timestr = Str$(TempTime) & ":" & Timestr
EstStoptime = Alarmtime + CVDate(Timestr)
TxtProgStop.Text = Format$(EstStoptime, "hh:nn:ss AM/PM")

Do Until (Time >= Alarmtime)
   DoEvents
   If Option1.Value = False Then
      Exit Sub
   End If
Loop

Print #FileNumber,
Print #FileNumber, "Equipment and experiment settings :"
Print #FileNumber, "Frequency meter setting :"; Tab(35); TxtFreqSetting.Text;
Tab(45); " GHz"
Print #FileNumber, "Attenuator setting :"; Tab(35); TxtAttenSetting.Text;
Tab(45); " dB"
Print #FileNumber, "Gunn current :"; Tab(35); TxtGunnCurrent.Text; Tab(45);
" A"
Print #FileNumber, "Gunn voltage :"; Tab(35); TxtGunnVoltage.Text; Tab(45);
" V"
Print #FileNumber, "Tuner angle :"; Tab(35); TxtTunerAngle.Text; Tab(45); "
deg."
Print #FileNumber, "Water density @ start temp. :"; Tab(35);
TxtWaterDen.Text; Tab(45); " g/cc"
```

```
Print #FileNumber, "Water viscosity @ start temp. :"; Tab(35);
TxtWaterVis.Text; Tab(45); " cp."
Print #FileNumber, "Oil density @ start temp. :"; Tab(35); TxtOilDen.Text;
Tab(45); " g/cc"
Print #FileNumber, "Oil viscosity @ start temp. :"; Tab(35); TxtOilVis.Text;
Tab(45); " cp."
Print #FileNumber, "Pump #1 water rate :"; Tab(35); TxtPumpRate1.Text;
Tab(45); " cc/hr."
Print #FileNumber, "Pump #2 oil rate :"; Tab(35); TxtPumpRate2.Text;
Tab(45); " cc/hr"
Print #FileNumber, "Starting Temperature :"; Tab(35); TxtStartTemp.Text;
Tab(45); " deg. C"
Print #FileNumber, "Porosity :"; Tab(35); TxtPorosity; Tab(45); " %"
Print #FileNumber, "Absolute permeability :"; Tab(35); TxtPermeability.Text;
Tab(45); " md."
Print #FileNumber, "Number of stations :"; Tab(35); TxtStation.Text
Print #FileNumber, "Number of scans :"; Tab(35); TxtScan.Text
Print #FileNumber, "Scan interval :"; Tab(35); TxtInterval.Text; Tab(45); "
sec."
Print #FileNumber, "Run number :"; Tab(35); TxtRun.Text
Print #FileNumber, "Core number :"; Tab(35); TxtCore.Text
Print #FileNumber,
Print #FileNumber, "Experiment started at :"; Tab(35); Time
Print #FileNumber,
Print #FileNumber, "Measured value :"

GraphPressure.RandomData = 0
GraphPressure.AutoInc = 0
GraphPressure.NumSets = 2
GraphPressure.NumPoints = 7
GraphPressure.YAxisStyle = 2
GraphPressure.YAxisTicks = 10
GraphPressure.YAxisMin = 0
GraphPressure.YAxisMax = 10
GraphPressure.IndexStyle = 1
GraphPressure.LegendStyle = 1

GraphPower.RandomData = 0
GraphPower.AutoInc = 0
GraphPower.NumSets = 1
```

```
GraphPower.NumPoints = 100
GraphPower.YAxisStyle = 2
GraphPower.YAxisTicks = 10
GraphPower.YAxisMin = 0
GraphPower.YAxisMax = .005
GraphPower.LegendStyle = 1

For K = 1 To GraphPressure.NumSets

   GraphPressure.ThisSet = K

   If K = 1 Then
      GraphPressure.LegendText = "Water-wet"
      GraphPressure.ColorData = 1
      GraphPressure.SymbolData = 3
      For L = 1 To GraphPressure.NumPoints
         GraphPressure.ThisPoint = L
         If L = 1 Then
            GraphPressure.XPosData = 6#
            GraphPressure.GraphData = 0
         ElseIf L = 2 Then
            GraphPressure.XPosData = 13.2
            GraphPressure.GraphData = 0

         ElseIf L = 3 Then
            GraphPressure.XPosData = 27.7
            GraphPressure.GraphData = 0
         ElseIf L = 4 Then
            GraphPressure.XPosData = 50.2
            GraphPressure.GraphData = 0
         ElseIf L = 5 Then
            GraphPressure.XPosData = 72.7
            GraphPressure.GraphData = 0
         ElseIf L = 6 Then
            GraphPressure.XPosData = 87.2
            GraphPressure.GraphData = 0
         ElseIf L = 7 Then
            GraphPressure.XPosData = 98.3
            GraphPressure.GraphData = 0
         End If
```

```
    Next L
  End If

  If K = 2 Then
    GraphPressure.LegendText = "Oil-wet"
    GraphPressure.ColorData = 4
    GraphPressure.SymbolData = 9
    For L = 1 To GraphPressure.NumPoints
      GraphPressure.ThisPoint = L
      If L = 1 Then
        GraphPressure.XPosData = 2.1
        GraphPressure.GraphData = 0
      ElseIf L = 2 Then
        GraphPressure.XPosData = 13.2
        GraphPressure.GraphData = 0
      ElseIf L = 3 Then
        GraphPressure.XPosData = 27.7
        GraphPressure.GraphData = 0
      ElseIf L = 4 Then
        GraphPressure.XPosData = 50.2
        GraphPressure.GraphData = 0
      ElseIf L = 5 Then
        GraphPressure.XPosData = 72.7
        GraphPressure.GraphData = 0
      ElseIf L = 6 Then
        GraphPressure.XPosData = 87.2
        GraphPressure.GraphData = 0
      ElseIf L = 7 Then
        GraphPressure.XPosData = 98.3
        GraphPressure.GraphData = 0
      End If
    Next L
  End If

Next K

GraphPower.ThisSet = GraphPower.NumSets
GraphPower.LegendText = "Attenuated power"
GraphPower.ColorData = 1
GraphPower.SymbolData = 3
```

```
For O = 1 To GraphPower.NumPoints
    GraphPower.ThisPoint = O
    GraphPower.XPosData = O
    GraphPower.GraphData = 0
Next O

FramePlotData.Visible = True
CmdYes.Enabled = True
CmdYes.Visible = True
CmdNo.Enabled = False
CmdNo.Visible = True

For I = 1 To Scan Step 1

    Duration = Timer + TimeInterval
    ScanStartTime = Time

    For J = 1 To Station Step 1
        Option2.Enabled = False
        Option3.Enabled = False
        Option4.Enabled = False
        Option5.Enabled = False
        CmdYes.Enabled = False
        CmdNo.Enabled = False
        GraphPressure.Enabled = False
        GraphPower.Enabled = False
        ActualTime(J) = Timer
        Call HP436ARead(J)
        Call HP432CRead(PortA2Addr, PortB2Addr, J)
        Call ForwardMove(PIO1Addr, IntervalPulse)
        DoEvents
        If Option1.Value = False Then
            Exit Sub
        End If
    Next J

    GraphPressure.Enabled = True
    GraphPower.Enabled = True
    CmdYes.Enabled = True
```

```
CmdNo.Enabled = True

ScanStopTime = Time
PressureRecordTime = Time
PressureRecordTimer = Timer
Call UPCRead(UPCBase)
Call Balance1Read
Call Balance2Read

Print #FileNumber,
Print #FileNumber, "Scan started at :"; ScanStartTime; Tab(35); "Seconds
since midnight :"; ActualTime(1); "sec."
Print #FileNumber,
Print #FileNumber, "Scan stopped at :"; ScanStopTime; Tab(35); "Seconds
since midnight :"; ActualTime(100); "sec."
Print #FileNumber,
Print #FileNumber, "Scan"; Tab(10); "Station"; Tab(20); "Delta Time(sec.)";
Tab(40); "Input watt"; Tab(55); "Attenuated watt"
Print #FileNumber,

For J = 1 To Station Step 1
   Call HP432CConversion(J, HP432CUMeasurment(J),
   HP432CLMeasurment(J))
   Call HP436AConversion(J, HP436ARaw(J))
   DeltaTime = Format(ActualTime(J) - ActualTime(1), "0.0000000000")
   PM432C(J) = Format$(HP432CMeasurement(J), "0.00E+00")
   PM436A(J) = Format$(HP436AMeasurement(J), "0.00E+00")
   Print #FileNumber, I; Tab(10); J; Tab(20); DeltaTime; Tab(40);
   PM432C(J); Tab(55); PM436A(J)
   DoEvents
   If Option1.Value = False Then
      Exit Sub
   End If
Next J

For M = 1 To 16 Step 1
   Pressure(M) = Format(Pressure(M), "0.0000")
Next M

Print #FileNumber,
```

```
Print #FileNumber, "Time"; Tab(13); "Scan"; Tab(20); "Weight1"; Tab(45);
"Weight2"
Print #FileNumber, PressureRecordTime; Tab(13); I; Tab(20); Weight1;
Tab(45); Weight2
Print #FileNumber,
Print #FileNumber, "Seconds since midnight :";
Print #FileNumber, PressureRecordTimer; "sec.";
Print #FileNumber,
Print #FileNumber, "Pres.1"; Tab(10); "Pres.2"; Tab(20); "Pres.3"; Tab(30);
"Pres.4"
Print #FileNumber, Pressure(1); Tab(10); Pressure(2); Tab(20); Pressure(3);
Tab(30); Pressure(4)
Print #FileNumber,
Print #FileNumber, "Pres.5"; Tab(10); "Pres.6"; Tab(20); "Pres.7"; Tab(30);
"Pres.8"
Print #FileNumber, Pressure(5); Tab(10); Pressure(6); Tab(20); Pressure(7);
Tab(30); Pressure(8)
Print #FileNumber,
Print #FileNumber, "Pres.9"; Tab(10); "Pres.10"; Tab(20); "Pres.11";
Tab(30); "Pres.12"
Print #FileNumber, Pressure(9); Tab(10); Pressure(10); Tab(20);
Pressure(11); Tab(30); Pressure(12)
Print #FileNumber,
Print #FileNumber, "Pres.13"; Tab(10); "Pres.14"; Tab(20); "Pres.15";
Tab(30); "Pres.16"
Print #FileNumber, Pressure(13); Tab(10); Pressure(14); Tab(20);
Pressure(15); Tab(30); Pressure(16)
Print #FileNumber,

Call ReverseMove(PIO1Addr, TotalPulse)

Option2.Enabled = True
Option3.Enabled = True
Option4.Enabled = True

For K = 1 To GraphPressure.NumSets

  GraphPressure.ThisSet = K

  If K = 1 Then
```

```
    GraphPressure.LegendText = "Water-wet"
    GraphPressure.ColorData = 1
    GraphPressure.SymbolData = 3
    For L = 1 To GraphPressure.NumPoints
        GraphPressure.ThisPoint = L
        If L = 1 Then
            GraphPressure.XPosData = 6#
            GraphPressure.GraphData = Pressure(2)
        ElseIf L = 2 Then
            GraphPressure.XPosData = 13.2
            GraphPressure.GraphData = Pressure(3)
        ElseIf L = 3 Then
            GraphPressure.XPosData = 27.7
            GraphPressure.GraphData = Pressure(4)
        ElseIf L = 4 Then
            GraphPressure.XPosData = 50.2
            GraphPressure.GraphData = Pressure(5)
        ElseIf L = 5 Then
            GraphPressure.XPosData = 72.7
            GraphPressure.GraphData = Pressure(6)
        ElseIf L = 6 Then
            GraphPressure.XPosData = 87.2
            GraphPressure.GraphData = Pressure(7)
        ElseIf L = 7 Then
            GraphPressure.XPosData = 98.3
            GraphPressure.GraphData = Pressure(8)
        End If
    Next L
End If

If K = 2 Then
    GraphPressure.LegendText = "Oil-wet"
    GraphPressure.ColorData = 4
    GraphPressure.SymbolData = 9
    For L = 1 To GraphPressure.NumPoints
        GraphPressure.ThisPoint = L
        If L = 1 Then
            GraphPressure.XPosData = 2.1
            GraphPressure.GraphData = Pressure(9)
        ElseIf L = 2 Then
```

```
            GraphPressure.XPosData = 13.2
            GraphPressure.GraphData = Pressure(10)
        ElseIf L = 3 Then
            GraphPressure.XPosData = 27.7
            GraphPressure.GraphData = Pressure(11)
        ElseIf L = 4 Then
            GraphPressure.XPosData = 50.2
            GraphPressure.GraphData = Pressure(12)
        ElseIf L = 5 Then
            GraphPressure.XPosData = 72.7
            GraphPressure.GraphData = Pressure(13)
        ElseIf L = 6 Then
            GraphPressure.XPosData = 87.2
            GraphPressure.GraphData = Pressure(14)
        ElseIf L = 7 Then
            GraphPressure.XPosData = 98.3
            GraphPressure.GraphData = Pressure(15)
        End If
    Next L
  End If

Next K

GraphPower.ThisSet = GraphPower.NumSets
GraphPower.LegendText = "Attenuated power"
GraphPower.ColorData = 1
GraphPower.SymbolData = 3
For O = 1 To GraphPower.NumPoints
  GraphPower.ThisPoint = O
  GraphPower.XPosData = O
  GraphPower.GraphData = PM436A(O)
Next O

label3.Caption = "Scans remaining :"
TxtScan.ForeColor = &HFF
TxtScan.Text = Int(TxtScan.Text) - 1

GraphPressure.DrawMode = 3
GraphPower.DrawMode = 3
```

```
    If TimeInterval >= 200 Then
        GraphPressure.DrawMode = 5
        GraphPower.DrawMode = 5
    End If

    If (Scan > 1) And (I < Scan) Then
        Do Until Timer > Duration
            DoEvents
            If Option1.Value = False Then
                Exit Sub
            End If
        Loop
    End If

Next I

Option1.Enabled = True
Option5.Enabled = True
Option2.Value = True

End Sub

Sub Option2_Click ()

    Dim Default As Variant
    Dim Msg As String
    Dim FinalTemp As Variant

    Set NewBox = Frame1
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = ""

    ' Option 2 finishes the experiment and unloads the form

    Option1.Enabled = True
    Option2.Enabled = True
    Option3.Enabled = True
    Option4.Enabled = True
    Option5.Enabled = True
```

```
Option1.Value = False

Msg = "Experiment finished."
Msg = Msg & "Enter final temperature in deg. C."
Default = TxtStartTemp.Text
FinalTemp = InputBox(Msg, "", Default)
If FinalTemp = "" Then
   FinalTemp = TxtStartTemp.Text
End If

Print #FileNumber,
Print #FileNumber, "Final Temperature : "; Tab(35); FinalTemp; Tab(45); " deg.
C"
Print #FileNumber, "Experiment finished at :"; Tab(35); Time

Close FileNumber

frmLogSet.Text1.Visible = True
frmLogSet.Text2.Visible = False
frmLogSet.Text3.Visible = False
frmLogSet.Command3D1.Visible = True
frmLogSet.Command3D2.Visible = True
frmLogSet.Command3D3.Visible = False
frmLogSet.Command3D4.Visible = False
frmLogSet.Command3D5.Visible = False
frmLogSet.Command3D6.Visible = False
frmLogSet.Label1.Visible = False
frmLogSet.Label44.Visible = False
frmLogSet.Label46.Visible = False
frmLogSet.Label48.Visible = False
frmLogSet.Label50.Visible = False
frmLogSet.Label51.Visible = False
frmLogSet.TxtCardBase1.Visible = False
frmLogSet.TxtCardBase2.Visible = False
frmLogSet.TxtHPIBAddr.Visible = False
frmLogSet.Txt601LAddress.Visible = False
frmLogSet.TxtBalance1.Visible = False
frmLogSet.TxtBalance2.Visible = False
```

```
    Option3.Value = True

    Unload frmFlow

End Sub

Sub Option3_Click ()

    ' Option 3 stops the experiment and resets the values
    ' in the text boxes to those found in the control file.

    Dim FlowFile As Integer
    Dim Char As Variant
    ReDim TextData(19) As Variant
    Dim I As Integer

    Option1.Enabled = True
    Option2.Enabled = True
    Option3.Enabled = True
    Option4.Enabled = True
    Option5.Enabled = True

    Option1.Value = False
    CmdYes.Visible = False
    CmdNo.Visible = False
    FramePlotData.Visible = False
    GraphPressure.Visible = False
    GraphPower.Visible = False

    Set NewBox = Frame1
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = ""

    FlowFile = FreeFile
    Open "Flow.ctl" For Input As FlowFile

    For I = 0 To 18
        Do
            Char = Input(1, FlowFile)
```

```
    Loop Until Char = Chr(58)
    TextData(I) = ""
    While Char <> Chr(13)
        Char = Input(1, FlowFile)
        TextData(I) = TextData(I) & Char
    Wend
Next I

Close FlowFile

TxtFreqSetting.Text = Left(TextData(0), Len(TextData(0)) - 1)
TxtAttenSetting.Text = Left(TextData(1), Len(TextData(1)) - 1)
TxtGunnCurrent.Text = Left(TextData(2), Len(TextData(2)) - 1)
TxtGunnVoltage.Text = Left(TextData(3), Len(TextData(3)) - 1)
TxtTunerAngle.Text = Left(TextData(4), Len(TextData(4)) - 1)
TxtWaterDen.Text = Left(TextData(5), Len(TextData(5)) - 1)
TxtWaterVis.Text = Left(TextData(6), Len(TextData(6)) - 1)
TxtOilDen.Text = Left(TextData(7), Len(TextData(7)) - 1)
TxtOilVis.Text = Left(TextData(8), Len(TextData(8)) - 1)
TxtPumpRate1.Text = Left(TextData(9), Len(TextData(9)) - 1)
TxtPumpRate2.Text = Left(TextData(10), Len(TextData(10)) - 1)
TxtStartTemp.Text = Left(TextData(11), Len(TextData(11)) - 1)
TxtPorosity.Text = Left(TextData(12), Len(TextData(12)) - 1)
TxtPermeability.Text = Left(TextData(13), Len(TextData(13)) - 1)
TxtStation.Text = Left(TextData(14), Len(TextData(14)) - 1)
TxtScan.Text = Left(TextData(15), Len(TextData(15)) - 1)
TxtInterval.Text = Left(TextData(16), Len(TextData(16)) - 1)
TxtRun.Text = Left(TextData(17), Len(TextData(17)) - 1)
TxtCore.Text = Left(TextData(18), Len(TextData(18)) - 1)

Delaytime = 0
TxtDelay.Text = "HH:MM:SS"
TxtProgStart.Text = "HH:MM:SS"
TxtRecStart.Text = "HH:MM:SS"
TxtProgElapse = "HH:MM:SS"
TxtProgStop = "HH:MM:SS"
label39.Caption = "Recording will start at:"
TxtRecStart.ForeColor = &HFF0000
Alarmsounded = False
```

```
    Option3.Value = False

End Sub

Sub Option4_Click ()

    ' Option 4 stops the experiment without unloading
    ' the form or resetting the text box values.

    Option1.Enabled = True
    Option2.Enabled = True
    Option3.Enabled = True
    Option4.Enabled = True
    Option5.Enabled = True

    Option1.Value = False
    FramePlotData.Visible = False
    CmdYes.Visible = False
    CmdNo.Visible = False
    GraphPressure.Visible = False
    GraphPower.Visible = False

    label3.Caption = "Number of scans :"
    TxtScan.ForeColor = &HFF0000

    label39.Caption = "Recording will start at:"
    TxtRecStart.ForeColor = &HFF0000
    Alarmsounded = False

    Set NewBox = Frame1
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = ""

End Sub

Sub Option5_Click ()

    Dim FlowFile As Integer
    Dim Response As Integer
```

```
Dim Msg As String
Dim Title As String

Set NewBox = Frame1
Call GFocus(OldBox, NewBox, OldText)
Set OldBox = NewBox
OldText = ""

' Option 5 saves the current values in the text boxes
' to the control file.

Msg = "Save these settings?"
Title = "Saving Flow Experiment Settings"
Response = MsgBox(Msg, 4 + 16 + 256, Title)
If Response = IDNO Then
   Option5.Value = False
   Exit Sub
End If

FlowFile = FreeFile

Open "Flow.ctl" For Output As FlowFile

Print #FlowFile,
Print #FlowFile, "Equipment and experiment settings control file."
Print #FlowFile, "Frequency meter setting (GHz):"; TxtFreqSetting.Text
Print #FlowFile, "Attenuator setting (dB):"; TxtAttenSetting.Text
Print #FlowFile, "Gunn current (A):"; TxtGunnCurrent.Text
Print #FlowFile, "Gunn voltage (V):"; TxtGunnVoltage.Text
Print #FlowFile, "Tuner angle (deg.):"; TxtTunerAngle.Text
Print #FlowFile, "Water density @ start temp. (g/cc):"; TxtWaterDen.Text
Print #FlowFile, "Water viscosity @ start temp. (cp.):"; TxtWaterVis.Text
Print #FlowFile, "Oil density @ start temp. (g/cc):"; TxtOilDen.Text
Print #FlowFile, "Oil viscosity @ start temp. (cp.):"; TxtOilVis.Text
Print #FlowFile, "Pump #1 water rate (cc/hr):"; TxtPumpRate1.Text
Print #FlowFile, "Pump #2 oil rate (cc/hr):"; TxtPumpRate2.Text
Print #FlowFile, "Starting Temperature (deg. C):"; TxtStartTemp.Text
Print #FlowFile, "Porosity (%):"; TxtPorosity
Print #FlowFile, "Absolute permeability (md.):"; TxtPermeability.Text
Print #FlowFile, "Number of stations :"; TxtStation.Text
```

```
    Print #FlowFile, "Number of scans :"; TxtScan.Text
    Print #FlowFile, "Scan interval (sec.):"; TxtInterval.Text
    Print #FlowFile, "Run number :"; TxtRun.Text
    Print #FlowFile, "Core number :"; TxtCore.Text
    Print #FlowFile,

    Close FlowFile

    Option5.Value = False

End Sub

Sub Timer1_Timer ()

    ' Timer 1 updates the date/time boxes and (if the
    ' experiment has been started) the elapsed time box
    ' every half second. It also keeps track of when the
    ' actual recording has started.

    TxtDate.Text = Date
    TxtTime.Text = Time

    If (Time >= Alarmtime) And (Not Alarmsounded) And (Option1.Value <>
    False) Then
        Beep
        TxtRecStart.Text = Time
        label39.Caption = "Recording started at:"
        TxtRecStart.ForeColor = &HFF&
        Alarmsounded = True
    End If

    If (Option1.Value <> False) Then
        TxtProgElapse.Text = Format$((Time - Starttime), "hh:nn:ss")
    End If


End Sub

Sub TxtAttenSetting_GotFocus ()
```

```
    Set NewBox = TxtAttenSetting
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtCore_GotFocus ()

    Set NewBox = TxtCore
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtDate_GotFocus ()

    Set NewBox = Frame1
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = ""

End Sub

Sub TxtDelay_GotFocus ()

    Set NewBox = Frame1
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = ""

    Delaytime = InputBox("Enter delay time in the format of HH:MM:SS",
    "Alarm", Alarmtime)
    If Delaytime = "" Then Exit Sub
    If Not IsDate(Delaytime) Then
        MsgBox "The time you entered was not valid."
    Else
        ' String returned from InputBox is a valid time
        ' so store as a date/time value
```

```
    TxtDelay.Text = Delaytime
  End If

End Sub

Sub TxtFreqSetting_GotFocus ()

  Set NewBox = TxtFreqSetting
  Call GFocus(OldBox, NewBox, OldText)
  Set OldBox = NewBox
  OldText = OldBox.Text

End Sub

Sub TxtGunnCurrent_GotFocus ()

  Set NewBox = TxtGunnCurrent
  Call GFocus(OldBox, NewBox, OldText)
  Set OldBox = NewBox
  OldText = OldBox.Text

End Sub

Sub TxtGunnVoltage_GotFocus ()

  Set NewBox = TxtGunnVoltage
  Call GFocus(OldBox, NewBox, OldText)
  Set OldBox = NewBox
  OldText = OldBox.Text

End Sub

Sub TxtInterval_GotFocus ()

  Set NewBox = TxtInterval
  Call GFocus(OldBox, NewBox, OldText)
  Set OldBox = NewBox
  OldText = OldBox.Text

End Sub
```

```
Sub TxtOilDen_GotFocus ()

    Set NewBox = TxtOilDen
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtOilVis_GotFocus ()

    Set NewBox = TxtOilVis
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtPermeability_GotFocus ()

    Set NewBox = TxtPermeability
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtPorosity_GotFocus ()

    Set NewBox = TxtPorosity
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtProgElapse_GotFocus ()

    Set NewBox = Frame1
```

```
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = ""

End Sub

Sub TxtProgStart_GotFocus ()

    Set NewBox = Frame1
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = ""

End Sub

Sub TxtProgStop_GotFocus ()

    Set NewBox = Frame1
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = ""

End Sub

Sub TxtPumpRate1_GotFocus ()

    Set NewBox = TxtPumpRate1
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtPumpRate2_GotFocus ()

    Set NewBox = TxtPumpRate2
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text
```

```
End Sub

Sub TxtRecStart_GotFocus ()

    Set NewBox = Frame1
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = ""

End Sub

Sub TxtRun_GotFocus ()

    Set NewBox = TxtRun
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtScan_GotFocus ()

    Set NewBox = TxtScan
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtStartTemp_GotFocus ()

    Set NewBox = TxtStartTemp
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtStation_GotFocus ()
```

```
    Set NewBox = TxtStation
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtTime_GotFocus ()

    Set NewBox = Frame1
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = ""

End Sub

Sub TxtTunerAngle_GotFocus ()

    Set NewBox = TxtTunerAngle
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub T[illegible]ocus ()

    [illegible]Den
    [illegible]wBox, OldText)

En[illegible]

Sub TxtWaterVis_GotFocus ()

    Set NewBox = TxtWaterVis
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text
```

```
End Sub
```

Appendix B - 15 LOGSET.FRM

```
'**********Equipment interfacing control ...m**********
'This form is written to control memory location for data communication.

Option Explicit


Sub CloseCards ()

   ' Close any open files
   Close FileNumber

   ' Close port used by BALANCE1
   If frmMain.MSComm1.PortOpen = True Then
         frmMain.MSComm1.PortOpen = False
         'Msg = "Communication port used by balance #1 is closed"
   End If

   ' Close port used by BALANCE2
   If frmMain.MSComm2.PortOpen = True Then
         frmMain.MSComm2.PortOpen = False
         'Msg = "Communication port used by balance #2 is closed"
   End If

   ' Close device session for power meter 436A
   If HPIBID <> 0 Then
         Call iclose(HPIBID)
         HPIBID = 0
         'Msg = "HPIB port closed"
   End If

End Sub

Sub Command3D1_Click ()

   Dim DgDef As Variant
   Dim Msg As String
   Dim Response As Variant
   Dim Title As Variant
   Dim CtlFile As Integer
   Dim Char As Variant
```

```
ReDim TextData(43) As Variant
Dim I As Integer

'If PassWordFlag = False Then
'   frmPassWord.Show 1
'End If

If List1.ListIndex = -1 Then
    Title = "Error"
    Msg = "Select one of the experiment types before proceeding,"
    Msg = Msg & " please retry."
    DgDef = MB_OK + MB_ICONSTOP
    Response = MsgBox(Msg, DgDef, Title)
    Exit Sub
End If

If List1.ListIndex = 2 And List2.ListIndex = -1 Then
    Title = "Error"
    Msg = "Select one of the flow types before proceeding,"
    Msg = Msg & " please retry."
    DgDef = MB_OK + MB_ICONSTOP
    Response = MsgBox(Msg, DgDef, Title)
    Exit Sub
End If

If List1.ListIndex = 3 And List2.ListIndex = -1 Then
    Title = "Error"
    Msg = "Select one of the flow types before proceeding,"
    Msg = Msg & " please retry."
    DgDef = MB_OK + MB_ICONSTOP
    Response = MsgBox(Msg, DgDef, Title)
    Exit Sub
End If

If TxtFileName.Text = "" Then
    Title = "Error"
    Msg = "Enter a valid file name to be opened before "
    Msg = Msg & "proceeding.  File will be created in "
    Msg = Msg & "the current drive and directory, please retry."
    DgDef = MB_OK + MB_ICONSTOP
```

```
        Response = MsgBox(Msg, DgDef, Title)
        Exit Sub
    End If

    If Dir$(frmLogSet.TxtFileName.Text) <> "" Then
        Title = "Warning"
        Msg = "You have chosen a file that already exists in the "
        Msg = Msg & "current directory. Do you want to choose another "
        Msg = Msg & "file name? Answer YES to choose a new file name "
        Msg = Msg & "or NO to overwrite the current file."
        DgDef = MB_YESNO + MB_ICONSTOP + MB_DEFBUTTON1
        Response = MsgBox(Msg, DgDef, Title)
        If Response = IDYES Then
          TxtFileName.SetFocus
          Exit Sub
        End If
    End If

    FileName = TxtFileName.Text
    FileNumber = FreeFile
    Open FileName For Output As FileNumber Len = 2048
    Print #FileNumber, "General information :"
    Print #FileNumber, "User :"; Tab(35); TxtUser.Text
    Print #FileNumber, "Date :"; Tab(35); TxtDate.Text
    Print #FileNumber, "Time :"; Tab(35); TxtTime.Text
    Print #FileNumber, "Experiment type :"; Tab(35); List1.Text
    Print #FileNumber, "Flow type :"; Tab(35); List2.Text
    Print #FileNumber, "File Name :"; Tab(35); TxtFileName.Text
    Print #FileNumber,

    CtlFile = FreeFile

    Open "Equip.ctl" For Input As CtlFile Len = 2048
    For I = 0 To 43
        Do
          Char = Input(1, CtlFile)
        Loop Until Char = Chr(58)
        TextData(I) = ""
        While Char <> Chr(13)
          Char = Input(1, CtlFile)
```

```
        TextData(I) = TextData(I) & Char
    Wend
Next I
Close CtlFile

Print #FileNumber, "Equipment serial numbers."
Print #FileNumber, "Pump #1 :"; Tab(35); TextData(0)
Print #FileNumber, "Pump #2 :"; Tab(35); TextData(1)
Print #FileNumber, "Stepping motor :"; Tab(35); TextData(2)
Print #FileNumber, "Stepping motor controller :"; Tab(35); TextData(3)
Print #FileNumber, "Balance #1 :"; Tab(35); TextData(4)
Print #FileNumber, "Balance #2 :"; Tab(35); TextData(5)
Print #FileNumber, "Gunn oscillator :"; Tab(35); TextData(6)
Print #FileNumber, "Power supply :"; Tab(35); TextData(7)
Print #FileNumber, "Low pass filter :"; Tab(35); TextData(8)
Print #FileNumber, "Frequency meter #1 :"; Tab(35); TextData(9)
Print #FileNumber, "Frequency meter #2 :"; Tab(35); TextData(10)
Print #FileNumber, "Attenuator :"; Tab(35); TextData(11)
Print #FileNumber, "Isolator :"; Tab(35); TextData(12)
Print #FileNumber, "Screw tuner :"; Tab(35); TextData(13)
Print #FileNumber, "Directional coupler #1 :"; Tab(35); TextData(14)
Print #FileNumber, "Directional coupler #2 :"; Tab(35); TextData(15)
Print #FileNumber, "Thermistor sensor #1 :"; Tab(35); TextData(16)
Print #FileNumber, "Thermistor sensor #2 :"; Tab(35); TextData(17)
Print #FileNumber, "Power sensor :"; Tab(35); TextData(18)
Print #FileNumber, "Power meter #1 :"; Tab(35); TextData(19)
Print #FileNumber, "Power meter #2 :"; Tab(35); TextData(20)
Print #FileNumber, "Power meter #3 :"; Tab(35); TextData(21)
Print #FileNumber, "PIO-12 card #1 :"; Tab(35); TextData(22)
Print #FileNumber, "PIO-12 card #2 :"; Tab(35); TextData(23)
Print #FileNumber, "601-L card :"; Tab(35); TextData(24)
Print #FileNumber, "HP-IB card :"; Tab(35); TextData(25)
Print #FileNumber, "Mercury barometer :"; Tab(35); TextData(26)
Print #FileNumber, "Heise gauge :"; Tab(35); TextData(27)
Print #FileNumber, "Transducer #1 :"; Tab(35); TextData(28)
Print #FileNumber, "Transducer #2 :"; Tab(35); TextData(29)
Print #FileNumber, "Transducer #3 :"; Tab(35); TextData(30)
Print #FileNumber, "Transducer #4 :"; Tab(35); TextData(31)
Print #FileNumber, "Transducer #5 :"; Tab(35); TextData(32)
Print #FileNumber, "Transducer #6 :"; Tab(35); TextData(33)
```

```
Print #FileNumber, "Transducer #7 :"; Tab(35); TextData(34)
Print #FileNumber, "Transducer #8 :"; Tab(35); TextData(35)
Print #FileNumber, "Transducer #9 :"; Tab(35); TextData(36)
Print #FileNumber, "Transducer #10 :"; Tab(35); TextData(37)
Print #FileNumber, "Transducer #11 :"; Tab(35); TextData(38)
Print #FileNumber, "Transducer #12 :"; Tab(35); TextData(39)
Print #FileNumber, "Transducer #13 :"; Tab(35); TextData(40)
Print #FileNumber, "Transducer #14 :"; Tab(35); TextData(41)
Print #FileNumber, "Transducer #15 :"; Tab(35); TextData(42)
Print #FileNumber, "Transducer #16 :"; Tab(35); TextData(43)
Print #FileNumber,

CtlFile = FreeFile
Open "Transcal.ctl" For Input As CtlFile Len = 2048
For I = 0 To 31
    Do
        Char = Input(1, CtlFile)
    Loop Until Char = Chr(58)
    TextData(I) = ""
    While Char <> Chr(13)
        Char = Input(1, CtlFile)
        TextData(I) = TextData(I) & Char
    Wend
Next I
Close CtlFile

Print #FileNumber, "Pressure transducer calibration numbers."
Print #FileNumber, "Transducer #1 gain :"; Tab(35); TextData(0)
Print #FileNumber, "Transducer #2 gain :"; Tab(35); TextData(1)
Print #FileNumber, "Transducer #3 gain :"; Tab(35); TextData(2)
Print #FileNumber, "Transducer #4 gain :"; Tab(35); TextData(3)
Print #FileNumber, "Transducer #5 gain :"; Tab(35); TextData(4)
Print #FileNumber, "Transducer #6 gain :"; Tab(35); TextData(5)
Print #FileNumber, "Transducer #7 gain :"; Tab(35); TextData(6)
Print #FileNumber, "Transducer #8 gain :"; Tab(35); TextData(7)
Print #FileNumber, "Transducer #9 gain :"; Tab(35); TextData(8)
Print #FileNumber, "Transducer #10 gain :"; Tab(35); TextData(9)
Print #FileNumber, "Transducer #11 gain :"; Tab(35); TextData(10)
Print #FileNumber, "Transducer #12 gain :"; Tab(35); TextData(11)
Print #FileNumber, "Transducer #13 gain :"; Tab(35); TextData(12)
```

```
Print #FileNumber, "Transducer #14 gain :"; Tab(35); TextData(13)
Print #FileNumber, "Transducer #15 gain :"; Tab(35); TextData(14)
Print #FileNumber, "Transducer #16 gain :"; Tab(35); TextData(15)
Print #FileNumber, "Transducer #1 offset :"; Tab(35); TextData(16)
Print #FileNumber, "Transducer #2 offset :"; Tab(35); TextData(17)
Print #FileNumber, "Transducer #3 offset :"; Tab(35); TextData(18)
Print #FileNumber, "Transducer #4 offset :"; Tab(35); TextData(19)
Print #FileNumber, "Transducer #5 offset :"; Tab(35); TextData(20)
Print #FileNumber, "Transducer #6 offset :"; Tab(35); TextData(21)
Print #FileNumber, "Transducer #7 offset :"; Tab(35); TextData(22)
Print #FileNumber, "Transducer #8 offset :"; Tab(35); TextData(23)
Print #FileNumber, "Transducer #9 offset :"; Tab(35); TextData(24)
Print #FileNumber, "Transducer #10 offset :"; Tab(35); TextData(25)
Print #FileNumber, "Transducer #11 offset :"; Tab(35); TextData(26)
Print #FileNumber, "Transducer #12 offset :"; Tab(35); TextData(27)
Print #FileNumber, "Transducer #13 offset :"; Tab(35); TextData(28)
Print #FileNumber, "Transducer #14 offset :"; Tab(35); TextData(29)
Print #FileNumber, "Transducer #15 offset :"; Tab(35); TextData(30)
Print #FileNumber, "Transducer #16 offset :"; Tab(35); TextData(31)

Text1.Visible = False
Command3D1.Visible = False
Command3d2.Visible = False

If List1.ListIndex = 0 Then
    Label1.Visible = True
    Label50.Visible = True
    Label51.Visible = True
    Txt601LAddress.Visible = True
    TxtBalance1.Visible = True
    TxtBalance2.Visible = True
    Text2.Visible = True
    Command3D3.Visible = True
    Command3D3.SetFocus
    Command3D4.Visible = True
End If

If List1.ListIndex = 1 Then
    Label1.Visible = True
    Label50.Visible = True
```

```
    Label51.Visible = True
    Txt601LAddress.Visible = True
    TxtBalance1.Visible = True
    TxtBalance2.Visible = True
    Text2.Visible = True
    Command3D3.Visible = True
    Command3D3.SetFocus
    Command3D4.Visible = True
End If

If List1.ListIndex = 2 Then
    Label1.Visible = True
    Label44.Visible = True
    Label46.Visible = True
    Label48.Visible = True
    Label50.Visible = True
    Label51.Visible = True
    TxtCardBase1.Visible = True
    TxtCardBase2.Visible = True
    TxtHPIBAddr.Visible = True
    Txt601LAddress.Visible = True
    TxtBalance1.Visible = True
    TxtBalance2.Visible = True
    Text2.Visible = True
    Command3D3.Visible = True
    Command3D3.SetFocus
    Command3D4.Visible = True
End If

If List1.ListIndex = 3 Then
    Label1.Visible = True
    Label44.Visible = True
    Label46.Visible = True
    Label48.Visible = True
    Label50.Visible = True
    Label51.Visible = True
    TxtCardBase1.Visible = True
    TxtCardBase2.Visible = True
    TxtHPIBAddr.Visible = True
    Txt601LAddress.Visible = True
```

```
        TxtBalance1.Visible = True
        TxtBalance2.Visible = True
        Text2.Visible = True
        Command3D3.Visible = True
        Command3D3.SetFocus
        Command3D4.Visible = True
    End If

    If List1.ListIndex = 4 Then
        Label44.Visible = True
        Label46.Visible = True
        Label48.Visible = True
        TxtCardBase1.Visible = True
        TxtCardBase2.Visible = True
        TxtHPIBAddr.Visible [illegible]e
        Text3.Visible = True
        Command3D5.Vis[illegible]e
        Command3D5.SetF[illegible]
        Command3D6.Visible = True
    End If

    If List1.ListIndex = 5 Then
        Label44.Visible = True
        Label46.Visible = True
        Label48.Visible = True
        TxtCardBase1.Visible = True
        TxtCardBase2.Visible = True
        TxtHPIBAddr.Visible = True
        Text3.Visible = True
        Command3D5.Visible = True
        Command3D5.SetFocus
        Command3D6.Visible = True
    End If

    If List1.ListIndex = 6 Then
        Label44.Visible = True
        Label46.Visible = True
        Label48.Visible = True
        TxtCardBase1.Visible = True
        TxtCardBase2.Visible = True
```

```
        TxtHPIBAddr.Visible = True
        Text3.Visible = True
        Command3D5.Visible = True
        Command3D5.SetFocus
        Command3D6.Visible = True
    End If

    If List1.ListIndex = 7 Then
        Label44.Visible = True
        Label46.Visible = True
        Label48.Visible = True
        TxtCardBase1.Visible = True
        TxtCardBase2.Visible = True
        TxtHPIBAddr.Visible = True
        Text3.Visible = True
        Command3D5.Visible = True
        Command3D5.SetFocus
        Command3D6.Visible = True
    End If

End Sub

Sub Command3D1_GotFocus ()

    Command3D1_Click

End Sub

Sub Command3D2_Click ()

    PassWordFlag = True
    Unload frmLogSet

End Sub

Sub Command3D3_Click ()

    Dim Comm1Settings As String
    Dim Comm2Settings As String
    Dim HPIBStrAddr As String
```

```
Dim LogFile As Integer

Screen.MousePointer = 11

LogFile = FreeFile

Open "Logset.ctl" For Output As LogFile

Print #LogFile,
Print #LogFile, "Logset settings control file."
Print #LogFile, "User :"; TxtUser.Text
Print #LogFile, "File Name :"; TxtFileName.Text
Print #LogFile, "VR interface address (Hex) :"; Txt601LAddress.Text
Print #LogFile, "Comm2 Balance #1 setup :"; TxtBalance1.Text
Print #LogFile, "Comm4 Balance #2 setup :"; TxtBalance2.Text
Print #LogFile, "Motor controller address (Hex) :"; TxtCardBase2.Text
Print #LogFile, "432C controller address (Hex) :"; TxtCardBase1.Text
Print #LogFile, "436A controller address (HPIB) :"; TxtHPIBAddr.Text
Print #LogFile,

Close LogFile

If List1.ListIndex = 0 Then
    Call UPCOn(UPCBase)
    Comm1Settings = TxtBalance1.Text
    Comm2Settings = TxtBalance2.Text
    Call RS2321Set(Comm1Settings)
    Call RS2322Set(Comm2Settings)
    Call Balance1PreSet
    Call Balance2PreSet
    frmPerm.Caption = "Absolute Permeability Measurement"
    Screen.MousePointer = 1
    frmPerm.Show 1
End If

If List1.ListIndex = 1 Then
    Call UPCOn(UPCBase)
    Comm1Settings = TxtBalance1.Text
    Comm2Settings = TxtBalance2.Text
    Call RS2321Set(Comm1Settings)
```

```
        Call RS2322Set(Comm2Settings)
        Call Balance1PreSet
        Call Balance2PreSet
        frmPerm.Caption = "Effective Permeability Measurement"
        Screen.MousePointer = 1
        frmPerm.Show 1
    End If

    If List1.ListIndex = 2 Then
        Comm1Settings = TxtBalance1.Text
        Comm2Settings = TxtBalance2.Text
        HPIBStrAddr = TxtHPIBAddr.Text
        Call InitPIO1(PIO1Addr)
        Call InitPIO2(PIO2Addr)
        Call HP432CPreSet(PIO2Addr)
        Call HP436ASet(HPIBStrAddr)
        Call HP432CAutoZero(PIO2Addr)
        Call UPCOn(UPCBase)
        Call RS2321Set(Comm1Settings)
        Call RS2322Set(Comm2Settings)
        Call Balance1PreSet
        Call Balance2PreSet
        frmFlow.Caption = "Steady State Experiment"
        Screen.MousePointer = 1
        frmFlow.Show 1
    End If

    If List1.ListIndex = 3 Then
        Comm1Settings = TxtBalance1.Text
        Comm2Settings = TxtBalance2.Text
        HPIBStrAddr = TxtHPIBAddr.Text
        Call InitPIO1(PIO1Addr)
        Call InitPIO2(PIO2Addr)
        Call HP432CPreSet(PIO2Addr)
        Call HP436ASet(HPIBStrAddr)
        Call HP432CAutoZero(PIO2Addr)
        Call UPCOn(UPCBase)
        Call RS2321Set(Comm1Settings)
        Call RS2322Set(Comm2Settings)
        Call Balance1PreSet
```

```
        Call Balance2PreSet
        frmFlow.Caption = "Unsteady State Experiment"
        Screen.MousePointer = 1
        frmFlow.Show 1
    End If

    Call CloseCards

End Sub

Sub Command3D4_Click ()

    Text1.Visible = True
    Command3D1.Visible = True
    Command3d2.Visible = True
    Text2.Visible = False
    Command3D3.Visible = False
    Command3D4.Visible = False
    TxtFileName.SetFocus

    Label1.Visible = False
    Label44.Visible = False
    Label46.Visible = False
    Label48.Visible = False
    Label50.Visible = False
    Label51.Visible = False
    TxtCardBase1.Visible = False
    TxtCardBase2.Visible = False
    TxtHPIBAddr.Visible = False
    Txt601LAddress.Visible = False
    TxtBalance1.Visible = False
    TxtBalance2.Visible = False

    Call CloseCards

End Sub

Sub Command3D5_Click ()

    Dim HPIBStrAddr As String
```

```
Dim LogFile As Integer

Screen.MousePointer = 11

LogFile = FreeFile

Open "Logset.ctl" For Output As LogFile

Print #LogFile,
Print #LogFile, "Logset settings control file."
Print #LogFile, "User :"; TxtUser.Text
Print #LogFile, "File Name :"; TxtFileName.Text
Print #LogFile, "VR interface address (Hex) :"; Txt601LAddress.Text
Print #LogFile, "Comm2 Balance #1 setup :"; TxtBalance1.Text
Print #LogFile, "Comm4 Balance #2 setup :"; TxtBalance2.Text
Print #LogFile, "Motor controller address (Hex) :"; TxtCardBase2.Text
Print #LogFile, "432C controller address (Hex) :"; TxtCardBase1.Text
Print #LogFile, "436A controller address (HPIB) :"; TxtHPIBAddr.Text
Print #LogFile,

Close LogFile

If List1.ListIndex = 4 Then
    HPIBStrAddr = TxtHPIBAddr.Text
    Call InitPIO1(PIO1Addr)
    Call InitPIO2(PIO2Addr)
    Call HP432CPreSet(PIO2Addr)
    Call HP436ASet(HPIBStrAddr)
    Call HP432CAutoZero(PIO2Addr)
    frmScan.Caption = "Meter Calibration @ Sw = 0.0 %"
    Screen.MousePointer = 1
    frmScan.Show 1
End If

If List1.ListIndex = 5 Then
    HPIBStrAddr = TxtHPIBAddr.Text
    Call InitPIO1(PIO1Addr)
    Call InitPIO2(PIO2Addr)
    Call HP432CPreSet(PIO2Addr)
    Call HP436ASet(HPIBStrAddr)
```

```
        Call HP432CAutoZero(PIO2Addr)
        frmScan.Caption = "Meter Calibration @ Sw = 100.0 %"
        Screen.MousePointer = 1
        frmScan.Show 1
    End If

    If List1.ListIndex = 6 Then
        HPIBStrAddr = TxtHPIBAddr.Text
        Call InitPIO1(PIO1Addr)
        Call InitPIO2(PIO2Addr)
        Call HP432CPreSet(PIO2Addr)
        Call HP436ASet(HPIBStrAddr)
        Call HP432CAutoZero(PIO2Addr)
        frmScan.Caption = "Meter Calibration @ Sw = Swirr %"
        Screen.MousePointer = 1
        frmScan.Show 1
    End If

    If List1.ListIndex = 7 Then
        HPIBStrAddr = TxtHPIBAddr.Text
        Call InitPIO1(PIO1Addr)
        Call InitPIO2(PIO2Addr)
        Call HP432CPreSet(PIO2Addr)
        Call HP436ASet(HPIBStrAddr)
        Call HP432CAutoZero(PIO2Addr)
        frmScan.Caption = "Meter Calibration @ Sw = Swm %"
        Screen.MousePointer = 1
        frmScan.Show 1
    End If

    Call CloseCards

End Sub

Sub Command3D6_Click ()

    Text1.Visible = True
    Command3D1.Visible = True
    Command3d2.Visible = True
    Text3.Visible = False
```

```
    Command3D5.Visible = False
    Command3D6.Visible = False
    TxtFileName.SetFocus

    Label44.Visible = False
    Label46.Visible = False
    Label48.Visible = False
    TxtCardBase1.Visible = False
    TxtCardBase2.Visible = False
    TxtHPIBAddr.Visible = False

    Call CloseCards

End Sub

Sub Form_Load ()

    FileExistanceFlag = True
    PassWordFlag = True

    ' The form is horizontally and vertically centered when loaded.
    Top = Screen.Height / 2 - Height / 2
    Left = Screen.Width / 2 - Width / 2

    Dim LogFile As Integer
    Dim Char As Variant
    ReDim TextData(8) As Variant
    Dim I As Integer

    LogFile = FreeFile

    Open "Logset.ctl" For Input As LogFile

    For I = 0 To 7
        Do
            Char = Input(1, LogFile)
        Loop Until Char = Chr(58)
        TextData(I) = ""
        While Char <> Chr(13)
            Char = Input(1, LogFile)
```

```
        TextData(I) = TextData(I) & Char
      Wend
   Next I

   Close LogFile

   TxtUser.Text = Left(TextData(0), Len(TextData(0)) - 1)
   TxtFileName.Text = Left(TextData(1), Len(TextData(1)) - 1)
   Txt601LAddress.Text = Left(TextData(2), Len(TextData(2)) - 1)
   TxtBalance1.Text = Left(TextData(3), Len(TextData(3)) - 1)
   TxtBalance2.Text = Left(TextData(4), Len(TextData(4)) - 1)
   TxtCardBase2.Text = Left(TextData(5), Len(TextData(5)) - 1)
   TxtCardBase1.Text = Left(TextData(6), Len(TextData(6)) - 1)
   TxtHPIBAddr.Text = Left(TextData(7), Len(TextData(7)) - 1)

   TxtDate.Text = Date$
   TxtTime.Text = Time$
   FileName = TxtFileName.Text
   PIO1Addr = CInt(Val("&H" & TxtCardBase2.Text))
   PIO2Addr = CInt(Val("&H" & TxtCardBase1.Text))
   UPCBase = CInt(Val("&H" & Txt601LAddress.Text))
   frmMain.MSComm1.Settings = TxtBalance1.Text
   frmMain.MSComm2.Settings = TxtBalance2.Text
   PIO1Init = False
   PIO2Init = False

   Call ListAdd
   Call INVISIBLE
```

End Sub

Sub Form_QueryUnload (Cancel As Integer, Unloadmode As Integer)

```
   Dim Title As String
   Dim Msg As String
   Dim DgDef As Integer
   Dim Response As Integer

   ' Put together a message box with all the proper components.
   Title = "Warning"
```

```
   Msg = "Do you really want to close the window?"
   DgDef = MB_YESNO + MB_ICONSTOP + MB_DEFBUTTON2

   Cancel = True

   ' Evaluate unload mode and give warning.
   If Unloadmode = 0 Then
         Response = MsgBox(Msg, DgDef, Title)
         If Response = IDYES Then
            Cancel = False
            Unload frmLogSet
         Else
            Cancel = True
         End If
   Else
            Cancel = False
   End If

End Sub

Sub Form_Unload (Cancel As Integer)

   FileExistanceFlag = True
   FileSearchEndFlag = False

End Sub

Sub INVISIBLE ()

   Text2.Visible = False
   Text3.Visible = False
   Command3D3.Visible = False
   Command3D4.Visible = False
   Command3D5.Visible = False
   Command3D6.Visible = False

   Txt601LAddress.Visible = False
   TxtHPIBAddr.Visible = False
   TxtBalance1.Visible = False
   TxtBalance2.Visible = False
```

```
    TxtCardBase1.Visible = False
    TxtCardBase2.Visible = False

    Label1.Visible = False
    Label44.Visible = False
    Label46.Visible = False
    Label48.Visible = False
    Label50.Visible = False
    Label51.Visible = False

End Sub

Sub List1_Click ()

    If List1.ListIndex = 2 Then
        List2.Clear
        List2.AddItem "Cocurrent flow", 0
        List2.AddItem "Countercurrent flow", 1
    End If

    If List1.ListIndex = 3 Then
        List2.Clear
        List2.AddItem "Cocurrent flow", 0
        List2.AddItem "Countercurrent flow", 1
    End If

    If List1.ListIndex = 0 Then
        List2.Clear
    End If

    If List1.ListIndex = 1 Then
        List2.Clear
    End If

    If List1.ListIndex = 4 Then
        List2.Clear
    End If

    If List1.ListIndex = 5 Then
        List2.Clear
```

```
    End If

    If List1.ListIndex = 6 Then
        List2.Clear
    End If

    If List1.ListIndex = 7 Then
        List2.Clear
    End If

End Sub

Sub ListAdd ()

    List1.AddItem "Abs. permeability measurement", 0
    List1.AddItem "Eff. permeability measurement", 1
    List1.AddItem "Steady state experiment", 2
    List1.AddItem "Unsteady state Experiment", 3
    List1.AddItem "Meter calibration @ Sw=0.0", 4
    List1.AddItem "Meter calibration @ Sw=1.0", 5
    List1.AddItem "Meter calibration @ Sw=Swirr", 6
    List1.AddItem "Meter calibration @ Sw=1-Sor", 7

End Sub

Sub Txt601LAddress_KeyPress (KeyAscii As Integer)

    Dim Tstr As String
    Dim Length As Integer
    Dim Ustr As String

    ' Scroll out the most significant digit when KeyAscii is a hex digit.
    If (InputHex(KeyAscii)) Then
        Tstr = Txt601LAddress.Text
        Length = 3
        Txt601LAddress.Text = Right$((Tstr + Chr$(KeyAscii)), Length)
        Ustr = Txt601LAddress.Text
        UPCBase = HexStrtoDec(Ustr)
    End If
```

```
    ' Intercept the character pressed if it is not hex character.
    KeyAscii = 0
    Beep

End Sub

Sub TxtCardBase1_KeyPress (KeyAscii As Integer)

    Dim Tstr As String
    Dim Length As Integer
    Dim Ustr As String

    ' Scroll out the most significant digit when KeyAscii is a hex digit.
    If (InputHex(KeyAscii)) Then
        Tstr = TxtCardBase1.Text
        Length = 3
        TxtCardBase1.Text = Right$((Tstr + Chr$(KeyAscii)), Length)
        Ustr = TxtCardBase1.Text
        PIO1Addr = HexStrtoDec(Ustr)
    End If

    ' Intercept the character pressed if it is not hex character.
    KeyAscii = 0
    Beep

End Sub

Sub TxtCardBase2_KeyPress (KeyAscii As Integer)

    Dim Tstr As String
    Dim Length As Integer
    Dim Ustr As String

    ' Scroll out the most significant digit when KeyAscii is a hex digit.
    If (InputHex(KeyAscii)) Then
        Tstr = TxtCardBase2.Text
        Length = 3
        TxtCardBase2.Text = Right$((Tstr + Chr$(KeyAscii)), Length)
        Ustr = TxtCardBase2.Text
        PIO1Addr = HexStrtoDec(Ustr)
```

```
    End If

    ' Intercept the character pressed if it is not hex character.
    KeyAscii = 0
    Beep

End Sub

Sub TxtDate_Click ()

    Dim Default, Msg, Newdate   ' Declare variables.

    Msg = "Enter a new date in the form mm-dd-yyyy"
    Default = Date  ' Current date.
    Newdate = InputBox(Msg, "", Default)   ' Get user input.
    If Len(Newdate) > 0 Then   ' Check if valid.
        Date = Newdate  ' Set date.
        Msg = "System date set to "
        Msg = Msg & Format(DateValue(Newdate), "dddd, mmmm d, yyyy")
    Else
        Msg = "You did not enter a valid date."
    End If
    MsgBox Msg  ' Display message.

End Sub

Sub TxtDate_DblClick ()

    TxtDate_Click

End Sub

Sub TxtDate_GotFocus ()

    ' Highlight the current entry.
    frmLogSet.TxtDate.SelStart = 0
    frmLogSet.TxtDate.SelLength = Len(frmLogSet.TxtDate.Text)

End Sub
```

```
Sub TxtFileName_Change ()

   FileExistanceFlag = True

End Sub

Sub TxtFileName_GotFocus ()

   ' Highlight the current entry.
   frmLogSet.TxtFileName.SelStart = 0
   frmLogSet.TxtFileName.SelLength = Len(frmLogSet.TxtFileName.Text)

End Sub

Sub TxtTime_Click ()

   Dim Msg, NewTime    ' Declare variables.
   Msg = "Enter a new time in the form hh:mm:ss"
   NewTime = InputBox(Msg, "", Time)   ' Get user input.
   If Len(NewTime) > 0 Then    ' Check if valid.
         Time = NewTime  ' Set time.
         Msg = "System time now set to "
         Msg = Msg & Time & "."  ' Put time in message.
   Else
         Msg = "You did not enter a valid time."
   End If
   MsgBox Msg  ' Display message.

End Sub

Sub TxtTime_DblClick ()

   TxtTime_Click

End Sub

Sub TxtUser_GotFocus ()

   ' Highlight the current entry.
   frmLogSet.TxtUser.SelStart = 0
```

```
   frmLogSet.TxtUser.SelLength = Len(frmLogSet.TxtUser.Text)

End Sub

Sub TxtUser_LostFocus ()

   If TxtUser.Text = "Samuel Chang" Then
         PassWordFlag = False
   End If

End Sub
```

Appendix B - 16 MAIN.BAS

```
'**********Global variables, library declarations and subprograms**********

' Declaration of global variables
' ***************************

Global FileName As String
Global PassWordFlag As Integer
Global FileExistanceFlag As Integer
Global FileSearchEndFlag As Integer
Global FileNumber As Integer
Global PIO1Addr As Integer
Global PIO2Addr As Integer
Global PIO1Init As Integer
Global PIO2Init As Integer
Global Cntrl1Addr As Integer
Global Cntrl2Addr As Integer
Global PortA1Addr As Integer
Global PortB1Addr As Integer
Global PortC1Addr As Integer
Global PortA2Addr As Integer
Global PortB2Addr As Integer
Global PortC2Addr As Integer
Global HPIBID As Integer
Global UPCBase As Integer
Global Pressure(16) As Single
Global HP432CMeasurement(200) As Single
Global HP436AMeasurement(200) As Single
Global HP432CUMeasurement(200) As Integer
Global HP432CLMeasurement(200) As Integer
Global HP436ARaw(200) As String
Dim HP436AData(200) As String
Global ExptIndex1 As Integer
Global ExptIndex2 As Integer
Global Weight1 As String * 20
Global Weight2 As String * 20
Global Range(200) As Integer
Global IntervalDistance As Single
Global Station As Integer
Global Scan As Integer
Global DistanceForward As Single
```

```
Global DistanceREverse As Single
Global IntervalPulse As Long
Global TotalPulse As Long
Global ActualTime(200) As Double
Global ArrayNum As Integer
Global PM432C(100) As Variant
Global PM436A(100) As Variant

Global Const SaturationTime# = 10#

' Declarations section for MHELP.VBX
' **********************************

Declare Function MhCtrlHwnd% Lib "mhelp.vbx" (A As Control)
Declare Function MhPeekByte% Lib "mhelp.vbx" (ByVal Segm%, ByVal
Ofst%)
Declare Sub MhPokeByte Lib "mhelp.vbx" (ByVal ByteVal%, ByVal Segm%,
ByVal Ofst%)
Declare Function Inp% Lib "mhelp.vbx" (ByVal PortNum%)
Declare Sub OUT Lib "mhelp.vbx" (ByVal PortNum%, ByVal DataByte%)
Declare Function VarPtr% Lib "mhelp.vbx" (Varbl As Any)
Declare Function VarSeg% Lib "mhelp.vbx" (Varbl As Any)
Declare Function VarSegPtr& Lib "mhelp.vbx" (Varbl As Any)
Declare Function SAdd% Lib "mhelp.vbx" (Lin$)
Declare Function SSeg% Lib "mhelp.vbx" (Lin$)
Declare Function SSegAdd& Lib "mhelp.vbx" (Lin$)


' Declaration section for PORTIO.DLL
' **********************************

Declare Function DasWordOut Lib "portio.dll" (ByVal port%, ByVal pdata%)
As Integer
Declare Function dasbyteout Lib "portio.dll" (ByVal port%, ByVal pdata%) As
Integer
Declare Function DasWordIn Lib "portio.dll" (ByVal port%) As Integer
Declare Function dasbytein Lib "portio.dll" (ByVal port%) As Integer

' Declaration section for HP Standard Instrument Control Library
' ***********************************************************
```

```
' Support levels:
Global Const I_SICL_REVISION = 38
Global Const I_SICL_LEVEL = 3

' Byte Ordering constants
Global Const I_ORDER_LE = True
Global Const I_ORDER_BE = False

' Session types
Global Const I_SESS_INTF = 1
Global Const I_SESS_DEV = 2
Global Const I_SESS_CMDR = 3

' Interface Types
Global Const I_INTF_NONE = 0
Global Const I_INTF_GPIB = 1
Global Const I_INTF_VXI = 2
Global Const I_INTF_RS232 = 3
' 5 is reserved -- don't use
Global Const I_INTF_USRDEF = 6
' 7 is reserved -- don't use

' iread termination conditions
Global Const I_TERM_MAXCNT = 1
Global Const I_TERM_CHR = 2
Global Const I_TERM_END = 4
Global Const I_TERM_NON_BLOCKED = 8

' ixtrig which values.
Global Const I_TRIG_STD = &H1
Global Const I_TRIG_ALL = &HFFFFFFFF
Global Const I_TRIG_TTL0 = &H1000
Global Const I_TRIG_TTL1 = &H2000
Global Const I_TRIG_TTL2 = &H4000
Global Const I_TRIG_TTL3 = &H8000
Global Const I_TRIG_TTL4 = &H10000
Global Const I_TRIG_TTL5 = &H20000
Global Const I_TRIG_TTL6 = &H40000
Global Const I_TRIG_TTL7 = &H80000
```

```
Global Const I_TRIG_ECL0 = &H100000
Global Const I_TRIG_ECL1 = &H200000
Global Const I_TRIG_ECL2 = &H400000
Global Const I_TRIG_ECL3 = &H800000
Global Const I_TRIG_EXT0 = &H1000000
Global Const I_TRIG_EXT1 = &H2000000
Global Const I_TRIG_EXT2 = &H4000000
Global Const I_TRIG_EXT3 = &H8000000
Global Const I_TRIG_CLK0 = &H10000000
Global Const I_TRIG_CLK1 = &H20000000
Global Const I_TRIG_CLK2 = &H40000000
Global Const I_TRIG_CLK10 = &H80000000
Global Const I_TRIG_CLK100 = &H800
Global Const I_TRIG_SERIAL_DTR = &H400
Global Const I_TRIG_SERIAL_RTS = &H200
Global Const I_TRIG_GPIB_CTL1 = &H80

' ihint values
Global Const I_HINT_DONTCARE = 0
Global Const I_HINT_USEDMA = 1
Global Const I_HINT_USEPOLL = 2
Global Const I_HINT_USEINTR = 3
Global Const I_HINT_SYSTEM = 4
Global Const I_HINT_IO = 5

' isetintr values.  1-15 are interface independant.
Global Const I_INTR_OFF = 0
Global Const I_INTR_INTFACT = 1
Global Const I_INTR_INTFDEACT = 2
Global Const I_INTR_TRIG = 3
Global Const I_INTR_STB = 4
Global Const I_INTR_DEVCLR = 5

' VXI Interrupts
Global Const I_INTR_VXI_SIGNAL = 16
Global Const I_INTR_VXI_SYSRESET = 17
Global Const I_INTR_VXI_VME = 18
Global Const I_INTR_VXI_LLOCK = 19
Global Const I_INTR_VXI_UKNSIG = 20
```

```
' GP-IB Interrupts
Global Const I_INTR_GPIB_IFC = 16
Global Const I_INTR_GPIB_PPOLLCONFIG = 17
Global Const I_INTR_GPIB_REMLOC = 18
Global Const I_INTR_GPIB_GET = 20
Global Const I_INTR_GPIB_TLAC = 21

' RS-232 Interrupts
Global Const I_INTR_SERIAL_DAV = 16
Global Const I_INTR_SERIAL_MSL = 17
Global Const I_INTR_SERIAL_BREAK = 18
Global Const I_INTR_SERIAL_ERROR = 19
Global Const I_INTR_SERIAL_TEMT = 20
Global Const I_INTR_SERIAL_MCL = 21

' 32 maximum isetintr values
Global Const I_INTR_MAX = 32

' ivxibusstatus values
Global Const I_VXI_BUS_TRIGGER = 0
Global Const I_VXI_BUS_LADDR = 1
Global Const I_VXI_BUS_SERVANT_AREA = 2
Global Const I_VXI_BUS_NORMOP = 3
Global Const I_VXI_BUS_CMDR_LADDR = 4
Global Const I_VXI_BUS_MAN_ID = 5
Global Const I_VXI_BUS_MODEL_ID = 6
Global Const I_VXI_BUS_PROTOCOL = 7
Global Const I_VXI_BUS_XPROT = 8
Global Const I_VXI_BUS_SHM_SIZE = 9
Global Const I_VXI_BUS_SHM_ADDR_SPACE = 10
Global Const I_VXI_SHM_PAGE = 11
Global Const I_VXI_BUS_VXIMXI = 12
Global Const I_VXI_BUS_TRIGSUPP = 13

' igpibbusstatus values
Global Const I_GPIB_BUS_REM = 1
Global Const I_GPIB_BUS_SRQ = 2
Global Const I_GPIB_BUS_NDAC = 3
Global Const I_GPIB_BUS_SYSCTLR = 4
Global Const I_GPIB_BUS_ACTCTLR = 5
```

```
Global Const I_GPIB_BUS_TALKER = 6
Global Const I_GPIB_BUS_LISTENER = 7
Global Const I_GPIB_BUS_ADDR = 8
Global Const I_GPIB_BUS_LINES = 9

' RS-232 values
Global Const I_SERIAL_BAUD = 1
Global Const I_SERIAL_PARITY = 2
Global Const I_SERIAL_STOP = 3
Global Const I_SERIAL_WIDTH = 4
Global Const I_SERIAL_FLOW_CTRL = 5
Global Const I_SERIAL_MSL = 6
Global Const I_SERIAL_STAT = 7
Global Const I_SERIAL_RESET = 9
Global Const I_SERIAL_READ_EOI = 10
Global Const I_SERIAL_WRITE_EOI = 11
Global Const I_SERIAL_DUPLEX = 12
Global Const I_SERIAL_READ_BUFSZ = 13
Global Const I_SERIAL_READ_DAV = 14

' RS-232 duplex modes
Global Const I_SERIAL_DUPLEX_HALF = 1
Global Const I_SERIAL_DUPLEX_FULL = 2

' RS-232 UART status
Global Const I_SERIAL_DAV = &H1
Global Const I_SERIAL_OVERFLOW = &H2
Global Const I_SERIAL_PARERR = &H4
Global Const I_SERIAL_FRAMING = &H8
Global Const I_SERIAL_BREAK = &H10
Global Const I_SERIAL_TEMT = &H20

' RS-232 flow control
Global Const I_SERIAL_FLOW_NONE = 0
Global Const I_SERIAL_FLOW_XON = 1
Global Const I_SERIAL_FLOW_RTS_CTS = 2
Global Const I_SERIAL_FLOW_DTR_DSR = 3

' RS-232 modem status lines
Global Const I_SERIAL_DCD = &H1
```

```
Global Const I_SERIAL_DSR = &H2
Global Const I_SERIAL_CTS = &H4
Global Const I_SERIAL_RI = &H8
Global Const I_SERIAL_D_DCD = &H10
Global Const I_SERIAL_D_DSR = &H20
Global Const I_SERIAL_D_CTS = &H40
Global Const I_SERIAL_D_TERI = &H80

' RS-232 modem control lines
Global Const I_SERIAL_RTS = &H1000
Global Const I_SERIAL_DTR = &H2000

' RS-232 parity values
Global Const I_SERIAL_PAR_NONE = 0
Global Const I_SERIAL_PAR_EVEN = 1
Global Const I_SERIAL_PAR_ODD = 2
Global Const I_SERIAL_PAR_MARK = 3
Global Const I_SERIAL_PAR_SPACE = 4
Global Const I_SERIAL_PAR_IGNORE = 5

' RS-232 stop-bit values
Global Const I_SERIAL_STOP_1 = 1
Global Const I_SERIAL_STOP_2 = 2

' RS-232 character width
Global Const I_SERIAL_CHAR_5 = 5
Global Const I_SERIAL_CHAR_6 = 6
Global Const I_SERIAL_CHAR_7 = 7
Global Const I_SERIAL_CHAR_8 = 8

' EOI support (used with the I_SERIAL_*_EOI command)
Global Const I_SERIAL_EOI_CHR = &H100
Global Const I_SERIAL_EOI_NONE = &H200
Global Const I_SERIAL_EOI_BIT8 = &H400

' imap mapspace values
Global Const I_MAP_A16 = &H0
Global Const I_MAP_A24 = &H1
Global Const I_MAP_A32 = &H2
Global Const I_MAP_VXIDEV = &H3
```

```
Global Const I_MAP_EXTEND = &H4
Global Const I_MAP_INTFREG = &H5
Global Const I_MAP_SHARED = &H6

' Error Codes
' NOTE that User Error Codes 32501-32630 are reserved
' for HP SICL.
Const SICL_ERR_BASE = 32501
Global Const I_ERR_NOERROR = 0
Global Const I_ERR_SYNTAX = SICL_ERR_BASE
Global Const I_ERR_SYMNAME = 1 + SICL_ERR_BASE
Global Const I_ERR_BADADDR = 2 + SICL_ERR_BASE
Global Const I_ERR_BADID = 3 + SICL_ERR_BASE
Global Const I_ERR_PARAM = 4 + SICL_ERR_BASE
Global Const I_ERR_NOCONN = 5 + SICL_ERR_BASE
Global Const I_ERR_NOPERM = 6 + SICL_ERR_BASE
Global Const I_ERR_NOTSUPP = 7 + SICL_ERR_BASE
Global Const I_ERR_NORSRC = 8 + SICL_ERR_BASE
Global Const I_ERR_NOINTF = 9 + SICL_ERR_BASE
Global Const I_ERR_LOCKED = 10 + SICL_ERR_BASE
Global Const I_ERR_NOLOCK = 11 + SICL_ERR_BASE
Global Const I_ERR_BADFMT = 12 + SICL_ERR_BASE
Global Const I_ERR_DATA = 13 + SICL_ERR_BASE
Global Const I_ERR_TIMEOUT = 14 + SICL_ERR_BASE
Global Const I_ERR_OVERFLOW = 15 + SICL_ERR_BASE
Global Const I_ERR_IO = 16 + SICL_ERR_BASE
Global Const I_ERR_OS = 17 + SICL_ERR_BASE
Global Const I_ERR_BADMAP = 18 + SICL_ERR_BASE
Global Const I_ERR_NODEV = 19 + SICL_ERR_BASE
Global Const I_ERR_INVLADDR = 20 + SICL_ERR_BASE
Global Const I_ERR_NOTIMPL = 21 + SICL_ERR_BASE
Global Const I_ERR_ABORTED = 22 + SICL_ERR_BASE
Global Const I_ERR_BADCONFIG = 23 + SICL_ERR_BASE
Global Const I_ERR_NOCMDR = 24 + SICL_ERR_BASE
Global Const I_ERR_VERSION = 25 + SICL_ERR_BASE
Global Const I_ERR_NESTEDIO = 26 + SICL_ERR_BASE
Global Const I_ERR_BUSY = 27 + SICL_ERR_BASE
Global Const I_ERR_INTERNAL = 127 + SICL_ERR_BASE
Global Const I_ERR_INTERRUPT = 128 + SICL_ERR_BASE
Global Const I_ERR_UNKNOWNERR = 129 + SICL_ERR_BASE
```

```
Global Const SICL_ERR_LAST = I_ERR_UNKNOWNERR

' Data Types used by SICL
Type lu_info
logical_unit As Long
symname As String
cardname As String
filler1 As Long
intftype As Long
Location As Long
busaddr As Long
hwarg(0 To 15) As String
filler2(0 To 11) As Long
End Type

Type vxiinfo
laddr As Integer
Name As String
manuf_name As String
model_name As String
man_id As Integer
model As Integer
devclass As Integer
selftest As Integer
cage_num As Integer
slot As Integer
protocol As Integer
x_protocol As Integer
servant_area As Integer
addrspace As Integer
memsize As Integer
memstart As Integer
slot0_laddr As Integer
cmdr_laddr As Integer
int_handler(0 To 7) As Integer
interrupter(0 To 7) As Integer
fill(0 To 9) As Integer
End Type

' Version Information
```

```
Declare Sub iversion Lib "sicl16.dll" (ByVal ID As Integer, specversion As
Integer, implversion As Integer)
Declare Sub idrvrversion Lib "sicl16.dll" (ByVal ID As Integer, specversion As
Integer, implversion As Integer)

' Open/Close
Declare Function iopen Lib "sicl16.dll" Alias "vbopen" (ByVal addr As String)
As Integer
Declare Sub iclose Lib "sicl16.dll" (ByVal ID As Integer)
Declare Function igetintfsess Lib "sicl16.dll" (ByVal ID As Integer) As Integer

' Write/Read
Declare Sub iwrite Lib "sicl16.dll" (ByVal ID As Integer, buf As Any, ByVal
datalen As Long, ByVal endi As Integer, Actual As Long)
Declare Sub iread Lib "sicl16.dll" (ByVal ID As Integer, buf As Any, ByVal
bufsize As Long, reason As Any, Actual As Long)
Declare Sub itermchr Lib "sicl16.dll" (ByVal ID As Integer, ByVal tchr As
Integer)
Declare Sub igettermchr Lib "sicl16.dll" (ByVal ID As Integer, tchr As Integer)

' Formatted I/O
Declare Function ivprintf Lib "sicl16.dll" Alias "vbvprintf" (ByVal ID As
Integer, ByVal fmt As String, ap As Any) As Integer
Declare Function isvprintf Lib "sicl16.dll" Alias "vbsvprintf" (ByVal user_buf
As String, ByVal fmt As String, ap As Any) As Integer
Declare Function ivscanf Lib "sicl16.dll" Alias "vbvscanf" (ByVal ID As Integer,
ByVal fmt As String, ap As Any) As Integer
Declare Function isvscanf Lib "sicl16.dll" Alias "vbsvscanf" (ByVal user_buf As
String, ByVal fmt As String, ap As Any) As Integer
Declare Sub ifwrite Lib "sicl16.dll" (ByVal ID As Integer, ByVal buf As Any,
ByVal datalen, ByVal endi As Integer, Actual As Long)
Declare Sub ifread Lib "sicl16.dll" (ByVal ID As Integer, ByVal buf As Any,
ByVal bufsize As Long, reason As Any, Actual As Long)
Declare Sub iflush Lib "sicl16.dll" (ByVal ID As Integer, ByVal mask As
Integer)
Declare Sub isetbuf Lib "sicl16.dll" (ByVal ID As Integer, ByVal mask As
Integer, ByVal size As Integer)

' Device/Interface Control
Declare Sub iclear Lib "sicl16.dll" (ByVal ID As Integer)
```

Declare Sub iabort Lib "sicl16.dll" (ByVal ID As Integer)
Declare Sub ilocal Lib "sicl16.dll" (ByVal ID As Integer)
Declare Sub iremote Lib "sicl16.dll" (ByVal ID As Integer)
Declare Sub ireadstb Lib "sicl16.dll" (ByVal ID As Integer, ByVal stb As String)
Declare Sub itrigger Lib "sicl16.dll" (ByVal ID As Integer)
Declare Sub ixtrig Lib "sicl16.dll" (ByVal ID As Integer, ByVal which As Long)
Declare Sub ihint Lib "sicl16.dll" (ByVal ID As Integer, ByVal hint As Integer)

' Commander Sessions
Declare Sub isetstb Lib "sicl16.dll" (ByVal ID As Integer, ByVal stb As Integer)

' Locking
Declare Sub ilock Lib "sicl16.dll" (ByVal ID As Integer)
Declare Sub iunlock Lib "sicl16.dll" (ByVal ID As Integer)
Declare Sub isetlockwait Lib "sicl16.dll" (ByVal ID As Integer, ByVal flag As Integer)
Declare Sub igetlockwait Lib "sicl16.dll" (ByVal ID As Integer, flag As Integer)

' Timeouts
Declare Sub itimeout Lib "sicl16.dll" (ByVal ID As Integer, ByVal tval As Long)
Declare Sub igettimeout Lib "sicl16.dll" (ByVal ID As Integer, tval As Long)

' Misc routines
Declare Sub igetaddr Lib "sicl16.dll" Alias "vbgetaddr" (ByVal ID As Integer, ByVal addr As String)
Declare Sub igetintftype Lib "sicl16.dll" (ByVal ID As Integer, pdata As Integer)
Declare Sub igetsesstype Lib "sicl16.dll" (ByVal ID As Integer, pdata As Integer)
Declare Sub igetdevaddr Lib "sicl16.dll" (ByVal ID As Integer, prim As Integer, sec As Integer)
Declare Sub igetlu Lib "sicl16.dll" (ByVal ID As Integer, lu As Integer)
Declare Sub ibeswap Lib "sicl16.dll" (addr As Any, ByVal length As Long, ByVal datasize As Integer)
Declare Sub ileswap Lib "sicl16.dll" (addr As Any, ByVal length As Long, ByVal datasize As Integer)
Declare Sub iswap Lib "sicl16.dll" (addr As Any, ByVal length As Long, ByVal

```
datasize As Integer)
Declare Sub igetlulist Lib "sicl16.dll" Alias "vbgetlulist" (list() As Integer)
Declare Sub igetluinfo Lib "sicl16.dll" Alias "vbgetluinfo" (ByVal lu As Integer,
result As lu_info)
Declare Sub igetgatewaytype Lib "sicl16.dll" (ByVal ID As Integer, pdata As
Integer)

' Error Handling
Declare Function igeterrno Lib "sicl16.dll" Alias "vbgeterrno" () As Integer
Declare Function igeterrstr Lib "sicl16.dll" Alias "vbgeterrstr" (ByVal errcode
As Integer) As String
Declare Sub icauseerr Lib "sicl16.dll" (ByVal ID As Integer, ByVal errcode As
Integer, ByVal flag As Integer)
Declare Sub vbsetsiclerrbase Lib "sicl16.dll" (ByVal errbase As Integer)

' RS-232 specific routines
Declare Sub iserialmclctrl Lib "sicl16.dll" (ByVal ID As Integer, ByVal sline As
Integer, ByVal state As Integer)
Declare Sub iserialmclstat Lib "sicl16.dll" (ByVal ID As Integer, ByVal sline As
Integer, state As Integer)
Declare Sub iserialctrl Lib "sicl16.dll" (ByVal ID As Integer, ByVal request As
Integer, ByVal Setting As Long)
Declare Sub iserialstat Lib "sicl16.dll" (ByVal ID As Integer, ByVal request As
Integer, result As Long)
Declare Sub iserialbreak Lib "sicl16.dll" (ByVal ID As Integer)

' VXI Specific routines
Declare Sub ivxibusstatus Lib "sicl16.dll" (ByVal ID As Integer, ByVal request
As Integer, result As Long)
Declare Sub ivxiwaitnormop Lib "sicl16.dll" (ByVal ID As Integer)
Declare Sub ivxitrigon Lib "sicl16.dll" (ByVal ID As Integer, ByVal which As
Long)
Declare Sub ivxitrigoff Lib "sicl16.dll" (ByVal ID As Integer, ByVal which As
Long)
Declare Sub ivxitrigroute Lib "sicl16.dll" (ByVal ID As Integer, ByVal
in_which As Long, ByVal out_which As Long)
Declare Sub ivxigettrigroute Lib "sicl16.dll" (ByVal ID As Integer, ByVal
which As Long, route As Long)
Declare Sub ivxiws Lib "sicl16.dll" (ByVal ID As Integer, ByVal wscmd As
Integer, wsresp As Integer, rpe As Integer)
```

```
Declare Sub ivxiservants Lib "sicl16.dll" Alias "vbvxiservants" (ByVal ID As
Integer, ByVal maxnum As Integer, list() As Integer)
Declare Sub ivxirminfo Lib "sicl16.dll" Alias "vbvxirminfo" (ByVal ID As
Integer, ByVal laddr As Integer, info As vxiinfo)

' GP-IB Specific Details
Declare Sub igpibbusstatus Lib "sicl16.dll" (ByVal ID As Integer, ByVal
request As Integer, result As Integer)
Declare Sub igpibppoll Lib "sicl16.dll" (ByVal ID As Integer, result As Integer)
Declare Sub igpibppollconfig Lib "sicl16.dll" (ByVal ID As Integer, ByVal cval
As Integer)
Declare Sub igpibppollresp Lib "sicl16.dll" (ByVal ID As Integer, ByVal sval
As Integer)
Declare Sub igpibpassctl Lib "sicl16.dll" (ByVal ID As Integer, ByVal busaddr
As Integer)
Declare Sub igpibrenctl Lib "sicl16.dll" (ByVal ID As Integer, ByVal ren As
Integer)
Declare Sub igpibatnctl Lib "sicl16.dll" (ByVal ID As Integer, ByVal atnval As
Integer)
Declare Sub igpibsendcmd Lib "sicl16.dll" (ByVal ID As Integer, ByVal buf As
String, ByVal length As Integer)
Declare Sub igpibllo Lib "sicl16.dll" (ByVal ID As Integer)
Declare Sub igpibbusaddr Lib "sicl16.dll" (ByVal ID As Integer, ByVal busaddr
As Integer)
Declare Sub igpibgett1delay Lib "sicl16.dll" (ByVal ID As Integer, Delay As
Integer)
Declare Sub igpibsett1delay Lib "sicl16.dll" (ByVal ID As Integer, ByVal Delay
As Integer)

' Map routines
Declare Function imap Lib "sicl16.dll" (ByVal ID As Integer, ByVal mapspace
As Integer, ByVal pagestart As Integer, ByVal pagecnt As Integer, ByVal
suggested As Long) As Long
Declare Sub iunmap Lib "sicl16.dll" (ByVal ID As Integer, ByVal addr As
Long, ByVal mapspace As Integer, ByVal pagestart As Integer, ByVal pagecnt
As Integer)
Declare Sub imapinfo Lib "sicl16.dll" (ByVal ID As Integer, ByVal mapspace
As Integer, numwindows As Integer, winsize As Integer)

' Block copy and fifo routines
```

```
Declare Sub ibblockcopy Lib "sicl16.dll" (ByVal ID As Integer, src As Any,
dest As Any, ByVal cnt As Long)
Declare Sub iwblockcopy Lib "sicl16.dll" (ByVal ID As Integer, src As Any,
dest As Any, ByVal cnt As Long, ByVal swap As Integer)
Declare Sub ilblockcopy Lib "sicl16.dll" (ByVal ID As Integer, src As Any, dest
As Any, ByVal cnt As Long, ByVal swap As Integer)
Declare Sub ibpushfifo Lib "sicl16.dll" (ByVal ID As Integer, src As Any, fifo
As Any, ByVal cnt As Long)
Declare Sub iwpushfifo Lib "sicl16.dll" (ByVal ID As Integer, src As Any, fifo
As Any, ByVal cnt As Long, ByVal swap As Integer)
Declare Sub ilpushfifo Lib "sicl16.dll" (ByVal ID As Integer, src As Any, fifo
As Any, ByVal cnt As Long, ByVal swap As Integer)
Declare Sub ibpopfifo Lib "sicl16.dll" (ByVal ID As Integer, fifo As Any, dest
As Any, ByVal cnt As Long)
Declare Sub iwpopfifo Lib "sicl16.dll" (ByVal ID As Integer, fifo As Any, dest
As Any, ByVal cnt As Long, ByVal swap As Integer)
Declare Sub ilpopfifo Lib "sicl16.dll" (ByVal ID As Integer, fifo As Any, dest
As Any, ByVal cnt As Long, ByVal swap As Integer)
Declare Sub icmd Lib "sicl16.dll" (ByVal ID As Integer, ByVal cmd As Long,
ByVal datalen As Integer, ByVal datawidth As Integer, pdata As Any)

' Windows 3.1 Cleanup routines
Declare Sub siclcleanup Lib "sicl16.dll" Alias "_siclcleanup" ()

' Windows 3.1 yield control routine
Declare Sub setsiclyield Lib "sicl16.dll" Alias "_setsiclyield" (ByVal
yield_option As Integer)

' Peek/Poke routines
Declare Sub ibpoke Lib "sicl16.dll" Alias "vbibpoke" (ByVal addr As Long,
ByVal Value As Integer)
Declare Sub iwpoke Lib "sicl16.dll" Alias "vbiwpoke" (ByVal addr As Long,
ByVal Value As Integer)
Declare Sub ilpoke Lib "sicl16.dll" Alias "ilpoke" (ByVal addr As Long, ByVal
Value As Long)
Declare Function ibpeek Lib "sicl16.dll" Alias "vbibpeek" (ByVal addr As Long)
As Integer
Declare Function iwpeek Lib "sicl16.dll" Alias "vbiwpeek" (ByVal addr As
Long) As Integer
Declare Function ilpeek Lib "sicl16.dll" Alias "ilpeek" (ByVal addr As Long) As
```

```
Long

' Visual Basic global constant section
' ********************************

' Clipboard formats
Global Const CF_LINK = &HBF00
Global Const CF_TEXT = 1
Global Const CF_BITMAP = 2
Global Const CF_METAFILE = 3
Global Const CF_DIB = 8
Global Const CF_PALETTE = 9

' DragOver
Global Const ENTER = 0
Global Const LEAVE = 1
Global Const OVER = 2

' Drag (controls)
Global Const CANCEL = 0
Global Const BEGIN_DRAG = 1
Global Const END_DRAG = 2

' Show parameters
Global Const MODAL = 1
Global Const MODELESS = 0

' Arrange Method
' for MDI Forms
Global Const CASCADE = 0
Global Const TILE_HORIZONTAL = 1
Global Const TILE_VERTICAL = 2
Global Const ARRANGE_ICONS = 3

'ZOrder Method
Global Const BRINGTOFRONT = 0
Global Const SENDTOBACK = 1

' Key Codes
```

```
Global Const KEY_LBUTTON = &H1
Global Const KEY_RBUTTON = &H2
Global Const KEY_CANCEL = &H3
Global Const KEY_MBUTTON = &H4    ' NOT contiguous with L &
RBUTTON
Global Const KEY_BACK = &H8
Global Const KEY_TAB = &H9
Global Const KEY_CLEAR = &HC
Global Const KEY_RETURN = &HD
Global Const KEY_SHIFT = &H10
Global Const KEY_CONTROL = &H11
Global Const KEY_MENU = &H12
Global Const KEY_PAUSE = &H13
Global Const KEY_CAPITAL = &H14
Global Const KEY_ESCAPE = &H1B
Global Const KEY_SPACE = &H20
Global Const KEY_PRIOR = &H21
Global Const KEY_NEXT = &H22
Global Const KEY_END = &H23
Global Const KEY_HOME = &H24
Global Const KEY_LEFT = &H25
Global Const KEY_UP = &H26
Global Const KEY_RIGHT = &H27
Global Const KEY_DOWN = &H28
Global Const KEY_SELECT = &H29
Global Const KEY_PRINT = &H2A
Global Const KEY_EXECUTE = &H2B
Global Const KEY_SNAPSHOT = &H2C
Global Const KEY_INSERT = &H2D
Global Const KEY_DELETE = &H2E
Global Const KEY_HELP = &H2F

' KEY_A thru KEY_Z are the same as their ASCII equivalents: 'A' thru 'Z'
' KEY_0 thru KEY_9 are the same as their ASCII equivalents: '0' thru '9'

Global Const KEY_NUMPAD0 = &H60
Global Const KEY_NUMPAD1 = &H61
Global Const KEY_NUMPAD2 = &H62
Global Const KEY_NUMPAD3 = &H63
Global Const KEY_NUMPAD4 = &H64
```

```
Global Const KEY_NUMPAD5 = &H65
Global Const KEY_NUMPAD6 = &H66
Global Const KEY_NUMPAD7 = &H67
Global Const KEY_NUMPAD8 = &H68
Global Const KEY_NUMPAD9 = &H69
Global Const KEY_MULTIPLY = &H6A
Global Const KEY_ADD = &H6B
Global Const KEY_SEPARATOR = &H6C
Global Const KEY_SUBTRACT = &H6D
Global Const KEY_DECIMAL = &H6E
Global Const KEY_DIVIDE = &H6F
Global Const KEY_F1 = &H70
Global Const KEY_F2 = &H71
Global Const KEY_F3 = &H72
Global Const KEY_F4 = &H73
Global Const KEY_F5 = &H74
Global Const KEY_F6 = &H75
Global Const KEY_F7 = &H76
Global Const KEY_F8 = &H77
Global Const KEY_F9 = &H78
Global Const KEY_F10 = &H79
Global Const KEY_F11 = &H7A
Global Const KEY_F12 = &H7B
Global Const KEY_F13 = &H7C
Global Const KEY_F14 = &H7D
Global Const KEY_F15 = &H7E
Global Const KEY_F16 = &H7F

Global Const KEY_NUMLOCK = &H90

' Variant VarType tags

Global Const V_EMPTY = 0
Global Const V_NULL = 1
Global Const V_INTEGER = 2
Global Const V_LONG = 3
Global Const V_SINGLE = 4
Global Const V_DOUBLE = 5
Global Const V_CURRENCY = 6
Global Const V_DATE = 7
```

```
Global Const V_STRING = 8

' Event Parameters

' ErrNum (LinkError)
Global Const WRONG_FORMAT = 1
Global Const DDE_SOURCE_CLOSED = 6
Global Const TOO_MANY_LINKS = 7
Global Const DATA_TRANSFER_FAILED = 8

' QueryUnload
Global Const FORM_CONTROLMENU = 0
Global Const FORM_CODE = 1
Global Const APP_WINDOWS = 2
Global Const APP_TASKMANAGER = 3
Global Const FORM_MDIFORM = 4

' Properties

' Colors
Global Const BLACK = &H0&
Global Const RED = &HFF&
Global Const GREEN = &HFF00&
Global Const YELLOW = &HFFFF&
Global Const BLUE = &HFF0000
Global Const MAGENTA = &HFF00FF
Global Const CYAN = &HFFFF00
Global Const WHITE = &HFFFFFF

' System Colors
Global Const SCROLL_BARS = &H80000000         ' Scroll-bars gray area.
Global Const DESKTOP = &H80000001             ' Desktop.
Global Const ACTIVE_TITLE_BAR = &H80000002     ' Active window
caption.
Global Const INACTIVE_TITLE_BAR = &H80000003   ' Inactive window
caption.
Global Const MENU_BAR = &H80000004             ' Menu background.
Global Const WINDOW_BACKGROUND = &H80000005    ' Window
background.
Global Const WINDOW_FRAME = &H80000006         ' Window frame.
```

```
Global Const MENU_TEXT = &H80000007          ' Text in menus.
Global Const WINDOW_TEXT = &H80000008         ' Text in windows.
Global Const TITLE_BAR_TEXT = &H80000009       ' Text in caption, size
box, scroll-bar arrow box..
Global Const ACTIVE_BORDER = &H8000000A        ' Active window
border.
Global Const INACTIVE_BORDER = &H8000000B      ' Inactive window
border.
Global Const APPLICATION_WORKSPACE = &H8000000C ' Background
color of multiple document interface (MDI) applications.
Global Const HIGHLIGHT = &H8000000D          ' Items selected item in a
control.
Global Const HIGHLIGHT_TEXT = &H8000000E       ' Text of item selected
in a control.
Global Const BUTTON_FACE = &H8000000F         ' Face shading on
command buttons.
Global Const BUTTON_SHADOW = &H80000010        ' Edge shading on
command buttons.
Global Const GRAY_TEXT = &H80000011          ' Grayed (disabled) text.
This color is set to 0 if the current display driver does not support a solid gray
color.
Global Const BUTTON_TEXT = &H80000012         ' Text on push buttons.

' Enumerated Types

' Align (picture box)
Global Const NONE = 0
Global Const ALIGN_TOP = 1
Global Const ALIGN_BOTTOM = 2

' Alignment
Global Const LEFT_JUSTIFY = 0  ' 0 - Left Justify
Global Const RIGHT_JUSTIFY = 1 ' 1 - Right Justify
Global Const CENTER = 2        ' 2 - Center

' BorderStyle (form)
'Global Const NONE = 0         ' 0 - None
Global Const FIXED_SINGLE = 1  ' 1 - Fixed Single
Global Const SIZABLE = 2       ' 2 - Sizable (Forms only)
Global Const FIXED_DOUBLE = 3  ' 3 - Fixed Double (Forms only)
```

```
' BorderStyle (Shape and Line)
'Global Const TRANSPARENT = 0   '0 - Transparent
'Global Const SOLID = 1         '1 - Solid
'Global Const DASH = 2          ' 2 - Dash
'Global Const DOT = 3           ' 3 - Dot
'Global Const DASH_DOT = 4      ' 4 - Dash-Dot
'Global Const DASH_DOT_DOT = 5 ' 5 - Dash-Dot-Dot
'Global Const INSIDE_SOLID = 6 ' 6 - Inside Solid

' MousePointer
Global Const DEFAULT = 0        ' 0 - Default
Global Const ARROW = 1          ' 1 - Arrow
Global Const CROSSHAIR = 2      ' 2 - Cross
Global Const IBEAM = 3          ' 3 - I-Beam
Global Const ICON_POINTER = 4   ' 4 - Icon
Global Const SIZE_POINTER = 5   ' 5 - Size
Global Const SIZE_NE_SW = 6     ' 6 - Size NE SW
Global Const SIZE_N_S = 7       ' 7 - Size N S
Global Const SIZE_NW_SE = 8     ' 8 - Size NW SE
Global Const SIZE_W_E = 9       ' 9 - Size W E
Global Const UP_ARROW = 10      ' 10 - Up Arrow
Global Const HOURGLASS = 11     ' 11 - Hourglass
Global Const NO_DROP = 12       ' 12 - No drop

' DragMode
Global Const MANUAL = 0    ' 0 - Manual
Global Const AUTOMATIC = 1 ' 1 - Automatic

' DrawMode
Global Const BLACKNESS = 1      ' 1 - Blackness
Global Const NOT_MERGE_PEN = 2  ' 2 - Not Merge Pen
Global Const MASK_NOT_PEN = 3   ' 3 - Mask Not Pen
Global Const NOT_COPY_PEN = 4   ' 4 - Not Copy Pen
Global Const MASK_PEN_NOT = 5   ' 5 - Mask Pen Not
Global Const INVERT = 6         ' 6 - Invert
Global Const XOR_PEN = 7        ' 7 - Xor Pen
Global Const NOT_MASK_PEN = 8   ' 8 - Not Mask Pen
Global Const MASK_PEN = 9       ' 9 - Mask Pen
Global Const NOT_XOR_PEN = 10   ' 10 - Not Xor Pen
```

```
Global Const NOP = 11          ' 11 - Nop
Global Const MERGE_NOT_PEN = 12 ' 12 - Merge Not Pen
Global Const COPY_PEN = 13      ' 13 - Copy Pen
Global Const MERGE_PEN_NOT = 14 ' 14 - Merge Pen Not
Global Const MERGE_PEN = 15     ' 15 - Merge Pen
Global Const WHITENESS = 16     ' 16 - Whiteness

' DrawStyle
Global Const SOLID = 0       ' 0 - Solid
Global Const DASH = 1        ' 1 - Dash
Global Const DOT = 2         ' 2 - Dot
Global Const DASH_DOT = 3    ' 3 - Dash-Dot
Global Const DASH_DOT_DOT = 4 ' 4 - Dash-Dot-Dot
Global Const INVISIBLE = 5   ' 5 - Invisible
Global Const INSIDE_SOLID = 6 ' 6 - Inside Solid

' FillStyle
' Global Const SOLID = 0          ' 0 - Solid
Global Const TRANSPARENT = 1      ' 1 - Transparent
Global Const HORIZONTAL_LINE = 2  ' 2 - Horizontal Line
Global Const VERTICAL_LINE = 3    ' 3 - Vertical Line
Global Const UPWARD_DIAGONAL = 4  ' 4 - Upward Diagonal
Global Const DOWNWARD_DIAGONAL = 5 ' 5 - Downward Diagonal
Global Const CROSS = 6            ' 6 - Cross
Global Const DIAGONAL_CROSS = 7   ' 7 - Diagonal Cross

' LinkMode (forms and controls)
' Global Const NONE = 0        ' 0 - None
Global Const LINK_SOURCE = 1    ' 1 - Source (forms only)
Global Const LINK_AUTOMATIC = 1 ' 1 - Automatic (controls only)
Global Const LINK_MANUAL = 2    ' 2 - Manual (controls only)
Global Const LINK_NOTIFY = 3    ' 3 - Notify (controls only)

' LinkMode (kept for VB1.0 compatibility, use new constants instead)
Global Const HOT = 1    ' 1 - Hot (controls only)
Global Const SERVER = 1 ' 1 - Server (forms only)
Global Const COLD = 2   ' 2 - Cold (controls only)

' ScaleMode
Global Const USER = 0      ' 0 - User
```

```
Global Const TWIPS = 1       ' 1 - Twip
Global Const POINTS = 2      ' 2 - Point
Global Const PIXELS = 3      ' 3 - Pixel
Global Const CHARACTERS = 4  ' 4 - Character
Global Const INCHES = 5      ' 5 - Inch
Global Const MILLIMETERS = 6 ' 6 - Millimeter
Global Const CENTIMETERS = 7 ' 7 - Centimeter

' ScrollBar
' Global Const NONE     = 0 ' 0 - None
Global Const HORIZONTAL = 1 ' 1 - Horizontal
Global Const VERTICAL = 2   ' 2 - Vertical
Global Const BOTH = 3       ' 3 - Both

' Shape
Global Const SHAPE_RECTANGLE = 0
Global Const SHAPE_SQUARE = 1
Global Const SHAPE_OVAL = 2
Global Const SHAPE_CIRCLE = 3
Global Const SHAPE_ROUNDED_RECTANGLE = 4
Global Const SHAPE_ROUNDED_SQUARE = 5

' WindowState
Global Const NORMAL = 0    ' 0 - Normal
Global Const MINIMIZED = 1 ' 1 - Minimized
Global Const MAXIMIZED = 2 ' 2 - Maximized

' Check Value
Global Const UNCHECKED = 0 ' 0 - Unchecked
Global Const CHECKED = 1   ' 1 - Checked
Global Const GRAYED = 2    ' 2 - Grayed

' Shift parameter masks
Global Const SHIFT_MASK = 1
Global Const CTRL_MASK = 2
Global Const ALT_MASK = 4

' Button parameter masks
Global Const LEFT_BUTTON = 1
Global Const RIGHT_BUTTON = 2
```

```
Global Const MIDDLE_BUTTON = 4

' Function Parameters
' MsgBox parameters
Global Const MB_OK = 0                ' OK button only
Global Const MB_OKCANCEL = 1          ' OK and Cancel buttons
Global Const MB_ABORTRETRYIGNORE = 2  ' Abort, Retry, and Ignore
buttons
Global Const MB_YESNOCANCEL = 3       ' Yes, No, and Cancel buttons
Global Const MB_YESNO = 4             ' Yes and No buttons
Global Const MB_RETRYCANCEL = 5       ' Retry and Cancel buttons

Global Const MB_ICONSTOP = 16         ' Critical message
Global Const MB_ICONQUESTION = 32     ' Warning query
Global Const MB_ICONEXCLAMATION = 48  ' Warning message
Global Const MB_ICONINFORMATION = 64  ' Information message

Global Const MB_APPLMODAL = 0         ' Application Modal Message Box
Global Const MB_DEFBUTTON1 = 0        ' First button is default
Global Const MB_DEFBUTTON2 = 256      ' Second button is default
Global Const MB_DEFBUTTON3 = 512      ' Third button is default
Global Const MB_SYSTEMMODAL = 4096    'System Modal

' MsgBox return values
Global Const IDOK = 1                 ' OK button pressed
Global Const IDCANCEL = 2             ' Cancel button pressed
Global Const IDABORT = 3              ' Abort button pressed
Global Const IDRETRY = 4              ' Retry button pressed
Global Const IDIGNORE = 5             ' Ignore button pressed
Global Const IDYES = 6                ' Yes button pressed
Global Const IDNO = 7                 ' No button pressed

' SetAttr, Dir, GetAttr functions
Global Const ATTR_NORMAL = 0
Global Const ATTR_READONLY = 1
Global Const ATTR_HIDDEN = 2
Global Const ATTR_SYSTEM = 4
Global Const ATTR_VOLUME = 8
Global Const ATTR_DIRECTORY = 16
Global Const ATTR_ARCHIVE = 32
```

```
'Grid
'ColAlignment,FixedAlignment Properties
Global Const GRID_ALIGNLEFT = 0
Global Const GRID_ALIGNRIGHT = 1
Global Const GRID_ALIGNCENTER = 2

'Fillstyle Property
Global Const GRID_SINGLE = 0
Global Const GRID_REPEAT = 1

'Data control
'Error event Response arguments
Global Const DATA_ERRCONTINUE = 0
Global Const DATA_ERRDISPLAY = 1

'Editmode property values
Global Const DATA_EDITNONE = 0
Global Const DATA_EDITMODE = 1
Global Const DATA_EDITADD = 2

' Options property values
Global Const DATA_DENYWRITE = &H1
Global Const DATA_DENYREAD = &H2
Global Const DATA_READONLY = &H4
Global Const DATA_APPENDONLY = &H8
Global Const DATA_INCONSISTENT = &H10
Global Const DATA_CONSISTENT = &H20
Global Const DATA_SQLPASSTHROUGH = &H40

'Validate event Action arguments
Global Const DATA_ACTIONCANCEL = 0
Global Const DATA_ACTIONMOVEFIRST = 1
Global Const DATA_ACTIONMOVEPREVIOUS = 2
Global Const DATA_ACTIONMOVENEXT = 3
Global Const DATA_ACTIONMOVELAST = 4
Global Const DATA_ACTIONADDNEW = 5
Global Const DATA_ACTIONUPDATE = 6
Global Const DATA_ACTIONDELETE = 7
Global Const DATA_ACTIONFIND = 8
```

```
Global Const DATA_ACTIONBOOKMARK = 9
Global Const DATA_ACTIONCLOSE = 10
Global Const DATA_ACTIONUNLOAD = 11

'OLE Client Control
'Actions
Global Const OLE_CREATE_EMBED = 0
Global Const OLE_CREATE_NEW = 0          'from ole1 control
Global Const OLE_CREATE_LINK = 1
Global Const OLE_CREATE_FROM_FILE = 1    'from ole1 control
Global Const OLE_COPY = 4
Global Const OLE_PASTE = 5
Global Const OLE_UPDATE = 6
Global Const OLE_ACTIVATE = 7
Global Const OLE_CLOSE = 9
Global Const OLE_DELETE = 10
Global Const OLE_SAVE_TO_FILE = 11
Global Const OLE_READ_FROM_FILE = 12
Global Const OLE_INSERT_OBJ_DLG = 14
Global Const OLE_PASTE_SPECIAL_DLG = 15
Global Const OLE_FETCH_VERBS = 17
Global Const OLE_SAVE_TO_OLE1FILE = 18

'OLEType
Global Const OLE_LINKED = 0
Global Const OLE_EMBEDDED = 1
Global Const OLE_NONE = 3

'OLETypeAllowed
Global Const OLE_EITHER = 2

'UpdateOptions
Global Const OLE_AUTOMATIC = 0
Global Const OLE_FROZEN = 1
Global Const OLE_MANUAL = 2

'AutoActivate modes
'Note that OLE_ACTIVATE_GETFOCUS only applies to objects that
'support "inside-out" activation. See related Verb notes below.
Global Const OLE_ACTIVATE_MANUAL = 0
```

```
Global Const OLE_ACTIVATE_GETFOCUS = 1
Global Const OLE_ACTIVATE_DOUBLECLICK = 2

'SizeModes
Global Const OLE_SIZE_CLIP = 0
Global Const OLE_SIZE_STRETCH = 1
Global Const OLE_SIZE_AUTOSIZE = 2

'DisplayTypes
Global Const OLE_DISPLAY_CONTENT = 0
Global Const OLE_DISPLAY_ICON = 1

'Update Event Constants
Global Const OLE_CHANGED = 0
Global Const OLE_SAVED = 1
Global Const OLE_CLOSED = 2
Global Const OLE_RENAMED = 3

'Special Verb Values
Global Const VERB_PRIMARY = 0
Global Const VERB_SHOW = -1
Global Const VERB_OPEN = -2
Global Const VERB_HIDE = -3
Global Const VERB_INPLACEUIACTIVATE = -4
Global Const VERB_INPLACEACTIVATE = -5
'The last two verbs are for objects that support "inside-out" activation,
'meaning they can be edited in-place, and that they support being left
'in-place-active even when the input focus moves to another control or form.
'These objects actually have 2 levels of being active. "InPlace Active"
'means that the object is ready for the user to click inside it and start
'working with it. "In-Place UI-Active" means that, in addition, if the object
'has any other UI associated with it, such as floating palette windows,
'that those windows are visible and ready for use. Any number of objects
'can be "In-Place Active" at a time, although only one can be
'"InPlace UI-Active".

'You can cause an object to move to either one of states programmatically by
'setting the Verb property to the appropriate verb and setting
'Action=OLE_ACTIVATE.
```

```
'Also, if you set AutoActivate = OLE_ACTIVATE_GETFOCUS, the server
'will automatically be put into "InPlace UI-Active" state when the user clicks
'on or tabs into the control.

'VerbFlag Bit Masks
Global Const VERBFLAG_GRAYED = &H1
Global Const VERBFLAG_DISABLED = &H2
Global Const VERBFLAG_CHECKED = &H8
Global Const VERBFLAG_SEPARATOR = &H800

'MiscFlag Bits - Or these together as desired for special behaviors

'MEMSTORAGE causes the control to use memory to store the object while
'       it is loaded. This is faster than the default (disk-tempfile),
'       but can consume a lot of memory for objects whose data takes
'       up a lot of space, such as the bitmap for a paint program.
Global Const OLE_MISCFLAG_MEMSTORAGE = &H1

'DISABLEINPLACE overrides the control's default behavior of allowing
'       in-place activation for objects that support it. If you
'       are having problems activating an object inplace, you can
'       force it to always activate in a separate window by setting this
'       bit
Global Const OLE_MISCFLAG_DISABLEINPLACE = &H2

'Common Dialog Control
'Action Property
Global Const DLG_FILE_OPEN = 1
Global Const DLG_FILE_SAVE = 2
Global Const DLG_COLOR = 3
Global Const DLG_FONT = 4
Global Const DLG_PRINT = 5
Global Const DLG_HELP = 6

'File Open/Save Dialog Flags
Global Const OFN_READONLY = &H1&
Global Const OFN_OVERWRITEPROMPT = &H2&
Global Const OFN_HIDEREADONLY = &H4&
Global Const OFN_NOCHANGEDIR = &H8&
Global Const OFN_SHOWHELP = &H10&
```

```
Global Const OFN_NOVALIDATE = &H100&
Global Const OFN_ALLOWMULTISELECT = &H200&
Global Const OFN_EXTENSIONDIFFERENT = &H400&
Global Const OFN_PATHMUSTEXIST = &H800&
Global Const OFN_FILEMUSTEXIST = &H1000&
Global Const OFN_CREATEPROMPT = &H2000&
Global Const OFN_SHAREAWARE = &H4000&
Global Const OFN_NOREADONLYRETURN = &H8000&

'Color Dialog Flags
Global Const CC_RGBINIT = &H1&
Global Const CC_FULLOPEN = &H2&
Global Const CC_PREVENTFULLOPEN = &H4&
Global Const CC_SHOWHELP = &H8&

'Fonts Dialog Flags
Global Const CF_SCREENFONTS = &H1&
Global Const CF_PRINTERFONTS = &H2&
Global Const CF_BOTH = &H3&
Global Const CF_SHOWHELP = &H4&
Global Const CF_INITTOLOGFONTSTRUCT = &H40&
Global Const CF_USESTYLE = &H80&
Global Const CF_EFFECTS = &H100&
Global Const CF_APPLY = &H200&
Global Const CF_ANSIONLY = &H400&
Global Const CF_NOVECTORFONTS = &H800&
Global Const CF_NOSIMULATIONS = &H1000&
Global Const CF_LIMITSIZE = &H2000&
Global Const CF_FIXEDPITCHONLY = &H4000&
Global Const CF_WYSIWYG = &H8000&        'must also have
CF_SCREENFONTS & CF_PRINTERFONTS
Global Const CF_FORCEFONTEXIST = &H10000
Global Const CF_SCALABLEONLY = &H20000
Global Const CF_TTONLY = &H40000
Global Const CF_NOFACESEL = &H80000
Global Const CF_NOSTYLESEL = &H100000
Global Const CF_NOSIZESEL = &H200000

'Printer Dialog Flags
Global Const PD_ALLPAGES = &H0&
```

```
Global Const PD_SELECTION = &H1&
Global Const PD_PAGENUMS = &H2&
Global Const PD_NOSELECTION = &H4&
Global Const PD_NOPAGENUMS = &H8&
Global Const PD_COLLATE = &H10&
Global Const PD_PRINTTOFILE = &H20&
Global Const PD_PRINTSETUP = &H40&
Global Const PD_NOWARNING = &H80&
Global Const PD_RETURNDC = &H100&
Global Const PD_RETURNIC = &H200&
Global Const PD_RETURNDEFAULT = &H400&
Global Const PD_SHOWHELP = &H800&
Global Const PD_USEDEVMODECOPIES = &H40000
Global Const PD_DISABLEPRINTTOFILE = &H80000
Global Const PD_HIDEPRINTTOFILE = &H100000

'Help Constants
Global Const HELP_CONTEXT = &H1        'Display topic in ulTopic
Global Const HELP_QUIT = &H2        'Terminate help
Global Const HELP_INDEX = &H3        'Display index
Global Const HELP_CONTENTS = &H3
Global Const HELP_HELPONHELP = &H4        'Display help on using help
Global Const HELP_SETINDEX = &H5        'Set the current Index for multi
index help
Global Const HELP_SETCONTENTS = &H5
Global Const HELP_CONTEXTPOPUP = &H8
Global Const HELP_FORCEFILE = &H9
Global Const HELP_KEY = &H101        'Display topic for keyword in
offabData
Global Const HELP_COMMAND = &H102
Global Const HELP_PARTIALKEY = &H105        'call the search engine in
winhelp

'Error Constants
Global Const CDERR_DIALOGFAILURE = -32768

Global Const CDERR_GENERALCODES = &H7FFF
Global Const CDERR_STRUCTSIZE = &H7FFE
Global Const CDERR_INITIALIZATION = &H7FFD
Global Const CDERR_NOTEMPLATE = &H7FFC
```

```
Global Const CDERR_NOHINSTANCE = &H7FFB
Global Const CDERR_LOADSTRFAILURE = &H7FFA
Global Const CDERR_FINDRESFAILURE = &H7FF9
Global Const CDERR_LOADRESFAILURE = &H7FF8
Global Const CDERR_LOCKRESFAILURE = &H7FF7
Global Const CDERR_MEMALLOCFAILURE = &H7FF6
Global Const CDERR_MEMLOCKFAILURE = &H7FF5
Global Const CDERR_NOHOOK = &H7FF4

'Added for CMDIALOG.VBX
Global Const CDERR_CANCEL = &H7FF3
Global Const CDERR_NODLL = &H7FF2
Global Const CDERR_ERRPROC = &H7FF1
Global Const CDERR_ALLOC = &H7FF0
Global Const CDERR_HELP = &H7FEF

Global Const PDERR_PRINTERCODES = &H6FFF
Global Const PDERR_SETUPFAILURE = &H6FFE
Global Const PDERR_PARSEFAILURE = &H6FFD
Global Const PDERR_RETDEFFAILURE = &H6FFC
Global Const PDERR_LOADDRVFAILURE = &H6FFB
Global Const PDERR_GETDEVMODEFAIL = &H6FFA
Global Const PDERR_INITFAILURE = &H6FF9
Global Const PDERR_NODEVICES = &H6FF8
Global Const PDERR_NODEFAULTPRN = &H6FF7
Global Const PDERR_DNDMMISMATCH = &H6FF6
Global Const PDERR_CREATEICFAILURE = &H6FF5
Global Const PDERR_PRINTERNOTFOUND = &H6FF4

Global Const CFERR_CHOOSEFONTCODES = &H5FFF
Global Const CFERR_NOFONTS = &H5FFE

Global Const FNERR_FILENAMECODES = &H4FFF
Global Const FNERR_SUBCLASSFAILURE = &H4FFE
Global Const FNERR_INVALIDFILENAME = &H4FFD
Global Const FNERR_BUFFERTOOSMALL = &H4FFC

Global Const FRERR_FINDREPLACECODES = &H3FFF
Global Const CCERR_CHOOSECOLORCODES = &H2FFF
```

```
'3D Controls
'Alignment (Check Box)
Global Const SSCB_TEXT_RIGHT = 0        '0 - Text to the right
Global Const SSCB_TEXT_LEFT = 1        '1 - Text to the left

'Alignment (Option Button)
Global Const SSOB_TEXT_RIGHT = 0        '0 - Text to the right
Global Const SSOB_TEXT_LEFT = 1        '1 - Text to the left

'Alignment (Frame)
Global Const SSFR_LEFT_JUSTIFY = 0      '0 - Left justify text
Global Const SSFR_RIGHT_JUSTIFY = 1     '1 - Right justify text
Global Const SSFR_CENTER = 2          '2 - Center text

'Alignment (Panel)
Global Const SSPN_LEFT_TOP = 0         '0 - Text to left and top
Global Const SSPN_LEFT_MIDDLE = 1       '1 - Text to left and middle
Global Const SSPN_LEFT_BOTTOM = 2       '2 - Text to left and bottom
Global Const SSPN_RIGHT_TOP = 3        '3 - Text to right and top
Global Const SSPN_RIGHT_MIDDLE = 4      '4 - Text to right and middle
Global Const SSPN_RIGHT_BOTTOM = 5      '5 - Text to right and bottom
Global Const SSPN_CENTER_TOP = 6       '6 - Text to center and top
Global Const SSPN_CENTER_MIDDLE = 7     '7 - Text to center and middle
Global Const SSPN_CENTER_BOTTOM = 8     '8 - Text to center and
bottom

'Autosize (Command Button)
Global Const SS_AUTOSIZE_NONE = 0       '0 - No Autosizing
Global Const SSPB_AUTOSIZE_PICTOBUT = 1  '0 - Autosize Picture to
Button
Global Const SSPB_AUTOSIZE_BUTTOPIC = 2  '0 - Autosize Button to
Picture

'Autosize (Ribbon Button)
'Global Const SS_AUTOSIZE_NONE     = 0  '0 - No Autosizing
Global Const SSRI_AUTOSIZE_PICTOBUT = 1  '0 - Autosize Picture to
Button
Global Const SSRI_AUTOSIZE_BUTTOPIC = 2  '0 - Autosize Button to
Picture
```

'Autosize (Panel)
'Global Const SS_AUTOSIZE_NONE = 0 '0 - No Autosizing
Global Const SSPN_AUTOSIZE_WIDTH = 1 '1 - Autosize Panel width to Caption
Global Const SSPN_AUTOSIZE_HEIGHT = 2 '2 - Autosize Panel height to Caption
Global Const SSPN_AUTOSIZE_CHILD = 3 '3 - Autosize Child to Panel

'BevelInner (Panel)
Global Const SS_BEVELINNER_NONE = 0 '0 - No Inner Bevel
Global Const SS_BEVELINNER_INSET = 1 '1 - Inset Inner Bevel
Global Const SS_BEVELINNER_RAISED = 2 '2 - Raised Inner Bevel

'BevelOuter (Panel)
Global Const SS_BEVELOUTER_NONE = 0 '0 - No Outer Bevel
Global Const SS_BEVELOUTER_INSET = 1 '1 - Inset Outer Bevel
Global Const SS_BEVELOUTER_RAISED = 2 '2 - Raised Outer Bevel

'FloodType (Panel)
Global Const SS_FLOODTYPE_NONE = 0 '0 - No flood
Global Const SS_FLOODTYPE_L_TO_R = 1 '1 - Left to light
Global Const SS_FLOODTYPE_R_TO_L = 2 '2 - Right to left
Global Const SS_FLOODTYPE_T_TO_B = 3 '3 - Top to bottom
Global Const SS_FLOODTYPE_B_TO_T = 4 '4 - Bottom to top
Global Const SS_FLOODTYPE_CIRCLE = 5 '5 - Widening circle

'Font3D (Panel, Command Button, Option Button, Check Box, Frame)
Global Const SS_FONT3D_NONE = 0 '0 - No 3-D text
Global Const SS_FONT3D_RAISED_LIGHT = 1 '1 - Raised with light shading
Global Const SS_FONT3D_RAISED_HEAVY = 2 '2 - Raised with heavy shading
Global Const SS_FONT3D_INSET_LIGHT = 3 '3 - Inset with light shading
Global Const SS_FONT3D_INSET_HEAVY = 4 '4 - Inset with heavy shading

'PictureDnChange (Ribbon Button)
Global Const SS_PICDN_NOCHANGE = 0 '0 - Use 'Up'bitmap with no change
Global Const SS_PICDN_DITHER = 1 '1 - Dither 'Up'bitmap
Global Const SS_PICDN_INVERT = 2 '2 - Invert 'Up'bitmap

```
'ShadowColor (Panel, Frame)
Global Const SS_SHADOW_DARKGREY = 0     '0 - Dark grey shadow
Global Const SS_SHADOW_BLACK = 1        '1 - Black shadow

'ShadowStyle (Frame)
Global Const SS_SHADOW_INSET = 0        '0 - Shadow inset
Global Const SS_SHADOW_RAISED = 1       '1 - Shadow raised

'Animated Button
'Cycle property
Global Const ANI_ANIMATED = 0
Global Const ANI_MULTISTATE = 1
Global Const ANI_TWO_STATE = 2

'Click Filter property
Global Const ANI_ANYWHERE = 0
Global Const ANI_IMAGE_AND_TEXT = 1
Global Const ANI_IMAGE = 2
Global Const ANI_TEXT = 3

'PicDrawMode Property
Global Const ANI_XPOS_YPOS = 0
Global Const ANI_AUTOSIZE = 1
Global Const ANI_STRETCH = 2

'SpecialOp Property
Global Const ANI_CLICK = 1

'TextPosition Property
Global Const ANI_CENTER = 0
Global Const ANI_LEFT = 1
Global Const ANI_RIGHT = 2
Global Const ANI_BOTTON = 3
Global Const ANI_TOP = 4

'GAUGE
'Style Property
Global Const GAUGE_HORIZ = 0
Global Const GAUGE_VERT = 1
```

```
Global Const GAUGE_SEMI = 2
Global Const GAUGE_FULL = 3

'Graph Control
'General
Global Const G_NONE = 0
Global Const G_DEFAULT = 0

Global Const G_OFF = 0
Global Const G_ON = 1

Global Const G_MONO = 0
Global Const G_COLOR = 1

'Graph Types
Global Const G_PIE2D = 1
Global Const G_PIE3D = 2
Global Const G_BAR2D = 3
Global Const G_BAR3D = 4
Global Const G_GANTT = 5
Global Const G_LINE = 6
Global Const G_LOGLIN = 7
Global Const G_AREA = 8
Global Const G_SCATTER = 9
Global Const G_POLAR = 10
Global Const G_HLC = 11

'Colors
Global Const G_BLACK = 0
Global Const G_BLUE = 1
Global Const G_GREEN = 2
Global Const G_CYAN = 3
Global Const G_RED = 4
Global Const G_MAGENTA = 5
Global Const G_BROWN = 6
Global Const G_LIGHT_GRAY = 7
Global Const G_DARK_GRAY = 8
Global Const G_LIGHT_BLUE = 9
Global Const G_LIGHT_GREEN = 10
Global Const G_LIGHT_CYAN = 11
```

```
Global Const G_LIGHT_RED = 12
Global Const G_LIGHT_MAGENTA = 13
Global Const G_YELLOW = 14
Global Const G_WHITE = 15
Global Const G_AUTOBW = 16

'Patterns
Global Const G_SOLID = 0
Global Const G_HOLLOW = 1
Global Const G_HATCH1 = 2
Global Const G_HATCH2 = 3
Global Const G_HATCH3 = 4
Global Const G_HATCH4 = 5
Global Const G_HATCH5 = 6
Global Const G_HATCH6 = 7
Global Const G_BITMAP1 = 16
Global Const G_BITMAP2 = 17
Global Const G_BITMAP3 = 18
Global Const G_BITMAP4 = 19
Global Const G_BITMAP5 = 20
Global Const G_BITMAP6 = 21
Global Const G_BITMAP7 = 22
Global Const G_BITMAP8 = 23
Global Const G_BITMAP9 = 24
Global Const G_BITMAP10 = 25
Global Const G_BITMAP11 = 26
Global Const G_BITMAP12 = 27
Global Const G_BITMAP13 = 28
Global Const G_BITMAP14 = 29
Global Const G_BITMAP15 = 30
Global Const G_BITMAP16 = 31

'Symbols
Global Const G_CROSS_PLUS = 0
Global Const G_CROSS_TIMES = 1
Global Const G_TRIANGLE_UP = 2
Global Const G_SOLID_TRIANGLE_UP = 3
Global Const G_TRIANGLE_DOWN = 4
Global Const G_SOLID_TRIANGLE_DOWN = 5
Global Const G_SQUARE = 6
```

```
Global Const G_SOLID_SQUARE = 7
Global Const G_DIAMOND = 8
Global Const G_SOLID_DIAMOND = 9

'Line Styles
'Global Const G_SOLID = 0
Global Const G_DASH = 1
Global Const G_DOT = 2
Global Const G_DASHDOT = 3
Global Const G_DASHDOTDOT = 4

'Grids
Global Const G_HORIZONTAL = 1
Global Const G_VERTICAL = 2

'Statistics
Global Const G_MEAN = 1
Global Const G_MIN_MAX = 2
Global Const G_STD_DEV = 4
Global Const G_BEST_FIT = 8

'Data Arrays
Global Const G_GRAPH_DATA = 1
Global Const G_COLOR_DATA = 2
Global Const G_EXTRA_DATA = 3
Global Const G_LABEL_TEXT = 4
Global Const G_LEGEND_TEXT = 5
Global Const G_PATTERN_DATA = 6
Global Const G_SYMBOL_DATA = 7
Global Const G_XPOS_DATA = 8
Global Const G_ALL_DATA = 9

'Draw Mode
Global Const G_NO_ACTION = 0
Global Const G_CLEAR = 1
Global Const G_DRAW = 2
Global Const G_BLIT = 3
Global Const G_COPY = 4
Global Const G_PRINT = 5
Global Const G_WRITE = 6
```

```
'Print Options
Global Const G_BORDER = 2

'Pie Chart Options        '
Global Const G_NO_LINES = 1
Global Const G_COLORED = 2
Global Const G_PERCENTS = 4

'Bar Chart Options        '
'Global Const G_HORIZONTAL = 1
Global Const G_STACKED = 2
Global Const G_PERCENTAGE = 4
Global Const G_Z_CLUSTERED = 6

'Gantt Chart Options      '
Global Const G_SPACED_BARS = 1

'Line/Polar Chart Options '
Global Const G_SYMBOLS = 1
Global Const G_STICKS = 2
Global Const G_LINES = 4

'Area Chart Options       '
Global Const G_ABSOLUTE = 1
Global Const G_PERCENT = 2

'HLC Chart Options        '
Global Const G_NO_CLOSE = 1
Global Const G_NO_HIGH_LOW = 2

'Key Status Control
'Style
Global Const KEYSTAT_CAPSLOCK = 0
Global Const KEYSTAT_NUMLOCK = 1
Global Const KEYSTAT_INSERT = 2
Global Const KEYSTAT_SCROLLLOCK = 3

'MCI Control (Multimedia)
'NOTE:
```

```
'Please use the updated Multimedia constants
'in the WINMMSYS.TXT file from the \VB\WINAPI
'subdirectory.

'Mode Property
'Global Const MCI_MODE_NOT_OPEN = 11
'Global Const MCI_MODE_STOP = 12
'Global Const MCI_MODE_PLAY = 13
'Global Const MCI_MODE_RECORD = 14
'Global Const MCI_MODE_SEEK = 15
'Global Const MCI_MODE_PAUSE = 16
'Global Const MCI_MODE_READY = 17

'NotifyValue Property
'Global Const MCI_NOTIFY_SUCCESSFUL = 1
'Global Const MCI_NOTIFY_SUPERSEDED = 2
'Global Const MCI_ABORTED = 4
'Global Const MCI_FAILURE = 8

'Orientation Property
'Global Const MCI_ORIENT_HORZ = 0
'Global Const MCI_ORIENT_VERT = 1

'RecordMode Porperty
'Global Const MCI_RECORD_INSERT = 0
'Global Const MCI_RECORD_OVERWRITE = 1

'TimeFormat Property
'Global Const MCI_FORMAT_MILLISECONDS = 0
'Global Const MCI_FORMAT_HMS = 1
'Global Const MCI_FORMAT_MSF = 2
'Global Const MCI_FORMAT_FRAMES = 3
'Global Const MCI_FORMAT_SMPTE_24 = 4
'Global Const MCI_FORMAT_SMPTE_25 = 5
'Global Const MCI_FORMAT_SMPTE_30 = 6
'Global Const MCI_FORMAT_SMPTE_30DROP = 7
'Global Const MCI_FORMAT_BYTES = 8
'Global Const MCI_FORMAT_SAMPLES = 9
'Global Const MCI_FORMAT_TMSF = 10
```

```
'Spin Button
'SpinOrientation
Global Const SPIN_VERTICAL = 0
Global Const SPIN_HORIZONTAL = 1

'Mas    ked Edit Control
'ClipMode
Global Const ME_INCLIT = 0
Global Const ME_EXCLIT = 1

'Comm Control
'Handshaking
Global Const MSCOMM_HANDSHAKE_NONE = 0
Global Const MSCOMM_HANDSHAKE_XONXOFF = 1
Global Const MSCOMM_HANDSHAKE_RTS = 2
Global Const MSCOMM_HANDSHAKE_RTSXONXOFF = 3

'Event constants
Global Const MSCOMM_EV_SEND = 1
Global Const MSCOMM_EV_RECEIVE = 2
Global Const MSCOMM_EV_CTS = 3
Global Const MSCOMM_EV_DSR = 4
Global Const MSCOMM_EV_CD = 5
Global Const MSCOMM_EV_RING = 6
Global Const MSCOMM_EV_EOF = 7

'Error code constants
Global Const MSCOMM_ER_BREAK = 1001
Global Const MSCOMM_ER_CTSTO = 1002
Global Const MSCOMM_ER_DSRTO = 1003
Global Const MSCOMM_ER_FRAME = 1004
Global Const MSCOMM_ER_OVERRUN = 1006
Global Const MSCOMM_ER_CDTO = 1007
Global Const MSCOMM_ER_RXOVER = 1008
Global Const MSCOMM_ER_RXPARITY = 1009
Global Const MSCOMM_ER_TXFULL = 1010

' MAPI SESSION CONTROL CONSTANTS
'Action
Global Const SESSION_SIGNON = 1
```

Global Const SESSION_SIGNOFF = 2

' MAPI MESSAGE CONTROL CONSTANTS
'Action
Global Const MESSAGE_FETCH = 1 ' Load all messages from message store
Global Const MESSAGE_SENDDLG = 2 ' Send mail bring up default mapi dialog
Global Const MESSAGE_SEND = 3 ' Send mail without default mapi dialog
Global Const MESSAGE_SAVEMSG = 4 ' Save message in the compose buffer
Global Const MESSAGE_COPY = 5 ' Copy current message to compose buffer
Global Const MESSAGE_COMPOSE = 6 ' Initialize compose buffer
Global Const MESSAGE_REPLY = 7 ' Fill Compose buffer as REPLY
Global Const MESSAGE_REPLYALL = 8 ' Fill Compose buffer as REPLY ALL
Global Const MESSAGE_FORWARD = 9 ' Fill Compose buffer as FORWARD
Global Const MESSAGE_DELETE = 10 ' Delete current message
Global Const MESSAGE_SHOWADBOOK = 11 ' Show Address book
Global Const MESSAGE_SHOWDETAILS = 12 ' Show details of the current recipient
Global Const MESSAGE_RESOLVENAME = 13 ' Resolve the display name of the recipient
Global Const RECIPIENT_DELETE = 14 ' Fill Compose buffer as FORWARD
Global Const ATTACHMENT_DELETE = 15 ' Delete current message

' ERROR CONSTANT DECLARATIONS (MAPI CONTROLS)
Global Const SUCCESS_SUCCESS = 32000
Global Const MAPI_USER_ABORT = 32001
Global Const MAPI_E_FAILURE = 32002
Global Const MAPI_E_LOGIN_FAILURE = 32003
Global Const MAPI_E_DISK_FULL = 32004
Global Const MAPI_E_INSUFFICIENT_MEMORY = 32005
Global Const MAPI_E_ACCESS_DENIED = 32006
Global Const MAPI_E_TOO_MANY_SESSIONS = 32008
Global Const MAPI_E_TOO_MANY_FILES = 32009

```
Global Const MAPI_E_TOO_MANY_RECIPIENTS = 32010
Global Const MAPI_E_ATTACHMENT_NOT_FOUND = 32011
Global Const MAPI_E_ATTACHMENT_OPEN_FAILURE = 32012
Global Const MAPI_E_ATTACHMENT_WRITE_FAILURE = 32013
Global Const MAPI_E_UNKNOWN_RECIPIENT = 32014
Global Const MAPI_E_BAD_RECIPTYPE = 32015
Global Const MAPI_E_NO_MESSAGES = 32016
Global Const MAPI_E_INVALID_MESSAGE = 32017
Global Const MAPI_E_TEXT_TOO_LARGE = 32018
Global Const MAPI_E_INVALID_SESSION = 32019
Global Const MAPI_E_TYPE_NOT_SUPPORTED = 32020
Global Const MAPI_E_AMBIGUOUS_RECIPIENT = 32021
Global Const MAPI_E_MESSAGE_IN_USE = 32022
Global Const MAPI_E_NETWORK_FAILURE = 32023
Global Const MAPI_E_INVALID_EDITFIELDS = 32024
Global Const MAPI_E_INVALID_RECIPS = 32025
Global Const MAPI_E_NOT_SUPPORTED = 32026

Global Const CONTROL_E_SESSION_EXISTS = 32050
Global Const CONTROL_E_INVALID_BUFFER = 32051
Global Const CONTROL_E_INVALID_READ_BUFFER_ACTION = 32052
Global Const CONTROL_E_NO_SESSION = 32053
Global Const CONTROL_E_INVALID_RECIPIENT = 32054
Global Const CONTROL_E_INVALID_COMPOSE_BUFFER_ACTION =
32055
Global Const CONTROL_E_FAILURE = 32056
Global Const CONTROL_E_NO_RECIPIENTS = 32057
Global Const CONTROL_E_NO_ATTACHMENTS = 32058

' MISCELLANEOUS GLOBAL CONSTANT DECLARATIONS (MAPI
CONTROLS)
Global Const RECIPTYPE_ORIG = 0
Global Const RECIPTYPE_TO = 1
Global Const RECIPTYPE_CC = 2
Global Const RECIPTYPE_BCC = 3

Global Const ATTACHTYPE_DATA = 0
Global Const ATTACHTYPE_EOLE = 1
Global Const ATTACHTYPE_SOLE = 2
```

```
' Outline
' PictureType
Global Const MSOUTLINE_PICTURE_CLOSED = 0
Global Const MSOUTLINE_PICTURE_OPEN = 1
Global Const MSOUTLINE_PICTURE_LEAF = 2

'Outline Control Error Constants
Global Const MSOUTLINE_BADPICFORMAT = 32000
Global Const MSOUTLINE_BADINDENTATION = 32001
Global Const MSOUTLINE_MEM = 32002
Global Const MSOUTLINE_PARENTNOTEXPANDED = 32003
```

Sub Balance1PreSet ()

```
Static Instruction(4) As String
Dim OutPutCmd As String
Dim Duration As Double
Dim Bal1Str As String

Instruction(1) = "@"
Instruction(2) = "D_W"
Instruction(3) = "Z"
Instruction(4) = "K_DD"

For I = 1 To 4 Step 1

' Open MSComm1 if not opened already.
If frmMain.MSComm1.PortOpen = False Then
    frmMain.MSComm1.PortOpen = True
End If

' Set up an error handler within this subroutine that will get
' called if a communication error occurs.
On Error GoTo ErrorHandler3

' Clear the transmitter buffer of MSComm1.
frmMain.MSComm1.OutBufferCount = 0

' Clear the receive buffer of MSComm1.
frmMain.MSComm1.InBufferCount = 0
```

```
' Set and return the number of characters the Input property reads from
' the receive buffer.
' Setting InputLen to 0 causes the communications control to read the
' entire contents of the receive buffer when Input is used.
frmMain.MSComm1.InputLen = 1

' Enable the Data Terminal Ready line during communications.  The
' Data Terminal Ready signal is sent by computer to device to indicate
' that the computer is ready to accept incoming transmission.
frmMain.MSComm1.DTREnable = True

' Enable Request To Send line on MSComm1.  Request permission
' to transmit data from computer to device.
' Set Request To Send line to high for RTS/CTS hardware handshaking.
frmMain.MSComm1.RTSEnable = True

' Determine whether data can be sent by querying the state of the
' Clear To Send line.  The Clear to Send signal is send from the
' device to computer to indicate that transmission can proceed.
' Determine the state of the Data Set Ready line.  Data Set Ready
' signal is sent by device to computer to indicate that it is
' ready to operate.

Do
Loop Until (frmMain.MSComm1.DSRHolding = True) And
(frmMain.MSComm1.CTSHolding = True)
OutPutCmd = Instruction(I) & Chr$(13) & Chr$(10)
frmMain.MSComm1.Output = OutPutCmd

'Make sure device has responded.
Do
Loop Until frmMain.MSComm1.InBufferCount > 0

' Delay timer for 0.3 second to handle timing problem.
Duration = Timer + .3
Do Until Timer > Duration
Loop

Next I
```

```
    frmMain.MSComm1.PortOpen = False

    Exit Sub

ErrorHandler3:

    ' Close the comm port if not already closed.
    If frmMain.MSComm1.PortOpen = True Then
        frmMain.MSComm1.PortOpen = False
    End If

    Exit Sub

End Sub
```

Sub Balance1Read ()

```
    Dim Instruction1 As String
    Dim OutPutCmd1 As String
    Dim Duration1 As Double
    Dim InputData1 As String
    Dim Crlfpos1 As Integer
    Dim Bal1Str As String

    Instruction1 = "SI"

    ' Open MSComm1 if not opened already.
    If frmMain.MSComm1.PortOpen = False Then
        frmMain.MSComm1.PortOpen = True
    End If

    ' Set up an error handler within this subroutine that will get
    ' called if a communication error occurs.
    On Error GoTo ErrorHandler5

    ' Clear the transmitter buffer of MSComm1.
    frmMain.MSComm1.OutBufferCount = 0

    ' Clear the receive buffer of MSComm1.
```

```
frmMain.MSComm1.InBufferCount = 0

' Set and return the number of characters the Input property reads from
' the receive buffer.
' Setting InputLen to 0 causes the communications control to read the
' entire contents of the receive buffer when Input is used.
frmMain.MSComm1.InputLen = 0

' Enable the Data Terminal Ready line during communications. The
' Data Terminal Ready signal is sent by computer to device to indicate
' that the computer is ready to accept incoming transmission.
frmMain.MSComm1.DTREnable = True

' Enable Request To Send line on MSComm1. Request permission
' to transmit data from computer to device.
' Set Request To Send line to high for RTS/CTS hardware handshaking.
frmMain.MSComm1.RTSEnable = True

' Determine whether data can be sent by querying the state of the
' Clear To Send line. The Clear to Send signal is send from the
' device to computer to indicate that transmission can proceed.
' Determine the state of the Data Set Ready line. Data Set Ready
' signal is sent by device to computer to indicate that it is
' ready to operate.

Do
Loop Until (frmMain.MSComm1.CTSHolding = True) And
(frmMain.MSComm1.DSRHolding = True)
OutPutCmd1 = Instruction1 & Chr$(13) & Chr$(10)
frmMain.MSComm1.Output = OutPutCmd1

'Make sure device has responded.
Do
Loop Until frmMain.MSComm1.InBufferCount > 0

' Delay timer for 0.3 second to handle timing problem.
Duration1 = Timer + .3
Do Until Timer > Duration1
Loop
```

```
    ' Read data out from the buffer & strip CR, LF.
    InputData1 = frmMain.MSComm1.Input
    Crlfpos1 = InStr(InputData1, Chr$(10))
    InputData1 = Left$(InputData1, Crlfpos1 - 2)

    Weight1 = InputData1

    frmMain.MSComm1.PortOpen = False

    Exit Sub

ErrorHandler5:

    ' Close the comm port if not already closed.
    If frmMain.MSComm1.PortOpen = True Then
        frmMain.MSComm1.PortOpen = False
    End If

    Exit Sub

End Sub

Sub Balance2PreSet ()

    Static Instruction(4) As String
    Dim OutPutCmd As String
    Dim Duration As Double
    Dim Bal2Str As String

    Instruction(1) = "@"
    Instruction(2) = "D_W"
    Instruction(3) = "Z"
    Instruction(4) = "K_DD"

    For I = 1 To 4 Step 1

    ' Open MSComm2 if not opened already.
    If frmMain.MSComm2.PortOpen = False Then
        frmMain.MSComm2.PortOpen = True
    End If
```

```
' Set up an error handler within this subroutine that will get
' called if a communication error occurs.
On Error GoTo ErrorHandler4

' Clear the transmitter buffer of MSComm2.
frmMain.MSComm2.OutBufferCount = 0

' Clear the receive buffer of MSComm2.
frmMain.MSComm2.InBufferCount = 0

' Set and return the number of characters the Input property reads from
' the receive buffer.
' Setting InputLen to 0 causes the communications control to read the
' entire contents of the receive buffer when Input is used.
frmMain.MSComm2.InputLen = 1

' Enable the Data Terminal Ready line during communications. The
' Data Terminal Ready signal is sent by computer to device to indicate
' that the computer is ready to accept incoming transmission.
frmMain.MSComm2.DTREnable = True

' Enable Request To Send line on MSComm2. Request permission
' to transmit data from computer to device.
' Set Request To Send line to high for RTS/CTS hardware handshaking.
frmMain.MSComm2.RTSEnable = True

' Determine whether data can be sent by querying the state of the
' Clear To Send line. The Clear to Send signal is send from the
' device to computer to indicate that transmission can proceed.
' Determine the state of the Data Set Ready line. Data Set Ready
' signal is sent by device to computer to indicate that it is
' ready to operate.

Do
Loop Until (frmMain.MSComm2.CTSHolding = True) And
(frmMain.MSComm2.DSRHolding = True)
OutPutCmd = Instruction(I) & Chr$(13) & Chr$(10)
frmMain.MSComm2.Output = OutPutCmd
```

```
    ' Make sure that device has responded
    Do
    Loop Until frmMain.MSComm2.InBufferCount > 0

    ' Delay timer for 0.3 second to handle timing problem.
    Duration = Timer + .3
    Do Until Timer > Duration
    Loop

    Next I

    frmMain.MSComm2.PortOpen = False

    Exit Sub

ErrorHandler4:

    ' Close the comm port if not already closed.
    If frmMain.MSComm2.PortOpen = True Then
        frmMain.MSComm2.PortOpen = False
    End If

    Exit Sub

End Sub

Sub Balance2Read ()

    Dim Instruction2 As String
    Dim OutPutCmd2 As String
    Dim Duration2 As Double
    Dim InputData2 As String
    Dim Crlfpos2 As Integer
    Dim Bal2Str As String

    Instruction2 = "SI"

    ' Open MSComm2 if not opened already.
    If frmMain.MSComm2.PortOpen = False Then
        frmMain.MSComm2.PortOpen = True
```

```
End If

' Set up an error handler within this subroutine that will get
' called if a communication error occurs.
On Error GoTo ErrorHandler6

' Clear the transmitter buffer of MSComm2.
frmMain.MSComm2.OutBufferCount = 0

' Clear the receive buffer of MSComm2.
frmMain.MSComm2.InBufferCount = 0

' Set and return the number of characters the Input property reads from
' the receive buffer.
' Setting InputLen to 0 causes the communications control to read the
' entire contents of the receive buffer when Input is used.
frmMain.MSComm2.InputLen = 0

' Enable the Data Terminal Ready line during communications.  The
' Data Terminal Ready signal is sent by computer to device to indicate
' that the computer is ready to accept incoming transmission.
frmMain.MSComm2.DTREnable = True

' Enable Request To Send line on MSComm2.  Request permission
' to transmit data from computer to device.
' Set Request To Send line to high for RTS/CTS hardware handshaking.
frmMain.MSComm2.RTSEnable = True

' Determine whether data can be sent by querying the state of the
' Clear To Send line.  The Clear to Send signal is send from the
' device to computer to indicate that transmission can proceed.
' Determine the state of the Data Set Ready line.  Data Set Ready
' signal is sent by device to computer to indicate that it is
' ready to operate.

Do
Loop Until (frmMain.MSComm2.CTSHolding = True) And
(frmMain.MSComm2.DSRHolding = True)
OutPutCmd2 = Instruction2 & Chr$(13) & Chr$(10)
frmMain.MSComm2.Output = OutPutCmd2
```

```
    ' Make sure that device has responded
    Do
    Loop Until frmMain.MSComm2.InBufferCount > 0

    ' Delay timer for 0.3 second to handle timing problem.
    Duration2 = Timer + .3
    Do Until Timer > Duration2
    Loop

    ' Read data out from the buffer & strip out CR, LF.
    InputData2 = frmMain.MSComm2.Input
    Crlfpos2 = InStr(InputData2, Chr$(10))
    InputData2 = Left$(InputData2, Crlfpos2 - 2)

    Weight2 = InputData2

    frmMain.MSComm2.PortOpen = False

    Exit Sub

ErrorHandler6:

    ' Close the comm port if not already closed.
    If frmMain.MSComm2.PortOpen = True Then
        frmMain.MSComm2.PortOpen = False
    End If

    Exit Sub

End Sub

Sub CloseFile (FileName As String)

    Dim F As Integer

    ' If there is an error, display the error message below.
    On Error GoTo CloseError

    F = FreeFile
```

```
    ' Otherwise, open the file name for output.
    ' Print the current text to the opened file.
    ' Close the file
    ' Reset the caption of the main form
    Open FileName For Output As F Len = 2048
    Print #F, frmMain.txtEdit.Text
    Close F
    FileName = "Untitled"
    Exit Sub

CloseError:

    MsgBox "Error occurred trying to close file, please retry.", 48
    Exit Sub
```

End Sub

Function DistanceToPulse (Distance As Single) As Long

```
    ' Function to convert distance in cm into be traveled into necessary
    ' number of pulse to be generated for stepping motor.

    DistanceToPulse = Int(367 * Distance)
```

End Function

Function DistanceTravelled (PulseCount As Long) As Single

```
    ' Function to calculate distance in cm traveled based on pulses
    ' already output to stepping motor.

    DistanceTravelled = PulseCount / 367
```

End Function

Sub ForwardMove (PIO1Addr As Integer, PulseCount As Long)

```
    ' Subroutine to generate pulses for stepping motor to move forward.
    ' Example :
```

```
' Call ForwardMove(PIO1Addr, PulseCount)
' Where : Pulse count is the number of pulses converted from distance
' required to be moved

Dim PortC1Addr As Integer
Dim OutData1 As Integer
Dim OutData2 As Integer
Dim ClearData As Integer
Dim Outer As Long
Dim DASErr As Integer
Dim Inner As Integer
Dim Dummy As Integer
Dim Tic As Integer

PortC1Addr = PIO1Addr + 2
OutData1 = &H2
OutData2 = &H3
ClearData = &H0

DASErr = dasbyteout(PortC1Addr, ClearData)

    For Outer = 1 To PulseCount Step 1
      DASErr = dasbyteout(PortC1Addr, OutData1)
      DASErr = dasbyteout(PortC1Addr, OutData2)
    Next Outer

DASErr = dasbyteout(PortC1Addr, ClearData)
```

End Sub

Sub GFocus (OldBox As Control, NewBox As Control, OldText As String)

```
' This routine and its parameters are used to mimic
' standard windows text box changing methods while
' retaining certain controls on the text allowed,
' such as the number of significant decimal places.
' Each time a change to a box is made, GFocus is
' called twice: once when the box is first clicked
' on (to hilight the text), and once when something
' else is clicked on (to verify the changes.)
```

```
' If the "something else" is not a text box, the
' relevant parts of this routine are suppressed.

Dim DecPos As Integer
Dim FormatStr As String
Dim I As Integer

'Hilight the text in the new box just clicked on
'Change the text in the last box clicked on

If TypeOf NewBox Is TextBox Then
    NewBox.SelStart = 0
    NewBox.SelLength = Len(NewBox.Text)
End If

If TypeOf OldBox Is TextBox Then
    ' If both numeric, make sure new text has correct
    ' number of decimal places.
    If IsNumeric(OldText) And IsNumeric(OldBox.Text) Then
      FormatStr = "#######0"
      DecPos = 0
      'DecPos is the number of sig. decimal places
      If InStr(OldText, ".") > 0 Then
          DecPos = Len(OldText) - InStr(OldText, ".")
      End If
      'Add the correct number of sig. decimal places
      If DecPos > 0 Then
          FormatStr = FormatStr & "."
          For I = 1 To DecPos
            FormatStr = FormatStr & "0"
          Next I
      End If
      'Format the new text (OldBox.Text) correctly
      OldBox.Text = Format$(OldBox.Text, FormatStr)
    ElseIf Not IsNumeric(OldText) And Not IsNumeric(OldBox.Text) Then
    ' If both non-numeric, change is Ok.
      Exit Sub
    Else
    ' New & old text are of different types; disallow
      OldBox.Text = OldText
```

```
        Exit Sub
    End If
End If

End Sub

Function HexStrtoDec (HexStr As String) As Integer

  ' This function converts hex string to decimal format.
  ' Convert A-F to 1 - 6 and add to 9.
  ' Example :
  ' DecNumber = HexStrToDec(HexStr)

  Dim N As Integer
  Dim I As Integer
  Dim Tstr As Variant

  For I = Len(HexStr) To 1 Step -1
      Tstr = Mid$(HexStr, I, 1)
      If (Tstr >= "A") Then
         N = N + (9 + (Asc(Tstr) - &H40)) * (16 ^ (Len(HexStr) - I))
      Else
         N = N + Val(Tstr) * (16 ^ (Len(HexStr) - I))
      End If
  Next I

  HexStrtoDec = N

End Function

Sub HP432CAutoZero (PIO2Addr As Integer)

  ' This subprogram auto zero HP432C meter
  ' Call HP432CAutoZero (PIO2Addr)
  ' Where : PIO2Addr is the starting address of PIO-12 Card #2

  Dim PortC2Data1 As Integer
  Dim PortC2Data2 As Integer
  Dim PortC2Addr As Integer
  Dim DASErr As Integer
```

```
    PortC2Data1 = &HC
    PortC2Data2 = &HF

    PortC2Addr = PIO2Addr + 2

    DASErr = dasbyteout(PortC2Addr, PortC2Data1)
    DASErr = dasbyteout(PortC2Addr, PortC2Data2)

End Sub
```

Sub HP432CConversion (I As Integer, FirstByte As Integer, SecondByte As Integer)

```
    ' This subprogram reads the stored data in memory, convert into Watt

    Dim RangeByte As Integer
    Dim RawData As Integer

    ' Convert data sent by meter into equivalent integer,
    ' then multiply by correct range value
    RawData = 100 * Val(Hex$(SecondByte And &H1F)) + Val(Hex$(FirstByte))

    'Multiply by 98.10121137 for the 20 dB coupler and insertion loss due to
    'frequency meter and attenuator
    'Determine measurement range and convert data into micro Watt.
    RangeByte = SecondByte And &HC0

    If RangeByte = &H0 Then
        If CSng(RawData) * .00000001 >= .00001 Then
            HP432CMeasurement(I) = CSng(RawData) * .00000001 *
            98.10121137
        Else
            HP432CMeasurement(I) = -999
        End If
    ElseIf RangeByte = &H40 Then
        HP432CMeasurement(I) = CSng(RawData) * .0000001 * 98.10121137
    ElseIf RangeByte = &H80 Then
        HP432CMeasurement(I) = CSng(RawData) * .000001 * 98.10121137
    ElseIf RangeByte = &HC0 Then
```

```
        If CSng(RawData) * .00001 <= .01 Then
            HP432CMeasurement(I) = CSng(RawData) * .00001 * 98.10121137
        Else
            HP432CMeasurement(I) = 999
        End If
    End If
```

End Sub

Sub HP432CPreSet (PIO2Addr As Integer)

```
    ' This subprogram preset PIO-12 Card #1 for motor controller
    ' Call HP432CPreSet (PIO2Addr)
    ' Where : PIO2Addr is the starting address of PIO-12 Card #2

    Dim PortC2Data1 As Integer
    Dim PortC2Addr As Integer
    Dim DASErr As Integer

    PortC2Data1 = &HF
    PortC2Addr = PIO2Addr + 2

    DASErr = dasbyteout(PortC2Addr, PortC2Data1)
```

End Sub

Sub HP432CRead (PortA2Addr As Integer, PortB2Addr As Integer, Station As Integer)

```
    ' This subprogram reads HP432C meter output, check for DSR bit
    ' Call HP432CRead (PIO2Addr)
    ' Where : PIO2Addr is the starting address of PIO-12 Card #2

    Dim DSRBit As Integer
    Dim DASErr As Integer

    DSRBit = &H0

    HP432CUMeasurement(Station) = dasbytein(PortA2Addr)
    HP432CLMeasurement(Station) = dasbytein(PortB2Addr)
```

```
End Sub

Sub HP436AConversion (Station As Integer, HP436AData As String)

   Dim Power As Single

   ' Get the single value buried in the string returned by meter
   ' Multiply by 1.01269811 because of insertion loss of waveguide

   Power = Mid(HP436AData, 4, 9)

   If Power >= .00000001 Then
         If Power <= .1 Then
            HP436AMeasurement(Station) = Mid(HP436AData, 4, 9) * 1.01269811
         Else
            HP436AMeasurement(Station) = 999
         End If
   'Else
   '   HP436AMeasurement(Station) = -999
   End If

End Sub

Sub HP436ARead (Station As Integer)

   Dim HP436AInput As String
   Dim Actual As Long
   Dim WaitTime As Double
   Dim LogMode As String
   On Error GoTo ErrorHandler8

   Call iclear(HPIBID)

   LogMode = "9A-T" & Chr$(10)
   Call iwrite(HPIBID, ByVal LogMode, 2, 1, Actual)

   WaitTime = Timer + SaturationTime
   Do Until Timer > WaitTime
   Loop
```

```
    'Get the measurement
    Call iread(HPIBID, ByVal HP436AInput, 16, ByVal 0&, Actual)
    HP436ARaw(Station) = HP436AInput

    Exit Sub

ErrorHandler8:

    'Give as a measurement an error flag value
    HP436AInput = "000-9999E-00" + Chr$(13) + Chr$(10)
    HP436ARaw(Station) = HP436AInput

    Exit Sub

End Sub

Sub HP436ASet (HPIBAddr As String)

    Dim Actual As Long

    On Error GoTo ErrorHandler9

    If HPIBID = 0 Then
        HPIBID = iopen(HPIBAddr)
    End If

    'Call ihint(HPIBID, I_HINT_SYSTEM)

    Call itimeout(HPIBID, 10000)
    Call iclear(HPIBID)
    Call iwrite(HPIBID, ByVal "Z" + Chr$(10), 2, 1, Actual)
    Call iwrite(HPIBID, ByVal "SYST:ERR?" + Chr$(10), 10, 1, Actual)

    Exit Sub

ErrorHandler9:

    If HPIBID = 0 Then
        MsgBox "Failure to open HPIB port!"
```

```
        Exit Sub
    End If

    ' Close the device session if iopen was successful.
    If HPIBID <> 0 Then
        Call iclose(HPIBID)
        MsgBox "HPIB port already open, or failure to write to port!"
    End If

End Sub

Sub InitPIO1 (PIO1Addr As Integer)

    ' This subprogram initialize PIO-12 card for motor controller
    ' Example :
    ' PIO1Init(PIO1Addr)
    ' Where: PIO1Addr is the starting address of PIO-12 Card #1

    Dim CntrlData As integer
    Dim ClearData As Integer
    Dim DASErr As Integer

    If PIO1Init = False Then
        PortA1Addr = PIO1Addr + 0
        PortB1Addr = PIO1Addr + 1
        PortC1Addr = PIO1Addr + 2
        Cntrl1Addr = PIO1Addr + 3
        CntrlData = &H80
        ClearData = &H0
        DASErr = dasbyteout(Cntrl1Addr, CntrlData)
        DASErr = dasbyteout(PortC1Addr, ClearData)
        PIO1Init = True
    End If

End Sub

Sub InitPIO2 (PIO2Addr As Integer)

    ' This subprogram initialize PIO-12 card for HP432C meter
    ' Example :
```

```
' PIO2Init(PIO2Addr)
' Where: PIO2Addr is the starting address of PIO-12 Card #2

Dim CntrlData As Integer
Dim ClearData As Integer
Dim DASErr As Integer

If PIO2Init = False Then
    PortA2Addr = PIO2Addr + 0
    PortB2Addr = PIO2Addr + 1
    PortC2Addr = PIO2Addr + 2
    Cntrl2Addr = PIO2Addr + 3
    ClearData = &H0
    CntrlData = &H92
    DASErr = dasbyteout(Cntrl2Addr, CntrlData)
    DASErr = dasbyteout(PortA2Addr, ClearData)
    DASErr = dasbyteout(PortB2Addr, ClearData)
    DASErr = dasbyteout(PortC2Addr, ClearData)
    PIO2Init = True
End If
```

End Sub

Function InputHex (KeyAscii As Integer) As Integer

```
' Check for 0 - 9, A-F, a-f.
' Convert a-f to A-F.
If (KeyAscii >= &H30 And KeyAscii <= &H39) Or (KeyAscii >= &H41 And
KeyAscii <= &H46) Or (KeyAscii >= &H61 And KeyAscii <= &H66) Then
    If (KeyAscii >= &H61) Then KeyAscii = KeyAscii - &H20
    InputHex = True
Else
    InputHex = False
End If
```

End Function

Sub OpenFile (FileName As String)

```
Dim F As Integer
```

```
 ' Avoid opening the file if already loaded.
 If "Text Editor: ' + FileName = frmMain.Caption Then
       Exit Sub
 Else
       ' Open file selected on File Open About.
       ' Read contents into text area & close file.
       ' Enable the Close menu items
       On Error GoTo ErrHandler
       F = FreeFile
       Open FileName For Input As F Len = 4096
       frmMain.txtEdit.Text = Input$(LOF(F), F)
       Close F
       frmMain.mnuFileItem(4).Enabled = True
       UpdateMenu
       frmMain.Caption = "Text Editor: " + FileName
       Exit Sub
 End If

ErrHandler:

       MsgBox "File too big to be opened or there are characters that cannot be
       recognized by this editor, please use other editors.", 48, "Text Editor"
       Close F
       Exit Sub
```

End Sub

Sub ReverseMove (PIO1Addr As Integer, PulseCount As Long)

```
 ' Subroutine to generate pulses for stepping motor to move reverse.
 ' Also call function to report distance moved.
 ' Example :
 ' Call ReverseMove(PulseCount)
 ' Where : Pulse count is the number of pulses converted from distance
 ' required to be moved

 Dim PortC1Addr As Integer
 Dim OutData1 As Integer
 Dim OutData2 As Integer
```

```
Dim ClearData As Integer
Dim Outer As Long
Dim DASErr As Integer
Dim Delay As Integer
Dim Inner As Integer
Dim Dummy As Integer
Dim Tic As Integer

PortC1Addr = PIO1Addr + 2
OutData1 = &H1
OutData2 = &H3
ClearData = &H0

Tic = 4

DASErr = dasbyteout(PortC1Addr, ClearData)

For Outer = 1 To PulseCount Step 1

    If Outer Mod 367 = 0 Then
      DoEvents
    End If

    DASErr = dasbyteout(PortC1Addr, OutData1)

    For Inner = 1 To Tic Step 1
      Dummy = Dummy + Dummy
    Next Inner

    DASErr = dasbyteout(PortC1Addr, OutData2)

    For Inner = 1 To Tic Step 1
      Dummy = Dummy + Dummy
    Next Inner

Next Outer

DASErr = dasbyteout(PortC1Addr, ClearData)

End Sub
```

```
Sub ReverseOut (PIO1Addr As Integer, PulseCount As Long)

    Dim PortC1Addr As Integer
    Dim OutData1 As Integer
    Dim OutData2 As Integer
    Dim ClearData As Integer
    Dim Outer As Long
    Dim DASErr As Integer

    PortC1Addr = PIO1Addr + 2
    OutData1 = &H1
    OutData2 = &H3
    ClearData = &H0

    DASErr = dasbyteout(PortC1Addr, ClearData)

        For Outer = 1 To PulseCount Step 1
            DASErr = dasbyteout(PortC1Addr, OutData1)
            DASErr = dasbyteout(PortC1Addr, OutData2)
        Next Outer

    DASErr = dasbyteout(PortC1Addr, ClearData)

End Sub

Sub RS2321Set (Setting As String)

    Dim Ball1Err As String

    If frmMain.MSComm1.PortOpen = False Then
        frmMain.MSComm1.Settings = Setting
        frmMain.MSComm1.PortOpen = True
    End If

    On Error GoTo ErrorHandler1
    frmMain.MSComm1.OutBufferCount = 0
    frmMain.MSComm1.InBufferCount = 0
    frmMain.MSComm1.InputLen = 0
    frmMain.MSComm1.DTREnable = True
```

```
  frmMain.MSComm1.RTSEnable = True

ErrorHandler1:

  Bal1Err = "Error : " + Error$
  If frmMain.MSComm1.PortOpen = True Then
      frmMain.MSComm1.PortOpen = False
  End If
```

End Sub

Sub RS2322Set (Setting As String)

```
  Dim Bal2Err As String

  If frmMain.MSComm2.PortOpen = False Then
      frmMain.MSComm2.Settings = Setting
      frmMain.MSComm2.PortOpen = True
  End If

  On Error GoTo ErrorHandler2
  frmMain.MSComm2.OutBufferCount = 0
  frmMain.MSComm2.InBufferCount = 0
  frmMain.MSComm2.InputLen = 0
  frmMain.MSComm2.DTREnable = True
  frmMain.MSComm2.RTSEnable = True

ErrorHandler2:

  Bal2Err = "Error : " + Error$
  If frmMain.MSComm2.PortOpen = True Then
      frmMain.MSComm2.PortOpen = False
  End If
```

End Sub

Sub UPCOff (UPCBase As Integer)

```
  ' This program turns off the sensor interface card so that conversions
  ' and other interrupt processing on the card ceases.
```

```
    ' Example :
    ' CALL UPCOff (UPCBase)
    ' Where: UPCBase = Card starting address in decimal notation

    Dim Page3 As Integer

    Dim CommandRegister As Integer
    Dim DataRegister As Integer
    Dim Command1 As Integer

    Page3 = UPCBase + 7

    CommandRegister = 254
    DataRegister = UPCBase
    Command1 = 250

    OUT Page3, CommandRegister
    OUT DataRegister, Command1
    OUT Page3, CommandRegister

    While Inp(DataRegister)
    Wend

End Sub

Sub UPCOn (UPCBase As Integer)

    ' This subprogram turns on the sensor interface card so that it
    ' implements the set-up table in transfer memory and begins
    ' converting anaiog data.
    ' Example :
    ' CALL UPCOn (UPCBase)
    ' Where: UPCBase = Card starting address in decimal notation
    ' Define PC I/O map address (portal address) reserved for
    ' access to four 256 byte Pages of UPC601-L Ram (transfer memory).

    Dim Page3 As Integer

    Dim CommandRegister As Integer
    Dim DataRegister As Integer
```

```
    Dim Command1 As Integer
    Dim Command2 As Integer

    Page3 = UPCBase + 7

    CommandRegister = 254
    DataRegister = UPCBase
    Command1 = 251
    Command2 = 255

    OUT Page3, CommandRegister
    OUT DataRegister, Command1

    While Inp(DataRegister)
    Wend

    OUT Page3, CommandRegister
    OUT DataRegister, Command2

    While Inp(DataRegister)
    Wend

End Sub

Sub UPCRead (UPCBase As Integer)

    ' Analog Input Subroutine for Sensor Interface Cards
    ' This subroutine access the analog input channels of sensor
    ' interface cards.
    ' Example:
    ' CALL UPCRead(UPCBase)
    ' Where: UPCBase is the starting address of the card in I/O map
    ' in decimal notation

    Dim Page2 As Integer
    Dim Page3 As Integer

    Dim CommandRegister As Integer
    Dim DataRegister As Integer
    Dim Command1 As Integer
```

```
    Dim Location As Integer
    Dim InChan As Single
    Dim Segment As Integer
    Dim OffSet As Integer
    Dim I As Integer
    Dim J As Integer
    Dim TmpOff As Integer

    Page2 = UPCBase + 6
    Page3 = UPCBase + 7

    CommandRegister = 254
    DataRegister = UPCBase
    Command1 = 222
    Location = 0

    InChan = 0
    Segment = VarSeg%(InChan)
    OffSet = VarPtr%(InChan)

    OUT Page3, CommandRegister
    OUT DataRegister, Command1
    OUT Page3, CommandRegister

    While Inp(DataRegister + 1)
    Wend

    OUT Page2, Location

    For I = 0 To 16
        For J = 0 To 3
            TmpOff = J
            MhPokeByte Inp%(DataRegister), Segment, OffSet + TmpOff
        Next J
        Pressure(I) = Format(InChan, "##.0000")
    Next I

End Sub

Sub UpdateMenu ()
```

```
' Make the initial element visible / display separator bar.
' Increment index property of control array.

Dim I As Integer

frmMain.mnuFileArray(0).Visible = True
ArrayNum = ArrayNum + 1

' Check to see if FileName is already on menu list.
For I = 0 To ArrayNum - 1
     If frmMain.mnuFileArray(I).Caption = FileName Then
        ArrayNum = ArrayNum - 1
        Exit Sub
     End If
Next I

' If FileName is not on menu list, add menu item.
' Create a new menu control.
' Set the caption of the new menu item.
' Make the new menu item visible.
Load frmMain.mnuFileArray(ArrayNum)
frmMain.mnuFileArray(ArrayNum).Caption = FileName
frmMain.mnuFileArray(ArrayNum).Visible = True

End Sub
```

Appendix B - 17 MAIN.FRM

```
'**********Master program display form**********
'This form is the master form of activating the program.

    Option Explicit

    Dim Msg As Variant
    Dim Response As Variant
    Dim DgDef As Variant
    Dim Title As Variant

    Dim ArrayNum As Integer
    Dim SuccessFlag As Integer

Sub Form_Load ()

    ' Change working directory to the directory
    ' where the application was executed.
    ChDir app.Path
    ChDrive app.Path

    ' Position the text box
    txtEdit.Move 0, 0

    ' The form is horizontally and vertically centered when loaded.
    Top = Screen.Height / 2 - Height / 2
    Left = Screen.Width / 2 - Width / 2

    SuccessFlag = False

    txtEdit.Text = "To calibrate the equipment, use Setup & Test under the
    Experiment heading." & Chr$(13) & Chr$(10)
    txtEdit.Text = txtEdit.Text & "Ideally, this should be done before every
    experiment." & Chr$(13) & Chr$(10)
    txtEdit.Text = txtEdit.Text & "To run an experiment, use Data Logging under
    the Experiment heading." & Chr$(13) & Chr$(10)
    txtEdit.Text = txtEdit.Text & "To process a data file for use with Excel, use the
    Create menu." & Chr$(13) & Chr$(10)

End Sub
```

```
Sub Form_QueryUnload (Cancel As Integer, Unloadmode As Integer)

    ' Put together a message box with all the proper components.
    Title = "Warning"
    Msg = "Do you really want to exit the application?"
    DgDef = MB_YESNO + MB_ICONSTOP + MB_DEFBUTTON2

    Cancel = True
    ' Evaluate unload mode and give warning.
    If Unloadmode = 0 Then
        Response = MsgBox(Msg, DgDef, Title)
        If Response = IDYES Then
            Cancel = False
            Unload frmMain
            End
        Else
            Cancel = True
        End If
    Else
            Cancel = False
    End If

End Sub

Sub Form_Resize (

    ' Automatical                    o fill the form Edit whenever the form
    ' is resized
    txtEdit.V
    txtEdit.

End Sub

Function Ma              ilename As String)

    Dim In As Integer
    Dim Out As Integer
    Dim OutFile As String
    Dim PeriodPos As Integer
    Dim TextLine As String
```

```
On Error GoTo ErrHandler41

'Construct the appropriate name for the output file
'If the input file is "temp.dat", the output file should
'be "temp.pwr"

If Len(Filename) Then
      PeriodPos = InStr(Filename, ".")
      If PeriodPos <> 0 Then
         PeriodPos = PeriodPos - 1
      Else
         PeriodPos = Len(Filename)
      End If
      OutFile = Left$(Filename, PeriodPos) & ".PWR"
      Msg = " Creating power data file " & OutFile
Else
      MakePower = False
      Msg = "Error, no file will be created."
End If

MsgBox Msg

'Open file selected.
In = FreeFile
Open Filename For Input Access Read Lock Read As In Len = 4096
Out = FreeFile
Open OutFile For Output Access Write Lock Write As Out Len = 4096

'Print output file header
Print #Out, "Scan"; Tab(8); ","; Tab(10); "Station"; Tab(18); ","; Tab(20);
"Delta time(sec.)"; Tab(38); ","; Tab(40); "Input watt"; Tab(53); ","; Tab(55);
"Attenuated watt"

Do While Not EOF(In)

      'Search for Power Info Header
      TextLine = ""
      Do While Not ((InStr(TextLine, "Scan") <> 0) And (InStr(TextLine,
      "Station") <> 0) And (InStr(TextLine, "Delta Time(sec.)") <> 0) And
```

```
        (InStr(TextLine,    ,put watt") <> 0) And (InStr(TextLine, "Attenuated
        watt") <> 0)) And Not EOF(In)
           Line Input #In, TextLine
        Loop

        'Get and print power Info, interspersed with commas
        If Not EOF(In) Then
           Line Input #In, TextLine
           Line Input #In, TextLine
           Do
                TextLine = Mid$(TextLine, 1, 4) & "   , " & Mid$(TextLine, 10, 4)
                & " , " & Mid$(TextLine, 20, 14) & "    ,  " & Mid$(TextLine, 40,
                10) & "  ,  " & Mid$(TextLine, 55, 10)
                Print #Out, TextLine
                Line Input #In, TextLine
           Loop Until TextLine = ""
        End If
    Loop

    Close In
    Close Out

    MakePower = True
    Exit Function

ErrHandler41:

    MsgBox "Error encountered while trying to open file, please retry.", 48, "Text
    Editor"
    Close In
    Close Out
    MakePower = False
    Exit Function

End Function

Function MakePressure (FilenName As String)

    Dim In As Integer
    Dim Out As Integer
```

```
Dim OutFile As String
Dim PeriodPos As Integer
Dim TextLine As String
Dim I As Integer
Dim OutLine As String
Dim StrTime As String
Dim StrScan As String

On Error GoTo ErrHandler42

'Construct the appropriate name for the output file
'If the input file is "temp.dat", the output file should
'be "temp.prs"

If Len(Filename) Then
      PeriodPos = InStr(Filename, ".")
      If PeriodPos <> 0 Then
         PeriodPos = PeriodPos - 1
      Else
         PeriodPos = Len(Filename)
      End If
      OutFile = Left$(Filename, PeriodPos) & ".PRS"
      Msg = " Creating pressure data file " & OutFile
Else
      MakePressure = False
      Msg = "Error, no file will be created."
End If

MsgBox Msg

'Open file selected.
In = FreeFile
Open Filename For Input Access Read Lock Read As In Len = 4096
Out = FreeFile
Open OutFile For Output Access Write Lock Write As Out Len = 4096

'Print output file header
Print #Out, "Time"; Tab(13); ","; Tab(15); "Scan"; Tab(21); ","; Tab(23);
"Pres.1"; Tab(31); ","; Tab(33); "Pres.2"; Tab(41); ","; Tab(43); "Pres.3";
Tab(51); ","; Tab(53); "Pres.4"; Tab(61); ","; Tab(63); "Pres.5"; Tab(71); ",";
```

```
Tab(73); "Pres.6"; Tab(81); ","; Tab(83); "Pres.7"; Tab(91); ","; Tab(93);
"Pres.8"; Tab(101); ","; Tab(103); "Pres.9"; Tab(111); ","; Tab(113); "Pres.10";
Tab(121); ","; Tab(123); "Pres.11"; Tab(131); ","; Tab(133); "Pres.12";
Tab(141); ","; Tab(143); "Pres.13"; Tab(151); ","; Tab(153); "Pres.14";
Tab(161); ","; Tab(163); "Pres.15"; Tab(171); ","; Tab(173); "Pres.16"; ""

Do While Not EOF(In)

    'Search for Time and Scan to add to pressure info
    TextLine = ""
    Do While Not ((InStr(TextLine, "Time") <> 0) And (InStr(TextLine,
    "Scan") <> 0) And (InStr(TextLine, "Weight1") <> 0) And
    (InStr(TextLine, "Weight2") <> 0)) And Not EOF(In)
      Line Input #In, TextLine
    Loop

    If Not EOF(In) Then
      Line Input #In, TextLine
      OutLine = Mid$(TextLine, 1, 12) & ", " & Mid$(TextLine, 13, 4)
    End If

    'Search for Pressure Info Header
    TextLine = ""
    Do While Not ((InStr(TextLine, "Pres.1") <> 0) And (InStr(TextLine,
    "Pres.2") <> 0) And (InStr(TextLine, "Pres.3") <> 0) And
    (InStr(TextLine, "Pres.4") <> 0)) And Not EOF(In)
      Line Input #In, TextLine
    Loop
    'Get and print pressure info, interspersed with commas
    If Not EOF(In) Then
      Line Input #In, TextLine
      OutLine = OutLine & "  , " & Mid$(TextLine, 1, 7) & " , " &
      Mid$(TextLine, 10, 7) & " , " & Mid$(TextLine, 20, 7) & " , " &
      Mid$(TextLine, 30, 7)
      For I = 1 To 3
          Line Input #In, TextLine
          Line Input #In, TextLine
          Line Input #In, TextLine
          OutLine = OutLine & " , " & Mid$(TextLine, 1, 7) & " , " &
           Mid$(TextLine, 10, 7) & " , " & Mid$(TextLine, 20, 7) & " , " &
```

```
            Mid$(TextLine, 30, 7)
        Next I
        Print #Out, OutLine
    End If
Loop

Close In
Close Out

MakePressure = True
Exit Function

ErrHandler42:

MsgBox "Error encountered while trying to open file, please retry.", 48, "Text
Editor"
Close In
Close Out
MakePressure = False
Exit Function
```

End Function

Function MakeWeight (Filename As String)

```
Dim In As Integer
Dim Out As Integer
Dim OutFile As String
Dim PeriodPos As Integer
Dim TextLine As String

On Error GoTo ErrHandler43

'Construct the appropriate name for the output file
'If the input file is "temp.dat", the output file should
'be "temp.wts"

If Len(Filename) Then
    PeriodPos = InStr(Filename, ".")
    If PeriodPos <> 0 Then
```

```
        PeriodPos = PeriodPos - 1
    Else
        PeriodPos = Len(Filename)
    End If
    OutFile = Left$(Filename, PeriodPos) & ".WTS"
    Msg = " Creating weight data file " & OutFile
Else
    MakeWeight = False
    Msg = "Error, no file will be created."
End If

MsgBox Msg

'Open file selected.
In = FreeFile
Open Filename For Input Access Read Lock Read As In Len = 4096
Out = FreeFile
Open OutFile For Output Access Write Lock Write As Out Len = 4096

'Print output file header
Print #Out, "Time"; Tab(13); ","; Tab(15); "Scan"; Tab(21); ","; Tab(23);
"Status1"; Tab(31); ","; Tab(33); "Weight1"; Tab(42); ","; Tab(44); "Status2";
Tab(52); ","; Tab(54); "Weight2 "

Do While Not EOF(In)

    'Search for Weight Info Header
    TextLine = ""
    Do While Not ((InStr(TextLine, "Time") <> 0) And (InStr(TextLine,
    "Scan") <> 0) And (InStr(TextLine, "Weight1") <> 0) And
    (InStr(TextLine, "Weight2") <> 0)) And Not EOF(In)
        Line Input #In, TextLine
    Loop

    'Get and print weight info, interspersed with commas
    If Not EOF(In) Then
        Line Input #In, TextLine
        TextLine = Mid$(TextLine, 1, 12) & ", " & Mid$(TextLine, 13, 4) & "  ,
        " & Mid$(TextLine, 20, 4) & "    , " & Mid$(TextLine, 27, 8) & " , " &
        Mid$(TextLine, 45, 4) & "    , " & Mid$(TextLine, 52, 8)
```

```
            Print #Out, TextLine
        End If
    Loop

    Close In
    Close Out

    MakeWeight = True
    Exit Function

ErrHandler43:

    MsgBox "Error encountered while trying to open file, please retry.", 48, "Text
    Editor"
    Close In
    Close Out
    MakeWeight = False
    Exit Function
```

End Function

Sub mnuEdit_Click ()

```
    ' Disable Cut and Copy if no text selected.
    mnuEditItem(0).Enabled = (txtEdit.SelLength > 0)
    mnuEditItem(1).Enabled = (txtEdit.SelLength > 0)
```

End Sub

Sub mnuEditItem_Click (Index As Integer)

```
    Select Case Index

    ' If Index = 0, user chose "Cut"
    ' Clear Clipboard.
    ' Copy selected text to Clipboard.
    ' Clear selected text from the document.
    Case 0
        Clipboard.Clear
        Clipboard.SetText txtEdit.SelText
```

```
        txtEdit.SelText = ""

    ' If Index = 1, user chose "Copy"
    ' Clear Clipboard.
    ' Copy selected text into clipboard.
    Case 1
        Clipboard.Clear
        Clipboard.SetText txtEdit.SelText

    ' If Index = 2, user chose "Paste"
    ' Paste Clipboard text (if any) into document.
    Case 2
        txtEdit.SelText = Clipboard.GetText()

    End Select

End Sub

Sub mnuEnvironmentColorItem_Click (Index As Integer)

    ' Set Cancel to True.
    CMDialog1.CancelError = True
    On Error GoTo ErrHandler1

    ' Set initial color selection for dialog.
    CMDialog1.Color = &HFF&

    ' Set the RGB_INIT flag.
    CMDialog1.Flags = &H1&

    ' Display the color dialog box.
    CMDialog1.Action = 3

    Select Case Index

    'If index = 0, user choose Background
    Case 0
        txtEdit.BackColor = CMDialog1.Color

    'If index = 1, user choose Foreground
```

```
    Case 1
        txtEdit.ForeColor = CMDialog1.Color

    End Select

ErrHandler1:

    ' User pressed Cancel button
    Exit Sub
```

End Sub

Sub mnuEnvironmentFont_Click (Index As Integer)

```
    ' CancelError is True
    CMDialog1.CancelError = True
    On Error GoTo ErrHandler2

    ' Set the CF_BOTH, CF_EFFECTS, CF_FORCEONTEXIST flags.
    CMDialog1.Flags = &H3& Or &H200& Or &H10000

    ' Display the Font Dialog box.
    CMDialog1.Action = 4

    ' Set text properties according to user's selections.
    txtEdit.FontName = CMDialog1.FontName
    txtEdit.FontSize = CMDialog1.FontSize
    txtEdit.FontBold = CMDialog1.FontBold
    txtEdit.FontItalic = CMDialog1.FontItalic
    txtEdit.FontUnderLine = CMDialog1.FontUnderLine
    txtEdit.FontStrikeThru = CMDialog1.FontStrikeThru
    txtEdit.ForeColor = txtEdit.ForeColor

ErrHandler2:

    ' User pressed cancel button
    Exit Sub
```

End Sub

```
Sub mnuExperimentDataLogging_Click ()

   ' The show method with style = 1 is used here to display the
   ' form Log as model form.
   frmLogSet.Show 1

End Sub

Sub mnuExperimentSetup_Click ()

   ' The show method with style = 1 is used here to display the
   ' form Log as model form.
   frmSetupTest.Show 1

End S

Su       eArray_Click (Index As Integer)

   ' Open the selected file.
   If Index >= 0 Then
        OpenFile (mnuFileArray(Index).Caption)
   End If

End Sub

Sub mnuFileItem_Click (Index As Integer)

   ' CancelError is True
   CMDialog1.CancelError = True
   On Error GoTo ErrHandler3

   ' Set OFN_ALLOWMULTISELECT, OFN_CREATEPROMPT,
   OFN_HIDEREADONLY, OFN_NOCHANFDIR,
   OFN_NOREADONLYRETURN, OFN_OVERWRITEPROMPT flags.
   CMDialog1.Flags = &H200& Or &H2000& Or &H4& Or &H8& Or
   &H8000& Or &H2&

   ' Check index value of selected menu item.
   Select Case Index
```

```
' If index = 0, the user chooses "New"
' Clear text box
' Set the title bar caption to "New File: Untitled"
Case 0
    txtEdit.Text = ""
    Filename = "Untitled"
    mnuFileItem(4).Enabled = False
    frmMain.Caption = "New File: " & Filename

' If index = 1, the user chooses "Open..."
' Set filters.
' Specify default filter.
' Display the File Open dialog.
Case 1
    CMDialog1.Filter = "All Files (*.*)|*.*|Control Files
    (*.ctl)|..\source\*.ctl|Data Files (*.dat)|..\data\*.dat|Weight Files
    (*.wts)|..\data\*.wts|Pressure Files (*.prs)|..\data\*.prs|Power Files
    (*.pwr)|..\data\*.pwr"
    CMDialog1.FilterIndex = 3
    CMDialog1.Action = 1
    Filename = CMDialog1.Filename
    OpenFile (Filename)
    frmMain.Caption = "Text Editor: " + Filename

' This menu item is a separator bar, no code needs to be written here
' because it cannot be selected and therefore cannot receive a Click event.
Case 2

' If index = 3, the user chooses "Save As..."
' Set filters.
' Specify default filter.
' Display the File Save As dialog
Case 3
    CMDialog1.Filter = "All Files (*.*)|*.*|Control Files
    (*.ctl)|..\source\*.ctl|Data Files (*.dat)|..\data\*.dat|Weight Files
    (*.wts)|..\data\*.wts|Pressure File (*.prs)|..\data\*.prs|Power Files
    (*.pwr)|..\data\*.pwr"
    CMDialog1.FilterIndex = 3
    CMDialog1.Action = 2
    Filename = CMDialog1.Filename
```

```
        CloseFile (Filename)
        frmMain.Caption = "Cocurrent and Countercurrent Flow Experiment
        Control Program"

' If index = 4, the user chooses "Close"
' Clear text box.
' Refresh caption of form.
' Disable this menu item.
' Close the current file.
Case 4
        txtEdit.Text = ""
        frmMain.Caption = "Cocurrent and Countercurrent Flow Experiment
        Control Program"
        mnuFileItem(4).Enabled = False

' This menu item is a separator bar, no code needs to be written here
' because it cannot be selected and therefore cannot receive a Click event.
Case 5

' If index = 6, the user chooses "Print"
' Display the Print dialog box.
' Get user-selected values from the dialog box.
Case 6

        printer.CurrentX = 0
        printer.CurrentY = 0
        printer.Print txtEdit.Text
        printer.Print
        printer.EndDoc

' If index = 7, the user chooses "Printer setup"
' This menu item is for setting up printer
Case 7

        Dim X
        X = Shell("c:\Windows\PrintMan.EXE")

' This menu item is a separator bar, no code needs to be written here
' because it cannot be selected and therefore cannot receive a Click event.
Case 8
```

```
   ' If index = 9, the user chooses "Exit"
   ' End this application and return to the Windows operating system.
   Case 9
        End

End Select

ErrHandler3:

   ' User pressed cancel button
   Exit Sub

End Sub

Sub mnuMiscAbout_Click ()

   ' All the CaptionTextn string variables below are concatenated together with
   ' the appropriate line feed characters to display text in the About dialog.

   Dim NL As String
   Dim CaptionText1 As String
   Dim CaptionText2 As String
   Dim CaptionText3 As String
   Dim CaptionText4 As String
   Dim CaptionText5 As String

   NL = Chr$(13) & Chr$(10)

   CaptionText1 = "Cocurrent and Countercurrent Flow Experiment Control
   Program" & NL & NL
   CaptionText2 = "--------------------------------------------------------------------
   -----------------------" & NL & NL
   CaptionText3 = "Written by Samuel Chang, Jacques Gibeau, and Cameron
   Bentsen" & NL & "Supervisor : Dr. R. G. Bentsen" & NL & "Date : January
   1995" & NL
   CaptionText4 = "Department of Mining, Metallurgy and Petroleum
   Engineering" & NL & "University of Alberta" & NL 'Room 606 Chemical and
   Mineral Engineering Building
   CaptionText5 = "Edmonton, Alberta, Canada" & NL & "T6G 2G6" & NL
```

```
    frmAbout.lblAbout.Caption = CaptionText1 & CaptionText2 & CaptionText3
    & CaptionText4 & CaptionText5

    ' The Show method with style = 1 is used here to display the dialog as
    ' modal. Unloading the dialog is handled in the forms cmdAbout_Click
    ' event procedure.
    frmAbout.Show 1

End Sub

Sub mnuMiscCalculator_Click ()

    frmCalculator.Show

End Sub

Sub mnuMiscCalendar_Click ()

    Dim X

    X = Shell("c:\Windows\Calendar.EXE")

End Sub

Sub mnuMiscClock_Click ()

    Dim X

    X = Shell("C:\Windows\Clock.Exe")

End Sub

Sub mnuMiscControl_Click ()

    Dim X

    X = Shell("C:\Windows\Control.Exe")

End Sub
```

Sub mnuMiscNotePad_Click ()

```
Dim X

X = Shell("c:\Windows\NOTEPAD.EXE")
```

End Sub

Sub mnuMiscPBrush_Click ()

```
Dim X

X = Shell("c:\Windows\pbrush.EXE")
```

End Sub

Sub mnuMiscWrite_Click ()

```
Dim X

X = Shell("C:\Windows\Write.Exe")
```

End Sub

Sub mnuProcessItem_Click (Index As Integer)

```
'Set SuccessFlag to default value
SuccessFlag = False

' CancelError is True
CMDialog1.CancelError = True
On Error GoTo ErrHandler40

' Set OFN_ALLOWMULTISELECT, OFN_CREATEPROMPT,
OFN_HIDEREADONLY, OFN_NOCHANFDIR,
OFN_NOREADONLYRETURN, OFN_OVERWRITEPROMPT flags.
CMDialog1.Flags = &H200& Or &H2000& Or &H4& Or &H8& Or
&H8000& Or &H2&
```

```
CMDialog1.Filter = "All Files (*.*)|*.*|Data Files (*.dat)|..\data\*.dat"
CMDialog1.FilterIndex = 2
CMDialog1.Action = 1
Filename = CMDialog1.Filename
frmMain.Caption = "Processing: " + Filename

' Check index value of selected menu item.

' Create weight file
If (Index = 0) Or (Index = 3) Then
    SuccessFlag = MakeWeight(Filename)
    If Not (SuccessFlag) Then
        MsgBox "Construction of Weight data file failed!"
    Else
        MsgBox "Weight data file construction successful."
    End If
    SuccessFlag = False
End If

' Create pressure file
If (Index = 1) Or (Index = 3) Then
    SuccessFlag = MakePressure(Filename)
    If Not (SuccessFlag) Then
        MsgBox "Construction of Pressure data file failed!"
    Else
        MsgBox "Pressure data file construction successful."
    End If
    SuccessFlag = False
End If

' Create power meter file
If (Index = 2) Or (Index = 3) Then
    SuccessFlag = MakePower(Filename)
    If Not (SuccessFlag) Then
        MsgBox "Construction of Power data file failed!"
    Else
        MsgBox "Power data file construction successful."
    End If
    SuccessFlag = False
End If
```

```
    Exit Sub

ErrHandler40:

    ' User pressed cancel button
    Exit Sub

End Sub
```

Appendix B - 18 PERM.FRM

```
'**********Absolute permeability measurement control form**********
'This form is written to control data collection for absolute permeability
measurement.

Option Explicit

  Dim OldBox As Control
  Dim NewBox As Control
  Dim OldText As String

Sub Form_Load ()

  ' The form is horizontally and vertically centered when loaded.
  Top = Screen.Height / 2 - Height / 2
  Left = Screen.Width / 2 - Width / 2

  ' These are used in conjunction with GFocus to mimic
  ' standard windows text box changing methods while
  ' retaining certain controls on the text allowed.
  Set NewBox = Nothing
  Set OldBox = Frame1
  OldText = ""

  ' Call reset to load the text boxes from the control file.
  Option1.Value = False
  Option2.Value = False
  Option4.Value = False
  Option3.Value = True

End Sub

Sub Form_QueryUnload (Cancel As Integer, Unloadmode As Integer)

  Dim Title As String
  Dim Msg As String
  Dim DgDef As Integer
  Dim Response As Integer

  ' Put together a message box with all the proper components.
  Title = "Warning"
```

```
Msg = "Do you really want to close the window?"
DgDef = MB_YESNO + MB_ICONSTOP + MB_DEFBUTTON2

Cancel = True

' Evaluate unload mode and give warning.
If Unloadmode = 0 Then
   Response = MsgBox(Msg, DgDef, Title)
   If Response = IDYES Then
      Cancel = False
      Unload frmPerm
   Else
      Cancel = True
   End If
Else
      Cancel = False
End If
```

End Sub

Sub Option1_Click ()

```
Dim I As Integer
Dim M As Integer
Dim Duration As Double
Dim Interval As Integer
Dim ScanNo As Integer

Set NewBox = Frame1
Call GFocus(OldBox, NewBox, OldText)
Set OldBox = NewBox
OldText = ""

Print #FileNumber,
Print #FileNumber, "Equipment and experiment settings :"
Print #FileNumber, "Flowing Fluid :"; Tab(35); TxtFluid.Text
Print #FileNumber, "Fluid density @ start temp. :"; Tab(35); TxtFluidDen.Text;
Tab(45); " g/cc"
Print #FileNumber, "Fluid viscosity @ start temp. :"; Tab(35);
TxtFluidVis.Text; Tab(45); " cp."
```

```
Print #FileNumber, "Starting Temperature :"; Tab(35); TxtStartTemp.Text;
Tab(45); " deg. C"
Print #FileNumber, "Pump number :"; Tab(35); TxtPumpNo.Text
Print #FileNumber, "Pump rate :"; Tab(35); TxtPumpRate.Text; Tab(45); "
cc/hr"
Print #FileNumber, "Recording interval :"; Tab(35); TxtInterval.Text; Tab(45);
" sec."
Print #FileNumber, "Balance number :"; Tab(35); TxtBalanceNo.Text
Print #FileNumber, "Porosity :"; Tab(35); TxtPorosity; Tab(45); " %"
Print #FileNumber, "Run number :"; Tab(35); TxtRun.Text
Print #FileNumber, "Core number :"; Tab(35); TxtCore.Text
Print #FileNumber,
Print #FileNumber, "Measured value :"

Interval = Int(Val(TxtInterval.Text))
Scan = Int(Val(TxtScan.Text))

For I = 1 To Scan Step 1
   ScanNo = I
   Duration = Timer + Interval

   Call UPCRead(UPCBase)
   For M = 1 To 16 Step 1
      Pressure(M) = Format(Pressure(M), "0.0000")
   Next M

   Call Balance1Read
   Call Balance2Read

   Print #FileNumber,
   Print #FileNumber, "Time"; Tab(13); "Scan"; Tab(20); "Weight1"; Tab(45);
   "Weight2"
   Print #FileNumber, Time; Tab(13); ScanNo; Tab(20); Weight1; Tab(45);
   Weight2
   Print #FileNumber,
   Print #FileNumber, "Pres.1"; Tab(10); "Pres.2"; Tab(20); "Pres.3"; Tab(30);
   "Pres.4"
   Print #FileNumber, Pressure(1); Tab(10); Pressure(2); Tab(20); Pressure(3);
   Tab(30); Pressure(4)
   Print #FileNumber,
```

```
      Print #FileNumber, "Pres.5"; Tab(10); "Pres.6"; Tab(20); "Pres.7"; Tab(30);
      "Pres.8"
      Print #FileNumber, Pressure(5); Tab(10); Pressure(6); Tab(20); Pressure(7);
      Tab(30); Pressure(8)
      Print #FileNumber,
      Print #FileNumber, "Pres.9"; Tab(10); "Pres.10"; Tab(20); "Pres.11";
      Tab(30); "Pres.12"
      Print #FileNumber, Pressure(9); Tab(10); Pressure(10); Tab(20);
      Pressure(11); Tab(30); Pressure(12)
      Print #FileNumber,
      Print #FileNumber, "Pres.13"; Tab(10); "Pres.14"; Tab(20); "Pres.15";
      Tab(30); "Pres.16"
      Print #FileNumber, Pressure(13); Tab(10); Pressure(14); Tab(20);
      Pressure(15); Tab(30); Pressure(16)
      Print #FileNumber,

      Do Until Timer > Duration
         DoEvents
         If Option1.Value = False Then
            Exit Sub
         End If
      Loop

   Next I

   Option2.Value = True
```

End Sub

Sub Option2_Click ()

```
   Dim Default As Variant
   Dim Msg As String
   Dim FinalTemp As Variant

   Option1.Value = False

   Set NewBox = Frame1
   Call GFocus(OldBox, NewBox, OldText)
   Set OldBox = NewBox
```

```
OldText = ""

' Option 2 ends the experiment and unloads the form.

Msg = "Experiment finished."
Msg = Msg & "Enter final temperature in deg. C."
Default = TxtStartTemp.Text
FinalTemp = InputBox(Msg, "", Default)
If FinalTemp = "" Then
   FinalTemp = TxtStartTemp.Text
End If

Print #FileNumber,
Print #FileNumber, "Final Temperature : "; Tab(35); FinalTemp; Tab(45); " deg.
C"
Close FileNumber

frmLogSet.Text1.Visible = True
frmLogSet.Text2.Visible = False
frmLogSet.Command3D1.Visible = True
frmLogSet.Command3D2.Visible = True
frmLogSet.Command3D3.Visible = False
frmLogSet.Command3D4.Visible = False
frmLogSet.Label1.Visible = False
frmLogSet.Label44.Visible = False
frmLogSet.Label46.Visible = False
frmLogSet.Label48.Visible = False
frmLogSet.Label50.Visible = False
frmLogSet.Label51.Visible = False
frmLogSet.TxtCardBase1.Visible = False
frmLogSet.TxtCardBase2.Visible = False
frmLogSet.TxtHPIBAddr.Visible = False
frmLogSet.Txt601LAddress.Visible = False
frmLogSet.TxtBalance1.Visible = False
frmLogSet.TxtBalance2.Visible = False

Option3.Value = True

Unload frmPerm
```

```
End Sub

Sub Option3_Click ()

    Option1.Value = False

    Dim PermFile As Integer
    Dim Char As Variant
    ReDim TextData(12) As Variant
    Dim I As Integer

    Set NewBox = Frame1
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = ""

    ' Option 3 resets the values in the text boxes to the
    ' default values (found in the control file.)

    PermFile = FreeFile

    Open "Perm.ctl" For Input As PermFile

    For I = 0 To 11
        Do
            Char = Input(1, PermFile)
        Loop Until Char = Chr(58)
        TextData(I) = ""
        While Char <> Chr(13)
            Char = Input(1, PermFile)
            TextData(I) = TextData(I) & Char
        Wend
    Next I

    Close PermFile

    TxtFluid = Left(TextData(0), Len(TextData(0)) - 1)
    TxtFluidDen = Left(TextData(1), Len(TextData(1)) - 1)
    TxtFluidVis = Left(TextData(2), Len(TextData(2)) - 1)
    TxtStartTemp = Left(TextData(3), Len(TextData(3)) - 1)
```

```
    TxtPumpNo = Left(TextData(4), Len(TextData(4)) - 1)
    TxtPumpRate = Left(TextData(5), Len(TextData(5)) - 1)
    TxtBalanceNo = Left(TextData(6), Len(TextData(6)) - 1)
    TxtPorosity = Left(TextData(7), Len(TextData(7)) - 1)
    TxtInterval = Left(TextData(8), Len(TextData(8)) - 1)
    TxtScan = Left(TextData(9), Len(TextData(9)) - 1)
    TxtRun = Left(TextData(10), Len(TextData(10)) - 1)
    TxtCore = Left(TextData(11), Len(TextData(11)) - 1)

    Option3.Value = False

End Sub

Sub Option4_Click ()

    Dim PermFile As Integer
    Dim Response As Integer
    Dim Msg As String
    Dim Title As String

    Set NewBox = Frame1
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = ""

    ' Option 4 saves the current values in the text boxes
    ' to the control file.

    Msg = "Save these settings?"
    Title = "Saving Permeability Settings"
    Response = MsgBox(Msg, 4 + 16 + 256, Title)
    If Response = IDNO Then
        Option1.Value = False
        Exit Sub
    End If

    PermFile = FreeFile
    Open "Perm.ctl" For Output As PermFile

    Print #PermFile,
```

```
    Print #PermFile, "Equipment and experiment settings control file."
    Print #PermFile, "Flowing Fluid :"; TxtFluid.Text
    Print #PermFile, "Fluid density @ start temp. (g/cc):"; TxtFluidDen.Text
    Print #PermFile, "Fluid viscosity @ start temp. (cp.):"; TxtFluidVis.Text
    Print #PermFile, "Starting Temperature (deg. C):"; TxtStartTemp.Text
    Print #PermFile, "Pump number :"; TxtPumpNo.Text
    Print #PermFile, "Pump rate (cc/hr):"; TxtPumpRate.Text
    Print #PermFile, "Balance number :"; TxtBalanceNo.Text
    Print #PermFile, "Porosity (%):"; TxtPorosity
    Print #PermFile, "Recording interval (sec.):"; TxtInterval.Text
    Print #PermFile, "Number of scans :"; TxtScan
    Print #PermFile, "Run number :"; TxtRun.Text
    Print #PermFile, "Core number :"; TxtCore.Text
    Print #PermFile,

    Close PermFile

    Option4.Value = False

End Sub

Sub TxtBalanceNo_GotFocus ()

    Set NewBox = TxtBalanceNo
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtCore_GotFocus ()

    Set NewBox = TxtCore
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtFluid_GotFocus ()
```

```
    Set NewBox = TxtFluid
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtFluidDen_GotFocus ()

    Set NewBox = TxtFluidDen
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtFluidVis_GotFocus ()

    Set NewBox = TxtFluidVis
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtInterval_GotFocus ()

    Set NewBox = TxtInterval
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtPorosity_GotFocus ()

    Set NewBox = TxtPorosity
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
```

```
    OldText = OldBox.Text

End Sub

Sub TxtPumpNo_GotFocus ()

    Set NewBox = TxtPumpNo
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtPumpRate_GotFocus ()

    Set NewBox = TxtPumpRate
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtRun_GotFocus ()

    Set NewBox = TxtRun
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtScan_GotFocus ()

    Set NewBox = TxtScan
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub
```

```
Sub TxtStartTemp_GotFocus ()

   Set NewBox = TxtStartTemp
   Call GFocus(OldBox, NewBox, OldText)
   Set OldBox = NewBox
   OldText = OldBox.Text

End Sub
```

Appendix B - 19 PMSET1.FRM

```
'**********Power meter #1 control form**********
'This form is written to configure the communication settings for power meter #1.

   Option Explicit

   Const BITS_OUT = 8

   Dim CardBase As Integer
   Dim DasErr As Integer
   Dim InitCard As Integer
   Dim Title As String
   Dim Msg As String
   Dim Response As Integer
   Dim DgDef As Integer
   Dim i As Integer


Sub BasAddr_KeyPress (KeyAscii As Integer)

   Dim tstr As String
   Dim ustr As String
   Dim Length As Integer

   ' only allow 0 - 9, A-F, a-f
   If (InputHex(KeyAscii)) Then
      tstr = BasAddr.Text
      Length = 3
      BasAddr.Text = Right$((tstr + Chr$(KeyAscii)), Length)
      ustr = BasAddr.Text
      CardBase = HexStrtoDec(ustr)
      InitCard = True
   End If

   KeyAscii = 0

End Sub

Sub closebutton_Click ()

   ' Unload frmPM1SetupTest from memory.
```

```
    Unload frmPM1SetupTest

    ' Activate command and option buttons on form SetupTest.
    frmSetupTest.Command1.Enabled = False
    frmSetupTest.Command2.Enabled = False
    frmSetupTest.Command3.Enabled = True
    frmSetupTest.Command4.Enabled = True
    frmSetupTest.Command5.Enabled = True

End Sub

Sub Form_Load ()

    Dim i As Integer

    InitCard = True
    CardBase = 768

    ' The form is horizontally and vertically centered when loaded.
    Top = Screen.Height / 2 - Height / 2
    Left = Screen.Width / 2 - Width / 2

    ' Ports are initially INPUT, disable their bits
    For i = 0 To BITS_OUT - 1
      PortABit(i).Enabled = False
    Next i

    For i = 0 To BITS_OUT - 1
      PortBBit(i).Enabled = False
    Next i

    For i = 0 To BITS_OUT - 1
      PortCBit(i).Enabled = True
    Next i

End Sub

Sub Form_QueryUnload (Cancel As Integer, Unloadmode As Integer)

    ' Put together a message box with all the proper components.
```

```
    Title = "Warning"
    Msg = "Do you really want to close the window?"
    DgDef = MB_YESNO + MB_ICONSTOP + MB_DEFBUTTON2

    Cancel = True
    ' Evaluate unload mode and give warning.
    If Unloadmode = 0 Then
       Response = MsgBox(Msg, DgDef, Title)
       If Response = IDYES Then
          Cancel = False
          Unload frmPM1SetupTest
       Else
          Cancel = True
       End If
    Else
          Cancel = False
    End If

    frmSetupTest.Command1.Enabled = False
    frmSetupTest.Command2.Enabled = False
    frmSetupTest.Command3.Enabled = True
    frmSetupTest.Command4.Enabled = True
    frmSetupTest.Command5.Enabled = True

End Sub

Function GetBase () As Integer

    Dim tstr As String

    tstr = BasAddr.Text
    GetBase = HexStrtoDec(tstr)

End Function

Function HexStrtoBitStr (HexStr As String, bitMax As Integer) As String

    Dim i As Integer
    Dim n As Integer
    Dim IntPart As Integer
```

```
   Dim bitStr As String

   ' Convert Hex string to binary string Pattern (MSB is first)
   n = HexStrtoDec(HexStr)

   ' Calculate bit pattern from MSB to LSB
   For i = bitMax - 1 To 0 Step -1
      IntPart = Fix(n / 2 ^ i)
      If (IntPart) Then
         bitStr = bitStr + "1"
         n = n - 2 ^ i
      Else
         bitStr = bitStr + "0"
      End If
   Next i

   HexStrtoBitStr = bitStr

End Function

Function HexStrtoDec (HexStr As String) As Integer

   Dim n As Integer
   Dim tstr As String
   Dim ustr As String

   ustr = UCase$(HexStr)

   ' Convert A-F to 1 - 6 and add to 9.
   ' "A" = &H41, - &H40 -> 1
   ' Assumes:  lower case hex not used (via InputHex)

   For i = Len(ustr) To 1 Step -1
      tstr = Mid$(ustr, i, 1)
      If (tstr >= "A") Then
         n = n + (9 + (Asc(tstr) - &H40)) * (16 ^ (Len(ustr) - i))
      Else
         n = n + Val(tstr) * (16 ^ (Len(ustr) - i))
      End If
   Next i
```

```
HexStrtoDec = n

End Function

Sub inbutton_Click ()

    Dim indata As Integer
    Dim tstr As String

    If InitCard = True Then DasErr = InitPIO()

    If PortADir.Caption = "Input" Then
        PortAHex.Text = Hex$(dasbytein(CardBase))
    End If

    If PortBDir.Caption = "Input" Then
        PortBHex.Text = Hex$(dasbytein(CardBase + 1))
    End If

    If PortCDir.Caption = "Input" Then
        PortCHex.Text = Hex$(dasbytein(CardBase + 2))
    End If

End Sub

Function InitPIO () As Integer

    Dim cntrl As Integer
    cntrl = &H80

    If PortADir.Caption = "Input" Then cntrl = cntrl + &H10
    If PortBDir.Caption = "Input" Then cntrl = cntrl + &H2
    If PortCDir.Caption = "Input" Then cntrl = cntrl + &H9

    DasErr = dasbyteout(CardBase + 3, cntrl)
    InitCard = False

End Function
```

```
Function InputHex (KeyAscii As Integer) As Integer

  ' Check for 0 - 9, A-F, a-f.  Convert a-f to A-F

  If (KeyAscii >= &H30 And KeyAscii <= &H39) Or (KeyAscii >= &H41 And
  KeyAscii <= &H46) Or (KeyAscii >= &H61 And KeyAscii <= &H66) Then
    If (KeyAscii >= &H61) Then KeyAscii = KeyAscii - &H20
      InputHex = True
    Else
      InputHex = False
  End If

End Function

Sub outbutton_Click ()

  Dim outdata As Integer
  Dim tstr As String

  If InitCard = True Then DasErr = InitPIO()

  If PortADir.Caption = "Output" Then
    tstr = PortAHex.Text
    outdata = HexStrtoDec(tstr)
    DasErr = dasbyteout(CardBase, outdata)
  End If

  If PortBDir.Caption = "Output" Then
    tstr = PortBHex.Text
    outdata = HexStrtoDec(tstr)
    DasErr = dasbyteout(CardBase + 1, outdata)
  End If

  If PortCDir.Caption = "Output" Then
    tstr = PortCHex.Text
    outdata = HexStrtoDec(tstr)
    DasErr = dasbyteout(CardBase + 2, outdata)
  End If

End Sub
```

```
Sub PortABit_Click (Index As Integer)

    Dim count As Integer
    Dim i As Integer

    count = 0
    For i = 0 To (BITS_OUT - 1)
        count = count + (2 ^ i) * (Abs(PortABit(i).Value))
    Next i

    PortAHex.Text = (Right$(Hex$(count), BITS_OUT / 4))

End Sub

Sub PortADir_Click ()

    Dim i As Integer
    Dim state As Integer

    If PortADir.Caption = "Output" Then
        PortADir.Caption = "Input"
        state = False
    Else
        PortADir.Caption = "Output"
        state = True
    End If

    For i = 0 To BITS_OUT - 1
        PortABit(i).Enabled = state
    Next i

    InitCard = True

End Sub

Sub PortAHex_Change ()

    Dim j As Integer
    Dim bitStr As String
```

```
If PortADir.Caption = "Input" Then
' Convert Binary Out Fields per Hex Output
   bitStr = HexStrtoBitStr((PortAHex.Text), BITS_OUT)
   j = 1
   For i = BITS_OUT - 1 To 0 Step -1
      PortABit(i).Value = Mid$(bitStr, j, 1)
      j = j + 1
   Next i
End If

End Sub

Sub PortAHex_KeyPress (KeyAscii As Integer)
  Dim cpos As Integer        ' cursor position
  Dim tstr As String
  Dim Length As Integer
  Dim i As Integer
  Dim j As Integer
  Dim bitStr As String

     ' only allow 0 - 9, A-F, a-f

  If (InputHex(KeyAscii) And PortADir.Caption = "Output") Then
     tstr = PortAHex.Text
     cpos = PortAHex.SelStart
     Length = BITS_OUT / 4

        PortAHex.Text = Right$((tstr + Chr$(KeyAscii)), Length)

           ' Convert Binary Out Fields per Hex Output
        bitStr = HexStrtoBitStr((PortAHex.Text), BITS_OUT)
        j = 1
        For i = BITS_OUT - 1 To 0 Step -1
           PortABit(i).Value = Mid$(bitStr, j, 1)
           j = j + 1
        Next i

  End If
```

```
   KeyAscii = 0

End Sub

Sub PortBBit_Click (Index As Integer)

   Dim count As Integer
   Dim i As Integer

   count = 0
   For i = 0 To (BITS_OUT - 1)
      count = count + (2 ^ i) * (Abs(PortBBit(i).Value))
   Next i

   PortBHex.Text = (Right$(Hex$(count), BITS_OUT / 4))

End Sub

Sub PortBDir_Click ()

Dim i As Integer

   Dim state As Integer

   If PortBDir.Caption = "Output" Then
      PortBDir.Caption = "Input"
      state = False
   Else
      PortBDir.Caption = "Output"
      state = True
   End If

   For i = 0 To BITS_OUT - 1
      PortBBit(i).Enabled = state
   Next i

   InitCard = True

End Sub
```

```
Sub PortBHex_Change ()

    Dim j As Integer
    Dim bitStr As String

    If PortBDir.Caption = "Input" Then
    ' Convert Binary Out Fields per Hex Output
       bitStr = HexStrtoBitStr((PortBHex.Text), BITS_OUT)
       j = 1
       For i = BITS_OUT - 1 To 0 Step -1
          PortBBit(i).Value = Mid$(bitStr, j, 1)
          j = j + 1
       Next i
    End If

End Sub

Sub PortBHex_KeyPress (KeyAscii As Integer)
    Dim cpos As Integer          ' cursor position
    Dim tstr As String
    Dim Length As Integer
    Dim i As Integer
    Dim j As Integer
    Dim bitStr As String

       ' only allow 0 - 9, A-F, a-f

    If (InputHex(KeyAscii) And PortBDir.Caption = "Output") Then
       tstr = PortBHex.Text
       cpos = PortBHex.SelStart
       Length = BITS_OUT / 4

          PortBHex.Text = Right$((tstr + Chr$(KeyAscii)), Length)

             ' Convert Binary Out Fields per Hex Output
          bitStr = HexStrtoBitStr((PortBHex.Text), BITS_OUT)
          j = 1
          For i = BITS_OUT - 1 To 0 Step -1
             PortBBit(i).Value = Mid$(bitStr, j, 1)
             j = j + 1
```

```
        Next i

    End If

    KeyAscii = 0


End Sub

Sub PortCBit_Click (Index As Integer)

    Dim count As Integer
    Dim i As Integer

    count = 0
    For i = 0 To (BITS_OUT - 1)
        count = count + (2 ^ i) * (Abs(PortCBit(i).Value))
    Next i

    PortCHex.Text = (Right$(Hex$(count), BITS_OUT / 4))

End Sub

Sub PortCDir_Click ()

    Dim i As Integer
    Dim state As Integer

    If PortCDir.Caption = "Output" Then
        PortCDir.Caption = "Input"
        state = False
    Else
        PortCDir.Caption = "Output"
        state = True
    End If

    For i = 0 To BITS_OUT - 1
        PortCBit(i).Enabled = state
    Next i
```

```
    InitCard = True

End Sub

Sub PortCHex_Change ()

    Dim j As Integer
    Dim bitStr As String

    If PortCDir.Caption = "Input" Then
    ' Convert Binary Out Fields per Hex Output
        bitStr = HexStrtoBitStr((PortCHex.Text), BITS_OUT)
        j = 1
        For i = BITS_OUT - 1 To 0 Step -1
            PortCBit(i).Value = Mid$(bitStr, j, 1)
            j = j + 1
        Next i
    End If

End Sub

Sub PortCHex_KeyPress (KeyAscii As Integer)

    Dim cpos As Integer         ' cursor position
    Dim tstr As String
    Dim Length As Integer
    Dim i As Integer
    Dim j As Integer
    Dim bitStr As String

        ' only allow 0 - 9, A-F, a-f

    If (InputHex(KeyAscii) And PortCDir.Caption = "Output") Then
        tstr = PortCHex.Text
        cpos = PortCHex.SelStart
        Length = BITS_OUT / 4

            PortCHex.Text = Right$((tstr + Chr$(KeyAscii)), Length)
```

```
        ' Convert Binary Out Fields per Hex Output
        bitStr = HexStrtoBitStr((PortCHex.Text), BITS_OUT)
        j = 1
        For i = BITS_OUT - 1 To 0 Step -1
            PortCBit(i).Value = Mid$(bitStr, j, 1)
            j = j + 1
        Next i

    End If

    KeyAscii = 0

End Sub
```

Appendix B - 20 PMSET2.FRM

```
' ********** Power meter #2 Setup and Testing program **********
' This program is written to utilities Hewlett Packard Standard Instrument
' Control Library installed to communicate to a HP 436A power meter using
' HP 82335 interface card.  The card was set at address 7 and the power meter
' was set at an address 13.

    Option Explicit

    Dim actual As Long              ' # of bytes xfrd
    Dim Inputdata As String * 128       ' buffer used for iread
    Dim Crlfpos As Integer          ' loc of CR's and LF's in readbuf
    Dim commandStr As String * 128   ' command passed to instrument
    Dim index As Integer            ' used to parse SCPI error message
    Dim Timeout As Variant
    Dim Duration As Integer
    Dim Instruction As String

    Dim DgDef
    Dim Msg
    Dim Response
    Dim Title

Sub cmdFinish_Click ()

    ' Unload frmPM2SetupTest from memory.
    Unload frmPM2SetupTest

    ' Activate command and option buttons on form SetupTest.
    frmSetupTest.Command1.Enabled = False
    frmSetupTest.Command2.Enabled = False
    frmSetupTest.Command3.Enabled = True
    frmSetupTest.Command4.Enabled = True
    frmSetupTest.Command5.Enabled = True

End Sub

Sub cmdSend_Click ()

    Dim HPIBAddr As String
```

```
' Set up an error handler within this subroutine that will get
' called if a SICL error occurs.
On Error GoTo ErrorHandler

' Disable the button used to initiate I/O while I/O is
' being performed.
cmdSend.Enabled = False

' Clear the response string in the txtResponse TextBox.
txtResponse.Text = ""

' Open a device session using the device address contained in
' the Text field of the txtInstAddr TextBox.

HPIBAddr = txtInstAddr.Text
If HPIBID = 0 Then
HPIBID = iopen(HPIBAddr)
End If

' Set the I/O time-out value for this session to user input in seconds.
Timeout = Val(txtTimeout.Text)
Call itimeout(HPIBID, Timeout * 1000)

' Write the command to the instrument terminated by a linefeed.
commandStr = Instruction + Chr$(10)
Call iwrite(HPIBID, ByVal commandStr, Len(Instruction) + 1, 1, actual)

' Query the instrument to see if the command was accepted
Call iwrite(HPIBID, ByVal "SYST:ERR?" + Chr$(10), 10, 1, actual)

' Read in the response to the "SYST:ERR?" query command.
Call iread(HPIBID, ByVal Inputdata, 16, ByVal 0&, actual)

Crlfpos = InStr(Inputdata, Chr$(13))
If Crlfpos Then
   Inputdata = Left(Inputdata, Crlfpos - 1)
End If
Crlfpos = InStr(Inputdata, Chr$(10))
If Crlfpos Then
```

```
        Inputdata = Left(Inputdata, Crlfpos - 1)
    End If
    txtResponse.Text = Inputdata

    ' Close the device session.
    If HPIBID <> 0 Then
        Call iclose(HPIBID)
        HPIBID = 0
    End If

    ' Enable the button used to initiate I/O
    cmdSend.Enabled = True

    Exit Sub

ErrorHandler:

    ' Display the error message in the txtResponse TextBox.
    txtResponse.Text = "Error : " + Error$

    ' Close the device session if iopen was successful.
    If HPIBID <> 0 Then
        Call iclose(HPIBID)
        HPIBID = 0
    End If

    ' Enable the button used to initiate I/O
    cmdSend.Enabled = True

    Exit Sub

End Sub

Sub Combo1_Click ()

    If Combo1.Text = "9A-T" Then
        Instruction$ = "9A-T"
    End If
    If Combo1.Text = "9A-R" Then
        Instruction$ = "9A-R"
```

```
End If
If Combo1.Text = "9A-V" Then
   Instruction$ = "9A-V"
End If

If Combo1.Text = "9A-I" Then
   Instruction$ = "9A-I"
End If

If Combo1.Text = "3A+I" Then
   Instruction$ = "3A+I"
End If

If Combo1.Text = "+T, disable(100%) cal. factor, trigger with settling time."
Then
   Instruction$ = "+T"
End If

If Combo1.Text = "Z9A, sensor auto zero mode, auto range, watt mode." Then
   Instruction$ = "Z9A"
End If

If Combo1.Text = "Z, sensor auto zero mode." Then
   Instruction$ = "Z"
End If

If Combo1.Text = "A, watt mode." Then
   Instruction$ = "A"
End If

If Combo1.Text = "B, dB(Rel) mode." Then
   Instruction$ = "B"
End If

If Combo1.Text = "C, dB(Ref) mode." Then
   Instruction$ = "C"
End If

If Combo1.Text = "D, dBm mode." Then
   Instruction$ = "D"
```

```
End If

If Combo1.Text = "+, disable(100%) cal factor." Then
   Instruction$ = "+"
End If

If Combo1.Text = "-, enable (front panel switch setting." Then
   Instruction$ = "-"
End If

If Combo1.Text = "H, hold measurement rate." Then
   Instruction$ = "H"
End If

If Combo1.Text = "T, trigger with settling time." Then
   Instruction$ = "T"
End If

If Combo1.Text = "I, trigger immediately." Then
   Instruction$ = "I"
End If

If Combo1.Text = "R, free run at maximum rate." Then
   Instruction$ = "R"
End If

If Combo1.Text = "V, free run with settling time." Then
   Instruction$ = "V"
End If

If Combo1.Text = "5, least sensitive range." Then
   Instruction$ = "5"
End If

If Combo1.Text = "4, next to least sensitive range." Then
   Instruction$ = "4"
End If

If Combo1.Text = "3, middle range." Then
   Instruction$ = "3"
```

```
    End If

    If Combo1.Text = "2, next to most sensitive range." Then
      Instruction$ = "2"
    End If

    If Combo1.Text = "1, most sensitive range." Then
      Instruction$ = "1"
    End If

    If Combo1.Text = "9, auto range." Then
      Instruction$ = "9"
    End If
```

End Sub

Sub Combo1_DblClick ()

```
    cmdSend_Click
```

End Sub

Sub Form_Load ()

```
    ' The form is horizontally and vertically centered when loaded.
    Top = Screen.Height / 2 - Height / 2
    Left = Screen.Width / 2 - Width / 2

    ' Add items to list for selection.
    Combo1.AddItem "Z9A, sensor auto zero mode, auto range, watt mode.", 0
    Combo1.AddItem "3A+I", 1
    Combo1.AddItem "9A-I", 2
    Combo1.AddItem "9A-T", 3
    Combo1.AddItem "9A-R", 4
    Combo1.AddItem "9A-V", 5
    Combo1.AddItem "+T, disable(100%) cal factor, trigger with settling time."
    Combo1.AddItem "A, watt mode."
    Combo1.AddItem "B, dB(Rel) mode."
    Combo1.AddItem "C, dB(Ref) mode."
    Combo1.AddItem "D, dBm mode."
```

```
    Combo1.AddItem "1, most sensitive range."
    Combo1.AddItem "2, next to most sensitive range."
    Combo1.AddItem "3, middle range."
    Combo1.AddItem "4, next to least sensitive range."
    Combo1.AddItem "5, least sensitive range."
    Combo1.AddItem "9, auto range."
    Combo1.AddItem "+, disable(100%) cal factor."
    Combo1.AddItem "-, enable (front panel switch setting."
    Combo1.AddItem "H, hold measurement rate."
    Combo1.AddItem "T, trigger with settling time."
    Combo1.AddItem "I, trigger immediately."
    Combo1.AddItem "R, free run at maximum rate."
    Combo1.AddItem "V, free run with settling time."
    Combo1.AddItem "Z, sensor auto zero mode."
```

End Sub

Sub Form_QueryUnload (Cancel As Integer, Unloadmode As Integer)

```
    ' Put together a message box with all the proper components.
    Title = "Warning"
    Msg = "Do you really want to close the window?"
    DgDef = MB_YESNO + MB_ICONSTOP + MB_DEFBUTTON2

    Cancel = True

    ' Evaluate unload mode and give warning.
    If Unloadmode = 0 Then
       Response = MsgBox(Msg, DgDef, Title)
       If Response = IDYES Then
          Cancel = False
          Unload frmPM2SetupTest
       Else
          Cancel = True
       End If
    Else
          Cancel = False
    End If

    frmSetupTest.Command1.Enabled = False
```

```
    frmSetupTest.Command2.Enabled = False
    frmSetupTest.Command3.Enabled = True
    frmSetupTest.Command4.Enabled = True
    frmSetupTest.Command5.Enabled = True

End Sub

Sub Form_Unload (Cancel As Integer)

   ' Tell SICL to clean up for this task
   Call siclcleanup

End Sub
```

Appendix B - 21 SCAN.FRM

```
'**********Microwave scanning control form**********
'This form is written to control data collection for microwave scans.

Option Explicit

   Dim OldBox As Control
   Dim NewBox As Control
   Dim OldText As String
```

Sub Form_Load ()

```
   ' The form is horizontally and vertically centered when loaded.
   Top = Screen.Height / 2 - Height / 2
   Left = Screen.Width / 2 - Width / 2

   ' These are used in conjunction with GFocus to mimic
   ' standard windows text box changing methods while
   ' retaining certain controls on the text allowed.
   Set NewBox = Nothing
   Set OldBox = Frame1
   OldText = ""

   ' Call reset to load the text boxes from the control file.
   Option1.Enabled = True
   Option2.Enabled = True
   Option3.Enabled = True
   Option4.Enabled = True

   Option1.Value = False
   Option2.Value = False
   Option4.Value = False
   Option3.Value = True
```

End Sub

Sub Form_QueryUnload (Cancel As Integer, Unloadmode As Integer)

```
   Dim Title As String
   Dim Msg As String
   Dim DgDef As Integer
```

```
    Dim Response As Integer

    ' Put together a message box with all the proper components.
    Title = "Warning"
    Msg = "Do you really want to close the window?"
    DgDef = MB_YESNO + MB_ICONSTOP + MB_DEFBUTTON2

    Cancel = True

    ' Evaluate unload mode and give warning.
    If Unloadmode = 0 Then
        Response = MsgBox(Msg, DgDef, Title)
        If Response = IDYES Then
            Cancel = False
            Unload frmScan
        Else
            Cancel = True
        End If
    Else
            Cancel = False
    End If

End Sub

Sub Form_Unload (Cancel As Integer)

    Call iclear(HPIBID)
    Call ilocal(HPIBID)
    Call iclose(HPIBID)

   ' Tell SICL to clean up for this task
   Call siclcleanup

End Sub

Sub Option1_Click ()

    Dim I As Integer
    Dim J As Integer
    Dim K As Integer
```

```
Dim L As Integer
Dim Scan As Integer
Dim Duration As Long
Dim TimeInterval As Long
Dim DeltaTime As Double
Dim WaitTime As Double

Set NewBox = Frame1
Call GFocus(OldBox, NewBox, OldText)
Set OldBox = NewBox
OldText = ""

Station = Int(Val(TxtStation.Text))
If Station <> 0 Then
  Option1.Value = True
Else
  MsgBox "# of stations must be greater than zero!"
  Option3.Value = True
  Exit Sub
End If

IntervalDistance = 100 / Station
Scan = Int(Val(TxtScan.Text))
IntervalPulse = DistanceToPulse(IntervalDistance)
TotalPulse = CLng(IntervalPulse * (Station + 0))
TimeInterval = Int(Val(TxtInterval.Text))

If Scan <> 0 Then
  Option1.Value = True
Else
  MsgBox "# of scan must be greater than zero!"
  Option3.Value = True
  Exit Sub
End If

If SaturationTime = .2 Then
  If TxtInterval.Text <= 60 Then
    MsgBox "Scan interval must be greater than 60 sec.!"
    Option3.Value = True
    Exit Sub
```

```
  End If
End If

If SaturationTime = 1 Then
  If TxtInterval.Text <= 200 Then
    MsgBox "Scan interval must be greater than 200 sec.!"
    Option3.Value = True
    Exit Sub
  End If
End If

If SaturationTime = 5 Then
  If TxtInterval.Text <= 600 Then
    MsgBox "Scan interval must be greater than 600 sec.!"
    Option3.Value = True
    Exit Sub
  End If
End If

If SaturationTime = 10 Then
  If TxtInterval.Text <= 1200 Then
    MsgBox "Scan interval must be greater than 1200 sec.!"
    Option3.Value = True
    Exit Sub
  End If
End If

Option1.Enabled = False

Print #FileNumber,
Print #FileNumber, "Equipment and experiment settings :"
Print #FileNumber, "Frequency meter setting :"; Tab(35); TxtFreqSetting.Text;
Tab(45); " GHz"
Print #FileNumber, "Attenuator setting :"; Tab(35); TxtAttenSetting.Text;
Tab(45); " dB"
Print #FileNumber, "Gunn current :"; Tab(35); TxtGunnCurrent.Text; Tab(45);
" A"
Print #FileNumber, "Gunn voltage :"; Tab(35); TxtGunnVoltage.Text; Tab(45);
" V"
Print #FileNumber, "Tuner angle :"; Tab(35); TxtTunerAngle.Text; Tab(45); "
```

```
deg."
Print #FileNumber, "Number of stations :"; Tab(35); TxtStation.Text
Print #FileNumber, "Number of scans :"; Tab(35); TxtScan.Text
Print #FileNumber, "Scan interval :"; Tab(35); TxtInterval.Text; Tab(45); "
sec."
Print #FileNumber, "Porosity :"; Tab(35); TxtPorosity.Text; Tab(45); " %"
Print #FileNumber, "Saturation :"; Tab(35); TxtSaturation.Text; Tab(45); " %"
Print #FileNumber, "Run number :"; Tab(35); TxtRun.Text
Print #FileNumber, "Core number :"; Tab(35); TxtCore.Text
Print #FileNumber,
Print #FileNumber, "Measured value :"

For I = 1 To Scan Step 1

  Duration = Timer + TimeInterval

  For J = 1 To Station Step 1
    Option3.Enabled = False
    Option4.Enabled = False
    ActualTime(J) = Timer
    Call HP436ARead(J)
    Call HP432CRead(PortA2Addr, PortB2Addr, J)
    Call ForwardMove(PIO1Addr, IntervalPulse)
    DoEvents
    If Option1.Value = False Then
      Exit Sub
    End If
  Next J

  Print #FileNumber,
  Print #FileNumber, "Scan"; Tab(10); "Station"; Tab(20); "Delta Time(sec.)";
  Tab(40); "Input watt"; Tab(55); "Attenuated watt"
  Print #FileNumber,

  For J = 1 To Station Step 1
    Call HP432CConversion(J, HP432CUMeasurment(J),
    HP432CLMeasurment(J))
    Call HP436AConversion(J, HP436ARaw(J))
    DeltaTime = ActualTime(J) - ActualTime(1)
    Print #FileNumber, I; Tab(10); J; Tab(20); DeltaTime; Tab(40);
```

```
      Format$(HP432CMeasurement(J), "0.00E+00"); Tab(55);
      Format$(HP436AMeasurement(J), "0.00E+00")
      DoEvents
      If Option1.Value = False Then
        Exit Sub
      End If
    Next J

    Call ReverseMove(PIO1Addr, TotalPulse)

    If (Scan > 1) And (I < Scan) Then
      Do Until Timer > Duration
        DoEvents
        If Option1.Value = False Then
          Exit Sub
        End If
      Loop
    End If

    Option3.Enabled = True

    label3.Caption = "Scans remaining :"
    TxtScan.ForeColor = &HFF
    TxtScan.Text = Int(TxtScan.Text) - 1

  Next I

  Option1.Enabled = True
  Option4.Enabled = True
  Option2.Value = True

End Sub

Sub Option2_Click ()

  Dim Default As Variant
  Dim Msg As String
  Dim FinalTemp As Variant

  ' Option 2 ends the experiment and unloads the form.
```

```
Option1.Enabled = True
Option2.Enabled = True
Option3.Enabled = True
Option4.Enabled = True

Option1.Value = False

Set NewBox = Frame1
Call GFocus(OldBox, NewBox, OldText)
Set OldBox = NewBox
OldText = ""

Msg = "Experiment finished."
Msg = Msg & "Enter final temperature in deg. C."
Default = TxtStartTemp.Text
FinalTemp = InputBox(Msg, "", Default)

Print #FileNumber,
Print #FileNumber, "Final Temperature : "; Tab(35); FinalTemp; Tab(45); " deg.
C"
Close FileNumber

frmLogSet.Text1.Visible = True
frmLogSet.Text3.Visible = False
frmLogSet.Command3D1.Visible = True
frmLogSet.Command3D2.Visible = True
frmLogSet.Command3D5.Visible = False
frmLogSet.Command3D6.Visible = False
frmLogSet.Label44.Visible = False
frmLogSet.Label46.Visible = False
frmLogSet.Label48.Visible = False
frmLogSet.TxtCardBase1.Visible = False
frmLogSet.TxtCardBase2.Visible = False
frmLogSet.TxtHPIBAddr.Visible = False

Option3.Value = True

Unload frmScan
```

```
End Sub

Sub Option3_Click ()

    Option1.Enabled = True
    Option1.Value = False

    Dim ScanFile As Integer
    Dim Char As Variant
    ReDim TextData(13) As Variant
    Dim I As Integer

    label3.Caption = "Number of scans :"
    TxtScan.ForeColor = &HFF0000

    Set NewBox = Frame1
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = ""

    ' Option 3 resets the values for all the text boxes to
    ' their default values (found in the control file)

    ScanFile = FreeFile

    Open "Scan.ctl" For Input As ScanFile

    For I = 0 To 12
        Do
            Char = Input(1, ScanFile)
        Loop Until Char = Chr(58)
        TextData(I) = ""
        While Char <> Chr(13)
            Char = Input(1, ScanFile)
            TextData(I) = TextData(I) & Char
        Wend
    Next I

    Close ScanFile
```

```
    TxtFreqSetting.Text = Left(TextData(0), Len(TextData(0)) - 1)
    TxtAttenSetting.Text = Left(TextData(1), Len(TextData(1)) - 1)
    TxtGunnCurrent.Text = Left(TextData(2), Len(TextData(2)) - 1)
    TxtGunnVoltage.Text = Left(TextData(3), Len(TextData(3)) - 1)
    TxtTunerAngle.Text = Left(TextData(4), Len(TextData(4)) - 1)
    TxtStartTemp.Text = Left(TextData(5), Len(TextData(5)) - 1)
    TxtScan.Text = Left(TextData(6), Len(TextData(6)) - 1)
    TxtStation.Text = Left(TextData(7), Len(TextData(7)) - 1)
    TxtInterval.Text = Left(TextData(8), Len(TextData(8)) - 1)
    TxtPorosity.Text = Left(TextData(9), Len(TextData(9)) - 1)
    TxtRun.Text = Left(TextData(10), Len(TextData(10)) - 1)
    TxtCore.Text = Left(TextData(11), Len(TextData(11)) - 1)
    TxtSaturation.Text = Left(TextData(12), Len(TextData(12)) - 1)

    Option3.Value = False

End Sub

Sub Option4_Click ()

    Dim ScanFile As Integer
    Dim Response As Integer
    Dim Msg As String
    Dim Title As String

    Set NewBox = Frame1
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = ""

    ' Option 4 saves the current values in the text boxes
    ' to the control file.

    Msg = "Save these settings?"
    Title = "Saving Scan Settings"
    Response = MsgBox(Msg, 4 + 16 + 256, Title)
    If Response = IDNO Then
        Option1.Value = False
        Exit Sub
    End If
```

```
    ScanFile = FreeFile

    Open "Scan.ctl" For Output As ScanFile

    Print #ScanFile,
    Print #ScanFile, "Equipment and experiment settings control file."
    Print #ScanFile, "Frequency meter setting (GHz):"; TxtFreqSetting.Text
    Print #ScanFile, "Attenuator setting (dB):"; TxtAttenSetting.Text
    Print #ScanFile, "Gunn current (A):"; TxtGunnCurrent.Text
    Print #ScanFile, "Gunn voltage (V):"; TxtGunnVoltage.Text
    Print #ScanFile, "Tuner angle (deg.):"; TxtTunerAngle.Text
    Print #ScanFile, "Starting temperature (deg. C):"; TxtStartTemp.Text
    Print #ScanFile, "Number of scans :"; TxtScan.Text
    Print #ScanFile, "Number of stations :"; TxtStation.Text
    Print #ScanFile, "Scan interval (sec.):"; TxtInterval.Text
    Print #ScanFile, "Porosity (%):"; TxtPorosity.Text
    Print #ScanFile, "Run number :"; TxtRun.Text
    Print #ScanFile, "Core number :"; TxtCore.Text
    Print #ScanFile, "Saturation :"; TxtSaturation.Text
    Print #ScanFile,

    Close ScanFile

    Option4.Value = False

End Sub

Sub TxtAttenSetting_GotFocus ()

    Set NewBox = TxtAttenSetting
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtCore_GotFocus ()

    Set NewBox = TxtCore
```

```
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtFreqSetting_GotFocus ()

    Set NewBox = TxtFreqSetting
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtGunnCurrent_GotFocus ()

    Set NewBox = TxtGunnCurrent
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtGunnVoltage_GotFocus ()

    Set NewBox = TxtGunnVoltage
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtInterval_GotFocus ()

    Set NewBox = TxtInterval
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text
```

```
End Sub

Sub TxtPorosity_GotFocus ()

    Set NewBox = TxtPorosity
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtRun_GotFocus ()

    Set NewBox = TxtRun
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtSaturation_GotFocus ()

    Set NewBox = TxtSaturation
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtScan_GotFocus ()

    Set NewBox = TxtScan
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtStartTemp_GotFocus ()
```

```
    Set NewBox = TxtStartTemp
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtStation_GotFocus ()

    Set NewBox = TxtStation
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub TxtTunerAngle_GotFocus ()

    Set NewBox = TxtTunerAngle
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub
```

Appendix B - 22 SETUPTES.FRM

```
'**********Equipment setup form**********
'This form is written to select proper equipment for data collection under different
'experimental requirement.

Option Explicit

   Dim DgDef As Variant
   Dim Msg As String
   Dim Response As Variant
   Dim Title As Variant


Sub Command1_Click ()

   Dim X

   Command2.Enabled = False
   Command3.Enabled = False

   If frmSetupTest.List1.ListIndex = 0 Then
      frmBalSet1.Show 1
   End If

   If frmSetupTest.List1.ListIndex = 1 Then
      frmBalSet2.Show 1
   End If

   If frmSetupTest.List1.ListIndex = 4 Then
      X = Shell("D:\FILES\EXPT\VALIDYNE\ES201\UPC.EXE", 3)
      frmSetupTest.Command1.Enabled = False
      frmSetupTest.Command3.Enabled = True
      frmSetupTest.Command4.Enabled = True
      frmSetupTest.Command5.Enabled = True
   End If

End Sub

Sub Command2_Click ()

   Command1.Enabled = False
```

```
    Command3.Enabled = False

    If frmSetupTest.List1.ListIndex = 0 Then
      frmBalTest1.Show 1
    End If

    If frmSetupTest.List1.ListIndex = 1 Then
      frmBalTest2.Show 1
    End If

    If frmSetupTest.List1.ListIndex = 4 Then
      frmTransducerTest.Show 1
    End If

    If frmSetupTest.List1.ListIndex = 2 Then
      frmPM1SetupTest.Show 1
    End If

    If frmSetupTest.List1.ListIndex = 3 Then
      frmPM2SetupTest.Show 1
    End If

    If frmSetupTest.List1.ListIndex = 5 Then
      frmSMSetupTest.Show 1
    End If

End Sub

Sub Command3_Click ()

    Unload frmSetupTest

End Sub

Sub Command4_Click ()

    List1.Enabled = False
    Command1.Enabled = False
    Command2.Enabled = False
    Command3.Enabled = False
```

```
    Command4.Enabled = False
    Command5.Enabled = False

    frmEquipmentRecord.Show 1

    List1.Enabled = True
    Command1.Enabled = False
    Command2.Enabled = False
    Command3.Enabled = True
    Command4.Enabled = True
    Command5.Enabled = True

End Sub

Sub Command5_Click ()

    List1.Enabled = False
    Command1.Enabled = False
    Command2.Enabled = False
    Command3.Enabled = False
    Command4.Enabled = False
    Command5.Enabled = False

    frmTransducerCali.Show 1

    List1.Enabled = True
    Command1.Enabled = False
    Command2.Enabled = False
    Command3.Enabled = True
    Command4.Enabled = True
    Command5.Enabled = True

End Sub

Sub Form_Load ()

    List1.Enabled = True

    Command1.Enabled = False
    Command2.Enabled = False
```

```
    Command3.Enabled = True
    Command4.Enabled = True
    Command5.Enabled = True

    ' The form is horizontally and vertically centered when loaded.
    Top = Screen.Height / 2 - Height / 2
    Left = Screen.Width / 2 - Width / 2

    List1.AddItem "Balance #1", 0
    List1.AddItem "Balance #2", 1
    List1.AddItem "Power Meter #1", 2
    List1.AddItem "Power Meter #2", 3
    List1.AddItem "Pressure Transducer", 4
    List1.AddItem "Stepping Motor", 5

End Sub

Sub Form_QueryUnload (Cancel As Integer, Unloadmode As Integer)

    ' Put together a message box with all the proper components.
    Title = "Warning"
    Msg = "Do you really want to close the window?"
    DgDef = MB_YESNO + MB_ICONSTOP + MB_DEFBUTTON2

    Cancel = True
    ' Evaluate unload mode and give warning.
    If Unloadmode = 0 Then
        Response = MsgBox(Msg, DgDef, Title)
        If Response = IDYES Then
            Cancel = False
            Unload frmSetupTest
        Else
            Cancel = True
        End If
    Else
            Cancel = False
    End If

End Sub
```

```
Sub List1_Click ()

   Command1.Enabled = True
   Command2.Enabled = True
   Command4.Enabled = False
   Command5.Enabled = False

   If frmSetupTest.List1.ListIndex = 2 Then
      Command1.Enabled = False
   End If

   If frmSetupTest.List1.ListIndex = 3 Then
      Command1.Enabled = False
   End If

   If frmSetupTest.List1.ListIndex = 5 Then
      Command1.Enabled = False
   End If

   If frmSetupTest.List1.ListIndex = 6 Then
      Command2.Enabled = False
   End If

End Sub

Sub List1_DblClick ()

   Command2_Click

End Sub
```

Appendix B - 23 SM.FRM

```
'**********Stepping motor control form**********
'This form is written to control stepping motor.

    Option Explicit

    Dim Title As Variant
    Dim Msg As String
    Dim DgDef As Variant
    Dim Response As Variant
    Dim Dummy As Integer
    Dim Inner As Integer
    Dim Tic As Integer

    Dim DASErr As Integer
    Dim PortCAddr As Integer
    Dim CntrlAddr As Integer
    Dim ClearData As Integer
    Dim Distance As Single

    Dim PulseCount As Long
    Dim Outer As Long
    Dim OutData1 As Integer
    Dim OutData2 As Integer

    Dim Tstr As String
    Dim Ustr As String
    Dim Length As Integer

    Dim InitCard As Integer
    Dim CardBase As Integer
    Dim Cntrl As Integer
```

Sub BasAddr_KeyPress (KeyAscii As Integer)

```
    Dim Length As Integer

    ' Scroll out the most significant digit when KeyAscii is a hex digit.
    If (InputHex(KeyAscii)) Then
        Tstr = BasAddr.Text
        Length = 3
```

```
      BasAddr.Text = Right$((Tstr + Chr$(KeyAscii)), Length)
      Ustr = BasAddr.Text
      CardBase = HexStrtoDec(Ustr)
      InitCard = True
   End If

   ' Intercept the character pressed if it is not hex character.
   KeyAscii = 0
```

End Sub

Sub cmdOK_Click ()

```
   If InitCard = False Then
      InitCard = True
   End If

   ' Unload frmSMLSSetupTest from memory.
   Unload frmSMSetupTest

   frmSetupTest.Command1.Enabled = False
   frmSetupTest.Command2.Enabled = False
   frmSetupTest.Command3.Enabled = True
   frmSetupTest.Command4.Enabled = True
   frmSetupTest.Command5.Enabled = True
```

End Sub

Function DistanceToPulse (Distance As Single) As Long

```
   DistanceToPulse = Int(367 * Distance)
```

End Function

Sub Form_Load ()

```
   ' The form is horizontally and vertically centered when loaded.
   Top = Screen.Height / 2 - Height / 2
   Left = Screen.Width / 2 - Width / 2
```

```
   frmSMSetupTest.List1.AddItem "Forward", 0
   frmSMSetupTest.List1.AddItem "Reverse", 1

   frmSMSetupTest.List1.Selected(0) = True

   InitCard = True
   CardBase = 512
   Option1.Value = False

End Sub

Sub Form_QueryUnload (Cancel As Integer, Unloadmode As Integer)

   ' Put together a message box with all the proper components.
   Title = "Warning"
   Msg = "Do you really want to close the window?"
   DgDef = MB_YESNO + MB_ICONSTOP + MB_DEFBUTTON2

   Cancel = True
   ' Evaluate unload mode and give warning.
   If Unloadmode = 0 Then
      Response = MsgBox(Msg, DgDef, Title)
      If Response = IDYES Then
         Cancel = False
         Unload frmSMSetupTest
      Else
         Cancel = True
      End If
   Else
         Cancel = False
   End If

   frmSetupTest.Command1.Enabled = False
   frmSetupTest.Command2.Enabled = False
   frmSetupTest.Command3.Enabled = True
   frmSetupTest.Command4.Enabled = True
   frmSetupTest.Command5.Enabled = True

End Sub
```

```
Function GetBase () As Integer

   ' Get the starting address for card, convert to decimal format from
   ' Hexadecimal format.

   Tstr = BasAddr.Text
   GetBase = HexStrtoDec(Tstr)

End Function

Function HexStrtoDec (HexStr As String) As Integer

   Dim n As Integer
   Dim i As Integer

   ' Convert hex string to decimal format.
   ' Convert A-F to 1 - 6 and add to 9.
   ' e.g. "A" = &H41,  "A" - &H40 -> 1

   For i = Len(Ustr) To 1 Step -1
      Tstr = Mid$(Ustr, i, 1)
      If (Tstr >= "A") Then
         n = n + (9 + (Asc(Tstr) - &H40)) * (16 ^ (Len(Ustr) - i))
      Else
         n = n + Val(Tstr) * (16 ^ (Len(Ustr) - i))
      End If
   Next i

   HexStrtoDec = n

End Function

Function InputHex (KeyAscii As Integer) As Integer

   ' Check for 0 - 9, A-F, a-f.
   ' Convert a-f to A-F.
   If (KeyAscii >= &H30 And KeyAscii <= &H39) Or (KeyAscii >= &H41 And
   KeyAscii <= &H46) Or (KeyAscii >= &H61 And KeyAscii <= &H66) Then
      If (KeyAscii >= &H61) Then KeyAscii = KeyAscii - &H20
      InputHex = True
```

```
    Else
        InputHex = False
    End If

End Function

Sub Option1_Click ()

    Tic = 2

    If Text3.Text <> 0# Then
        Option1.Value = True
    End If

    Text2.Text = 0#

    CntrlAddr = CardBase + 3
    PortCAddr = CardBase + 2
    Cntrl = &H80
    ClearData = &H0

    If frmSMSetupTest.List1.Text = "Forward" Then
        OutData1 = &H2
        OutData2 = &H3
    End If

    If frmSMSetupTest.List1.Text = "Reverse" Then
        OutData1 = &H1
        OutData2 = &H3
    End If

    If InitCard = True Then
        DASErr = DASByteOut(CntrlAddr, Cntrl)
        DASErr = DASByteOut(PortCAddr, ClearData)
    End If

    PulseCount = DistanceToPulse(Val(Text3.Text))

    If PulseCount >= 8000 Then
        Tic = 3
```

```
    End If

    DASErr = DASByteOut(PortCAddr, ClearData)

    For Outer = 1 To PulseCount Step 1
        If Outer Mod 367 = 0 Then
            DoEvents
            Distance = Format(CSng(Outer) / 367, "#,##0.00;;;Nil")
            Text2.Text = Distance
        End If
        DASErr = DASByteOut(PortCAddr, OutData1)
        For Inner = 1 To Tic Step 1
            Dummy = Dummy + Dummy
        Next Inner
        DASErr = DASByteOut(PortCAddr, OutData2)
        For Inner = 1 To Tic Step 1
            Dummy = Dummy + Dummy
        Next Inner
    Next Outer

    Option1.Value = False
    Option2.Value = True

End Sub

Sub Option2_Click ()

    Option1.Value = False

End Sub

Sub Option3_Click ()

    Option1.Value = False
    Text2.Text = "0"
    Text3.Text = "0"

End Sub
```

Appendix B - 24 TRANCALI.FRM

```
'**********Transducer calibration form**********
'This form is written to display transducer calibration coefficients.

Option Explicit

    Dim Title As Variant
    Dim Msg As Variant
    Dim DgDef As Variant
    Dim Response As Integer

    Dim OldBox As Control
    Dim NewBox As Control
    Dim OldText As String

Sub Form_Load ()

    ' The form is horizontally and vertically centered when loaded.
    Top = Screen.Height / 2 - Height / 2
    Left = Screen.Width / 2 - Width / 2

    ' These are used in conjunction with GFocus to mimic
    ' standard windows text box changing methods while
    ' retaining certain controls on the text allowed.
    Set NewBox = Nothing
    Set OldBox = Frame1
    OldText = ""

    Frame1.Enabled = True
    Option1.Enabled = True
    Option2.Enabled = True
    Option3.Enabled = True

    ' Call reset to load the text boxes from the control file.
    Option1.Value = False
    Option3.Value = False
    Option2.Value = True

End Sub

Sub Form_QueryUnload (Cancel As Integer, Unloadmode As Integer)
```

```
' Put together a message box with all the proper components.
Title = "Warning"
Msg = "Do you really want to close the window?"
DgDef = MB_YESNO + MB_ICONSTOP + MB_DEFBUTTON2

Cancel = True
' Evaluate unload mode and give warning.
If Unloadmode = 0 Then
   Response = MsgBox(Msg, DgDef, Title)
   If Response = IDYES Then
      Cancel = False
      Unload frmTransducerCali
   Else
      Cancel = True
   End If
Else
      Cancel = False
End If

frmSetUPTest.Command1.Enabled = False
frmSetUPTest.Command2.Enabled = False
frmSetUPTest.Command3.Enabled = True
frmSetUPTest.Command4.Enabled = True
frmSetUPTest.Command5.Enabled = True
frmSetUPTest.List1.Enabled = True
```

End Sub

Sub Option1_Click ()

```
Dim TransFile As Integer
Dim Response As Integer
Dim Msg As String
Dim Title As String

Set NewBox = Frame1
Call GFocus(OldBox, NewBox, OldText)
Set OldBox = NewBox
OldText = ""
```

```
' Option 1 saves the gains & offsets to a control
' file, then unloads the form.

Msg = "Save these settings?"
Title = "Saving Transducer Calibration Record"
Response = MsgBox(Msg, 4 + 16 + 256, Title)
If Response = IDNO Then
   Option1.Value = False
   Exit Sub
End If

TransFile = FreeFile

Open "Transcal.ctl" For Output As TransFile

Print #TransFile,
Print #TransFile, "Pressure transducer calibration numbers control file."
Print #TransFile, "Transducer #1 gain :"; Text1.Text
Print #TransFile, "Transducer #2 gain :"; Text2.Text
Print #TransFile, "Transducer #3 gain :"; Text3.Text
Print #TransFile, "Transducer #4 gain :"; Text4.Text
Print #TransFile, "Transducer #5 gain :"; Text5.Text
Print #TransFile, "Transducer #6 gain :"; Text6.Text
Print #TransFile, "Transducer #7 gain :"; Text7.Text
Print #TransFile, "Transducer #8 gain :"; Text8.Text
Print #TransFile, "Transducer #9 gain :"; Text9.Text
Print #TransFile, "Transducer #10 gain :"; Text10.Text
Print #TransFile, "Transducer #11 gain :"; Text11.Text
Print #TransFile, "Transducer #12 gain :"; Text12.Text
Print #TransFile, "Transducer #13 gain :"; Text13.Text
Print #TransFile, "Transducer #14 gain :"; Text14.Text
Print #TransFile, "Transducer #15 gain :"; Text15.Text
Print #TransFile, "Transducer #16 gain :"; Text16.Text
Print #TransFile, "Transducer #1 offset :"; Text17.Text
Print #TransFile, "Transducer #2 offset :"; Text18.Text
Print #TransFile, "Transducer #3 offset :"; Text19.Text
Print #TransFile, "Transducer #4 offset :"; Text20.Text
Print #TransFile, "Transducer #5 offset :"; Text21.Text
Print #TransFile, "Transducer #6 offset :"; Text22.Text
```

```
    Print #TransFile, "Transducer #7 offset :"; Text23.Text
    Print #TransFile, "Transducer #8 offset :"; Text24.Text
    Print #TransFile, "Transducer #9 offset :"; Text25.Text
    Print #TransFile, "Transducer #10 offset :"; Text26.Text
    Print #TransFile, "Transducer #11 offset :"; Text27.Text
    Print #TransFile, "Transducer #12 offset :"; Text28.Text
    Print #TransFile, "Transducer #13 offset :"; Text29.Text
    Print #TransFile, "Transducer #14 offset :"; Text30.Text
    Print #TransFile, "Transducer #15 offset :"; Text31.Text
    Print #TransFile, "Transducer #16 offset :"; Text32.Text
    Print #TransFile,

    Close TransFile

    Unload frmTransducerCali

End Sub

Sub Option2_Click ()

    Dim TransFile As Integer
    Dim Char As Variant
    ReDim TextData(32) As Variant
    Dim I As Integer

    Set NewBox = Frame1
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = ""

    ' Option 2 loads the gains & offsets from the control
    ' file into the text boxes.

    TransFile = FreeFile

    Open "Transcal.ctl" For Input As TransFile

    For I = 0 To 31
      Do
        Char = Input(1, TransFile)
```

```
    Loop Until Char = Chr(58)
    TextData(I) = ""
    While Char <> Chr(13)
        Char = Input(1, TransFile)
        TextData(I) = TextData(I) & Char
    Wend
Next I

Close TransFile

Text1.Text = Left(TextData(0), Len(TextData(0)) - 1)
Text2.Text = Left(TextData(1), Len(TextData(1)) - 1)
Text3.Text = Left(TextData(2), Len(TextData(2)) - 1)
Text4.Text = Left(TextData(3), Len(TextData(3)) - 1)
Text5.Text = Left(TextData(4), Len(TextData(4)) - 1)
Text6.Text = Left(TextData(5), Len(TextData(5)) - 1)
Text7.Text = Left(TextData(6), Len(TextData(6)) - 1)
Text8.Text = Left(TextData(7), Len(TextData(7)) - 1)
Text9.Text = Left(TextData(8), Len(TextData(8)) - 1)
Text10.Text = Left(TextData(9), Len(TextData(9)) - 1)
Text11.Text = Left(TextData(10), Len(TextData(10)) - 1)
Text12.Text = Left(TextData(11), Len(TextData(11)) - 1)
Text13.Text = Left(TextData(12), Len(TextData(12)) - 1)
Text14.Text = Left(TextData(13), Len(TextData(13)) - 1)
Text15.Text = Left(TextData(14), Len(TextData(14)) - 1)
Text16.Text = Left(TextData(15), Len(TextData(15)) - 1)
Text17.Text = Left(TextData(16), Len(TextData(16)) - 1)
Text18.Text = Left(TextData(17), Len(TextData(17)) - 1)
Text19.Text = Left(TextData(18), Len(TextData(18)) - 1)
Text20.Text = Left(TextData(19), Len(TextData(19)) - 1)
Text21.Text = Left(TextData(20), Len(TextData(20)) - 1)
Text22.Text = Left(TextData(21), Len(TextData(21)) - 1)
Text23.Text = Left(TextData(22), Len(TextData(22)) - 1)
Text24.Text = Left(TextData(23), Len(TextData(23)) - 1)
Text25.Text = Left(TextData(24), Len(TextData(24)) - 1)
Text26.Text = Left(TextData(25), Len(TextData(25)) - 1)
Text27.Text = Left(TextData(26), Len(TextData(26)) - 1)
Text28.Text = Left(TextData(27), Len(TextData(27)) - 1)
Text29.Text = Left(TextData(28), Len(TextData(28)) - 1)
Text30.Text = Left(TextData(29), Len(TextData(29)) - 1)
```

```
    Text31.Text = Left(TextData(30), Len(TextData(30)) - 1)
    Text32.Text = Left(TextData(31), Len(TextData(31)) - 1)

    Option2.Value = False

End Sub

Sub Option3_Click ()

    Set NewBox = Frame1
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = ""

    ' Option 3 resets the gains & offsets

    Option2.Value = True

    Unload frmTransducerCali

End Sub

Sub Text1_GotFocus ()

    Set NewBox = Text1
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub Text10_GotFocus ()

    Set NewBox = Text10
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub
```

```
Sub Text11_GotFocus ()

    Set NewBox = Text11
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub Text12_GotFocus ()

    Set NewBox = Text12
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub Text13_GotFocus ()

    Set NewBox = Text13
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub Text14_GotFocus ()

    Set NewBox = Text14
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub Text15_GotFocus ()

    Set NewBox = Text15
    Call GFocus(OldBox, NewBox, OldText)
```

```
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub Text16_GotFocus ()

    Set NewBox = Text16
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub Text17_GotFocus ()

    Set NewBox = Text17
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub Text18_GotFocus ()

    Set NewBox = Text18
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub Text19_GotFocus ()

    Set NewBox = Text19
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub
```

```
Sub Text2_GotFocus ()

    Set NewBox = Text2
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub Text20_GotFocus ()

    Set NewBox = Text20
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub Text21_GotFocus ()

    Set NewBox = Text21
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub Text22_GotFocus ()

    Set NewBox = Text22
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub Text23_GotFocus ()

    Set NewBox = Text23
```

```
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub Text24_GotFocus ()

    Set NewBox = Text24
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub Text25_GotFocus ()

    Set NewBox = Text25
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub Text26_GotFocus ()

    Set NewBox = Text26
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub Text27_GotFocus ()

    Set NewBox = Text27
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text
```

```
End Sub

Sub Text28_GotFocus ()

    Set NewBox = Text28
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub Text29_GotFocus ()

    Set NewBox = Text29
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub Text3_GotFocus ()

    Set NewBox = Text3
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub Text30_GotFocus ()

    Set NewBox = Text30
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub Text31_GotFocus ()
```

```
    Set NewBox = Text31
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub Text32_GotFocus ()

    Set NewBox = Text32
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub Text4_GotFocus ()

    Set NewBox = Text4
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub Text5_GotFocus ()

    Set NewBox = Text5
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub Text6_GotFocus ()

    Set NewBox = Text6
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text
```

```
End Sub

Sub Text7_GotFocus ()

    Set NewBox = Text7
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub Text8_GotFocus ()

    Set NewBox = Text8
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub

Sub Text9_GotFocus ()

    Set NewBox = Text9
    Call GFocus(OldBox, NewBox, OldText)
    Set OldBox = NewBox
    OldText = OldBox.Text

End Sub
```

Appendix B - 25 TRANSDUC.FRM

```
'**********Transducer control form**********
'This form is written to test interfacing and control data collection for pressure
'transducers.

Option Explicit

    Dim Opened As Integer

    Dim StartAdd As Integer
    Dim MaxItem As Integer
    Dim Items(16) As Variant
    Dim CommandRegister As Integer
    Dim DataRegister As Integer
    Dim Order1 As Integer

    Dim InChan As Single
    Dim Offset As Integer
    Dim Segment As Integer

    Dim I As Integer
    Dim J As Integer
    Dim K As Integer

    Dim Page0 As Integer
    Dim Page1 As Integer
    Dim Page2 As Integer
    Dim Page3 As Integer

    Dim TmpOff As Integer

    Dim Title As String
    Dim Msg As String
    Dim DgDef As Integer
    Dim Response As Integer

    Dim Duration As Integer
```

Sub Command1_Click ()

```
InChan = 0
Segment = VarSeg%(InChan)
Offset = VarPtr%(InChan)
OUT Page3, CommandRegister
OUT DataRegister, Order1
OUT Page3, CommandRegister
While Inp%(StartAdd + 1)
Wend
OUT Page2, 0
For I = 0 To MaxItem Step 1
  For J = 0 To 3 Step 1
    TmpOff = J
    MhPokeByte Inp%(DataRegister), Segment, Offset + TmpOff
  Next J
  Items(I) = InChan
Next I

Opened = True
For K = 0 To MaxItem Step 1
  Items(K) = Format(Items(K), "00.0000")
Next K

Label2.Caption = Items(1)
Label3.Caption = Items(2)
Label4.Caption = Items(3)
Label5.Caption = Items(4)
Label6.Caption = Items(5)
Label7.Caption = Items(6)
Label8.Caption = Items(7)
Label9.Caption = Items(8)
Label10.Caption = Items(9)
Label11.Caption = Items(10)
Label12.Caption = Items(11)
Label13.Caption = Items(12)
Label14.Caption = Items(13)
Label15.Caption = Items(14)
Label16.Caption = Items(15)
Label17.Caption = Items(16)

End Sub
```

```
Sub Command2_Click ()

    Label2.Caption = ""
    Label3.Caption = ""
    Label4.Caption = ""
    Label5.Caption = ""
    Label6.Caption = ""
    Label7.Caption = ""
    Label8.Caption = ""
    Label9.Caption = ""
    Label10.Caption = ""
    Label11.Caption = ""
    Label12.Caption = ""
    Label13.Caption = ""
    Label14.Caption = ""
    Label15.Caption = ""
    Label16.Caption = ""
    Label17.Caption = ""

End Sub

Sub Command3_Click ()

            rmTransducerTest

            Test.Command2.Enabled = False
             Test.Command3.Enabled = True
             est.Command4.Enabled = True
              st.Command5.Enabled = True



Sub Form_Load ()

    ' The form is horizontally and vertically centered when loaded.
    Top = Screen.Height / 2 - Height / 2
    Left = Screen.Width / 2 - Width / 2

    Opened = False
```

```
    StartAdd = Val(frmTransducerTest.Text19.Text)

    MaxItem = Val(frmTransducerTest.Text18.Text)

    ' PC I/O address.
    Page0 = StartAdd + 4
    Page1 = StartAdd + 5
    Page2 = StartAdd + 6
    Page3 = StartAdd + 7
    CommandRegister = 254
    DataRegister = StartAdd
    Order1 = 222
```

End Sub

Sub Form_QueryUnload (Cancel As Integer, Unloadmode As Integer)

```
    ' Close card if not closed already.
    If Opened = True Then
       OUT Page3, CommandRegister
       OUT DataRegister, 250
       OUT Page3, CommandRegister
       While Inp(DataRegister)
       Wend
       Opened = False
    End If

    ' Put together a message box with all the proper components.
    Title = "Warning"
    Msg = "Do you really want to close the window?"
    DgDef = MB_YESNO + MB_ICONSTOP + MB_DEFBUTTON2

    Cancel = True

    ' Evaluate unload mode and give warning.
    If Unloadmode = 0 Then
       Response = MsgBox(Msg, DgDef, Title)
       If Response = IDYES Then
          Cancel = False
```

```
        Unload frmTransducerTest
    Else
        Cancel = True
    End If
Else
        Cancel = False
End If

frmSetupTest.Command2.Enabled = False
frmSetupTest.Command3.Enabled = True
frmSetupTest.Command4.Enabled = True
frmSetupTest.Command5.Enabled = True

End Sub
```