Lehigh University
Lehigh Preserve

Theses and Dissertations

1985

An automatic data acquisition system for microelectronic test structure evaluation /

Phillip Mark Goldman
Lehigh University

Follow this and additional works at: https://preserve.lehigh.edu/etd

Part of the Electrical and Computer Engineering Commons

Recommended Citation

Goldman, Phillip Mark, "An automatic data acquisition system for microelectronic test structure evaluation /" (1985). *Theses and Dissertations*. 4587.
https://preserve.lehigh.edu/etd/4587

This Thesis is brought to you for free and open access by Lehigh Preserve. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of Lehigh Preserve. For more information, please contact preserve@lehigh.edu.

AN AUTOMATIC DATA ACQUISITION SYSTEM FOR MICROELECTRONIC TEST STRUCTURE EVALUATION

by

Phillip Mark Goldman

A Thesis
Presented to the Graduate Committee
of Lehigh University
in Candidacy for the Degree of
Master of Science
in
Electrical Engineering
December 9, 1985

Certificate of Approval

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

December 10, 1985
(date)

Professor in Charge

Chairman of Department

Acknowledgments

I would like to thank Dr. Marvin White for his support, both academic and financial, without whom this work would not have been possible. Thanks are also extended to the following people: Dr. Robert Vogel and Floyd Miller for the knowledge of semiconductor processing; Ebrahim Khalily of Hewlett-Packard Company for enduring the endless questions about the original TECAP program; Tom Krutsick for supplying the completed wafers measured here; Rich Booth for consulting about general programming; and all of the graduate students of Sherman Fairchild Labs for their frequent yet constructive criticisms.

Table of Contents

1. Introduction **2**

2. Common Test System Strategies **4**

  2.1 Why Measure At All? 4
  2.2 Manual versus Automatic Measurement 5
  2.3 Automatic Measurement System Strategies 6
  2.4 Typical Structure of a Programmable Measurement System 8
  2.5 Software Strategies 10

3. Sherman Fairchild Laboratory Measurement Systems **13**

  3.1 System Diagrams and Component Description 14
  3.2 TECAP2 Program Capabilities 17
  3.3 Prober Control Modification to TECAP2 21
  3.4 Mapping Extensions to TECAP2 22
    3.4.1 Map Data Internal Structure 23
    3.4.2 Command Structure of the Map Extension 24
    3.4.3 Map Reporting Formats 26
  3.5 Test Pattern Design 27

4. Detailed Operation of the TECAP Extensions **29**

5. Sample Measurements from TECAP2 Mapping **34**

  5.1 Models and Extraction Methods used in TECAP2 34
  5.2 Substrate Doping NSUB 36
  5.3 Threshold Voltage VTO 37
  5.4 Surface mobility U0 37
  5.5 Channel narrowing due to lateral diffusion, the WD parameter 38

6. Capacitance-Voltage Measurement System Development **40**

7. Conclusions and Recommendations **42**

References **44**

Appendix A. Measured Wafer Data **46**

Appendix B. C-V Station Program Ver 1.2 User's Manual **68**

  B.1 Introduction To C-V Version 1.2 68
  B.2 Quickee User's Guide 70
  B.3 Detailed Operation Guide 71
  B.4 Sample Outputs 76
    B.4.1 Plot and Results 76
    B.4.2 Printout of Actual Data 77
  B.5 Equations Used in C-V 78
  B.6 Program Listings 80

Appendix C. TECAP2 Prober Control Update Manual - RK681 - for TECAP 10.00 **84**

  C.1 Routines Needed To Add An Unsupported Prober To TECAP 85
  C.2 Prober Driver Routines for the RK681 Prober 87
  C.3 File for Linking Prober to System 90

Appendix D. User Module Code **92**
Vita **117**

List of Figures

Figure 2-1: Typical automated test system configurations 8
Figure 2-2: Components of a Comprehensive Software System 11
Figure 3-1: Block Diagram of System Hardware 15
Figure 3-2: Sherman Fairchild Laboratory Measurement System 16
Figure 3-3: TECAP2 Internal Data Structure and Data Paths 19
Figure 3-4: NMOS Test Pattern for Mapping 28
Figure A-1: Wafer B7 Substrate Doping Distribution (NSUB) 48
Figure A-2: Wafer B7 Substrate Doping Map (NSUB) 49
Figure A-3: Wafer B2 Substrate Doping Distribution (NSUB) 51
Figure A-4: Wafer B2 Substrate Doping Map (NSUB) 52
Figure A-5: Wafer B7 Threshold Voltage Distribution (VTO) 54
Figure A-6: Wafer B7 Threshold Voltage Map (VTO) 55
Figure A-7: Wafer B2 Threshold Voltage Distribution (VTO) 57
Figure A-8: Wafer B2 Threshold Voltage Map (VTO) 58
Figure A-9: Wafer B7 Bulk Mobility Distribution (U0) 60
Figure A-10: Wafer B7 Bulk Mobility Map (U0) 61
Figure A-11: Wafer B2 Bulk Mobility Distribution (U0) 63
Figure A-12: Wafer B2 Bulk Mobility Map (U0) 64
Figure A-13: Wafer B7 Channel Narrowing Distribution (WD) 66
Figure A-14: Wafer B7 Channel Narrowing Map (WD) 67

List of Tables

Table 3-1: Resolutions of Major System Components [7] 18
Table A-1: Wafer B7 Substrate Doping Data and Statistics (NSUB) 47
Table A-2: Wafer B2 Substrate Doping Data and Statistics (NSUB) 50
Table A-3: Wafer B7 Threshold Voltage Data and Statistics (VTO) 53
Table A-4: Wafer B2 Threshold Voltage Data and Statistics (VTO) 56
Table A-5: Wafer B7 Bulk Mobility Data and Statistics (U0) 59
Table A-6: Wafer B2 Bulk Mobility Data and Statistics (U0) 62
Table A-7: Wafer B7 Channel Narrowing Data and Statistics (WD) 65

Abstract

A microcomputer-based full wafer characterization system has been implemented based on TECAP2, an interactive CAD/CAE software package from Hewlett-Packard Company. The system consists of a HP9836 Computer, RK681 Computer Controlled Wafer Prober, and a HP4145 Semiconductor Parameter Analyzer. This system enables the user in a research environment to fully measure and evaluate the DC simulation parameters on finished wafers from a semiconductor line. The characterizations are performed over the entire surface of a finished wafer, with the data presented as a distribution plot (wafer map) or statistical plot (histogram) at the user's request.

In addition, software has been written to drive an automatic capacitance-voltage measurement system in the processing lab itself. This software is also interactive in nature, and provides full C-V and bias-temperature stress measurement capabilities. It is based on an MSI Data Station and an HP85 Desktop computer. A manual for its operation is provided.

This thesis presents the operational procedures for the wafer mapping system, extensions to the TECAP2 manual for such operation, and sample measurements demonstrating the usefulness of such a program.

Chapter 1
Introduction

With the increasing complexity and variability of semiconductor manufacturing there must be convenient and flexible systems for device and process characterization. DC parametric testing is one very common, if time consuming way to insure the quality and consistency of a fabrication facility's output.

Nowhere is this more critical than in a small research facility, where equipment budgets are low and processing and design are subject to rapid variation. A small laboratory must have a system sufficiently flexible to allow many different types of measurement to take place, but not so flexible as to be overly complex and therefore too expensive. For a laboratory with a changing staff, like a university lab, the software must also be as self-explanatory as possible to allow each new staff member to be able to use the system with a minimum of training time.

Historically systems such as this have been built up as "modular" systems made up of many different components. These are then interconnected together and controlled through a small computer to take the data from a device and store it. The data is then transferred to a more powerful computer for data reduction and report generation. Because of the many translations of data this method is prone to error and data loss. If an error occurs during measurement it may not be found until the data is transferred and analyzed, which is often enough time to insure the conditions of the experiment cannot be recreated. Often, both machines must translate data into a communications format which is awkward for the systems to handle , again slowing the process of analyzing the data.

With the growth in power of microcomputers or desktop minicomputers comes a new way to implement these systems. Now it is practical, from both a time and a budget point of view, to use the same computer which controls the instrumentation to store and analyze the data. This would eliminate the need for the slow data transfers, and allows the results of a test to be known in time to correct errors in the procedure. Also, by keeping the data in a constant convenient format, the task of comparing or analyzing data from different measurements is simplified. The measurement environment becomes more interactive, and the system user becomes more involved in the workings of the system making it much more adaptable to device conditions.

This thesis describes the selection and extension of such a system for transistor characterization. Fully menu driven, modular in construction, and capable of full measurement and parameter extraction, this system meets all of the standards mentioned above. It allows tracking of such critical parameters as threshold voltage, leakage current, or substrate doping. It produces reports on these data in various formats, from simple list data to wafer mapping.

First, some background information about such characterization systems is given. Next, a detailed explanation of the new developments for this thesis are give. Finally, some examples of the measurements performed by this system are given. The appendixes contain the detailed information not included in the main body of the thesis. This document should provide adequate background for the user familiar with device characteristics to use the programs and their extensions.

Chapter 2
Common Test System Strategies

Recently the specification of test systems for tracking the variations in process parameters has been a common topic in many of the professional journals. Most often it is concerned with tracking variations in the actual processing which will affect the yield of a commercial product line. This document concentrates developing a system suitable for a research environment where these changes are induced in the process to determine their effect on the final devices, or where the precise values of theses parameters is needed for comparison with other research results. Much of what has been written is directly applicable to this task; these design ideas will be discussed in this chapter.

2.1 Why Measure At All?

If one makes a product, it is only logical that one would like to know how well he (or she) is making that product. Measurement, or parametric testing, is one way to assure the quality of the product. DC parametric testing is used at various points in the process sequence to determine if the intermediate steps have proceeded correctly; measurements made after the process is complete screen out the defective chips from a lot before the costly process of packaging is begun. These results are the used to improve the yield in subsequent process runs.

Most commonly parametric tests are performed on specially designed test dice placed at strategic places on the actual product wafer. This pattern may include transistors, resistors, capacitors, or specialized process monitoring structures. These few chips placed within a run of product chips indicate the relative quality of the chips on the wafer. Variations from lot to lot are noted, and adjustments are made to keep yield as high as possible. [6]

That scheme is mainly to characterize yield in a commercial line. In a research environment the final product is information; what variations in processing do to the wafers in the end product. For this type of situation dedicated test wafers are used. These are wafers fully covered with test patterns to evaluate the desired parameters. Because there are many of these (typically more than 30) even on a small wafer, statistically significant data may be taken on the processing variations within a wafer instead of only lot by lot. [11]

2.2 Manual versus Automatic Measurement

The question of manual or automatic measurement is a central question to this topic. Obviously since this paper is about automatic full wafer characterization there is a preference for automatic measurement here, but why? To understand this an example of each is given below.

A manual measurement system generally consists of a collection of sources, meters, and generally a curve tracer, strip chart recorder, or X-Y plotter. Each system generally consists of only the equipment needed to perform a given task; when the task is completed the equipment is used in another task. Each experiment is constructed individually and often requires as much set-up time as actual measurement time. Taking data requires the attention of the tester throughout, since the tester (the human tester) must record each data point by hand, or make adjustments to the system as the device under test warrants. Since this is a manpower-intensive, repetitive system, it is also prone to human error. The amount of time involved often precludes the accumulation of large quantities of data. Finally, data taken in this manner is awkward to store, retrieve, and analyze.

By comparison, an automatic measurement system, regardless of approach, consists of a controller (commonly a desktop computer), sources and measurement units under the control of this unit, some programmable way of making the device connections to allow for flexibility, and some way of storing the data. Often too, this system can perform some fairly complex data reduction without the aid of a larger system. The system is generally able to handle a great many measurement tasks due to the flexibility gained by using a computer, thus cutting down drastically on setup time. Many data points may be taken and stored conveniently (and exactly) without the tedium of involving the human user in point-by-point recording. Several different approaches are used to implement an automatic test system, whether it is for functional (go/no go) testing or detailed parametric analysis; each approach maintains its advantages over manual testing.

2.3 Automatic Measurement System Strategies

Three common strategies exist in developing automated testing. Since this thesis is more concerned with parametric testing, however, only those approaches which pertain directly to parametric testing will be considered. These can be summarize as 1) use of functional testers to evaluate device parameters, 2) design and construction of custom systems meant for a specific measurement task, and 3) a general system consisting of "smart instruments" connected to a controller to allow for more general purpose activity.

Often an integrated circuit manufacturer wishes to perform parametric testing on the output of his plant in order to track the quality and consistency of his product. Since very often he has functional testers already in-house for quality assurance purposes he wishes to make this very expensive equipment do extra duty. By reprogramming these systems it is sometimes possible to allow this, but often the functional tester lacks the necessary sensitivity to perform the parametric analysis. It is designed to drive large voltages and currents (normal operation conditions for the products under test). In addition, revision time on software for such a system could easily rival the demands of the manual approach.

Design and construction of a custom system will overcome the sensitivity problem, but will often require many man-hours to implement, still with no guarantee that the system will perform the desired task. In addition, the equipment used in such a system is then dedicated to one task for its useful lifetime. Finally, duplication of a working system for a similar operation may be as time-consuming as the original design, especially if the system contains much custom hardware. This approach may be appropriate if the system will be used in a dedicated fashion, such as a capacitance-voltage measurement station used to track the quality of oxides in a production facility. This approach fails to meet the needs of a small research environment though, where flexibility is a prerequisite of a useful measurement system.

The most viable solution for the research lab is using some of the "smart instruments" available for purchase in the instrumentation catalogs. These pieces of equipment often provide some data storage or analysis capability in addition to the flexible programming of its operation. When combined with a small computer as a controller/coordinator, a smart system becomes an extremely useful, often indispensable tool for parametric analysis.

2.4 Typical Structure of a Programmable Measurement System

Shown below are two examples of typical programmable parametric test systems. On the left is a diagram of the system as a whole, including the common software modules used. [10] On the right is a block diagram of the hardware and interconnections involved. [6] The typical elements of a measurement system consist of:

1. Sources: Voltage, Current
2. Measurement Instruments: Voltage, Current, Capacitance
3. Switching Matrix
4. Probing Equipment to allow contact with the DUT
5. Controller (commonly a desktop computer)
6. Disk or Tape Storage
7. Comprehensive software for measurement, storage, and data reduction.

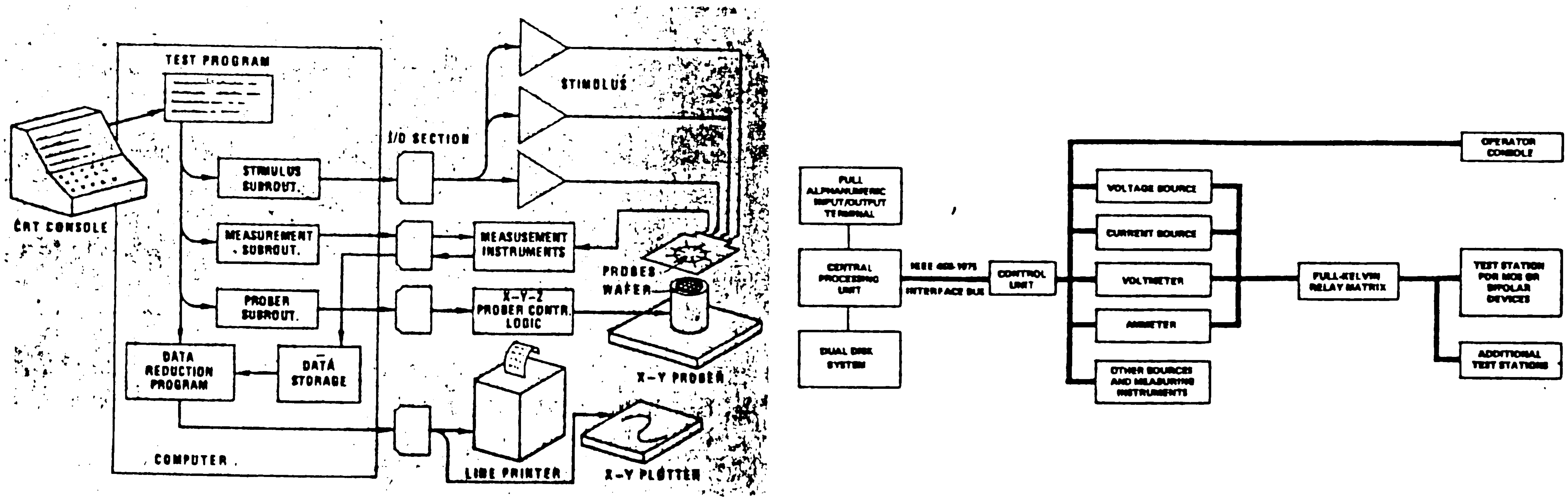


Figure 2-1: Typical automated test system configurations

The sources are used to force the known conditions on the device under test to perform the desired analysis. For DC parametric analysis the current supplies should have a precision typically on the order of 100 picoamperes minimum (or smaller) with a full scale current capability of 100 milliamperes or more. All six decades of range should be selectable by software, to a precision of about 0.1% of full scale. Similarly, the voltage supplies should have a resolution of 1 to 10 millivolts, with a supply range swing of 50 to 300 volts.

The measurement instruments should be able to handle the full range of the forcing sources with a precision of better than 0.05% of full scale. In an intelligent instrument system these devices should also be auto-scaling and self-calibrating.

A switching matrix is used to make the connections from the sources and measurement units to the device under test. This generally contains a series of low noise relays under program control. Low noise refers to the low thermoelectric noise and current offsets, which may limit the sensitivity of the measurement system. This matrix (ideally) should also make some provisions for avoiding "hot switching" (moving the contacts under operational conditions) to preserve the life of the relays under high current or voltage conditions.

The heart of the measurement system is the central processing unit. It is typically a desktop computer with the ability to communicate easily with all of the equipment in the measurement system. It should have the capability of a higher level language (BASIC or FORTRAN are most common) to allow for the flexible programming of new measurement schemes. It should also possess adequate data storage media to contain the necessary programs and data for detailed measurement.

The probing equipment is necessary when analyzing devices which have not yet been packaged. A probing station often has the capability of moving the sample under test in precise steps, controlling the temperature of the sample, and, in some cases, load and unload the sample automatically. The probing system also must provide some method for aligning the wafer along the axes of movement of the stage, most commonly a microscope or television camera and monitor.

Finally, the system software is where all the pieces come together. This software allows for the flexible configuration of all of the devices in the measurement system, the data acquisition (sometimes on a real-time basis) and the storage of data. Sophisticated software may even reduce the data on the controlling computer without having to transmit that data to a larger system. This software is considered in more detail.

2.5 Software Strategies

A comprehensive software system is needed to control and coordinate all of the operations of a measurement system. It must control the system equipment, transfer data to and from the peripherals, disks, and user data base, and allow for the creation of test programs. A diagram of the system software is given below.

Several components of the software bear elaboration. Under the real-time operating system exist the programs that may be used for system resource management and language support. Disk operations, file management, report generation, and measurement system configuration are carried out by the operating system. The programming languages which control the measurement system may also be expected to make heavy use of the operating system's facilities.

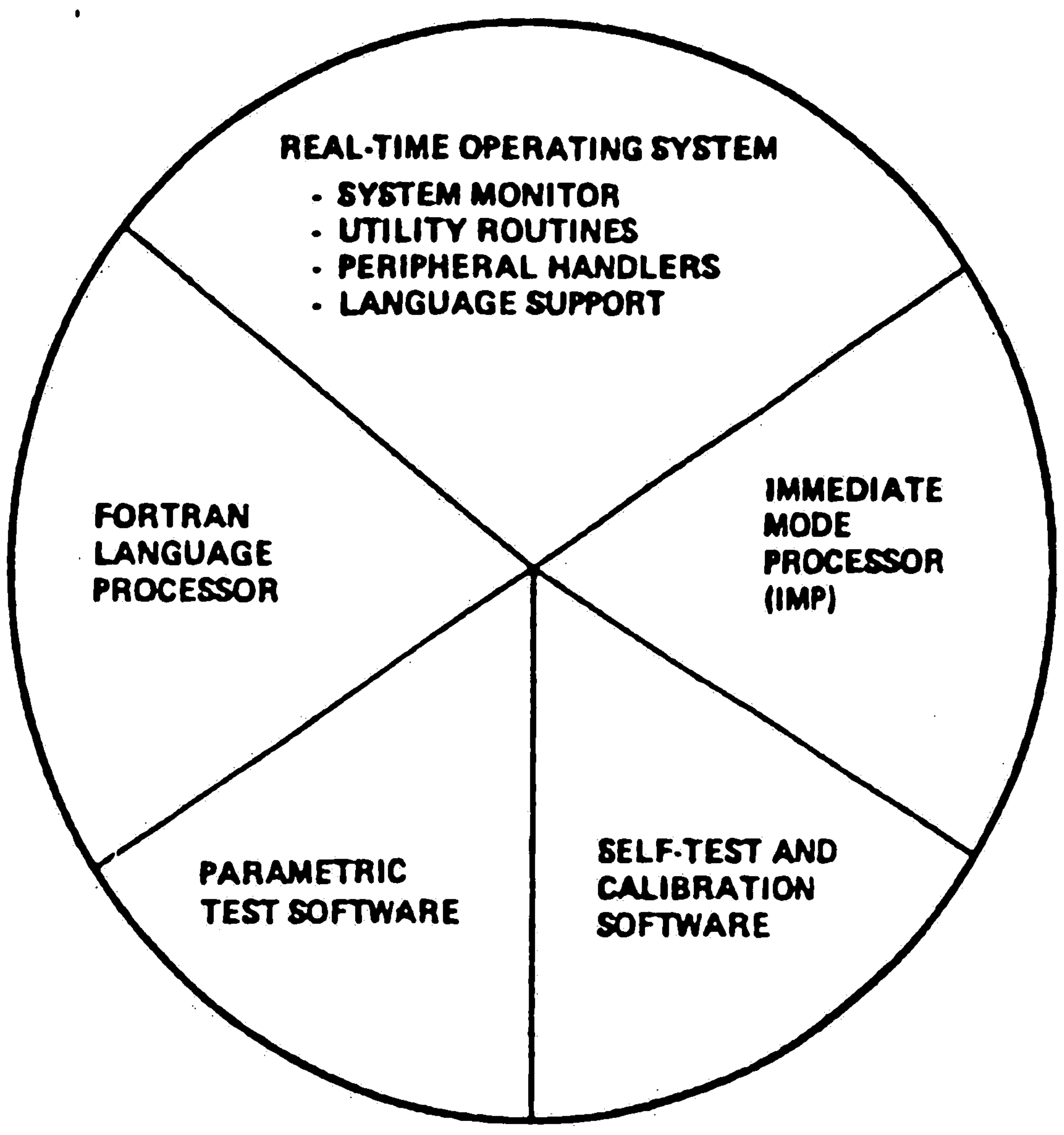


Figure 2-2: Components of a Comprehensive Software System

Listed in the figure is a FORTRAN language processor. This is not the only type of language environment used, but it is typical. The high level language supplied with the controller is augmented with new commands to handle the measurement devices, or a library of pre-tested routines are made available. Another alternative is to write an new measurement language, but this is a very manpower-intensive operation, and is generally not justified by the complexity of the measurements to be made.

Most of the time involved in writing the parametric test software is therefore involved in writing the subroutines to perform the basic measurement system functions. These functions can be classified into four main categories: CONNECTIONS, which instruct the relay matrix to make or break certain connections within the measurement system; FORCING, which sets the source units to the values needed to make a give measurement; MEASUREMENT, which instruct the instruments to take data and return it to the controller; and DELAY, which allows for time dependent operations to take place, like settling time or long-term applied stresses. These routines are generally standard FORTRAN subroutines which may be called directly from small FORTRAN code, which in turn may also contain analysis routines. All that is required of a test then is a knowledge of the tests to be made, and some basic high-level language programming.

Chapter 3
Sherman Fairchild Laboratory Measurement Systems

Now that several possible design ideas for a measurement system have been considered, the topic of this work, the structure of the existing measurement system may be discussed. Each topic will be taken in turn, resolving the differences between our measurement system and those ideals discussed previously.

Obviously, from the title of this document, the system is an automatic measurement arrangement. What should be kept in mind is that the system is primarily used by graduate students in aiding them in doing original research. These graduate students require the system to be immediately useable, self-explanatory, and consistent.

TECAP2 [16], a program from Hewlett-Packard Design Aids Division, was found to possess many of the qualities so useful in a research environment. This program was supplied to us in compiled form from Hewlett-Packard, and, along with the capability to add user control procedures and models, allowed the project to be completed. This chapter will focus on the structure of the measurement system, its operations, and extensions for use in Sherman Fairchild Laboratory.

3.1 System Diagrams and Component Description

Figure 3.1 shows the layout of the Sherman Fairchild Laboratory Measurement system. Its components are:

1. **Sources and Measurement Units:** A HP4145A Semiconductor Parameter Analyzer provides the source and measurement unit capability. The unit is essentially a "smart curve tracer" consisting of four Stimulus-Measurement Units (SMU's). Each SMU is capable of supplying a constant voltage or current and monitoring the current or voltage flowing through them. The resolution of the SMU's is given in Table 3-1 . The HP4145 has a test fixture associated with it which can handle packaged devices in a variety of housings, along with managing the connection of the SMU's to the pins. The SMU's are assignable through program control at the unit's front panel, or, as in this system, under computer control.

2. **Switching Systems**: Currently only the software assignments of the four SMU's is permitted. This is discussed more fully under Future Recommendations.

3. **Probing Equipment**: The probing system consists of a Rucker and Kolls Model RK681A Computer Controlled Prober. The system is controlled by the computer over the interface bus exclusively; parameters are again set up in the main TECAP2 program. It is capable of stepping in the X and Y directions as well as raising and lowering the probes to the wafer surface. Resolution of movement is

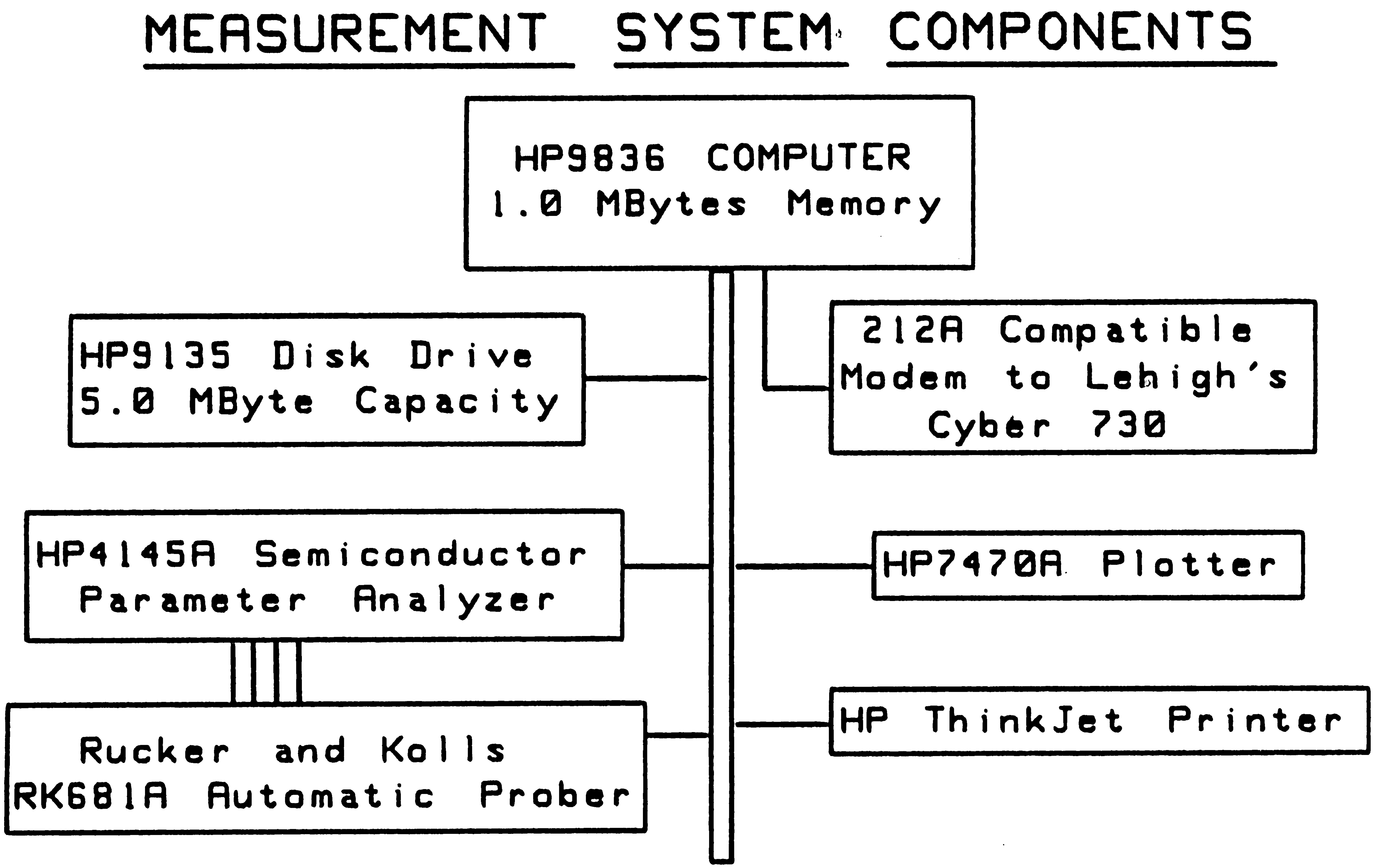


Figure 3-1: Block Diagram of System Hardware

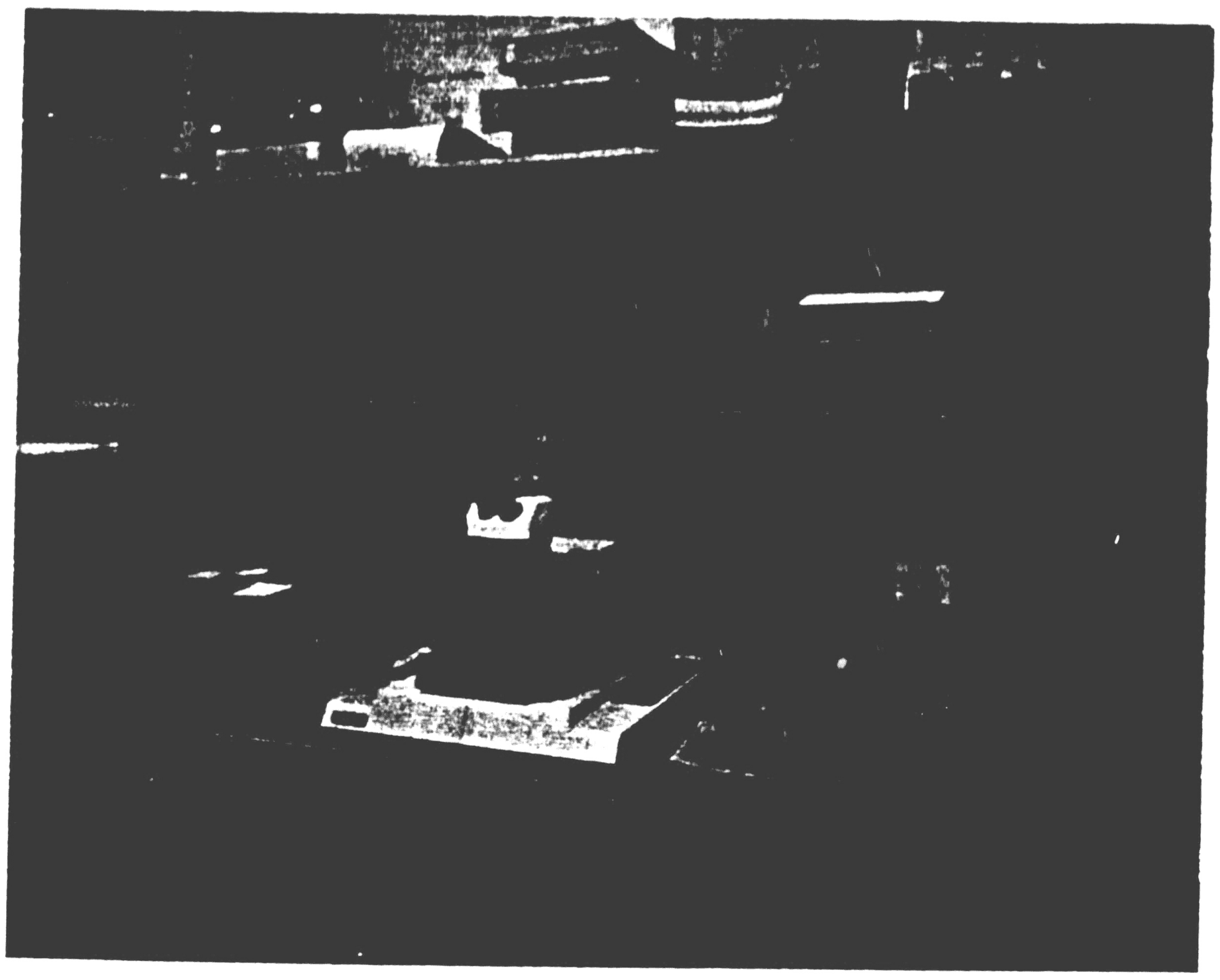


Figure 3-2: Sherman Fairchild Laboratory Measurement System

.001 mm. in both the x and y directions. New control software was written to handle this instrument.

4. **Controller**: The system controller is a Hewlett-Packard HP9836 Desktop Computer running the HP-Pascal Operating System Version 2.0. The system contains 1.0 megabytes of main system memory, two floppy disk drives, and a black-and-white combined text/graphics display CRT. The system controls the external instruments by way of the HPIB (Hewlett - Packard Interface Bus), a version of the IEEE-488-1975 bus specification.

5. **Disk Storage**: A HP9136 Winchester Hard Disk Drive stores the program and configuration files, Pascal utilites, and help files for ease

of access and speed. The drive has a 5 megabyte Winchester fixed-media disk and a 5.25 inch floppy disk for backup purposes.

6. **Software**: The main body of software is supplied as TECAP2, which will be discussed in detail.

In addition to the basic measurement system, a modem and RS232C capability has been added to communicate with the Cyber 730 computer belonging to the Lehigh University Computing Center. First, TECAP2 is used to extract the SPICE program parameters. Through a virtual terminal program data or parameters may be transfered to the mainframe for circuit simulation. In this way we may verify the parameter extraction or try to predict the reaction of devices under test and their operation in finished circuit.

3.2 TECAP2 Program Capabilities

TECAP2 is a product of Hewlett-Packard's Design Aids/ Engineering Productivity Division in Palo Alto, California [16]. It is designed for designers and process engineers to measure semiconductor test structures and extract device model parameters for circuit simulators. The program has many capabilities which are convenient for the researcher in a small lab to use. Modularity of design allows for ease of user modification and understanding. The Internal Data Base is used by all functions of TECAP. This section will discuss the original internal structure of TECAP and its capabilities.

TECAP2 provides the capability of 1) Precise and flexible measurement of DC characteristics of transistors, 2) Extraction of model parameters, directly finding the HPSPICE model parameters, and using an optimizing simulation for parameters of user-specified models, and 3) Simulating the device performance

Table 3-1: Resolutions of Major System Components [7]

Current Range	Resolution	Accuracy	Max. Voltage
±100mA	100µA	0.3%+(0.1+0.2*Vo/100)%	20V (>50mA) 40V (>20mA)
±10mA	10µA		100V (≤20mA)
±1000µA	1µA		
±100µA	100nA		
±10µA	10nA		
±1000nA	1nA	0.5%+(0.1+0.2*Vo/100)%	
±100nA	100pA		
±10nA	10pA	1%+(0.1+0.2*Vo/100)%+5pA	
±1000pA	1pA		

Voltage Range	Resolution	Accuracy	Max. Current
±20V	1mV	0.1%+0.05%+0.4Ω*Io	100mA
±40V	2mV		50mA
±100V	5mV		20mA

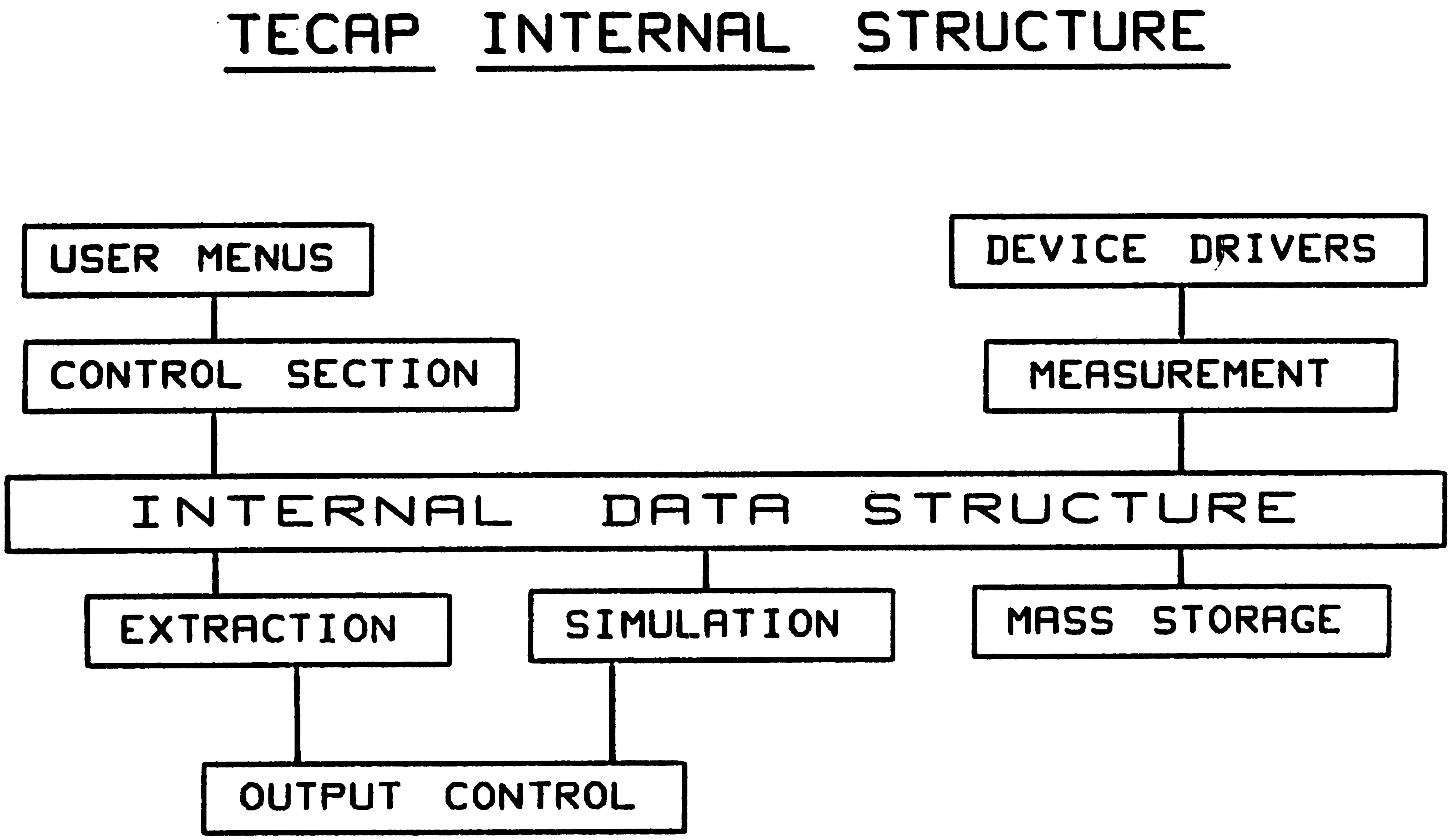


Figure 3-3: TECAP2 Internal Data Structure and Data Paths

using the models built in. TECAP2 supports the HPSPICE models for MOS, BJT, and DIODE equations. The extracted parameters can be printed in the form of a SPICE .MODEL card for simulation on a mainframe computer.

The modular internal structure of the TECAP2 program allows for the flexibility and ease of programming which TECAP exhibits. This data structure is shown in Figure 3-3 . The internal data structure contains the setup, measurement, simulation, and filing information being used by each phase of the TECAP program.

On the upper left of the structure is the user interface section of the program. All measurement, filing, data management, and simulation commands are issued through this section. Each selection is menu driven or graphically displayed on the screen. The various routines check the internal data base and store the user's requests in the data base. Parameters such as voltage sweep settings, current parameter values for a given model, which devices are connected to the bus, and where the current data is coming from are stored there. When the commands to perform a certain operation are given, the data is taken from the data structure instead of repeatedly polling the user. Any data in this structure may be stored to disk for later usage.

The User Interface also provides a fundamental programming capability. Commands given at the keyboard may be a string of command characters which can be stored to disk for later use. There is a single branch condition and a basic looping structure for repeated commands to be used. This feature was used for the parameter mapping extensions made at Fairchild Labs.

On the upper right of Figure 3-3 are the subroutines which drive the individual measurement devices. They are analogous to the specialized subroutines presented as part of the FORTRAN code in Chapter 2. Each device has its own set of device drivers, which allow the user to set parameters and retrieve data from each device without having to know the command structure for the individual device. The measurement subroutines extract the necessary data from the internal data base to accomplish the measurements and stores the measured data back in the data structure.

On the lower left of the diagram are the extraction and simulation routines. Both sets of programs use the setup data (voltage sweeps, compliances, etc.) to measure transistors, but the simulation routines measure a "software transistor", a built-in subroutine which mimics a transistor with the given parameters from the database. Parameter extraction is accomplished by minimizing the RMS error between measured data and simulated results, changing the simulation parameters to accomplish the agreements. These move on to the output control where the data is printed, plotted, or written out in a format which a circuit simulator (not a part of TECAP) can use.

These program capabilities work for a single device or single set of devices for a given set of parameters. They give no indication of device to device or wafer to wafer variations. Adding this capability is the purpose of this thesis work. First, the capability to examine an entire wafer needed to be added.

3.3 Prober Control Modification to TECAP2

As supplied, TECAP2 Ver 1C.00 supports only the Rucker and Kolls 1032 Probing Station. This station is much too expensive for a university research environment and as such could not be used for our measurement system. A much less expensive option, the Rucker and Kolls 681 Prober, would also meet our needs, but was not supported by the system software. Furthermore, the actual code for the RK1032 drivers is for Hewlett-Packard Internal Use Only and could not be obtained for modification. This necessitated writing new code drivers for TECAP2 and the RK681 code, substituting this code into the already compiled code of Hewlett-Packard without doing damage, and re-loading the software.

The modification was done with only a minimum on knowledge of the internal structure of the program. When disassembling the import and export tables of the module PROBE_DRIVER (the sections which establish the linking conventions with outside routines) the names of the several prober-dependent routines were found. Mr. Ebrahim Khalily was kind enough to specify a partial functional description of these modules, which allowed us to write Pascal code to mimic these operations for the RK681 prober.

The final code version of the new PROBE_DRIVER along with the installation manual and linking code are provided in the appendixes of this document.

3.4 Mapping Extensions to TECAP2

As currently supplied, version 1C.00 of TECAP2 has no provision for creating a wafer map of the devices on a wafer even though it does have the capability of stepping across the surface of a wafer to measure it. Because of this limitation there is no insight provided into the device to device or wafer to wafer variation of parameters, a useful observation to someone who is putting together a new process or evaluating the quality of an established on. A purpose of this thesis was to install the foundations of such a mapping system into TECAP2.

3.4.1 Map Data Internal Structure

The data structure for the map is very simple to use and understand. First and foremost, one must store the data to be mapped in the database. Along with this some useful information like which parameter is being mapped is good. Finally, some indication of the validity of the given data point is necessary, since not all wafer coordinates will yield useful data.

The declaration for the map types of data is as follows:

```
type

    {  points to data for map in the heap - dynamic }
         map_ptr = ^map_data;
    {  data structure for the map itself:
                     - .data    contains the numbers
                     - .setflg  whether the data is valid   }
         map_data = record
                          data:array[0..map_x_size, 0..map_y_size] of
                                real;
                          setflg:array [0..map_x_size,0..map_y_size]
                                of boolean;
                     end;

  var

         map_array :              map_ptr;
         map_par_num:             integer;
         map_allocated:           boolean;
```

These variables in tandem with those already declared in the original TECAP DATA BASE determine the mapping parameters for the new routines. The data are kept in memory with a pointer to the data array to minimize memory usage when mapping of parameters is not taking place. MAP_ARRAY^ points to the data, which is only valid when MAP_ALLOCATED is true. Each of the mapping routines refers to this variable when it begins to access the map data segment.

MAP_PAR_NUM is an integer which points to the current mapping parameter in the current active model. Both the model name and its units are already stored in TECAP2's main data base.

The actual data array consists of two parallel arrays of data. The .DATA field contains the value of the parameter to be mapped; its corresponding .SETFLG field indicates if the data field has been set to valid data. These fields are set to 0.0000 and FALSE at initialization, respectively.

Each of these fields is saved in the disk file, along with the user's name, current active model number (with the string for readability), device geometry and type, and a flag indicating it is map data. Note procedures *store_map_data* and *fetch_map_data* in the appendix listing of these programs.

3.4.2 Command Structure of the Map Extension

This provides a brief overview of the functions of the mapping extensions. For a user's guide view of the added commands, please see the chapter later in this document, *Addendum to the TECAP2 User's Guide.*

The auxiliary menu for TECAP2 now appears as:

```
A)  Store map data
A1) Select map param
A2) Initialize map
A3) Print map data
A4) Print stat data
A5) Statistics plot
A6) Wafer Surf. Plot
A7) Save/Fetch Map
A8) Release Prober

A10) Set supply vals
A11) Time delay (s)
```

The commands perform as follows:

A) Store map data

This procedure copies the parameter from the internal transistor parameter array in active_model and stores it in the map data array. It first checks to see if the array has been allocated.

A1) Select map param

Allows the user to select the parameter from the current model which will be mapped. Defaults to the current parameter, initially 1.

A2) Initialize map

Allocates the map data array and zeroes the elements (clears the map). Warns if data already exist.

A3) Print map data

Lists all of the current map data to the current listing device (set in O5-O7).

A4) Print stat data

Prints the mean, standard deviation, 95% confidence interval (assuming normal distribution) [4], minimum and maximum values of the map data to the current list device.

A5) Statistics plot

Plot a histogram (auto_scaled) of the current map data with a normal distribution curve superimposed. Labelled for easy identification.

A6) Wafer Surf. plot

Plot a wafer map (auto_scaled) of the current map data, with symbols representing data in ten steps. Grids can be toggled to appear or not according to the grid_flag set in the P menu.

A7) Store/Fetch map

Used to retrieve and store the current map data.

A10) Set supply values

Will set any SMU or voltage source connected to the system to a given value. Prompts for all answers.

A11) Time delay (s)

Will delay doing anything for a given number of seconds. Prints a period on the screen every two seconds to let the user know the system is still running.

3.4.3 Map Reporting Formats

There are four ways to report the data from the extensions to TECAP2: 1) A direct printing of the data, 2) Printouts of the statistical variations, 3) plots of the distributions (histogram) and 4) a wafer map of the parameters. See the section *Sample Data from TECAP2 Mapping* for more detailed explanations.

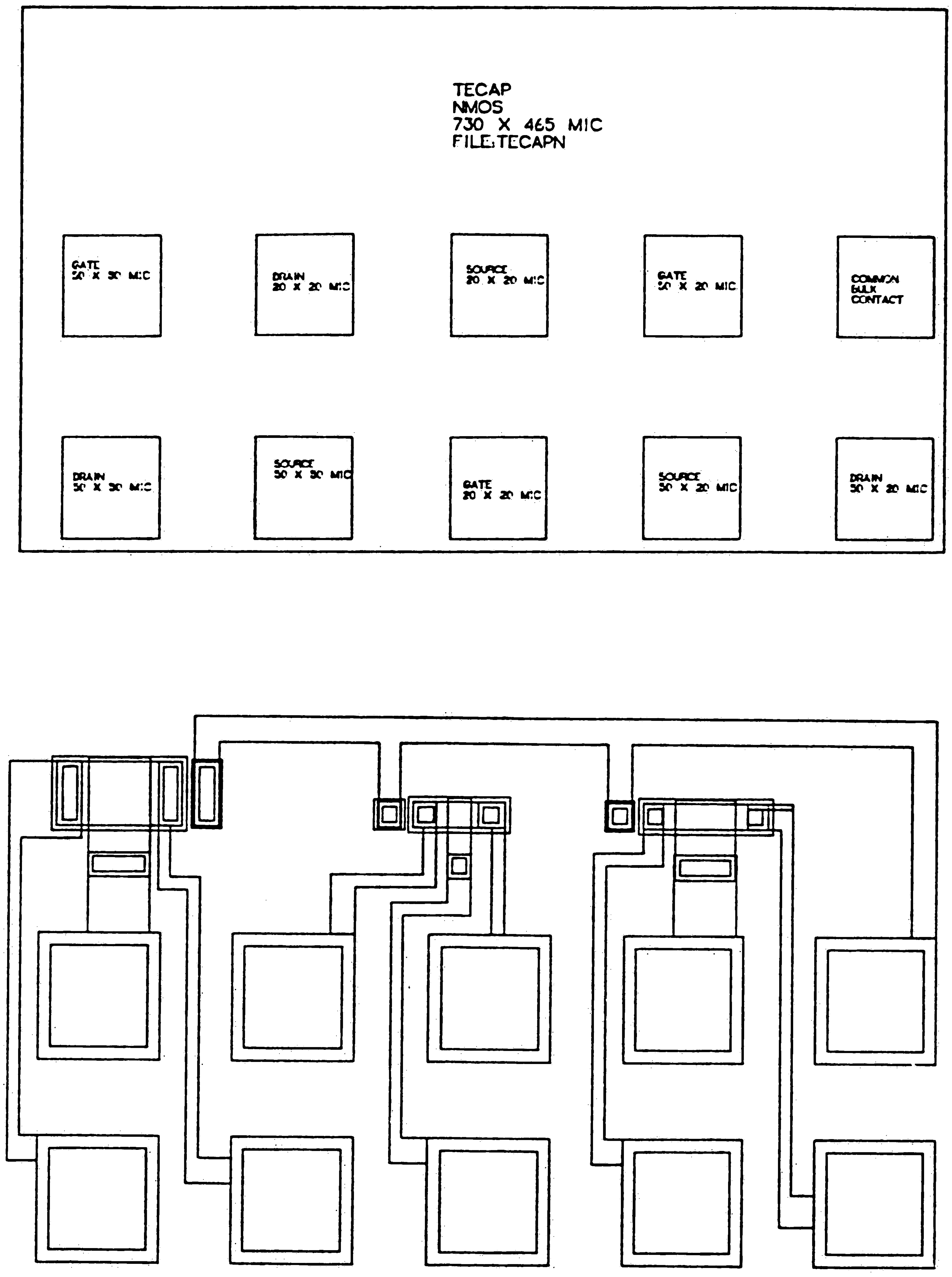
3.5 Test Pattern Design

To fully extract the DC parameters from the device with TECAP a collection of three devices of different geometries is needed. For the purpose of generating maps of the final wafers produced at Sherman Fairchild Labs, a test pattern of three PMOS and three NMOS devices was designed. These are present once in each test chip, 28 times across the new wafer. Some working samples of this pattern were made near the end of this research, but proved to be unsuitable for automatic probing because of the thinness of the aluminum deposited.

To extract the classical MOS device parameters I_d versus V_g data for a large transistor are needed. For the channel width parameters the same measurements are needed on a narrow channel device to force the effects to become dominant. The channel shortening effects, then, require a device with a very short channel.

The transistor dimensions (specified as gate width to length dimensions) are 50 x 50 μm, 50 x 20 μm and 20 x 20 μm. These devices, though they do not press the processing into severe precision, should be sufficient to characterize the processing of the wafers. Slight alignment or patterning problems should not prevent them from operating; rather, they should then be able to point out the errors in the processing. Shown below is the pattern of the NMOS test array; the PMOS pattern is completely analogous.

Figure 3-4: NMOS Test Pattern for Mapping



Chapter 4

Detailed Operation of the TECAP Extensions

Supplement to the TECAP2 User's Guide and System Designer's Guides

This chapter contains the complete modifications which must be made to the TECAP2 User's Guide and System Designer's Guide. All attempts were made to keep the format already present in the 1C.00 version of the manuals, so these pages may be added directly to the manuals. Reference to their locations is made before each new section.

TECAP2 REFERENCE MANUAL ADDENDUM
For section 3.1 Auxiliary Commands

Commands A1 through A7, A10, and A11 have been implemented as part of the the user module code, and allow for the manipulation of the map data.

A) Store map data

This command allows the user to store the extracted parameter into the map data structure. This data will be plotted or printed according to the commands A3 through A6. When the command is given the program reports the current position of the probes (and where the data will be stored) along with the value of the parameter being stored. The display is maintained for a few seconds to allow for reading. The flag indicating this position contains valid data will also be set. An error message will be generated if no space for map data is allocated.

*** WARNING! ****

This routine overwrites the data in the current map data base at the given location. Care should be taken that this data is correct and desired.

A1) Select Map Parameter

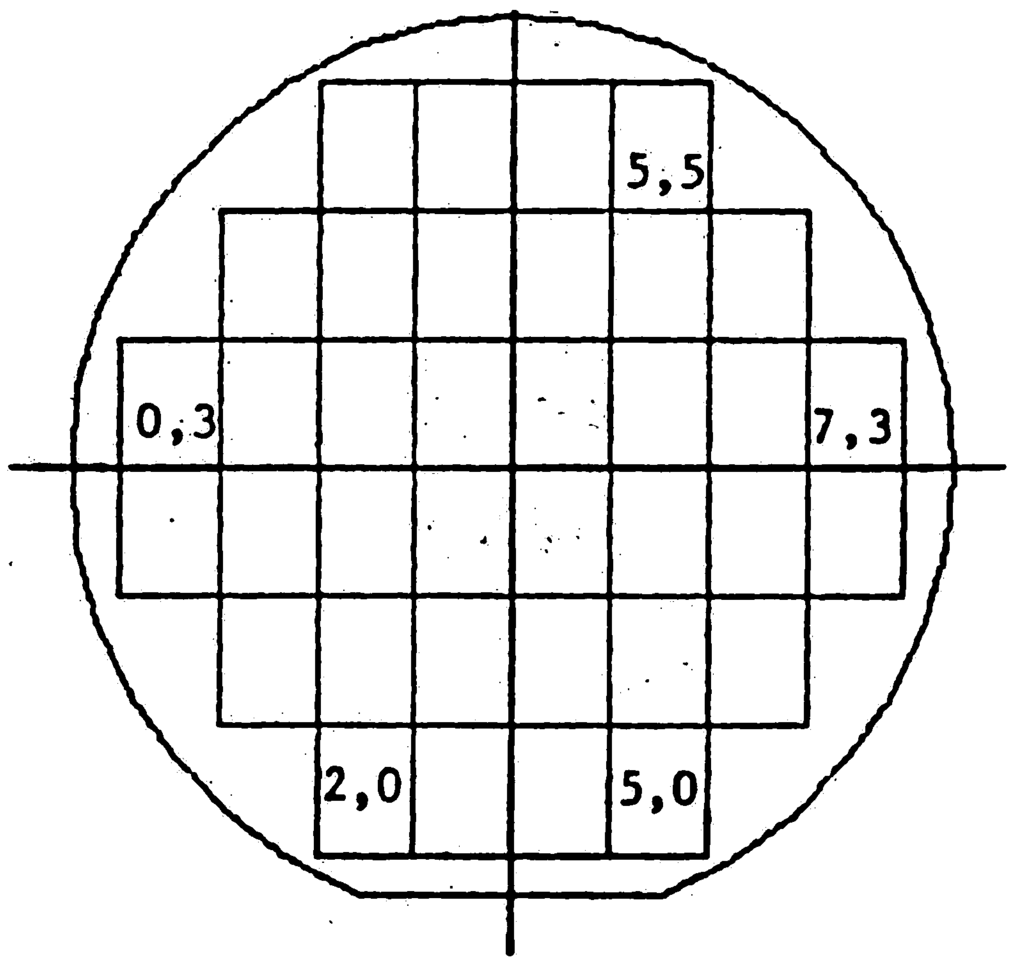
When this command is executed a list of the parameters available from the current active model is displayed and the user is prompted for the number of the parameter to be mapped. The parameter already in effect is the default. If the active model is changed the parameter is reset to number 1. This command does not change or reset the map data structure in any way; the data from previous assignments is still there - use command A2 to reset the data base.

A2) Initialize Map Data

This command will allocate memory space for the map data if it already has not been done. If data space is already set aside the routine asks the user if he really wants to destroy the existing data, and will abort by default to preserve data. The number of chips horizontally and vertically is calculated, the data spaces in the map are set to 0.0 and all flags are set to indicate no valid data. None of these parameters except the map data are set by any other operations. See command A1 for storing map data.

A3) Print Map Data

This command prints the data stored in the map data base to the current text output device (screen, printer, or file). If data has not been set, the data field will read 'Not set'. The coordinates referred to are mapped as follows:



A4) Print Stat Data

This will print the mean, standard deviation, value to a 95% confidence interval of the parameter of interest (assuming a normal distribution) and the minimum and maximum value of the mapped parameter. The values will be displayed at the current text output device.

A5) Plot Stat Data

This command prints a histogram of the distribution of the map data in ten steps (auto scaled and rounded) with mean and standard deviation indicated, along with the corresponding normal data distribution curve for the data (assuming a normal distribution and adjusted for the amount of valid data present in the system. The plot will be performed on the current plot output device.)

A6) Wafer Surface Plot

This command plots a wafer map of the data stored in the current map data base. The data is printed in ten intervals (corres--ponding to the steps in the histogram plot, with the same limits). The ranges of the symbols are plotted nearby with units. The plot is sent to the current plotting device (see commands 01..03)

A7) Fetch and Store Map Data

This sequence will save the current map data base to disk according to the system defaults (drive, extension, name). If the operation is storage, the system will prompt for file name. The '.M' suffix will be appended as with other TECAP2 files. (See the System Designer's Guide for file internal format). If the operation is fetch, the system will execute the A2) Init Map command before loading.

A8) Release prober

This command releases the Rucker and Kolls 681 prober to be operated from its own front panel instead of strictly under TECAP2 control. Useful for alignment of the wafer.

A10) Set Supply Values

Allows any of the SMU's, VS's, VM's or CMU's in the measurement system to be set to a given value. Useful for making non-standard measurements (stress or aging measurements, programming devices, etc.) All values are prompted, the <stop> key will leave all values unchanged.

A11) Time Delay

Simply does nothing for the number of seconds specified. The value is trimmed to the nearest millisecond, and for periods longer than 2 seconds a period is displayed to indicate the computer's continued operation and give an idea of elapsed time.

TECAP2 SYSTEM DESIGNER'S MANUAL ADDENDUM

Under Chapter 5: FILE DESCRIPTION

Add: PROBE_681.TEXT	contains the RK681 prober drivers
PMG_USER.TEXT	contains the map extension routines
PROBE_681.CODE	contains the RK681 prober drivers
PMG_USER.CODE	contains the map extension routines
Stream file LINK_681	links modules with the RK681 drivers
Stream file LINK_PMG	installs the map modifications with the drivers for the RK681

Under Chapter 6: SYSTEM CONFIGURATION

DEVEL: LINK_681.TEXT
PROBE_681.TEXT
PROBE_681.CODE
PMG_USER.TEXT
PMG_USER.CODE
LINK_PMG.TEXT

Under Chapter 17: DATA FILE STRUCTURE

Map Data File:

```
!     MEASURED file, TECAP2 : 1C.02  {MAP DATA FILE
!     User_name
!     ORD(Device_type)
!     Device_name
!     Wafer_name
!     Wafer_comment
!     Device_Length  Device_Width
!     Source_Area    Drain_Area  Source_Perim  Drain_Perim
!     0.00000E+000  0.00000E+000
```

```
!     Active_Model_Numer          Active_Model_Name
!     Map_Parameter_Number        Map_Parameter_Name
!     X_Chip_Size                 Y_Chip_Size
!     1st_Data_point (0,0)
!     2nd_Data_point (0,1)
                .

                .
!     Last_data_point (Xsize,Ysize)
!     End-of-file
```

Chapter 5
Sample Measurements from TECAP2 Mapping

Presented here are the first set of measurements made with the mapping extensions to TECAP2. Regretfully, the working samples of the test pattern designed for this purpose proved unsuitable for automatic probing because of the thinness of the aluminum layer deposited. The metal lifted away from the field oxide whenever the probe card tips touched the surface of the metal. Manual probing was still possible, which showed that the devices worked, though the samples had very high leakage currents. The tests presented here were made on three wafers from TP200 (the second of the Sherman Fairchild Labs Student Project Wafers), supplied by Thomas Krutsick. The threshold voltage VTO, substrate doping NSUB, and surface mobility U0 were extracted from a 50μm by 50μm PMOS transistor in the transistor array designed by Richard Booth and Thomas Krutsick. The channel width narrowing parameter WD was measured on the same PMOS transistor array, devices measuring 50μm by 24μm.

5.1 Models and Extraction Methods used in TECAP2

The parameters extracted by TECAP2 are the standard parameters for HPSPICE [HPSPICE], a modification of SPICE Ver 2 from U.C. Berkeley. Extracted in this thesis are the so-called Level 1 parameters, the classical MOS parameters. The equations used are listed below.

$$V_T = V_{TO} + \lambda f \sqrt{Vsb' + 2\Phi_f} - \lambda f \sqrt{2\Phi_f}$$

where

$$2\Phi_f = \frac{2KT}{q}\ln\left(\frac{N_{sub}}{n_i}\right)$$

$$\lambda = \frac{\sqrt{2\epsilon_{si} q N_{sub}}}{C_{ox}}$$

$$f = \left[1 - \frac{X_j}{L}\left(\sqrt{1 + 2W'/X_j} - 1\right)\right]$$

$$W' = \frac{\sqrt{2\epsilon_{si}}}{\sqrt{qN_{sub}}}\sqrt{Vsb' + 2\Phi_f}$$

$$I_d = 0$$

for $V_{gs}' < V_T$

$$I_d = \frac{\mu C_{ox} W}{L}\left\{\left(V_{gs}' - V_{FB} - 2\Phi_f - V_{DS}'/2\right)V_{ds}' - \frac{2}{3}\lambda f\left(\left(V_{DS}' + V_{SB}' + 2\Phi_f\right)^{\frac{3}{2}} - \left(V_{SB}' + 2\Phi_f\right)^{\frac{3}{2}}\right)\right\}$$

for $V_{GS}' > V_T$ and $V_{DS}' < V_{DSAT}$

$$I_d = \frac{\mu C_{ox} W}{L}\left\{\left(V_{gs}' - V_{FB} - 2\Phi_f - V_{DSAT}/2\right)V_{DSAT} - \frac{2}{3}\lambda f\left(\left(V_{DSAT} + V_{SB}' + 2\Phi_f\right)^{\frac{3}{2}} - \left(V_{SB}' + 2\Phi_f\right)^{\frac{3}{2}}\right)\right\}$$

for $V_{GS}' > V_T$ and $V_{DS} > V_{DSAT}$

Note: L and W are device effective channel length and width; i.e., after subtracting 2xWD and 2xLD from device parameters.

V_{DSAT} is determined iteratively to be V_{DS} when $L \times E_{crit} = \frac{I_d}{dI_d/dV_{DS}}$

$$\mu = \mu_0\left\{\frac{1}{1 + \frac{Vgs' - Vt}{Vnorm}}\right\}\left\{\frac{1}{1 + \frac{Vds'}{L \times Etra}}\right\}$$

Each of the parameters is used to compute the drain current for the "software transistor" given the measurement settings for the actual transistor. The simulation procedure is iterated until the error between the simulated and measured data is minimized, with the parameters of interest used to fit the data.

The algorithm used to extract these parameters is the Levenberg-Marquardt method which uses the first derivative of the function of interest. It combines the method of Steepest Descent and Gauss-Newton described below to minimize the function I_d as a function of the paramters to be extracted.

The method of Steepest Descent is an iterative algorithm which senses the direction of the steepest negative gradient in a function or array of points and proceeds to move the solution to that point. This is a very fast, but not very accurate method; it converges very quickly for initial conditions far from the solution.

Gauss-Newton is a method for solving the same type of system in the neighborhood of the final value. It assumes a quadratic function of the variables near the solution and uses the first three terms of its Taylor series to evaluate the minimum point. The first and second derivatives are approximated from the data itself.

The Levenberg-Marquardt algorithm combines these methods to speed the solution. The method of steepest descent is used to find the neighborhood of the solution and the answer is refined by Gauss-Newton. [TECAP2]

5.2 Substrate Doping NSUB

The doping characteristics were extracted by minimum the difference the simulated and measured threshold voltages, using the doping as a fitting parameter. The substrate doping extracted from the electrical parameters agreed fairly well with what was specified in the original material (5 ohm-cm material). However, the values extracted varied greatly, with the standard deviation equal to half that of the mean value. This data is probably not very meaningful numerically, the distribution over the wafer surface is informative. In wafer B7 the values on the upper half of the chip are fairly constant, increasing as one goes down and to the left, while the data for wafer B2, processed at the same time, shows a marked difference in pattern. Some parameter of the doping steps seems to have been uneven here, but because there are no records of the orientation of the wafers during processing, no conclusions can be reached. For future runs with the test chips, orientation will have to be monitored.

5.3 Threshold Voltage VTO

This parameter was extracted by a simple linear fit through an approximated linear region of the I-V curve and is found at the x intercept. The solution here is also iterated to find the best fit, and, along with NSUB and U0, determines the fit of the extracted to measured curves. The threshold voltage control on both wafers seems very good. Both are centered at about -1.1 volts, and the variations across the wafer are typically a few millivolts. Some of the devices show a marked difference in the threshold, but this is probably due to some local defects and can thus be ignored for this treatment. Note however that the data is bunched around the mean and tails off toward the more negative threshold voltage. The curve resembles the distribution of dopants as they diffuse into a substrate; this could show the profile of the threshold voltage implant, that is, how well the depth of the implant is controlled across the surface of the wafer.

5.4 Surface mobility U0

This is again used to fit the data to the simulation. The parameters of interest are interrelated as in the equations above. The extracted surface mobility is fairly constant across the wafer, which is not surprising since the PMOS devices are fabricated into the perfect or nearly perfect N substrate. There is no reason to expect the same good fortune for the NMOS devices. For wafer B7 the mean of the data is very close to that reported in elementary

/

textbooks of 450 cm^2/V-s. The variation is also not great from the mean reported. The value for this parameter on wafer B2 is much much lower that for B7, but there does not seem to be a difference between the measurement approaches. Several repetitions of this experiment and extraction yielded the same results. If we assume the data from wafer B7 to be correct, then we have verified (statistically) the surface mobility for holes in silicon.

5.5 Channel narrowing due to lateral diffusion, the WD parameter

This is a measure of the width reduction parameter of the finshed wafer with respect to the designed width. The channel width is expressed as $W_{effective} = W_{designed} - 2 \times WD$. The data seems to show that the dopant from the surrounding material narrows the channels by 8μm on a side for a total of 16μm. This would suggest that a channel cannot exist in any device designed to be narrower than 16μm.

This data is probably not valid, however. The extractions were performed on devices which were made in a non-self-aligned process, so the assumptions of the gate geometry made for this may not be true. Also, since data for intermediate parameters was made on physically different devices, the parameters may be incorrect for the location mapped. These possible sources of error add up to a rather unbelievable result as far as this goes. This would indicate a misalignment of about 2.5-5 μm during the masking sequences, something which clearly did not happen.

This is an example of what the mapping system can accomplish. The data measurement, reduction, and report generation took six hours to complete. This involved the measurement of the equivalent of four full wafers of data and the extraction of four separate parameter sets. These parameters point out some of the processing ideas in these wafers, but since they were not as carefully tracked as one would like, only a few conclusions regarding processing could be drawn.

The parameters were normally distributed or at least nearly so (a bit of a wide skew due to the "freaks" of the distribution), justifying the use of that distribution in this analysis. Another similar distribution, the Pearson IV distribution (commonly used to model implantation depth) might also be used for these one-sided variations in extracted parameters. Local irregularities (lower that normal threshold voltage for example) are most likely caused by impurities introduced in processing or some local crystal flaw, and can be ignored. However, regions where several devices show the same tendencies (such as the substrate doping) can be used to draw conclusions about the underlying population distributions.

Chapter 6
Capacitance-Voltage Measurement System Development

Another useful, but currently unrelated, automatic measurement system which exists in Sherman Fairchild Laboratory is a Capacitance-Voltage measurement system based on an MSI Electronics Data Station and an HP85 controller. The software currently in use was written at Lehigh to rectify several deficiencies in the program supplied with the station.

The worst of these deficiencies was a consistent tendency to destroy the device under test with excessive voltage. The software would set a maximum voltage and search for the accumulation region. This works fine if there is a clear-cut accumulation region, but if there is some region where the accumulation wanders, the program will not find it, will increase the voltage, and search again. This eventually will reach a point where the voltage will exceed the strength of the device and it will be destroyed.

Other nice features which the new program contains are:

- **Default Parameters:** The new software contains the most commonly used measurement parameters for our lab.

- **Menu Driven:** Even a novice user may be able to use the program without a manual.

- **Asymmetric Sweep:** In the old software the voltage sweep needed to be symmetric about zero. The new program allows arbitrary sweep size.

- **Data Management:** Results of measurements may be saved and retrieved for later examination.

Future enhancements should include different gate materials, variable duration temperature stress (currently it is limited to the cycle time of the chuck temperature, about 5 minutes), and a pulsed-capacitance measurement capability to provide such features as doping profiling.

A complete user's manual and system guide is included in the appendixes of this document.

Chapter 7
Conclusions and Recommendations

In this paper it has been shown that, for a small laboratory, a flexible and inexpensive DC parametric analysis system is an indispensable tool. From this general discussion the merits of the TECAP2 program were discussed, and the lack of device to device comparison and data reporting were noted.

Program extensions were made to TECAP2 to remedy the deficiencies in the program with respect to the comparison of devices with respect to location on a wafer. These extensions were tested, and the results reinforced the need for such characterizations. Variations were seen in several classical parameters, and in the channel narrowing effects. The channel narrowing effects as extracted were also related to other observed effects.

A second program, for in-lab C-V measurement, was also discussed. The complete operations manual is included in the appendixes to this document.

Recommendations for future development include:

- Three-dimensional plotting of wafer parameter or contour plotting to even more clearly show the variations across a wafer. This could be used as a processing monitor, carried along through the process to control variations.

- A provision for storing and plotting multiple parameters for the same wafer, easing the extraction burden.

- Acquisition of a full-Kelvin matrix switching system to allow for more than a four-port measurement. At least 20 ports are needed for an effective system. This is currently the system's major weakness, since the need to reprobe the devices causes destruction of devices and loss of time.

- Addition of some variety of capacitance meter to the system, preferably the HP4280, already supported by the software. This would eliminate the need for approximating C_{ox} as constant across the wafer.

- A thermal chuck for the prober, to allow for bias-temperature stress measurement and study.

- Several different distribution models should be considered in the statistical reporting. A 10 or 15% trimmed mean should be reported, thus eliminating some of the misleading distribution data caused by local variations outside the normal problems.

- A low current capability coupled with low noise probes could aid in the characterization of even smaller geometry devices.

- Some method for adding of user models to the system should be developed. Some work has been done to this end, but is not complete enough to present in this work. Any parameters supplied to the user model could be extracted via the optimization/minimization scheme already built into TECAP2. The description of the terminal voltages and currents are all that is necessary. This could be done such that the equations are entered into a utility program which would compose the appropriate Pascal code to implement the model.

References

[1] Buehler, Martin G., and Loren W. Linholm.
Role of Test Chips in Coordinating Logic and Circuit Design and Layout Aids for VLSI.
Solid State Technology Sept:68-74, 1981.

[2] Cheyney, Ward and David Kincaid.
Numerical Mathematics and Computing.
Brooks/Cole, Monterey, CA, 1980.

[3] Hewlett-Packard.
Analysis of Semiconductor Capacitance Characteristics Using the 4280A High Frequency Capacitance Meter.
Application Note 322. Palo Alto, CA, 1983.

[4] Devore, Jay L.
Probablility and Statistics for Engineering and the Physical Sciences.
Brooks/Cole, Monterey, CA, 1982.

[5] Gordon, Barton J.
A Microprocessor-Based Semiconductor Measurement System.
Solid State Technology July:43-47, 1978.

[6] Howard, John S. and Jinet Nahourai.
Improvement in LSI Production using an Automated Parametric Test System.
Solid State Technology July:48-52, 1978.

[7] HPDA.
Hewlett-Packard 4145A Semiconductor Parameter Analyzer User's Manual
Palo Alto, CA, 1982.
Instrument Operations Guide.

[8] Hewlett-Packard Design Aids.
HPSPICE User's Manual.
Technical Report, Hewlett-Packard Company, 1980.

[9] Hewlett-Packard Design Aids.
SPICE Model Equations.
Technical Report, Hewlett-Packard Company, 1980.

[10] Kaempf, Ulrich .
Automated parametric testers to monitor the integrated circuit process.
Solid State Technol. Sept:81-87, 1981.

[11] Perloff, D. S., C. L. Mallory, F. E. Wahl, and S. W. Mylroie.
Microelectronic Test Chips in Integrated Circuit Manufacturing.
Solid State Technology Sept:75-80, 1981.

[12] Streetman, Ben G.
Solid State Electronic Series: Solid State Electronic Devices.
Prentice-Hall, Inc., Englewood Cliffs, NJ, 1980.

[13] Sze, S. M.
Physics of Semiconductor Devices.
John Wiley and Sons, New York, NY, 1981.

[14] Sze, S.M.
VLSI Technology.
McGraw-Hill, New York, NY, 1983.

[15] Khalily, Ebrahim.
TECAP - An automated characterization system.
Technical Report SEL-79-004, Stanford University, Mar, 1979.

[16] Khalily, E., Paul Decker, Irving Klein, and Darryl Teergarden.
TECAP2-Transistor Electrical Characterization and Analysis Program
Hewlett-Packard Design Aids Division, Palo Alto, CA, 1980.

[17] Fang, Robert C.Y., Robert D. Rung, Kit M. Cham.
An Improved Automatic Test System for VLSI Parametric Testing.
IEEE Trans. Instru. Meas. IM-31(3):198-205, Sept, 1982.

Appendix A
Measured Wafer Data

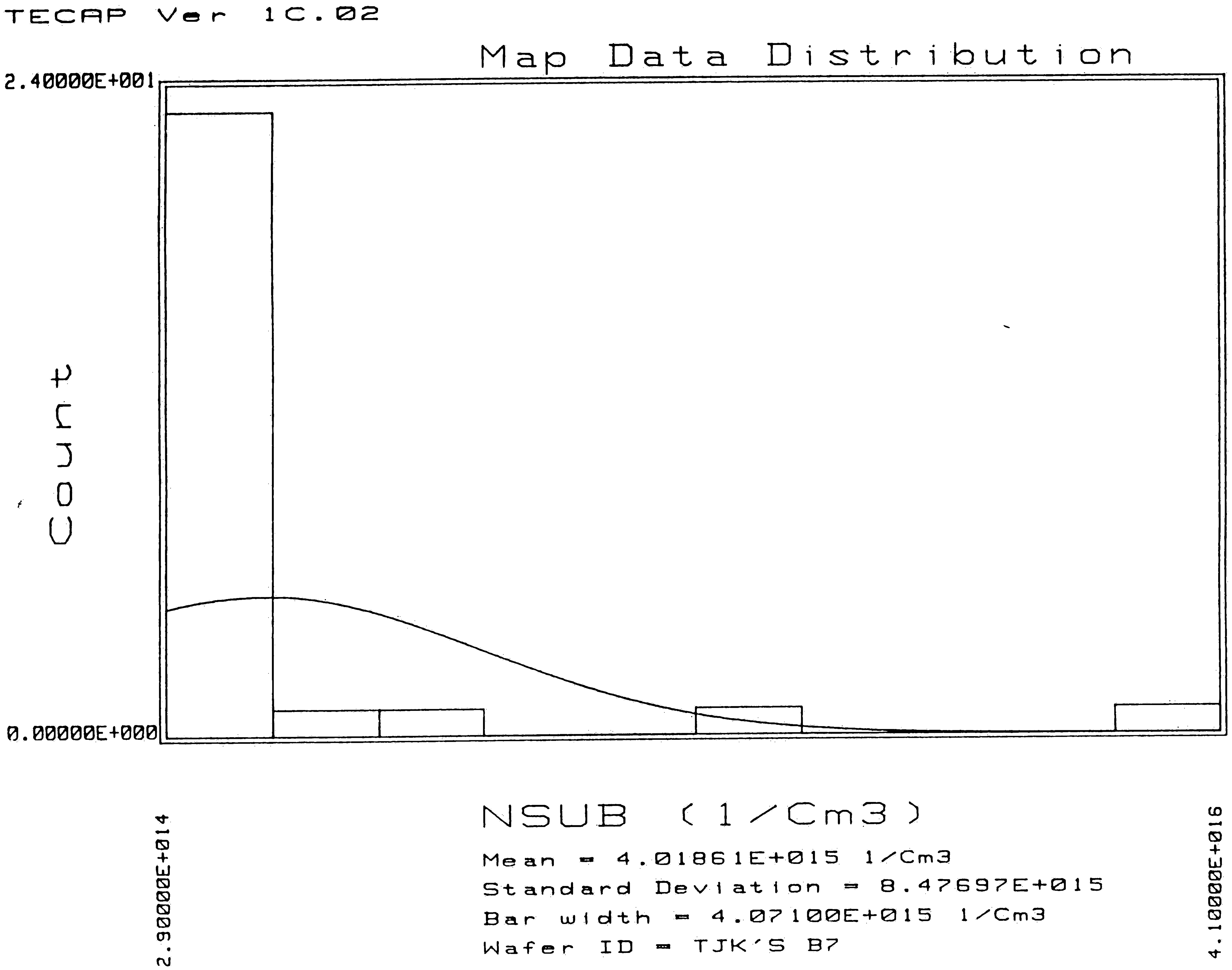
The data contained in this section was measured from devices supplied by Thomas Krutisck. Evaluation of this data may be found in the main body of this thesis.

Source: RATARRAY PMOS Transistors, supplied by Thomas Krutsick

Table A-1: Wafer B7 Substrate Doping Data and Statistics (NSUB)

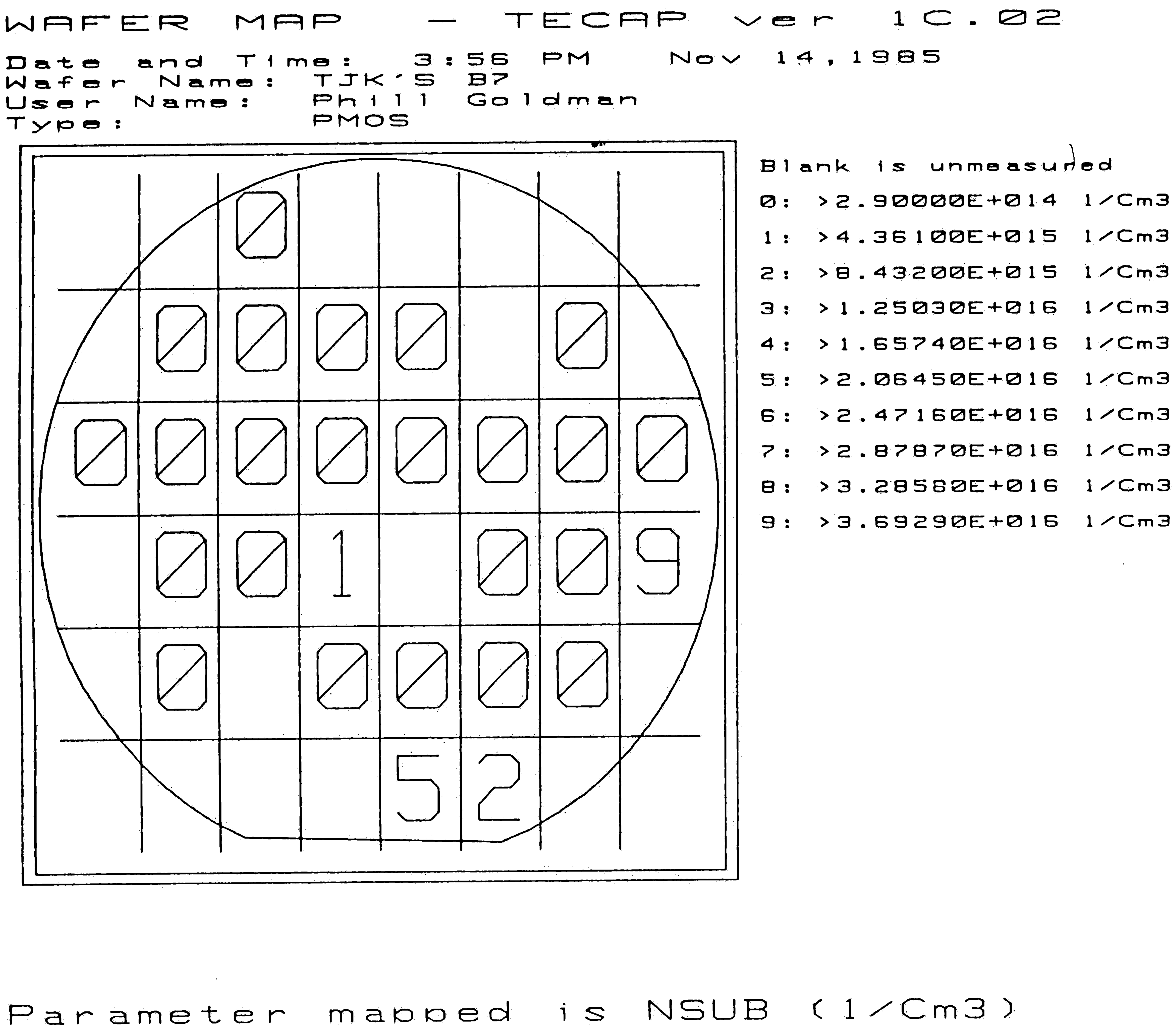
```
Print Stored Map Data
---------------------

Wafer ID is TJK'S B7
User Name is Phill Goldman
Date and Time:  3:59 PM   Nov 14,1985

Active Model is HPSPICE-MOS
Parameter Mapped is NSUB  (1/Cm3)

--------------------------------------------
| X position       Y position         Value |
--------------------------------------------
       0                0          Not set
       0                1          Not set
       0                2          Not set
       0                3          1.23143E+015
       0                4          Not set
       0                5          Not set
       1                0          Not set
       1                1          1.02535E+015
       1                2          7.38023E+014
       1                3          7.13797E+014
       1                4          6.89194E+014
       1                5          Not set
       2                0          Not set
       2                1          Not set
       2                2          6.11342E+014
       2                3          6.84010E+014
       2                4          7.49834E+014
       2                5          9.07993E+014
       3                0          Not set
       3                1          1.41245E+015
       3                2          5.22303E+015
       3                3          1.27094E+015
       3                4          1.67646E+015
       3                5          Not set
       4                0          2.16300E+016
       4                1          1.17459E+015
       4                2          Not set
       4                3          1.53457E+015
       4                4          2.93809E+014
       4                5          Not set
       5                0          8.59892E+015
       5                1          2.61200E+015
       5                2          1.21715E+015
       5                3          1.72209E+015
       5                4          Not set
       5                5          Not set
       6                0          Not set
       6                1          4.10297E+015
       6                2          2.90167E+015
       6                3          1.76191E+015
       6                4          1.61730E+015
       6                5          Not set
       7                0          Not set
       7                1          Not set
       7                2          4.09179E+016
       7                3          1.48362E+015
       7                4          Not set
       7                5          Not set
```

```
Wafer ID is TJK'S B7
User Name is Phill Goldman
Date and Time:  4: 0 PM   Nov 14,1985

Parameter Mapped is NSUB  (1/Cm3)
Statistical Data for Current Map
--------------------------------

 Mean                          = 4.01861E+015
 Standard Deviation            = 8.47697E+015
 Value of parameter            = 4.01861E+015  +/-  2.39815E+015

 Minimum data value            = 2.93809E+014
       at location (4, 4)
 Maximum data value            = 4.09179E+016
       at location (7, 2)
```



Mean = 4.01861E+015 1/Cm3
Standard Deviation = 8.47697E+015
Bar width = 4.07100E+015 1/Cm3
Wafer ID = TJK'S B7

Figure A-1: Wafer B7 Substrate Doping Distribution (NSUB)



```
WAFER MAP   - TECAP ver 1C.02
Date and Time:   3:56 PM    Nov 14,1985
Wafer Name:  TJK'S B7
User Name:   Phill Goldman
Type:        PMOS

Blank is unmeasured
0: >2.90000E+014 1/Cm3
1: >4.36100E+015 1/Cm3
2: >8.43200E+015 1/Cm3
3: >1.25030E+016 1/Cm3
4: >1.65740E+016 1/Cm3
5: >2.06450E+016 1/Cm3
6: >2.47160E+016 1/Cm3
7: >2.87870E+016 1/Cm3
8: >3.28580E+016 1/Cm3
9: >3.69290E+016 1/Cm3

Parameter mapped is NSUB (1/Cm3)
```

Figure A-2: Wafer B7 Substrate Doping Map (NSUB)

Table A-2: Wafer B2 Substrate Doping Data and Statistics (NSUB)

```
Print Stored Map Data
---------------------

Wafer ID is TJK'S B2
User Name is Phill Goldman
Date and Time:  3:38 PM   Nov 14,1985

Active Model is HFSPICE-MOS
Parameter Mapped is NSUB  (1/Cm3)

---------------------------------------------
| X position       Y position          Value |
---------------------------------------------
      0                0           Not set
      0                1           Not set
      0                2           1.00000E+013
      0                3           2.04370E+015
      0                4           Not set
      0                5           Not set
      1                0           Not set
      1                1           3.07383E+015
      1                2           Not set
      1                3           Not set
      1                4           2.14929E+015
      1                5           Not set
      2                0           5.40163E+015
      2                1           1.04597E+014
      2                2           2.19926E+015
      2                3           2.73484E+015
      2                4           Not set
      2                5           2.96110E+015
      3                0           8.58870E+015
      3                1           2.79409E+015
      3                2           1.99707E+015
      3                3           Not set
      3                4           2.71188E+015
      3                5           2.12907E+015
      4                0           5.54192E+014
      4                1           6.32756E+015
      4                2           Not set
      4                3           Not set
      4                4           Not set
      4                5           Not set
      5                0           Not set
      5                1           1.01063E+016
      5                2           4.40343E+015
      5                3           3.46803E+015
      5                4           4.21696E+015
      5                5           4.28820E+015
      6                0           Not set
      6                1         - 1.70512E+016
      6                2           Not set
      6                3           6.02572E+015
      6                4           3.32647E+016
      6                5           Not set
      7                0           Not set
      7                1           Not set
      7                2           Not set
      7                3           1.00002E+013
      7                4           Not set
      7                5           Not set
```

```
Wafer ID is TJK'S B2
User Name is Phill Goldman
Date and Time:  3:39 PM   Nov 14,1985

Parameter Mapped is NSUB  (1/Cm3)
Statistical Data for Current Map
--------------------------------

 Mean                         = 5.14461E+015
 Standard Deviation           = 6.91603E+015
 Value of parameter           = 5.14461E+015  +/-  1.95655E+015

 Minimum data value           = 1.00000E+013
        at location (0, 2)
 Maximum data value           = 3.32647E+016
        at location (6, 4)
```

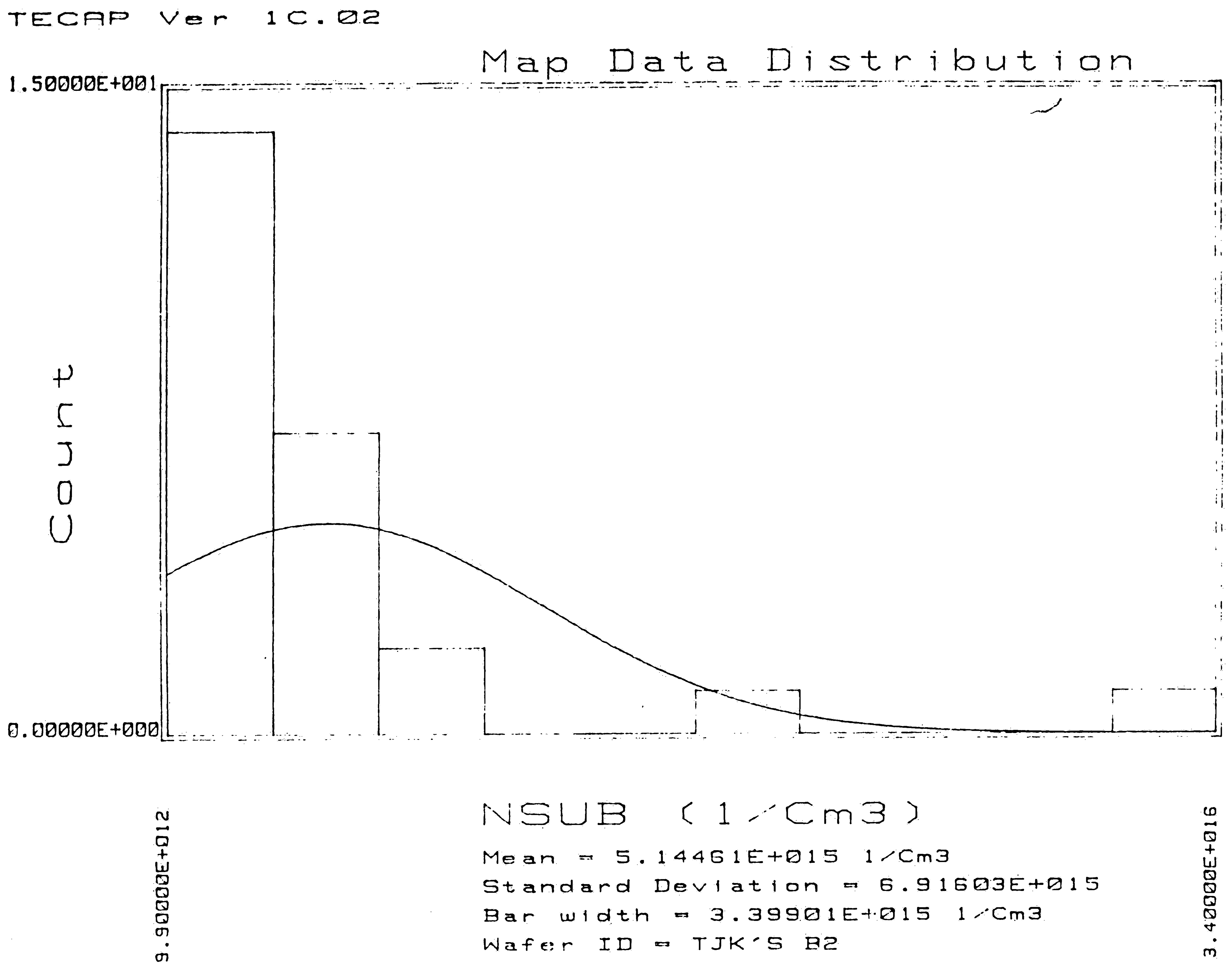
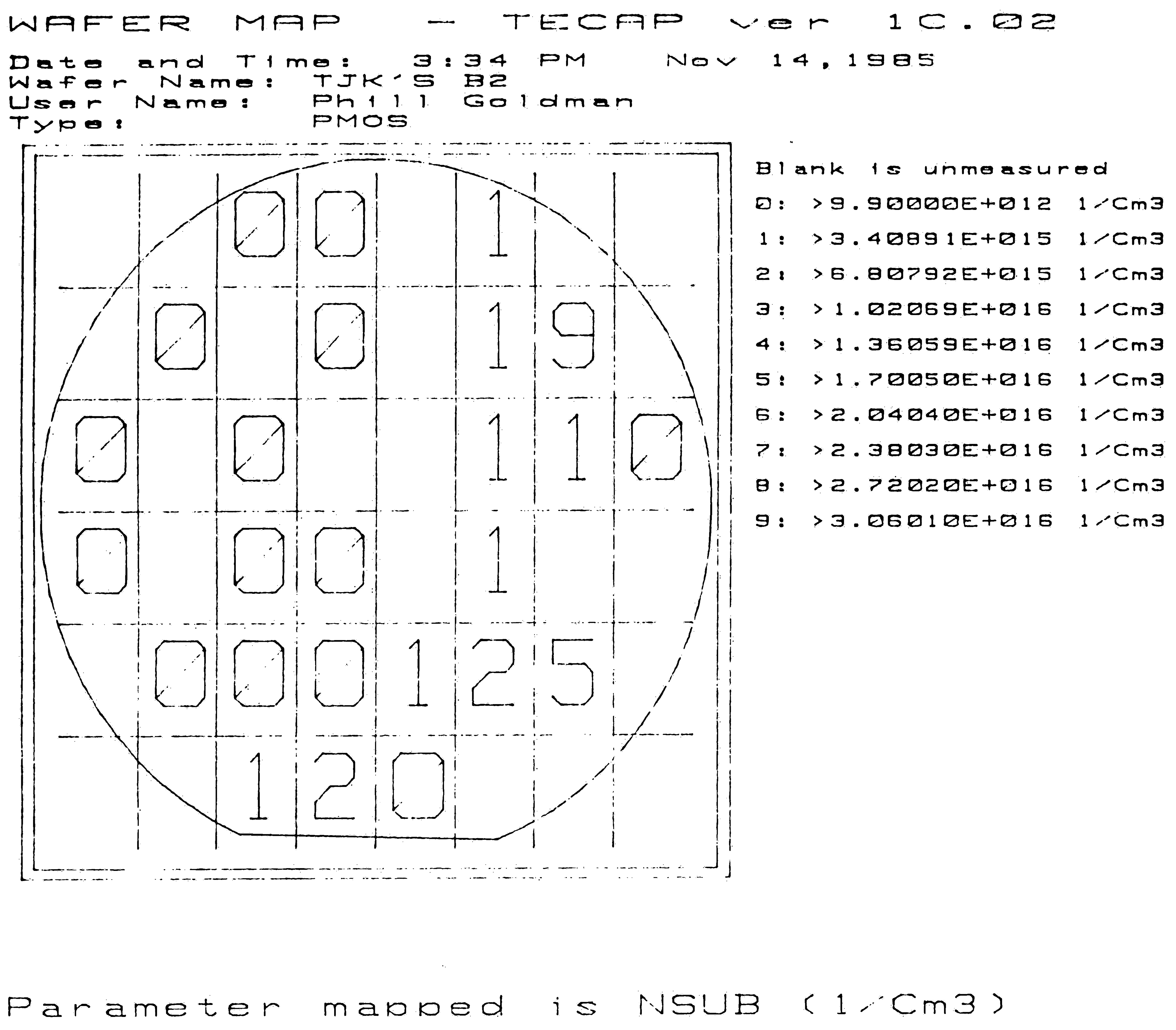


Figure A-3: Wafer B2 Substrate Doping Distribution (NSUB)



```
WAFER MAP   - TECAP ver 1C.02
Date and Time:   3:34 PM    Nov 14,1985
Wafer Name:  TJK'S B2
User Name:   Phill Goldman
Type:        PMOS
```

		0	0		1		
	0		0		1	9	
0		0			1	1	0
0		0	0		1		
	0	0	0	1	2	5	
		1	2	0			

```
Blank is unmeasured
0: >9.90000E+012 1/Cm3
1: >3.40891E+015 1/Cm3
2: >6.80792E+015 1/Cm3
3: >1.02069E+016 1/Cm3
4: >1.36059E+016 1/Cm3
5: >1.70050E+016 1/Cm3
6: >2.04040E+016 1/Cm3
7: >2.38030E+016 1/Cm3
8: >2.72020E+016 1/Cm3
9: >3.06010E+016 1/Cm3
```

Parameter mapped is NSUB (1/Cm3)

Figure A-4: Wafer B2 Substrate Doping Map (NSUB)

Table A-3: Wafer B7 Threshold Voltage Data and Statistics (VTO)

```
Print Stored Map Data
---------------------

Wafer ID is TJK'S B7
User Name is Phill Goldman
Date and Time:  3:51 PM   Nov 14,1985

Active Model is HPSPICE-MOS
Parameter Mapped is VTO  (Volt)

---------------------------------------------
| X position       Y position          Value |
---------------------------------------------
       0               0         Not set
       0               1         Not set
       0               2         Not set
       0               3         -1.0261E+000
       0               4         Not set
       0               5         Not set
       1               0         Not set
       1               1         -1.0087E+000
       1               2         -9.6880E-001
       1               3         -9.5831E-001
       1               4         -9.4556E-001
       1               5         Not set
       2               0         Not set
       2               1         Not set
       2               2         -9.6323E-001
       2               3         -9.5766E-001
       2               4         -9.3213E-001
       2               5         -9.5549E-001
       3               0         Not set
       3               1         -1.0076E+000
       3               2         -2.0417E+000
       3               3         -1.0048E+000
       3               4         -1.0009E+000
       3               5         Not set
       4               0         -1.3771E+000
       4               1         -1.0561E+000
       4               2         Not set
       4               3         -1.0130E+000
       4               4         -1.1783E+000
       4               5         Not set
       5               0         -1.2619E+000
       5               1         -1.1071E+000
       5               2         -1.1526E+000
       5               3         -1.0405E+000
       5               4         Not set
       5               5         Not set
       6               0         Not set
       6               1         -1.1614E+000
       6               2         -1.1010E+000
       6               3         -1.0528E+000
       6               4         -1.0362E+000
       6               5         Not set
       7               0         Not set
       7               1         Not set
       7               2         -1.4265E+000
       7               3         -1.0508E+000
       7               4         Not set
       7               5         Not set
```

```
Wafer ID is TJK'S B7
User Name is Phill Goldman
Date and Time:  3:50 PM   Nov 14,1985

Parameter Mapped is VTO  (Volt)
Statistical Data for Current Map
--------------------------------

 Mean                          = -1.1032E+000
 Standard Deviation            = 2.25404E-001
 Value of parameter            = -1.1032E+000  +/-  6.37673E-002

 Minimum data value            = -2.0417E+000
       at location (3, 2)
 Maximum data value            = -9.3213E-001
       at location (2, 4)
```

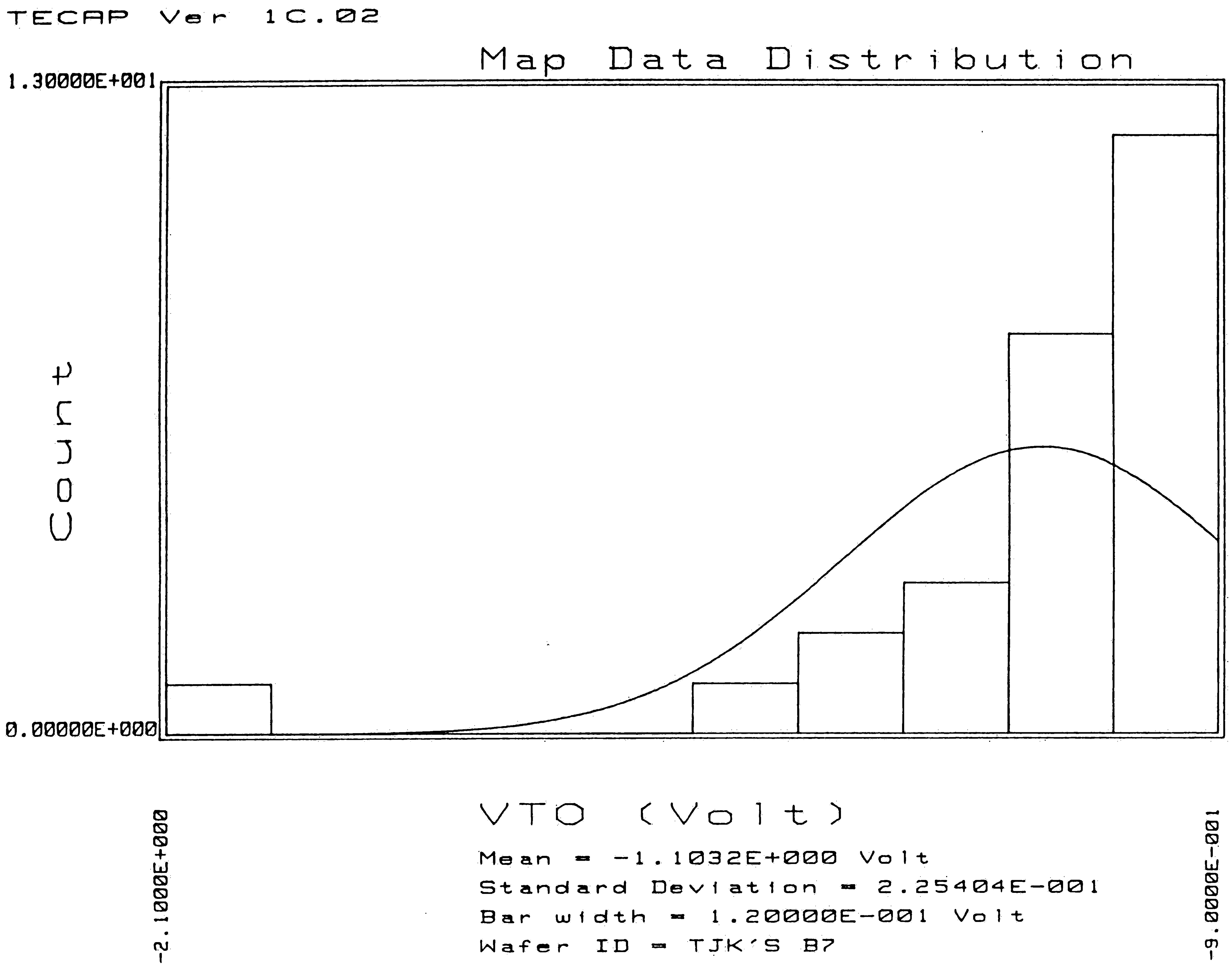
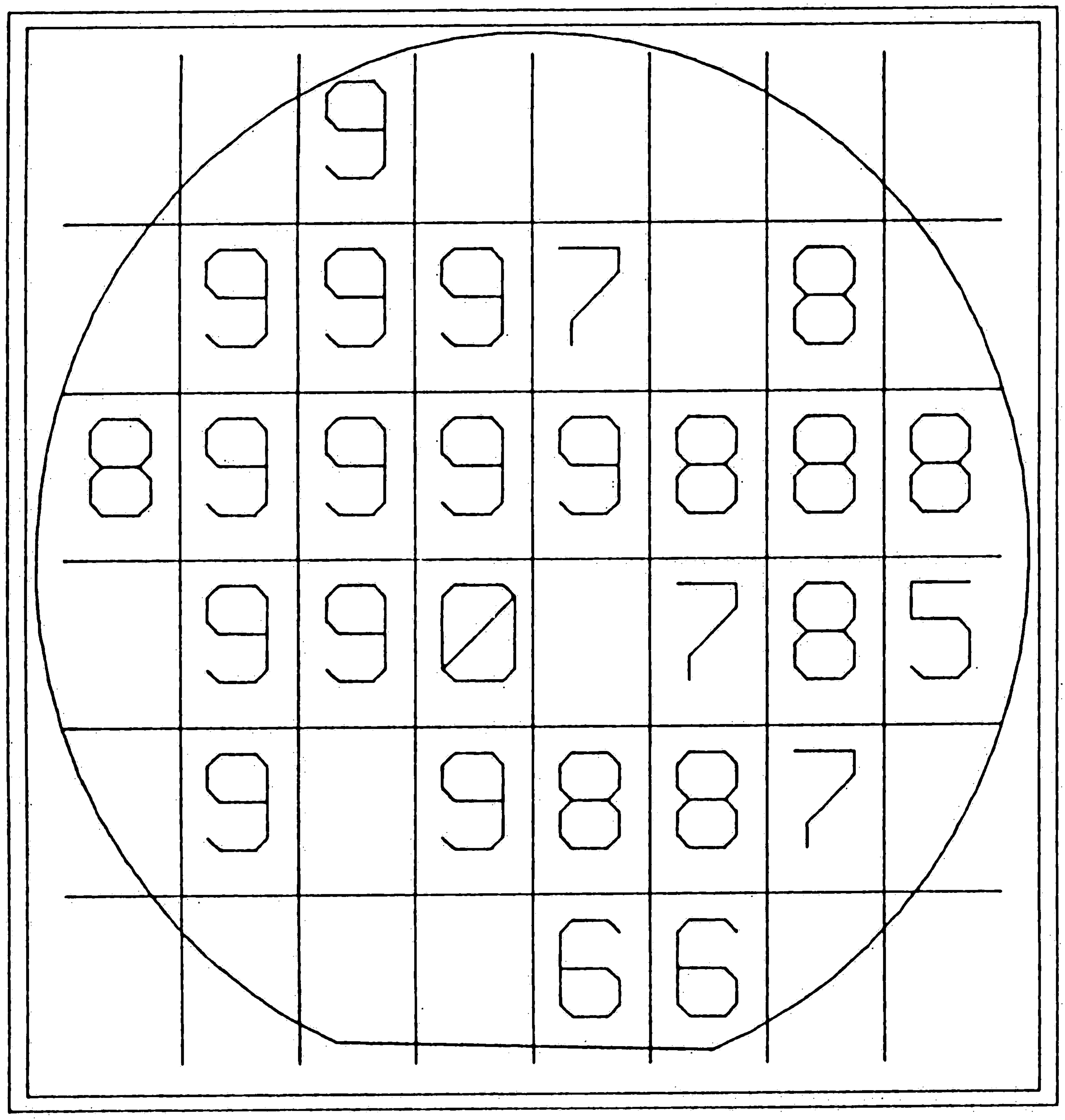


Figure A-5: Wafer B7 Threshold Voltage Distribution (VTO)

```
WAFER MAP   - TECAP ver 1C.02
Date and Time:   3:48 PM   Nov 14,1985
Wafer Name:  TJK'S B7
User Name:   Phill Goldman
Type:        PMOS
```



		9					
	9	9	9	7		8	
8	9	9	9	9	8	8	8
	9	9	0		7	8	5
	9		9	8	8	7	
				6	6		

```
Blank is unmeasured
0: >-2.1000E+000 Volt
1: >-1.9800E+000 Volt
2: >-1.8600E+000 Volt
3: >-1.7400E+000 Volt
4: >-1.6200E+000 Volt
5: >-1.5000E+000 Volt
6: >-1.3800E+000 Volt
7: >-1.2600E+000 Volt
8: >-1.1400E+000 Volt
9: >-1.0200E+000 Volt
```

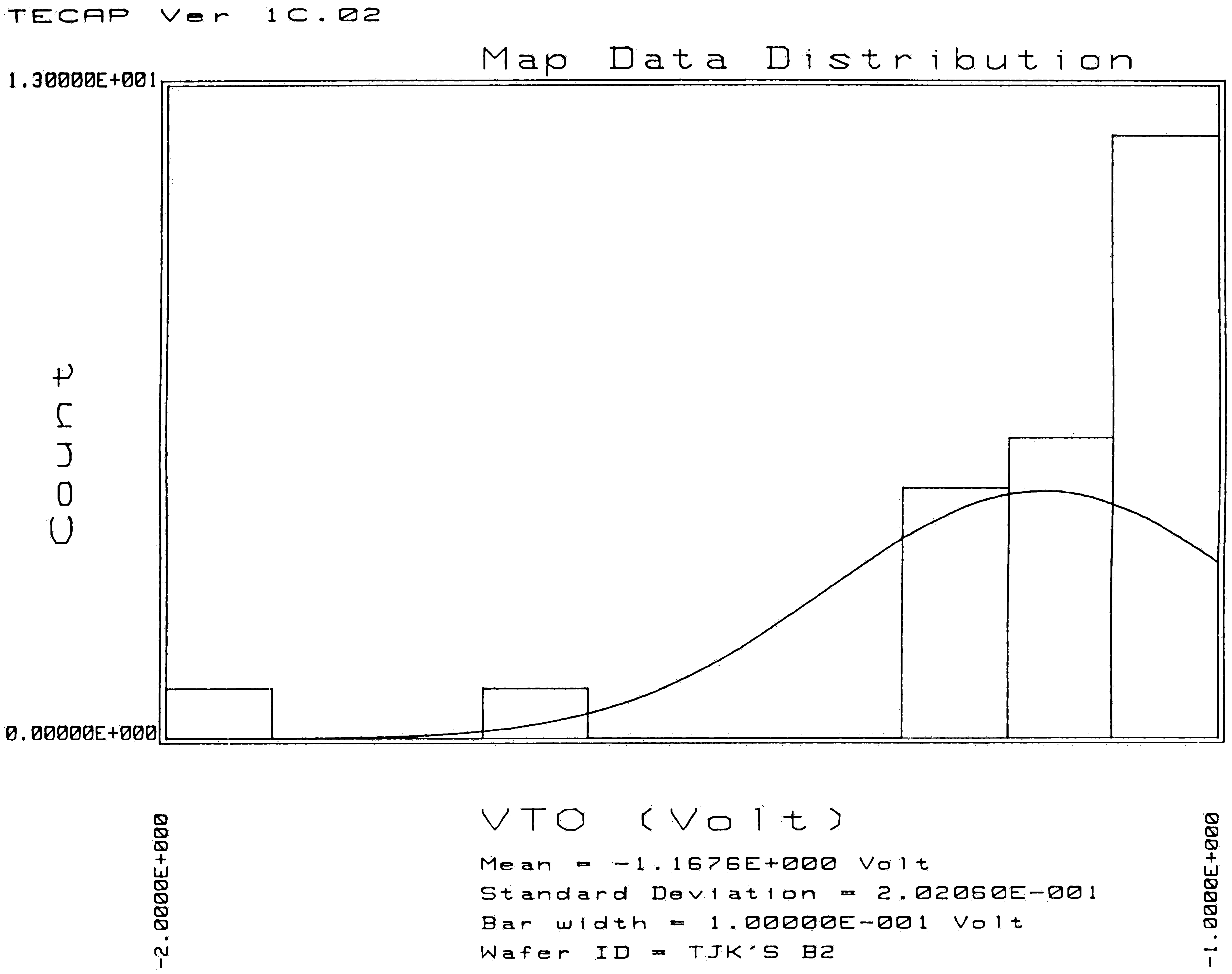
```
Parameter mapped is VTO (Volt)
```

Figure A-6: Wafer B7 Threshold Voltage Map (VTO)

Table A-4: Wafer B2 Threshold Voltage Data and Statistics (VTO)

```
Print Stored Map Data
---------------------

Wafer ID is TJK'S B2
User Name is Phill Goldman
Date and Time:  3:45 PM   Nov 14,1985

Active Model is HPSPICE-MOS
Parameter Mapped is VTO  (Volt)

----------------------------------------------
| X position       Y position         Value |
----------------------------------------------
      0                 0          Not set
      0                 1          Not set
      0                 2          -1.1087E+000
      0                 3          -1.0662E+000
      0                 4          Not set
      0                 5          Not set
      1                 0          Not set
      1                 1          -1.0759E+000
      1                 2          Not set
      1                 3          Not set
      1                 4          -1.0534E+000
      1                 5          Not set
      2                 0          -1.1439E+000
      2                 1          -1.0760E+000
      2                 2          -1.0287E+000
      2                 3          -1.0475E+000
      2                 4          Not set
      2                 5          -1.0795E+000
      3                 0          -1.2087E+000
      3                 1          -1.0414E+000
      3                 2          -1.0258E+000
      3                 3          Not set
      3                 4          -1.0669E+000
      3                 5          -1.0672E+000
      4                 0          -1.6250E+000
      4                 1          -1.0891E+000
      4                 2          Not set
      4                 3          Not set
      4                 4          Not set
      4                 5          Not set
      5                 0          Not set
      5                 1          -1.2128E+000
      5                 2          -1.1063E+000
      5                 3          -1.1038E+000
      5                 4          -1.1132E+000
      5                 5          -1.1738E+000
      6                 0          Not set
      6                 1          -1.2078E+000
      6                 2          Not set
      6                 3          -1.2451E+000
      6                 4          -1.2903E+000
      6                 5          Not set
      7                 0          Not set
      7                 1          Not set
      7                 2          Not set
      7                 3          -1.9327E+000
      7                 4          Not set
      7                 5          Not set
```

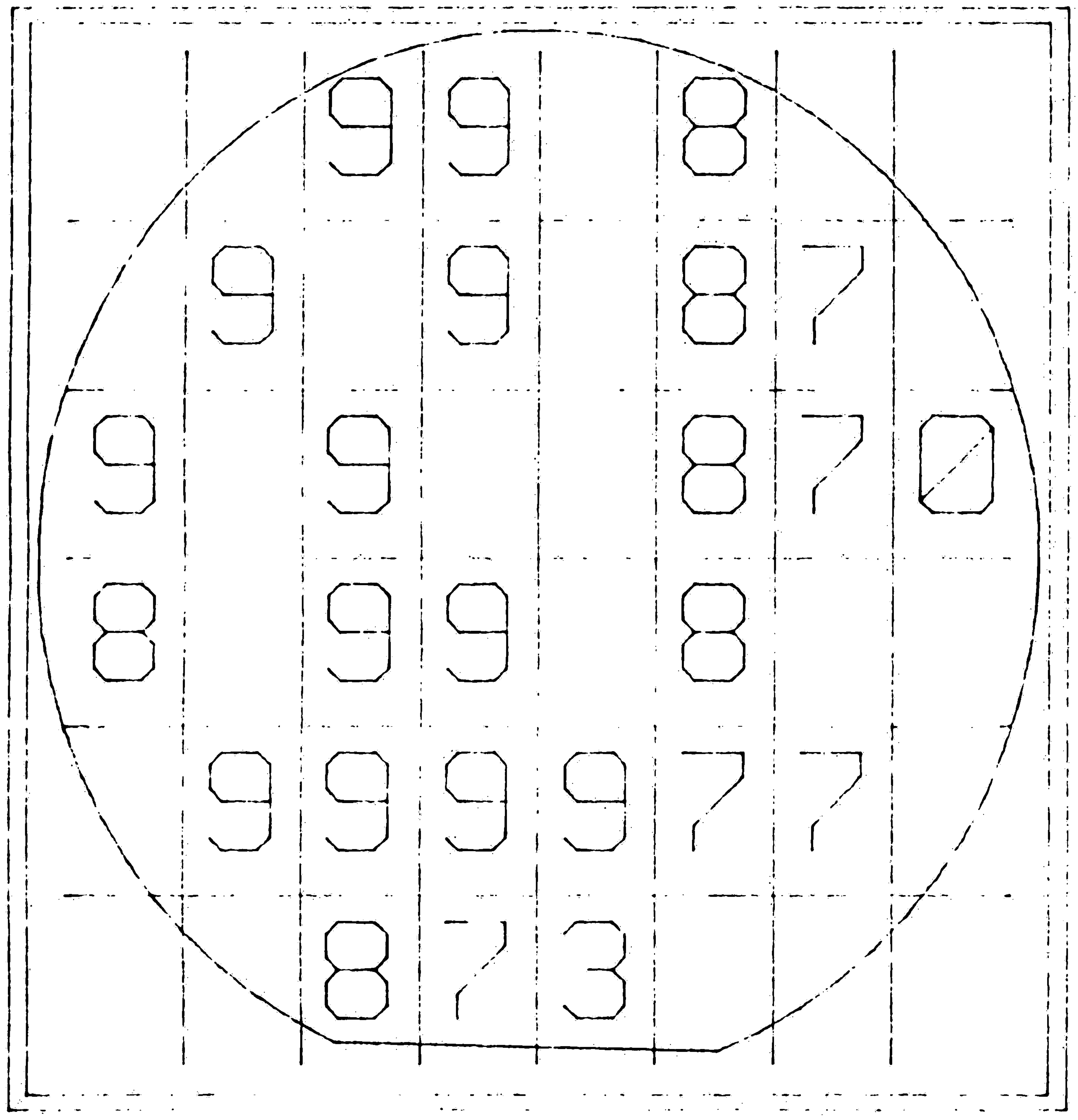
```
Wafer ID is TJK'S B2
User Name is Phill Goldman
Date and Time:  3:45 PM   Nov 14,1985

Parameter Mapped is VTO  (Volt)
Statistical Data for Current Map
--------------------------------

 Mean                           = -1.1676E+000
 Standard Deviation             = 2.02060E-001
 Value of parameter             = -1.1676E+000  +/-  5.71631E-002

 Minimum data value             = -1.9327E+000
        at location (7, 3)
 Maximum data value             = -1.0258E+000
        at location (3, 2)
```



Mean = -1.1676E+000 Volt
Standard Deviation = 2.02060E-001
Bar width = 1.00000E-001 Volt
Wafer ID = TJK'S B2

Figure A-7: Wafer B2 Threshold Voltage Distribution (VTO)

```
WAFER MAP    -- TECAP ver 1C.02
Date and Time:   3:43 PM    Nov 14,1985
Wafer Name:  TJK'S B2
User Name:   Phill Goldman
Type:        PMOS
```



```
Blank is unmeasured
0: >-2.0000E+000 Volt
1: >-1.9000E+000 Volt
2: >-1.8000E+000 Volt
3: >-1.7000E+000 Volt
4: >-1.6000E+000 Volt
5: >-1.5000E+000 Volt
6: >-1.4000E+000 Volt
7: >-1.3000E+000 Volt
8: >-1.2000E+000 Volt
9: >-1.1000E+000 Volt
```

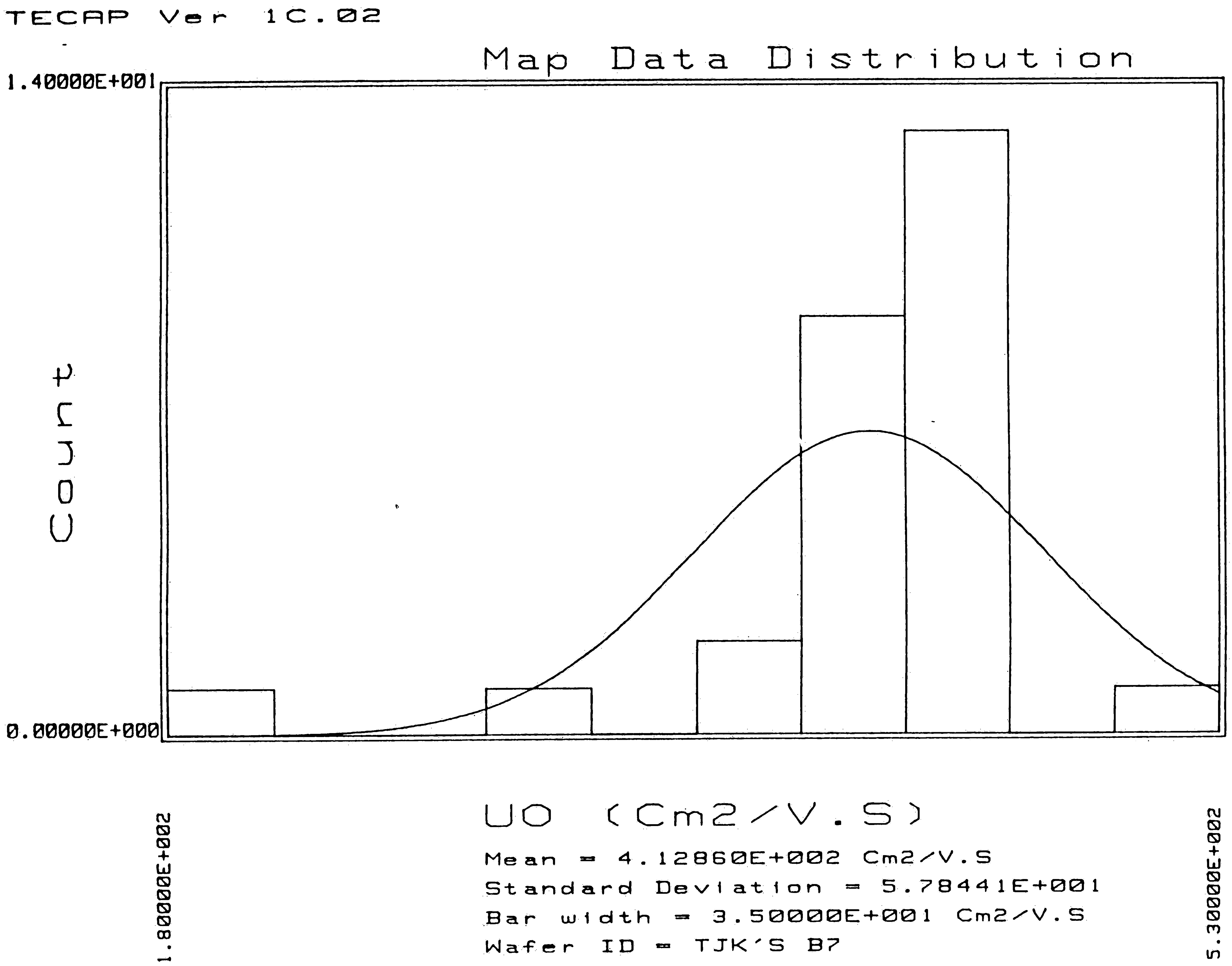
```
Parameter mapped is VTO (Volt)
```

Figure A-8: Wafer B2 Threshold Voltage Map (VTO)

Table A-5: Wafer B7 Bulk Mobility Data and Statistics (U0)

```
Print Stored Map Data
---------------------

Wafer ID is TJK'S B7
User Name is Phill Goldman
Date and Time:  3:54 PM   Nov 14,1985

Active Model is HPSPICE-MOS
Parameter Mapped is U0  (Cm2/V.S)

----------------------------------------------
! X position      Y position          Value !
----------------------------------------------
      0               0           Not set
      0               1           Not set
      0               2           Not set
      0               3           4.08614E+002
      0               4           Not set
      0               5           Not set
      1               0           Not set
      1               1           4.21298E+002
      1               2           4.34238E+002
      1               3           4.39365E+002
      1               4           4.41613E+002
      1               5           Not set
      2               0           Not set
      2               1           Not set
      2               2           4.45406E+002
      2               3           4.46918E+002
      2               4           4.53898E+002
      2               5           4.32993E+002
      3               0           Not set
      3               1           4.35249E+002
      3               2           1.89080E+002
      3               3           4.34531E+002
      3               4           4.39957E+002
      3               5           Not set
      4               0           5.28021E+002
      4               1           4.20328E+002
      4               2           Not set
      4               3           4.33929E+002
      4               4           3.97774E+002
      4               5           Not set
      5               0           3.61165E+002
      5               1           3.98305E+002
      5               2           3.98060E+002
      5               3           4.26154E+002
      5               4           Not set
      5               5           Not set
      6               0           Not set
      6               1           3.76719E+002
      6               2           4.06579E+002
      6               3           4.18280E+002
      6               4           4.32403E+002
      6               5           Not set
      7               0           Not set
      7               1           Not set
      7               2           3.13858E+002
      7               3           4.12494E+002
      7               4           Not set
      7               5           Not set
```

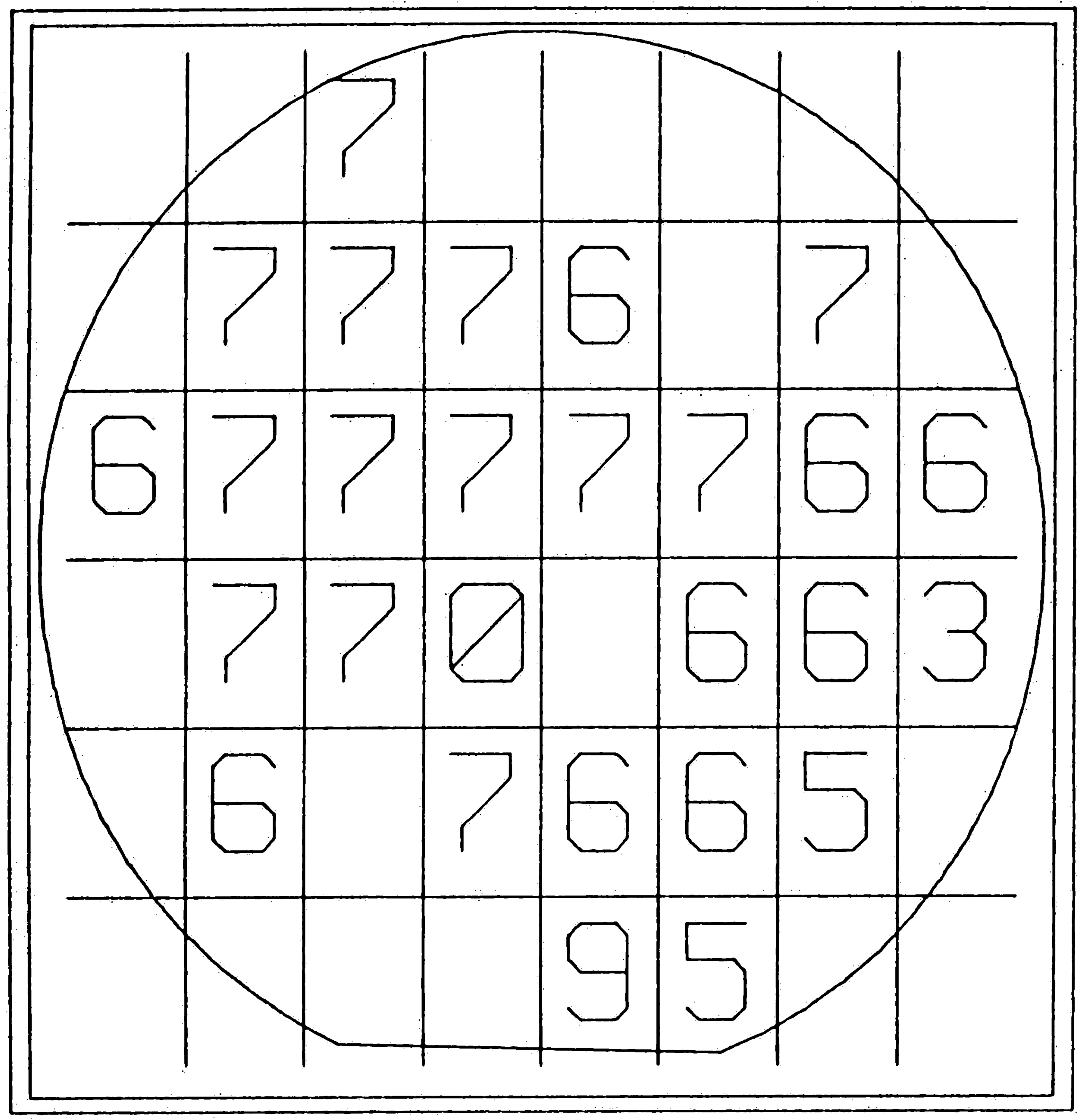
```
Wafer ID is TJK'S B7
User Name is Phill Goldman
Date and Time:  3:56 PM   Nov 14,1985

Parameter Mapped is U0  (Cm2/V.S)
Statistical Data for Current Map
--------------------------------

 Mean                           = 4.12860E+002
 Standard Deviation             = 5.78441E+001
 Value of parameter             = 4.12860E+002  +/-   1.63642E+001

 Minimum data value             = 1.89080E+002
       at location (3, 2)
 Maximum data value             = 5.28021E+002
       at location (4, 0)
```



Mean = 4.12860E+002 Cm2/V.S
Standard Deviation = 5.78441E+001
Bar width = 3.50000E+001 Cm2/V.S
Wafer ID = TJK'S B7

Figure A-9: Wafer B7 Bulk Mobility Distribution (U0)

```
WAFER MAP   - TECAP ver 1C.02
Date and Time:   3:52 PM    Nov 14,1985
Wafer Name:  TJK'S B7
User Name:   Phill Goldman
Type:        PMOS
```



```
    7
 7 7 7 6   7
6 7 7 7 7 7 6 6
 7 7 0   6 6 3
 6   7 6 6 5
       9 5
```

```
Blank is unmeasured
0: >1.80000E+002 Cm2/V.S
1: >2.15000E+002 Cm2/V.S
2: >2.50000E+002 Cm2/V.S
3: >2.85000E+002 Cm2/V.S
4: >3.20000E+002 Cm2/V.S
5: >3.55000E+002 Cm2/V.S
6: >3.90000E+002 Cm2/V.S
7: >4.25000E+002 Cm2/V.S
8: >4.60000E+002 Cm2/V.S
9: >4.95000E+002 Cm2/V.S
```

```
Parameter mapped is U0 (Cm2/V.S)
```

Figure A-10: Wafer B7 Bulk Mobility Map (U0)

Table A-6: Wafer B2 Bulk Mobility Data and Statistics (U0)

```
Print Stored Map Data
---------------------

Wafer ID is TJK'S B2
User Name is Phill Goldman
Date and Time:  3:31 PM   Nov 14,1985

Active Model is HPSPICE-MOS
Parameter Mapped is UO  (Cm2/V.S)

---------------------------------------------
| X position      Y position          Value |
---------------------------------------------
        0               0           Not set
        0               1           Not set
        0               2           2.90341E+002
        0               3           2.71099E+002
        0               4           Not set
        0               5           Not set
        1               0           Not set
        1               1           2.74116E+002
        1               2           Not set
        1               3           Not set
        1               4           2.76836E+002
        1               5           Not set
        2               0           2.61033E+002
        2               1           3.00049E+002
        2               2           2.88276E+002
        2               3           2.76225E+002
        2               4           Not set
        2               5           2.82259E+002
        3               0           2.57815E+002
        3               1           2.83016E+002
        3               2           2.85366E+002
        3               3           Not set
        3               4           2.87149E+002
        3               5           2.80656E+002
        4               0           2.47706E+002
        4               1           2.87779E+002
        4               2           Not set
        4               3           Not set
        4               4           Not set
        4               5           Not set
        5               0           Not set
        5               1           2.67500E+002
        5               2           2.83002E+002
        5               3           2.81524E+002
        5               4           2.86797E+002
        5               5           2.75053E+002
        6               0           Not set
        6               1           2.22509E+002
        6               2           Not set
        6               3           2.40092E+002
        6               4           2.34404E+002
        6               5           Not set
        7               0           Not set
        7               1           Not set
        7               2           Not set
        7               3           1.39569E+002
        7               4           Not set
        7               5           Not set
```

```
Wafer ID is TJK'S B2
User Name is Phill Goldman
Date and Time:  3:31 PM   Nov 14,1985

Parameter Mapped is UO  (Cm2/V.S)
Statistical Data for Current Map
--------------------------------

 Mean                          = 2.67207E+002
 Standard Deviation            = 3.26661E+001
 Value of parameter            = 2.67207E+002  +/-  9.24128E+000

 Minimum data value            = 1.39569E+002
       at location (7, 3)
 Maximum data value            = 3.00049E+002
       at location (2, 1)
```

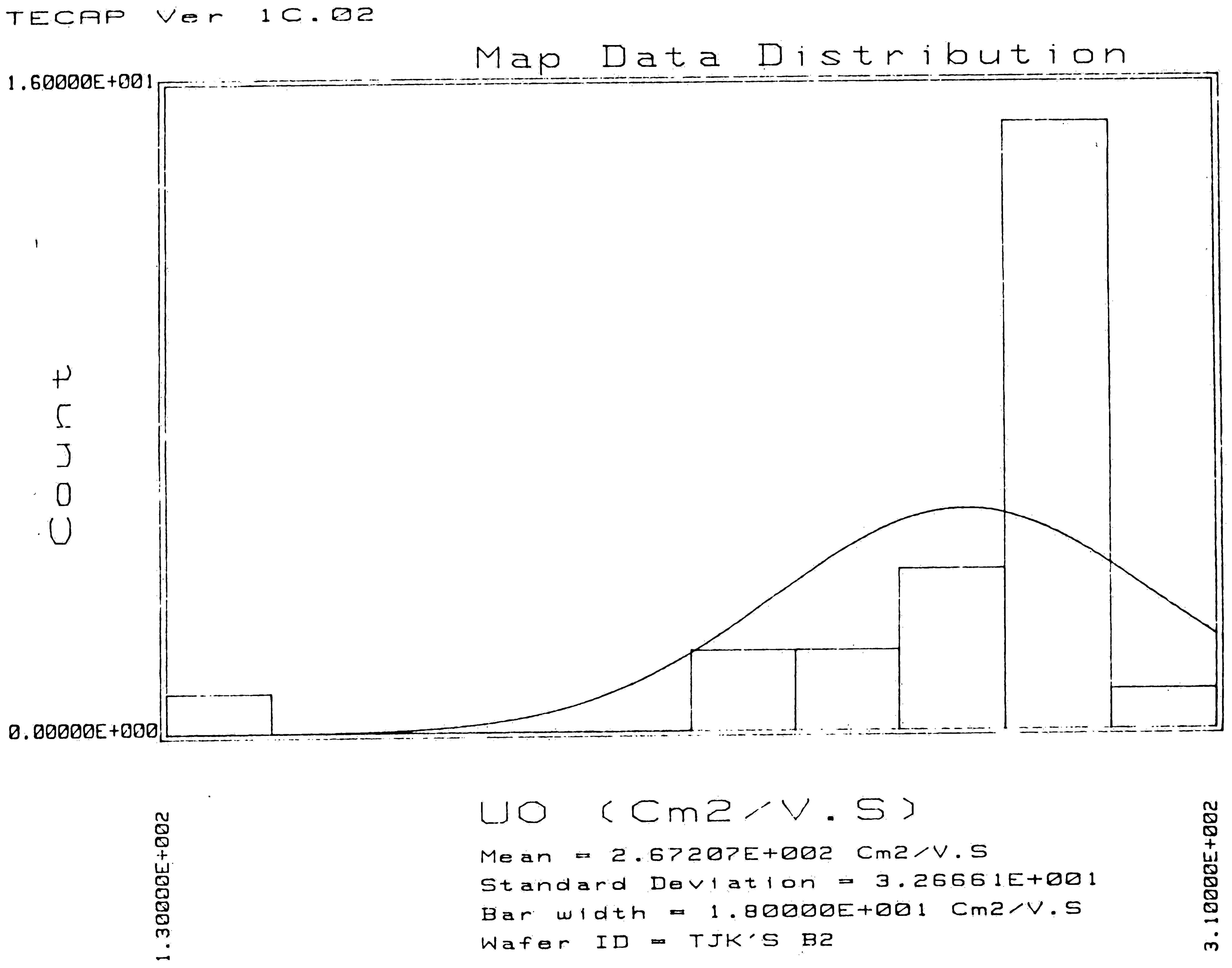
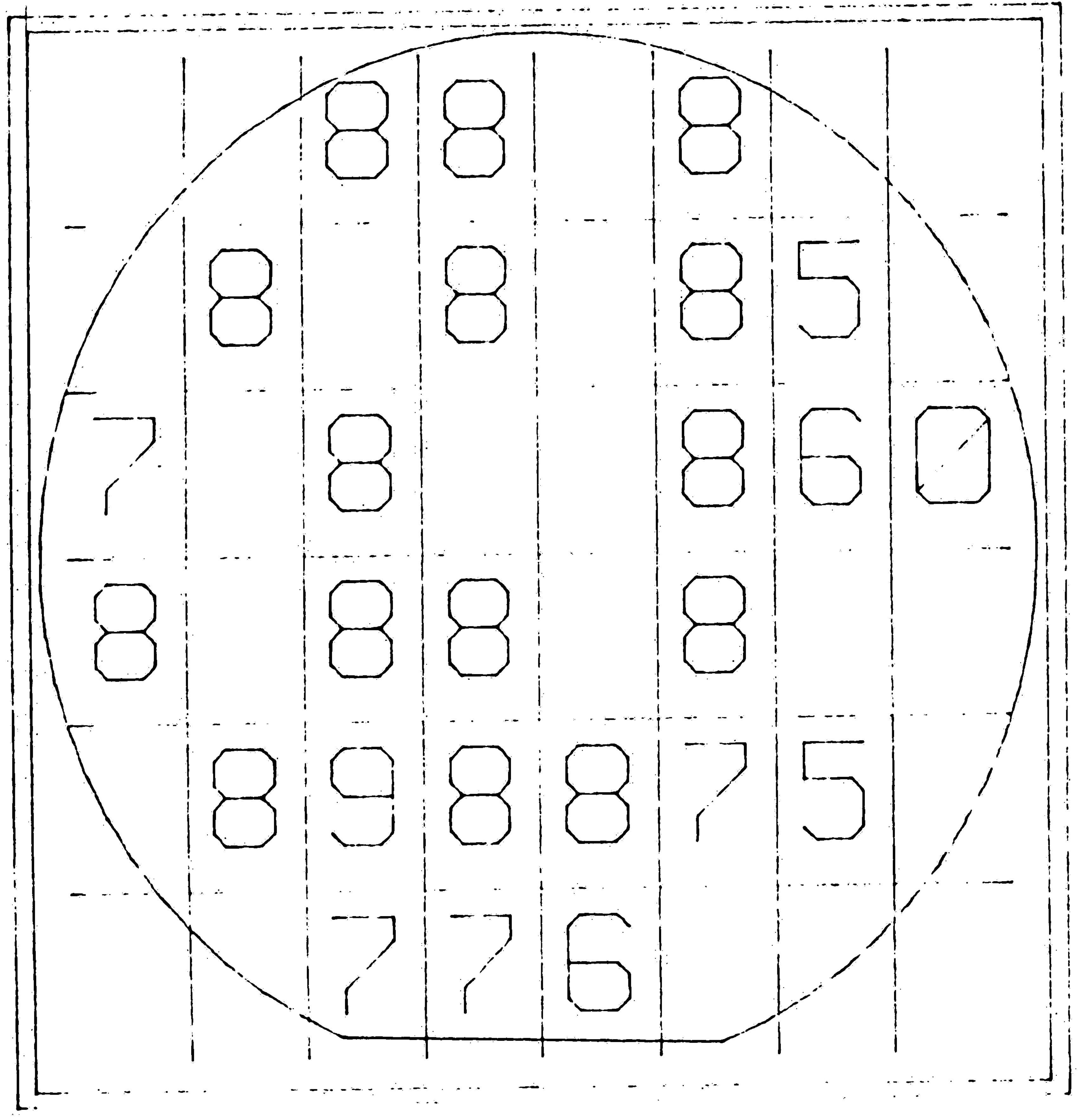


Figure A-11: Wafer B2 Bulk Mobility Distribution (U0)

```
WAFER MAP   - TECAP ver  1C.02
Date and Time:    3:40 PM    Nov 14,1985
Wafer Name:    TJK'S B2
User Name:     Phill Goldman
Type:          PMOS
```



```
Blank is unmeasured
0: >1.30000E+002 Cm2/V.S
1: >1.48000E+002 Cm2/V.S
2: >1.66000E+002 Cm2/V.S
3: >1.84000E+002 Cm2/V.S
4: >2.02000E+002 Cm2/V.S
5: >2.20000E+002 Cm2/V.S
6: >2.38000E+002 Cm2/V.S
7: >2.56000E+002 Cm2/V.S
8: >2.74000E+002 Cm2/V.S
9: >2.92000E+002 Cm2/V.S

Parameter mapped is UO (Cm2/V.S)
```

Figure A-12: Wafer B2 Bulk Mobility Map (U0)

Table A-7: Wafer B7 Channel Narrowing Data and Statistics (WD)

```
Print Stored Map Data
---------------------

Wafer ID is TJK'S B7
User Name is Phill Goldman
Date and Time:  4: 4 PM   Nov 14,1985

Active Model is HPSPICE-MOS
Parameter Mapped is WD  (Meter)

--------------------------------------------
! X position      Y position         Value !
--------------------------------------------
      0               0          Not set
      0               1          Not set
      0               2          8.27443E-006
      0               3          8.28141E-006
      0               4          Not set
      0               5          Not set
      1               0          Not set
      1               1          8.25268E-006
      1               2          8.17716E-006
      1               3          8.22408E-006
      1               4          8.21849E-006
      1               5          Not set
      2               0          8.28874E-006
      2               1          Not set
      2               2          8.15325E-006
      2               3          8.15888E-006
      2               4          8.10701E-006
      2               5          8.18065E-006
      3               0          8.33451E-006
      3               1          8.19754E-006
      3               2          8.18133E-006
      3               3          Not set
      3               4          8.23912E-006
      3               5          8.21089E-006
      4               0          8.48315E-006
      4               1          8.21161E-006
      4               2          8.21961E-006
      4               3          9.12941E-006
      4               4          8.21566E-006
      4               5          8.21059E-006
      5               0          8.48339E-006
      5               1          8.31589E-006
      5               2          8.27824E-006
      5               3          8.27202E-006
      5               4          8.29372E-006
      5               5          8.45251E-006
      6               0          Not set
      6               1          8.42883E-006
      6               2          Not set
      6               3          8.32612E-006
      6               4          Not set
      6               5          Not set
      7               0          Not set
      7               1          Not set
      7               2          8.66828E-006
      7               3          8.31791E-006
      7               4          Not set
      7               5          Not set
```

```
Wafer ID is TJK'S B7
User Name is Phill Goldman
Date and Time:  4: 7 PM   Nov 14,1985

Parameter Mapped is WD  (Meter)
Statistical Data for Current Map
--------------------------------

 Mean                          = 8.30585E-006
 Standard Deviation            = 1.90301E-007
 Value of parameter            = 8.30585E-006  +/-   5.38364E-008

 Minimum data value            = 8.10701E-006
        at location (2, 4)
 Maximum data value            = 9.12941E-006
        at location (4, 3)
```

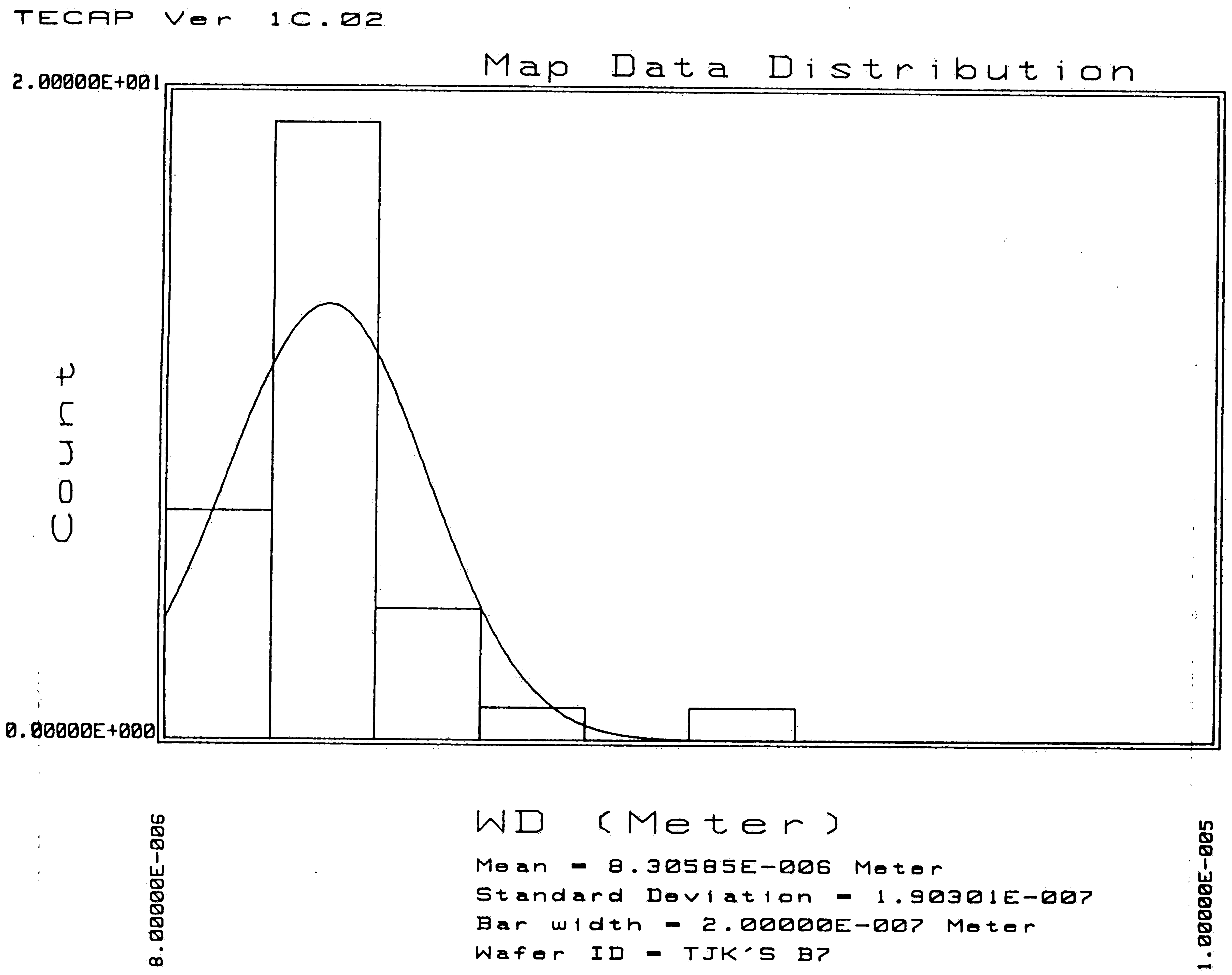
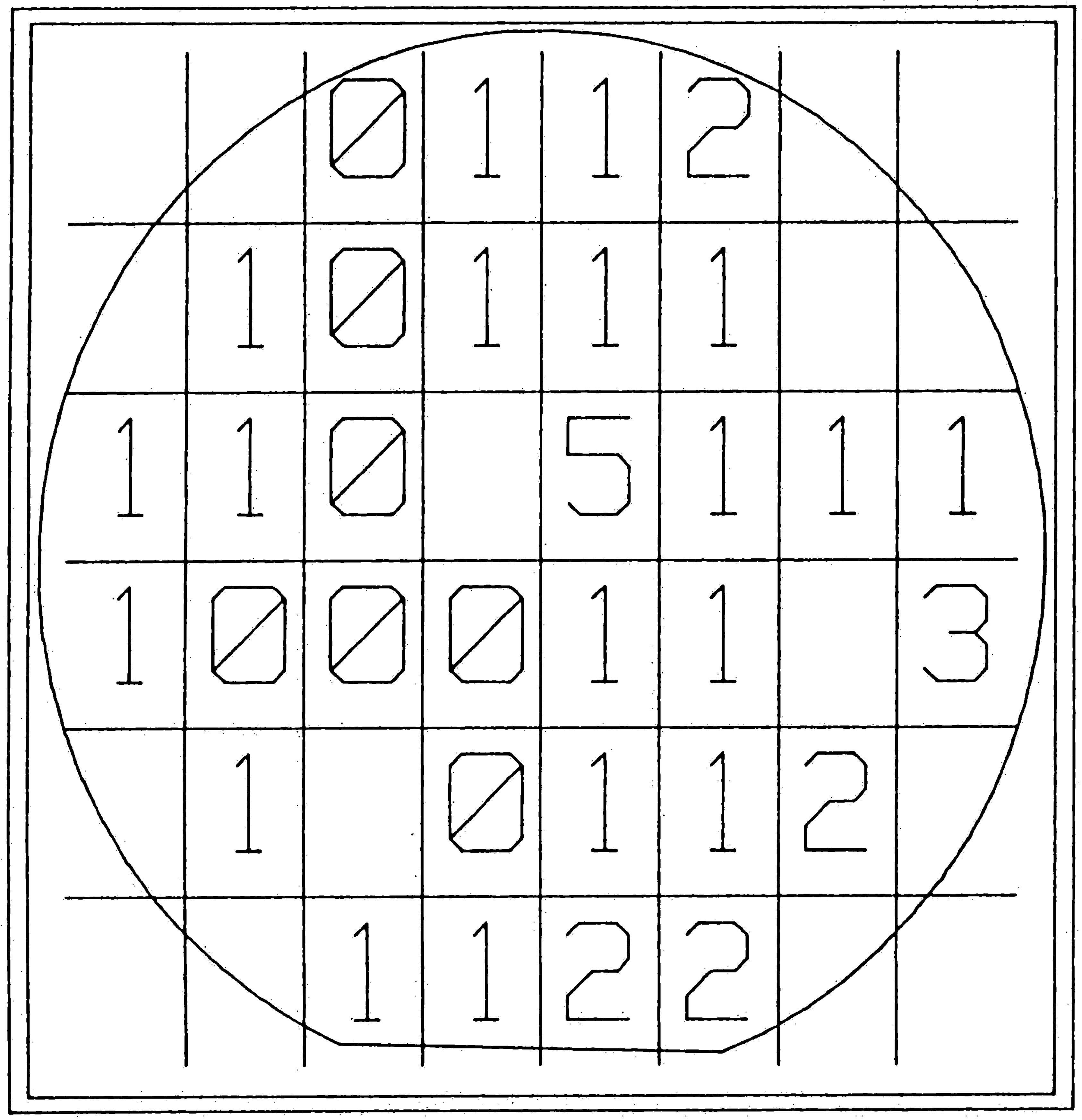


Figure A-13: Wafer B7 Channel Narrowing Distribution (WD)

WAFER MAP - TECAP ver 1C.02

Date and Time: 4: 2 PM Nov 14, 1985
Wafer Name: TJK'S B7
User Name: Phill Goldman
Type: PMOS



		0	1	1	2		
	1	0	1	1	1		
1	1	0		5	1	1	1
1	0	0	0	1	1		3
	1		0	1	1	2	
		1	1	2	2		

Blank is unmeasured
0: >8.00000E-006 Meter
1: >8.20000E-006 Meter
2: >8.40000E-006 Meter
3: >8.60000E-006 Meter
4: >8.80000E-006 Meter
5: >9.00000E-006 Meter
6: >9.20000E-006 Meter
7: >9.40000E-006 Meter
8: >9.60000E-006 Meter
9: >9.80000E-006 Meter

Parameter mapped is WD (Meter)

Figure A-14: Wafer B7 Channel Narrowing Map (WD)

Appendix B
C-V Station Program Ver 1.2 User's Manual

Updated: August 29, 1985

B.1 Introduction To C-V Version 1.2

This manual covers the operation and theory behind the updated C-V measurement procedure involving the HP85 computer and the MSI Electronics Programmable C-V meter. The program was written to allow the most common options used in these measurements to be the default while allowing menu driven selection of changes in these defaults.

The program provides the following features:

- Default parameters sufficient for many purposes.
- Menu-oriented parameter selection using the function keys.
- Audible feedback when the probe meets the device under test. This may help avoid damage caused by excessive pressure of probe on the device surface.
- Voltage limits are strictly set by the user to avoid the ugly problem of destroying samples with excess voltage.
- Up and Back voltage sweeps are implemented to check direction-sensitive devices.
- Bias / Temperature Stress is also implemented.
- Voltage range may be asymmetric.
- Data can be saved on tape for later calculation or comparison with new data. The Data Cartridge need not be left in the drive after start-up.

Part One of this manual is intended for the casual user who wishes to make a "fast measurement" without learning even the few details this program involves. Part Two is a more detailed and complete documentation of the procedures involved. Part Three explains the mathematics behind the calculations. The final section contains a complete program listing.

****** NOTE: DO NOT CHANGE THE C-V PROGRAM WITHOUT BACKING IT UP!!!!! THERE ARE NO OTHER COPIES OTHER THAN ON THE TAPE "C-V DEVELOPMENT" ******

Release Information:

Date	Number	Fix(es)
8408.17	1.0	Initial Code
8411.29	1.1	Fix errors in math, add print option
8508.29	1.2	BTS Fix, Manual recover in Cfb

B.2 Quickee User's Guide

This is a quick guide for using the C-V program. For a more detailed guide, see Part Two of the C-V manual.

The steps for execution are as follows:

Locate the tape labelled "C-V Development" and place it in the tape slot of the HP85 and turn on the computer and the MSI Programmable C-V Meter. Press the RESET button on the Meter.

The computer will then take several minutes to load the necessary routines to do the measurements. When the system is done it will ask you to press <K1> to do measurements. You may remove the program cartridge from the slot now if you wish.

Press <K1> to continue the program. Next, select what you wish to do; in the case of a new sample, press MEASURE.

The default selections will appear in menu form on the screen. If these are acceptable, press Measure again and the analysis will begin.

Probe the device on the stand and press <K1> when you are satisfied with the contact. The uncompensated capacitance measurement will be displayed during the probing time.

When the measurement is finished, press ANALYZE to do the calculations on the current data. Again, the default parameters are displayed and can be editted. Press GO to plot and calculate the data, press PRINT to print out the Voltage/Capacitance data.

B.3 Detailed Operation Guide

This C-V program was written to be as largely self-explanatory as possible, so this section of the manual will largely be a complete application example. The steps here are true of most of the measurements.

First, all of the hardware must be turned on and initialized. To do this , insert the tape labelled "C-V DEVELOPMENT" into the tape slot and turn on the HP85. While the program is loading into memory, turn on the MSI Electronics C-V Meter, the Temperature Controller, and the small gray vacuum pump behind the probe box. Be sure the pump is connected to the probe box. Press RESET on the C-V meter, then press COOL on the temperature controller. The equipment is now ready for use.

When first powered up the computer screen will show

```
      C-V STATION
      MEASUREMENT
        Ver 1.2
    Revised 08/29/85

     Please wait...
```

Then the screen will blank while the necessary routines are loaded from tape to the internal program memory. When the loading is complete, the screen will show

```
             Select Function

  <K1> C-V Measurement



  ----------------------------------
                               EXIT
  C-V
```

At this point the tape is no longer necessary, and it may be removed from the tape slot, and you may insert your own tape to save the data which you gather. Press <K1> to continue with the measurement. (NOTE: The other function keys are reserved for future use.)

After pressing <K1> and a few seconds pass, the screen will read

```
                C-V STATION
                MEASUREMENT

            Select Function with
            <k1> thru <k3>
            Exit with <k8>


----------------------------------
        ANALYZE          EXIT
MEASURE         FILER
```

In this example we will measure a sample one sweep, save the data, then go on and analyze it. Different measurements will follow the same pattern, with changes being made at the user menus.

First, press <K1> to go on to the measurement menu, which appears like this:

```
CURRENT PARAMTER SETTINGS

Channel Type       N-chan(P-Sub)
Sweep Direction    + to -
Measurement Type   One Sweep
Upper Voltage      10 volts
Lower Voltage     -10 volts
Ramp Rate (mV/s)   1000
Probe Type         A1

     <k1> to <k7> to set,
       <k8> to measure.

--------------------------------
LOWER-V RAMP RT PROBE   C-TYPE
Measure SWEEP   M-TYPE  UPPER-V
```

The parameters shown above are the default parameters set by the program at power-up, and may be sufficient for you to make a measurement. If you are willing to use them, simply press the Measure function key to make the test. Here though, we wish to make some changes. First, our device is on an N-type substrate, which means it is a P-channel device. Press <K8> to toggle the change in channel. Next, we wish to sweep from -5 volts to +5 volts. Pressing function keys <K5> and <K4> prompt us for the lower and upper voltages, which then appear in the menu. After our changes the menu appears as

```
        CURRENT PARAMTER SETTINGS

Channel Type        P-chan(N-Sub)
Sweep Direction     + to -
Measurement Type    One Sweep
Upper Voltage       5 volts
Lower Voltage      -5 volts
Ramp Rate (mV/s)    1000
Probe Type          A1

     <k1> to <k7> to set,
       <k8> to measure.

--------------------------------
LOWER-V RAMP RT PROBE   C-TYPE
Measure SWEEP   M-TYPE  UPPER-V
```

Now that we have the menu set for our measurements we press <K1> to make the test. After pressing the key the computer will ask if you really want to make this measurement. If "N" the system will return to the top menu in the C-V system. If "Y", the computer will instruct you to probe the device. Place your sample in the probe box and place the probes on the device under test. When contact is made the computer will BEEP to tell you that you are in contact. During the probing the system will display the uncompensated capacitance read from the probes. After you are satisfied with the probe contact, press <K1> and the measurement will be made.

During the measurement you should not interrupt the computer or bump the probe box. The measurements are time-sensitive in some cases and the probes most definitely are motion sensitive. While the measurement is being made a rough plot of the incoming data is displayed on the computer screen. This is a good indication of the quality of the data you have received. If it is not satisfactory, allow the computer to re-measure the sample with the same measurement parameters. NOTE: This will destroy the first set of data.

After measurement the data may be saved and/or analyzed. In this example, we will do both, first save it, then perform the calculations.

From the main menu now we press <K3> for FILER. This routine's only purpose is to transfer data between the tape and the program's internal data structure. The menu appears as

```
          DATA FILER (ON TAPE)
         (All work done w.r.t. data
          in memory and on tape)

      <1> Load from tape
      <2> Save to tape
      <3> Tape Catalog
      <4> End Filing


 -------------------------------

 LOAD      STORE   CATALOG END
```

We press <K2> for the Save operation. We will save the data under the name EXAMP for now. This will take quite a bit of time if you are used to the floppy disk systems on other computers, but it's the best this system has. We can check to see if the file is saved with <K3>, a tape catalog. Our data is now saved. Next, we analyze the data. Press <K4> to end the FILER routine, and press <K2>, for ANALYZE data.

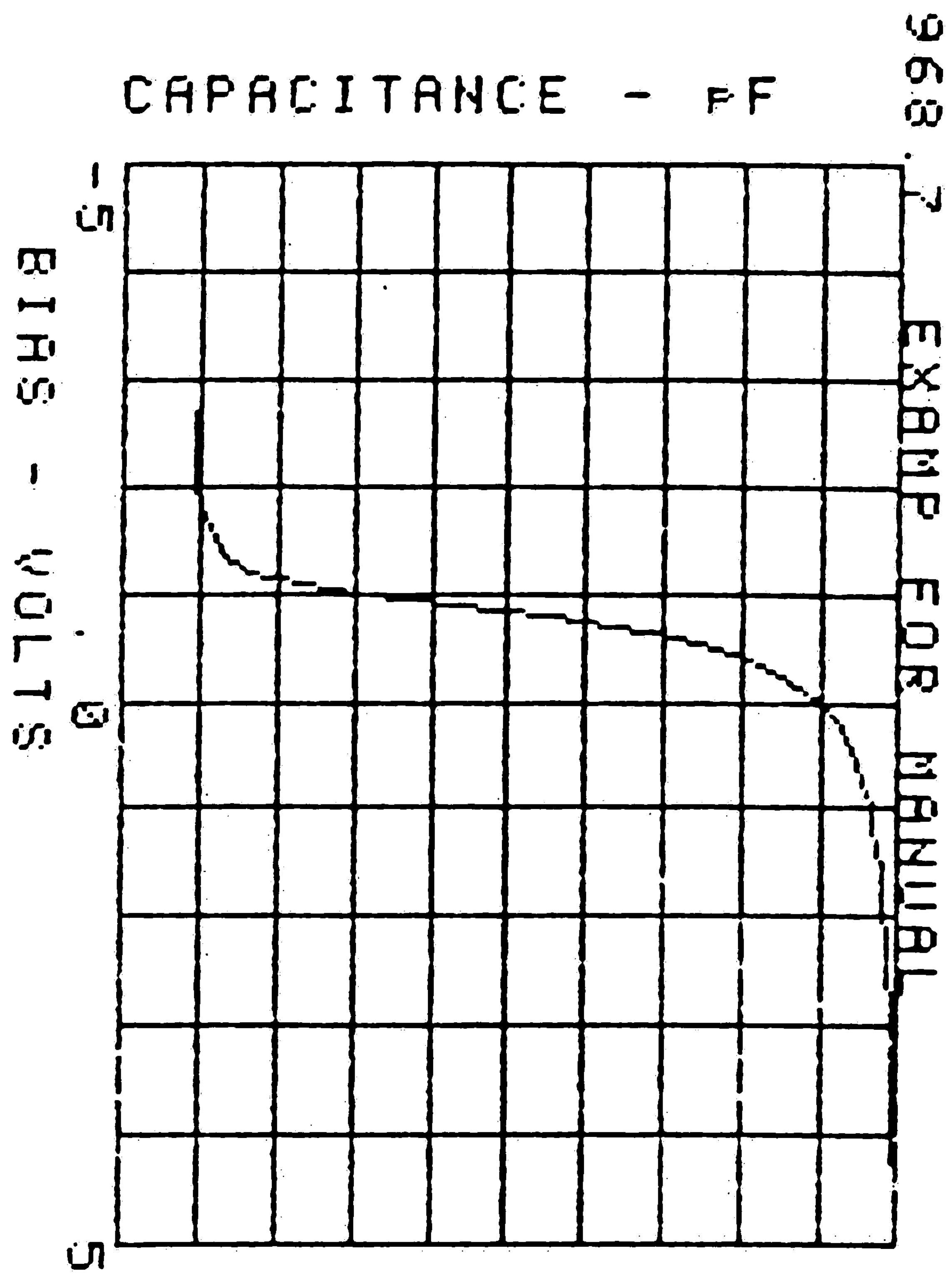
A startup menu for the analysis section will appear on the screen, like this

```
      C-V CALCULATIONS

Device Diameter is  1  mm.
Device Area is     .7854 mm^2
Curve# for calc is  1
Curve# for stress is 2
Type of calculation is One Sweep

--------------------------------
TYPE    PRINT   CURVE#  DIAM
GO      AREA    MENU    STRESS#
```

The default parameters shown are for the most commonly measured structure, the one millimeter dot on the surface in a one sweep calculation. Since this is exactly what we measured in EXAMP we will not change any parameters. If you had a different device, you could enter either the new diameter or the new area and the other parameter would be adjusted automatically for the calculation. The device type, i.e. N- or P- channel, is carried along with the measured or stored data and need not be entered. Press <K6> to print out the data actually measured, or <K1> to plot the data and analyze it. The order does not matter. Following is the sample output from our test measurement.

B.4 Sample Outputs

B.4.1 Plot and Results



```
EXAMP FOR MANUAL

Cox   =  968.7  pF
Xox   =  274   A
Nsub  = +9.307E+014 cm^-3
Cfb   =  606.44 pF
Vfb   =    -.71 volts
Nf    = +1.374E+011 cm^2
Vth   =  -1.284 volts
```

B.4.2 Printout of Actual Data

```
Voltage       Capacitance
-------       -----------
 -4.800        95.80pF
 -4.600        95.70pF
 -4.400        95.70pF
 -4.200        95.60pF
 -4.000        95.60pF
 -3.800        95.30pF
 -3.600        95.00pF
 -3.400        94.90pF
 -3.200        94.50pF
 -3.000        94.10pF
 -2.800        93.40pF
 -2.600        92.30pF
 -2.400        91.00pF
 -2.200        91.20pF
 -2.000        93.00pF
 -1.800        98.00pF
 -1.600       106.40pF
 -1.400       120.40pF
 -1.200       155.30pF
 -1.000       298.70pF
   -.800      531.50pF
   -.600      696.30pF
   -.400      784.60pF
   -.200      836.20pF
 +0.000       869.20pF
   +.200      892.30pF
   +.400      908.00pF
   +.600      919.80pF
   +.800      927.40pF
 +1.000       935.40pF
 +1.200       939.70pF
 +1.400       943.30pF
 +1.600       948.30pF
 +1.800       951.00pF
 +2.000       953.00pF
 +2.200       955.10pF
 +2.400       957.10pF
 +2.600       958.80pF
 +2.800       960.90pF
 +3.000       961.10pF
 +3.200       961.60pF
 +3.400       962.30pF
 +3.600       963.20pF
 +3.800       964.40pF
 +4.000       965.60pF
 +4.200       966.00pF
 +4.400       967.10pF
 +4.600       967.80pF
 +4.800       968.70pF
 +5.000       967.80pF
```

B.5 Equations Used in C-V

The equations on this page were used in the calculation of the sample parameters in the analyze section of the program. Given are the theoretical expression and how it appears in the program line. [3]

C_{ox} = *maximum (oxide) capacitance*

```
2640 C(C5,52)
```

$$X_{ox}=\frac{\epsilon_o\epsilon_{si}d^2}{C_{ox}}$$

```
2660 x1=507100000*KO*D1*D1/C(C5,52)
```

$$N_{sub}=\frac{4\phi_f}{q\epsilon_o\epsilon_{si}}(\frac{C_{smin}}{A})^2$$

where

$$\phi_f=\pm\frac{kT}{q}\ln(\frac{N_{sub}}{N_o})\quad -=\text{p-type},\ +=\text{n-type}$$

```
2680-2750  Uses the Secant rule
           (modified Newton-Raphson) to find
           the solution. [2]
```

$$C_{fb}=\frac{C_{ox}C_{sfb}}{C_{ox}+C_{sfb}}$$

where

$$C_{sfb}=\frac{\sqrt{2}A\epsilon_o\epsilon_{si}}{\lambda}$$

and

$$\lambda=\frac{\sqrt{2kT\epsilon_o\epsilon_{si}}}{\sqrt{q^2N_{sub}}}$$

V_{fb} *is found by interpolation.*

2800-2820 L8=λ
C9=C_{sfb}
C8=C_{fb}
2850-2890 F=V_{fb}

$$N_f=\frac{Q_{ss}}{q}=\frac{C_{ox}}{Aq}|\Phi_{MS}-V_{fb}|$$

where

$$\Phi=-.6-\phi_f$$

```
2920  Q1=C(C5,52)/1E12/(A/100)/1.6E-19*ABS(.6-F4-F)
```

$$V_{th}=V_{fb}+(2\phi_f-\frac{AQ_b}{C_{ox}})$$

where

$$Q_b=\pm qN_{sub}\frac{\epsilon_o\epsilon_{si}A}{C_{ox}} \quad -=n-type, \ +=p-type$$

```
2960-2980   V4=F+(2*F4-A/100*Q2/(C(C5,52)*1E-12)
```

$$N_m=\frac{1.2e-6\times C_{ox}\times|dV_{fb}|}{d^2}$$

```
2580   12.E-6*C(C5,52)*ABS(F2-F3)/D1/D1
```

B.6 Program Listings

```
 10 REM ***********************
 20 REM **  C-V STATION MEAS- **
 30 REM **  UREMENT PROGRAM   **
 40 REM **                    **
 50 REM **   LEHIGH U. 1984   **
 60 REM ** AT FAIRCHILD LABS  **
 70 REM **         BY         **
 80 REM **     P. GOLDMAN     **
 90 REM ***********************
100 OPTION BASE 0
110 DIM B(3,52),B2(3,50)
120 COM C(3,52),V(3,50),S0
130 V0=-10 @ V1=10 @ R=1000 @ C0
    =0 @ P0=0 @ D0=0 @ T0=300
140 D=1 @ T1=0
150 B1=10
160 V4$="(none)"
170 CLEAR @ GCLEAR
180 DISP "         C-V STATION"
190 DISP "         ";HGL$("MEASURE
    MENT")
200 DISP @ DISP @ DISP "      Sel
    ect Function with"
210 DISP "     <k1> thru <k3>"
220 DISP "     Exit with <k8>"
230 ON KEY# 1,"MEASURE" GOTO 320
240 ON KEY# 6,"ANALYZE" GOTO 330
250 ON KEY# 2 GOTO 330
260 ON KEY# 3,"FILER" GOTO 360
270 ON KEY# 4 GOTO 340
280 ON KEY# 8,"EXIT" GOTO 340
290 OFF KEY# 5 @ OFF KEY# 7
300 KEY LABEL
310 GOTO 230
320 GOTO 890
330 GOTO 1930
340 CLEAR @ MASS STORAGE IS ":T"
350 DISP @ DISP "      Program ru
    n complete." @ PAUSE
360 REM ***********************
370 REM **   FILER SEGMENT   **
380 REM ***********************
390 MASS STORAGE IS ":T"
400 ALPHA 1 @ CLEAR
410 DISP "    ";HGL$("DATA FILER
     (ON TAPE)")
420 DISP "   (All work done w.r.
    t. data" @ DISP "    in memo
    ry and on tape)" @ DISP
430 DISP "<1> Load from tape"
440 DISP "<2> Save to tape"
450 DISP "<3> Tape Catalog"
460 DISP "<4> End filing"
470 ON KEY# 1,"LOAD" GOTO 570
480 ON KEY# 2,"STORE" GOTO 690
490 ON KEY# 3,"CATALOG" GOTO 790
500 ON KEY# 4,"END" GOTO 670
510 ON KEY# 8,"" GOTO 470
520 ON KEY# 6,"" GOTO 470
530 OFF KEY# 5 @ OFF KEY# 7
540 KEY LABEL
550 GOTO 470
560 REM ----------------------
570 CLEAR @ DISP HGL$("Loading D
    ata..") @ DISP
580 DISP "What TAPE file to load
    " @ INPUT F$
590 ASSIGN# 1 TO F$
600 READ# 1 ; V0,V1
610 READ# 1 ; C0
620 READ# 1 ; C(,)
630 READ# 1 ; V(,)
640 ASSIGN# 1 TO *
650 CLEAR @ GOTO 360
660 REM ----------------------
670 MASS STORAGE IS ".ED" @ GOTO
     170
680 REM ----------------------
690 CLEAR @ DISP HGL$("Saving da
    ta..")
700 DISP @ DISP "What data file"
     @ INPUT F$
710 CREATE F$,510,8
720 ASSIGN# 1 TO F$
730 PRINT# 1 ; V0,V1
740 PRINT# 1 ; C0
750 PRINT# 1 ; C(,)
760 PRINT# 1 ; V(,)
770 ASSIGN# 1 TO *
780 GOTO 360
790 CLEAR @ CAT ":T"
800 DISP @ DISP "Press <END LINE
    > to go on"
810 INPUT A$
820 GOTO 360
830 REM ----------------------
840 REM ***********************
850 REM **   MEASUREMENT SEG **
860 REM ***********************
870 REM
880 DIM P$[8],D$[18],T$[27],C$[4
    0]
890 P$="AlumMerc"
900 D$="+ TO -- TO +      "
910 T$="One SweepUp & BackB - T
    - S"
920 C$="N-chan(P-Sub)P-chan(N-Su
    b)Unknown Type "
930 L0$="         "
940 ALPHA 1 @ CLEAR @ GCLEAR
950 DISP " ";HGL$("CURRENT PARME
    TER SETTINGS")
960 DISP @ GOTO 1040
970 ALPHA 3,1 @ DISP "Channel Ty
    pe      ";C$[C0*13+1,C0*13+1
    3] @ RETURN
980 ALPHA 4,1 @ DISP "Sweep Dire
    ction   ";D$[D0*6+1,D0*6+6]
    @ RETURN
990 ALPHA 5,1 @ DISP "Measuremen
    t Type  ";T$[T1*9+1,T1*9+9]
    @ RETURN
```

```
1000 ALPHA 9,1 @ DISP "Probe Typ
     e        ";P$[P0*4+1,P0*4+4
     ] @ RETURN
1010 ALPHA 8,1 @ DISP "Ramp Rate
      (mV/s)  ";R @ RETURN
1020 ALPHA 7,1 @ DISP "Lower Vol
     tage     ";V0;" volts" @ RE
     TURN
1030 ALPHA 6,1 @ DISP "Upper Vol
     tage     ";V1;" volts" @ RE
     TURN
1040 GOSUB 970 @ GOSUB 980 @ GOS
     UB 990 @ GOSUB 1030 @ GOSUB
      1020 @ GOSUB 1010 @ GOSUB
     1000
1050 ALPHA 11,1 @ DISP "     <k1
     > to <k7> to set," @ DISP "
            <k8> to measure." @
     DISP
1060 REM ----------------------
1070 ON KEY# 8,"C-TYPE" GOSUB 11
     80
1080 ON KEY# 2,"SWEEP" GOSUB 119
     0
1090 ON KEY# 3,"M-TYPE" GOSUB 12
     00
1100 ON KEY# 4,"UPPER-V" GOSUB 1
     210
1110 ON KEY# 5,"LOWER-V" GOSUB 1
     230
1120 ON KEY# 6,"RAMP RT" GOSUB 1
     270
1130 ON KEY# 7,"PROBE" GOSUB 129
     0
1140 ON KEY# 1,"Measure" GOTO 13
     10
1150 KEY LABEL
1160 GOTO 1070
1170 REM ----------------------
1180 C0=(C0+1) MOD 3 @ GOSUB 970
      @ RETURN
1190 D0=(D0+1) MOD 3 @ GOSUB 980
      @ RETURN
1200 T1=(T1+1) MOD 3 @ GOSUB 990
      @ RETURN
1210 ALPHA 11,1 @ DISP @ DISP @
     ALPHA 11,1 @ DISP "What upp
     er voltage" @ INPUT V1
1220 GOSUB 1030 @ GOTO 1050
1230 ALPHA 11,1 @ DISP @ DISP @
     ALPHA 11,1 @ DISP "What low
     er voltage" @ INPUT V0
1240 GOSUB 1020 @ GOTO 1050
1250 RETURN
1260 ALPHA 8,19 @ DISP L0$ @ INP
     UT V0@ ALPHA 8,1 @ GOSUB 10
     20 @ RETURN
1270 ALPHA 11,1 @ DISP @ DISP @
     ALPHA 11,1 @ DISP "What ram
     p rate" @ INPUT R
1280 GOSUB 1010 @ GOTO 1050
1290 ALPHA 6,1 @ P0=(P0+1) MOD 2
      @ GOSUB 1000 @ RETURN
1300 REM ----------------------
1310 REM *** MEASURE IT HERE
1320 CLEAR
1330 DISP "Press <ENDLINE> to go
     " @ DISP "on, anything else
      and <ENDLINE> " @ DISP "to
      abort."
1340 INPUT A$@ IF A$<>"" THEN 17
     0
1350 IF P0=0 THEN S0=.8 ELSE S0=
     2.5
1360 ON T1+1 GOTO 1370,1440,1550
1370 REM *** SINGLE SWEEP ****
1380 P=0
1390 IF C0=D0 THEN L=1 ELSE L=0
1400 N=1
1410 GOSUB 1840
1420 GOTO 1870
1430 REM ----------------------
1440 REM up and back
1450 P=0
1460 IF C0=D0 THEN L=1 ELSE L=0
1470 N=1 @ GOSUB 1840
1480 D0=NOT D0
1490 L=0
1500 N=2 @ GOSUB 1840
1510 D0=NOT D0
1520 BEEP
1530 GOTO 1870
1540 REM ----------------------
1550 REM bias temp stress
1560 P=0
1570 CLEAR @ DISP "Measuring Vir
     gin curve"
1580 IF C0=D0 THEN L=1 ELSE L=0
1590 N=1
1600 GOSUB 1840
1610 DISP "Positive bias stress"
1620 CLEAR @ DISP "What stress v
     oltage?" @ DISP "(Default i
     s :";B1;") " @ INPUT B1
1630 IF B1=0 THEN B1=B0 ELSE B0=
     B1
1640 DISP "(Set temp, press <K1>
     )"
1650 ON KEY# 1 GOTO 1660 @ GOTO
     1650
1660 OUTPUT 714 ;"BS ";-B1 @ GOS
     UB 1730 ! cycle
1670 L=0
1680 N=2 @ GOSUB 1840
1690 DISP "Negative Bias stress"
1700 N=3 @ OUTPUT 714 ;"BS ";-B1
      @ GOSUB 1730 @ GOSUB 1840
1710 GOTO 1870
1720 REM ----------------------
1730 REM temperature cycle
```

```
1740 ALPHA 10,8 @ DISP HGL$("STA
     TUS")
1750 OUTPUT 714 ;"TC"
1760 OUTPUT 714 ;"TS"
1770 ENTER 714 ; A$
1780 ALPHA 11,8 @ DISP A$
1790 IF A$<>UPC$(A$) THEN DISP "
     SET CONTROLLER IN REMOTE" @
      GOTO 1750
1800 IF A$<>"LOW " THEN 1760
1810 DISP "Heating cycle done.."
1820 WAIT 50 @ RETURN
1830 REM ----------------------
1840 REM *** MEASURE IT ***
1850 IF D0=0 THEN CALL "AcqCV" (
      V1,V0,R,N,L,P ) ELSE CALL
     "AcqCV" ( V0,V1,R,N,L,P ) @
      RETURN
1860 RETURN
1870 CLEAR @ DISP "     Do you w
     ant to" @ DISP "   repeat a
      measurement" @ INPUT A$
1880 IF A$[1,1]="Y" THEN GOTO 94
     0 ELSE GOTO 170
1890 REM **********************
1900 REM ** MATH SECTION,    **
1910 REM ** FINDS PARAMS     **
1920 REM **********************
1930 DIM T6$[27]
1940 T6$="One SweepUp & BackB -
     T - S"
1950 K0=.338 @ K1=1.036E-12 @ N0
     =14000000000
1960 T0=300
1970 S=(V1-V0)/50
1980 CLEAR @ GCLEAR
1990 A=PI*(D/2)^2 @ C5=1 @ C6=2
     @ T9=0
2000 REM ----------------------
2010 ALPHA 1 @ CLEAR @ ALPHA 2,2
      @ DISP "     ";HGL$("C-V C
     ALCULATIONS")
2020 DISP
2030 ALPHA 4,1 @ DISP "Device Di
     ameter is ";D;" mm."
2040 IMAGE "Device Area is ",MDD
     .DDDD," mm^2"
2050 ALPHA 5,1 @ DISP USING 2040
      ; A
2060 ALPHA 6,1 @ DISP "Curve# fo
     r calc is ";C5
2070 ALPHA 7,1 @ DISP "Curve# fo
     r stress calc is ";C6
2080 ALPHA 8,1 @ DISP "Type of c
     alculation is ";T6$[1+T9*9,
     T9*9+9] @ DISP
2090 REM ----------------------
2100 ON KEY# 8,"DIAM" GOTO 2210
2110 ON KEY# 2,"AREA" GOTO 2230
2120 ON KEY# 7,"CURVE#" GOTO 224
     0
2130 ON KEY# 4,"STRESS#" GOTO 22
     70
2140 ON KEY# 5,"TYPE" GOTO 2300
2150 ON KEY# 1,"GO" GOTO 2310
2160 ON KEY# 3,"MENU" GOTO 170
2170 ON KEY# 6,"PRINT" GOTO 3050
2180 KEY LABEL
2190 GOTO 2100
2200 REM ----------------------
2210 ALPHA 10,3 @ DISP "New Devi
     ce diameter";@ INPUT D
2220 GOTO 1990
2230 ALPHA 10,3 @ DISP "What new
      area";@ INPUT A@ GOTO 2010
2240 ALPHA 10,3 @ DISP "What new
      curve #";@ INPUT C5
2250 IF C5>3 OR C5<1 THEN BEEP @
      GOTO 2240
2260 GOTO 2010
2270 ALPHA 10,3 @ DISP "What new
      stress curve #";@ INPUT C6
2280 IF C6>3 OR C6<1 THEN BEEP @
      GOTO 2270
2290 GOTO 2010
2300 T9=(T9+1) MOD 3 @ GOTO 2010
2310 CLEAR @ DISP "Title for Pri
     ntouts";@ INPUT V4$
2320 REM ----------------------
2330 P1=(V1-V0)/7 @ P2=C(C5,52)/
     7
2340 GCLEAR
2350 SCALE V0-P1,V1+P1,-P2,1.143
     *C(C5,52)
2360 FOR I=0 TO 10 @ MOVE V0,I*P
     2/10*7 @ DRAW V1,I*P2/10*7
     @ NEXT I
2370 FOR I=0 TO 10 @ MOVE I*P1/1
     0*7+V0,0 @ DRAW I*P1/10*7+V
     0,C(C5,52) @ NEXT I
2380 MOVE V0+P1,C(C5,52) @ LABEL
      V4$
2390 MOVE V0,-P2 @ LABEL "  BIAS
      - VOLTS"
2400 MOVE V0-P1/3,0 @ LDIR 90 @
     LABEL "CAPACITANCE - pF" @
     LDIR 0
2410 MOVE V0,-P2/2 @ LABEL VAL$(
     V0)
2420 MOVE 0,-P2/2 @ LABEL "0"
2430 MOVE V1,-P2/2 @ LABEL VAL$(
     V1)
2440 MOVE V0-P1,C(C5,52) @ LABEL
      VAL$(C(C5,52))
2450 PENUP
2460 FOR I=1 TO 50
2470 PLOT V(C5,I),C(C5,I)
2480 NEXT I
2490 PENUP
2500 IF T9=0 THEN 2540
2510 IF T9=1 THEN 2530
```

```
2520 FOR I=1 TO 50 @ PLOT V(C5,I
     ),C(C6,I) @ NEXT I @ GOTO 2
     540
2530 FOR I=1 TO 50 @ PLOT V(C5,5
     1-I),C(C6,I) @ NEXT I
2540 COPY @ REM ---------------
2550 ON T9+1 GOTO 2570,2580,2600
2560 DISP "ERROR T9" @ BEEP @ PA
     USE
2570 GOSUB 2650 @ GOTO 2010
2580 GOSUB 2650 @ F3=F @ C7=C5 @
      C5=C6 @ GOSUB 2650 @ C5=C7
      @ F2=F
2590 IF T9=1 THEN GOTO 2010 ELSE
      RETURN
2600 GOSUB 2580 @ PRINT "dVfB =
     ";F2-F3;" volts"
2610 N8=1200000*C(C5,52)*ABS(F2-
     F3)/D1/D1
2620 IMAGE "Nm   = ",D.DDDE," cm
     ^2"
2630 PRINT USING 2620 ; N8
2640 GOTO 2010
2650 REM ---------------------
2660 PRINT @ PRINT @ PRINT V4$
2670 PRINT @ PRINT "Cox  = ";C(C
     5,52);" pF"
2680 D1=D/25.4 @ A1=(SQR(A/PI)/2
     5.4)^2*PI
2690 X1=507100000*K0*D1*D1/C(C5,
     52)
2700 PRINT "Xox  = ";INT(X1);" A
     "
2710 B=T0*2.081E27 @ C=(C(C5,51)
     *.000000000001/A*100)^2
2720 N1=1.E21 @ N2=1.E20
2730 F1=N1-B*C*LOG(N1/N0)
2740 F2=N2-B*C*LOG(N2/N0)
2750 IF ABS(F2)<=.0001 THEN 2790
2760 N3=N2-F2*(N2-N1)/(F2-F1)
2770 N1=N2 @ N2=N3
2780 GOTO 2730
2790 REM ! N2 IS Nsub
2800 IMAGE "Nsub = ",SD.DDDE," c
     m^-3"
2810 PRINT USING 2800 ; N2
2820 F4=1.38E-23*T0/1.6E-19*LOG(
     N2/N0) @ ! F4 IS PHI SUB F
2830 L8=SQR(2*1.38E-23*T0*K1/(1.
     6E-19^2*N2))
2840 C9=SQR(2)*A*K1/100/L8 @ C8=
     C(C5,52)*C9/(C(C5,52)+C9)
2850 C8=C8*1.E12
2860 PRINT USING 2870 ; C8
2870 IMAGE "Cfb  = ",DDDD.DD," p
     F"
2880 FOR I=1 TO 49
2890 IF C(C5,I)>C8 AND C(C5,I+1)
     <C8 THEN 2920
2900 IF C(C5,I)<C8 AND C(C5,I+1)
     >C8 THEN 2920
2910 NEXT I @ PRINT "NO OCCURENC
     E OF Cfb IN ARRAY.." @ DISP
      "Press CONT." @ PAUSE @ GO
     TO 3110
2920 F=V(C5,I)+(V(C5,I+1)-V(C5,I
     ))*(C8-C(C5,I))/(C(C5,I+1)-
     C(C5,I))
2930 PRINT USING 2940 ; F
2940 IMAGE "Vfb  = ",SDD.DD," vo
     lts"
2950 Q1=C(C5,52)/1.E12/(A/100)/1
     .6E-19*ABS(-.6-F4-F)
2960 PRINT USING 2970 ; Q1
2970 IMAGE "Nf   = ",SD.DDDE," c
     m^2"
2980 Q2=1.6E-19*N2*K1*(A/100)/C(
     C5,52*.000000000001)
2990 IF C0=1 THEN F4=-F4
3000 IF C0=0 THEN Q2=-Q2
3010 V4=F+(2*F4-A/100*Q2/(C(C5,5
     2)*.000000000001))
3020 PRINT USING 3030 ; V4
3030 IMAGE "Vth  = ",SDD.DDD," v
     olts"
3040 RETURN
3050 C$="N-chan(P-Sub)P-chan(N-S
     ub)Unknown Type " @ PRINT @
      PRINT
3060 IMAGE "    ",SDD.DDD,"    "
     ,DDDD.DD,"pF"
3070 PRINT "Name:" @ PRINT V4$ @
      PRINT "Channel type is: ";
     C$[C0*13+1,C0*13+13]
3080 PRINT @ PRINT "    Voltage
         Capacitance" @ PRINT "
       --------    -----------"
3090 FOR I=1 TO 50 @ PRINT USING
      3060 ; V(C5,I),C(C5,I) @ N
     EXT I
3100 GOTO 2010
3110 ALPHA 1 @ CLEAR @ DISP HGL$
     ("Manual Recovery")
3120 DISP "Enter the correct par
     ms"
3130 ALPHA 6,1 @ DISP "Cmin
     = ";C(C5,51);" pF"
3140 ALPHA 7,1 @ DISP "Cox
     = ";C(C5,52);" pF"
3150 ON KEY# 1,"Go" GOTO 2670
3160 ON KEY# 2,"Cmin" GOTO 3220
3170 ON KEY# 3,"Cox" GOTO 3230
3180 ON KEY# 8,"Menu" GOTO 2000
3190 OFF KEY# 4 @ OFF KEY# 5 @ O
     FF KEY# 6 @ OFF KEY# 7
3200 KEY LABEL
3210 GOTO 3150
3220 ALPHA 10,3 @ DISP "What new
      Cmin (pF)" @ INPUT C(C5,51
     )@ GOTO 3110
3230 ALPHA 10,3 @ DISP "What new
      Cox (pF)" @ INPUT C(C5,52)
     @ GOTO 3110
```

Appendix C
TECAP2 Prober Control Update Manual - RK681 - for TECAP 1C.00

A Summary of Support for the RK681A Prober

To incorporate a new probe station under TECAP2 you, the end user may replace the existing module PROBE_DRIVER in LIB4.CODE with your own version of this module tailored to your specific probe station. One function and four subroutines from this module are called directly by TECAP2. Any other routines needed for operation may be included and need not be EXPORTed to TECAP. The routines needed are:

```
function P_STATUS_CHECK(bit_number: integer):boolean;
procedure P_UPCHUCK;
procedure P_DOWNCHUCK;
procedure P_ORIG( xvar, yvar: real);
procedure P_MOVE_RELATIVE(xvar, yvar: real);
```

These routines are generally very simple to write, but the interface to TECAP2 must take into account the fact that the only prober TECAP can handle is the Rucker and Kolls 1032. If your prober can act like the 1032 in its remote operation mode, then the routines are simple.

This document is a general description of what the prober driver routines are supposed to do and how the 1032 expects to be handled. Also please note that the textfile and codefile versions of PROBE_DRIVER for the RK681 are included on this disk as well as a stream file which will link a version of TECAP containing the new probe driver. This is an adaptation of the LINK_ANY stream file supplied with the standard release. This document is in file RK681_DOC.TEXT on the same disk, volume label DAT681:

C.1 Routines Needed To Add An Unsupported Prober To TECAP

The routines needed for the operation of a prober are:

FUNCTION P_STATUS_CHECK(BIT_NUMBER: INTEGER):BOOLEAN;

This function returns the condition of the prober to TECAP. It is only called with the argument '3'. Normally this routine carries out a serial poll of the prober to determine the status. Bit 3 of the serial response is used to determine whether the START button on the prober had been pressed. The program simply waits for the value of the function to become true. If you have such a button, or have one you wish to use for this function, arrange for this routine to return the value FALSE until that button is pressed. The routine need not be a serial poll. If you wish to use the keyboard for input, then simply make this routine return TRUE always and use the PAUSE function from T_UTIL wherever you need it. This is called in the commands C12, C13, and C14.

PROCEDURE P_UPCHUCK;

This is a simple command; it instructs the prober to make contact with the wafer under test. The name UPCHUCK may be a bit confusing at first until you realize that the 1032 brought the chuck up to the probes, not the probes down to the chuck. Be careful how you use this one. It is also called in C12, C13, and C14.

PROCEDURE P_DOWNCHUCK;

By the same logic as above, the DOWNCHUCK routine instructs the prober to move the wafer away from the probes (or the probes away from the wafer). This routine is nice to have if your prober is not smart enough to break contact before moving, but is never called explicitly from TECAP.

PROCEDURE P_ORIG(XVAR, YVAR: REAL);

This routine does a bit more. It is used to define the home position to the prober. The values which come in through the arguments are the location (in microns) from the current location where the home location is to be set. TECAP really doesn't use the home reference since all of its moves are relative to the current probe position instead of a home position, but this routine provides a good opportunity to make sure the wafer is where it should be. Control of the prober can be released to the front panel here to allow for placement and alignment. Depending on how your prober chooses to use the home location, you may want to set it here.

PROCEDURE P_MOVE_RELATIVE(XVAR, YVAR: REAL);

Finally, the routine that does the most work. This routine moves the chuck in the x and y directions relative to the CURRENT PROBE LOCATION according to the arguments xvar and yvar. The arguments are in microns, so your procedure must convert them to centimeters, inches, steps, or whatever your prober needs. Convert them accurately, because any error will add up because of the relative move.

You will probably want to add a library of routines to format the commands you need for the system. As far as error handling goes, if you set the HPIB timeout before each bus use, the TECAP program will trap the bus errors and print out a message.

On this disk is a version of PROBE_DRIVER which will control the RK 681 prober (note its simplicity), and a stream file to install it. This is compiled under Pascal 2.0 and assumes that the TECAP libraries are on DATCP: and the finished program goes into TECAP:.

C.2 Prober Driver Routines for the RK681 Prober

```
{*************************************************************
 *     MODULE PROBE_DRIVER TO ADD SUPPORT FOR                 *
 *  RUCKER AND KOLLS 680 SERIES PROBER - WRITTEN FOR 681A     *
 *     Phillip M. Goldman, Lehigh University, March 1985      *
 **************************************************************

        The following file is the necessary routines for adding
   drivers for the Rucker and Kolls 680 series probers to
   TECAP2 Ver 1C.00.  This file replaces the module PROBE_DRIVER
   already supplied as part of LIB4.CODE in the 1C.00 release.
   It does not need to become part of the user module, however,
   it may be compiled with the user module. This is written and
   tested under Pascal 2.0.

    See the file RK681_DOC.TEXT on DAT681: for more detail. }


MODULE PROBE_DRIVER;

$SEARCH 'DATCP:LIB2' ,'*INTERFACE.' ,'*LIBRARY.'$

IMPORT
     iodeclarations,
     general_1,
     general_2,
     hpib_1,
     hpib_2,
     dgl_lib,
     tecap_data_base,
     tecap_utility;

EXPORT
     function p_status_check(bit_number: integer):boolean;
     procedure p_upchuck;
     procedure p_downchuck;
     procedure p_orig(xval, yval: real);
     procedure p_move_relative(xval, yval: real);

{============================================================}

IMPLEMENT


{ These first six routines are for the use of this module.
  They are never called directly from TECAP2 }


 procedure p_681_send_command( command: string_3);
 { This procedure sends the alphabetic command contained
   in 'command' to the prober at HPIB=7, RK_1032_ADD=7.
   string_3 is defined in the TECAP data base as type
   string[3] }

   begin
      set_timeout(hpib, 2.5);
      listen(hpib, rk_1032_add);
```

```
      talk(hpib, my_address(rk_1032_add));
      {send the first character}
      writechar(hpib, command[1]);
      {send the second if non-blank}
      if command[2] <> ' ' then
          writechar(hpib, command[2]);
   end;


procedure p_681_send_number(intnum : integer);
{This sends a number to the prober in HPIB format.  }

   begin
       set_timeout(hpib, 2.5);
       listen(hpib, rk_1032_add);
       talk(hpib, my_address(rk_1032_add));
       writeword(hpib, intnum);
   end;


procedure p_681_move_distance(x, y: integer);
{ This routine sends the command to move the chuck to
  the points given by x, y with respect to the currently
  set home position. The x and y values sent to the prober
  are negative to agree with the direction definitions
  of the 681 prober. }

     begin
         p_681_send_command('M  ');
         p_681_send_number(-x);
         p_681_send_number(-y);
     end;


procedure p_681_set_new_home;
{  This routine tells the printer to define the current
   location as the home location. All moves made are with
   respect to the current home position defined in this way.  }

   begin
      p_681_send_command('SH ');
   end;


procedure p_681_local;
{ This command enables the front panel controls of the prober
 to allow the chuck to be moved }

   begin
      set_timeout(hpib, 2.5);
      local(hpib*100+rk_1032_add);
   end;


procedure p_681_remote;
{  This command re-asserts the computer control of the
   prober via the bus }
```

```
    begin
       set_timeout(hpib, 2.5);
       remote(hpib*100+rk_1032_add);
    end;


{==========  ROUTINES THAT GO TO THE OUTSIDE WORLD ===========}


function p_status_check(bit_number: integer):boolean;
{ Returns the 'status' of the prober.  For the RK1032 this
  is used to find out if the user pressed the START button.
  Since the 681 doesn't have one,  waiting for it would be
  useless.  In the places necessary, a keyboard
  PAUSE in included }

  begin
      p_status_check := true;
  end;


procedure p_upchuck;
{  Commands the prober to make contact with the wafer.
   In the RK1032 the chuck rises to meet the probes,
   but the 681 probes move to meet the wafer, so the obvious
   meanings of UPCHUCK and DOWNCHUCK are reversed.  Pay it no
   attention. }

  begin
      p_681_send_command('D  ');
  end;


procedure p_downchuck;
{  Commands the prober to come off of the wafer }

  begin
      p_681_send_command('U  ');
  end;


procedure p_orig(xval, yval: real);
{ Allows the current position of the wafer to be defined.
  This is called in command C12) Define Position.   The 1032
  would allow TECAP to command it to move to the defined home
  position via the xval, yval parameters, but the 681 isn't that
  smart.  This allows the current location to be known. Since
  TECAP calculates all of the next moves relative to the current
  position, only the current position need be set in the prober.}

  begin
    writeln(output,
     '----- Place the probes on the position just specified. ----');
    writeln(output,
     '                    Use the prober front panel controls ');
    p_681_local;
    pause;
    p_681_set_new_home;
```

```
end;


procedure p_move_relative(xval, yval: real);
{  This routine moves the probes xval, yval microns from
   the current position. The current position is defined
   here for surety and the divide by 10's in the move is
   to take care of the different step sizes of the 1032
   and 681.  All moves are calculated by TECAP with respect
   to the current position and the 681 moves with respect
   to the home location, home is redefined before the move}

  begin
    p_681_remote;
    p_downchuck;
    p_681_set_new_home;
    p_681_move_distance( trunc(xval/10), trunc(yval/10));
  end;


end {MODULE PROBE_DRIVER for RK681 prober}.
```

C.3 File for Linking Prober to System

```
*******************************************************************
*                                                                 *
* TECAP2  MODULE  LINK               1C.00                        *
*       for RK681 support                                         *
*                                                                 *
*  Any of the following modules may be deleted in the object code.*
*                                                                 *
*    TECAP_CV        :  capacitance measurement                   *
*    TECAP_4145      :  HP 4145 drivers                           *
*    TECAP_MEASURE   :  measurement module                        *
*    TECAP_CV_MOD    :  capacitance models (PN-CAP & MOS-CAP)     *
*    TECAP_HPMOS     :  hpspice mosfet model                      *
* X  TECAP_MATRIX    :  connection matrix                         *
*    PRBDRV02        :  prober interface for 681                  *
*    TECAP_PROBER    :  prober sequence                           *
*    TECAP_HP_BJT    :  hpspice bipolar model                     *
*    TECAP_DIODE     :  hpspice diode model                       *
*    TECAP_SIMULATE  :  simulator                                 *
*    TECAP_OPTIMIZER :  optimizer                                 *
* X  T_USER          :  user defined commands                     *
*                                                                 *
*  Each module may be deleted by deleting the 't' character       *
*  on the next line after the module name. Modules with 'X' are   *
*  not included yet.                                              *
*                                                                 *
*******************************************************************
frTECAP:TECAP2.CODE
q
lh97
oTECAP:TECAP2
iDATCP:LIB1
a
iDATCP:LIB2
```

```
a
iDATCP:LIB3
mTECAP_CHECK
t
mTECAP_CV
t
mTECAP_4145
t
mTECAP_MEASURE
t
mTECAP_SETUP
t
mTECAP_CV_MOD
t
mTECAP_HP_MOS
t
iDATCP:LIB4
mTECAP_MATRIX

iDAT681:PROBE_681
mPROBE_DRIVER
t
iDATCP:LIB4
mTECAP_PROBER
t
iDATCP:T_USER
mUSER_MODULE

iDATCP:MAIN
mTECAP_HP_BJT
t
mTECAP_DIODE
t
mTECAP_SIMULATE
t
mTECAP_OPTIMIZER
t
mTECAP_CONTROL
t
mTECAP
t
kq
```

Appendix D
User Module Code

This appendix contains the complete listings for the user module extension to TECAP which allow the mapping of wafer parameters (all of the commands in the A menu of TECAP2).

```
$LINES 55$
$REF 45$

 {PRINT DATE : 8511.12}

 {******************** TECAP 2 USER'S MODULE ********************

      This module contains the user code to automate the full
wafer surface gathering of data, plotting the distribution of
that data, and saving and retrieving it.  It was written as part
of the Master's Thesis of Phillip Mark Goldman, MS EE Jan '86,
Lehigh University.

      Each procedure is commented rather thoroughly and should be
self-documenting.  More background on the justification and
results obatined by this routine may be found in the main body
of the Thesis document.  The models (3, 14, and 15) at the end
have not been modified by the author in any way.

      The author would like to thank Mr. Ebrahim Khalily and all
those at Hewlett-Packard who made this work possible through
donations of software, equipment, expertise, patience, and
understanding.

                      -Phillip M. Goldman
****************************************************************}
module  user_module;

 { ****** Thesis Mod. Version 1C.02  Date: 11-12-85 ****** }
 { ****** User Mod.  Version: 1C.01  Date:  4-10-84 ****** }
 { ******     IMR    Version: 1C.00  Date:  3-07-84 ****** }
 { ******     IMR    Version: 1B.00  Date:  7-28-83 ****** }
 { ****** Prototype  Version: 1A.00  Date:  5-03-83 ****** }

$search 'DATCP:LIB2','DATCP:LIB3','*INTERFACE.'$

import
    { Loads some of the more useful libraries provided with TECAP }

                           tecap_data_base
                         , tecap_utility
                         , tecap_plot
                         , tecap_4145
                         , tecap_4141
                         , tecap_cv
                         , tecap_measure
```

```
                          , tecap_check
                          , hpib_2
                          , iodeclarations
                          , dgl_lib ;


export

  {  Tell the world what is in this program segment }

  const
      map_x_size = 30;
      map_y_size = 30;


  type  menu_array = array[0..14] of string[20];

        {-------MAP DATA TYPES-------------}

    {  points to data for map in the heap - dynamic }
        map_ptr = ^map_data;
    {  data structure for the map itself:
                  - .data    contains the numbers
                  - .setflg  whether the data is valid   }
        map_data = record
                      data:array[0..map_x_size, 0..map_y_size] of
                            real;
                      setflg:array [0..map_x_size,0..map_y_size]
                            of boolean;
                   end;


  procedure user_init(var m : menu_array);
  procedure a0;
  procedure a1;
  procedure a2;
  procedure a3;
  procedure a4;
  procedure a5;
  procedure a6;
  procedure a7;
  procedure a8;
  procedure a9;
  procedure a10;
  procedure a11;
  procedure a12;
  procedure a13;
  procedure a14;
  procedure model_three(var p    : par_type;
                        var vpin : array8;
                        var ipin : array8;
                        var qpin : array8;
                        var gmat : array88;
                        var xmat : array88;
                        var nod  : arraynn;
                        var lim  : integer8;
                        var corner,termal : real;
                            infoflag,initflag,dflag,acflag,
```

```
                                areaflag : boolean);
  procedure model_fourteen (var p     : par_type;
                            var vpin : array8;
                            var ipin : array8;
                            var qpin : array8;
                            var gmat : array88;
                            var xmat : array88;
                            var nod  : arraynn;
                            var lim  : integer8;
                            var corner,termal : real;
                                infoflag,initflag,dflag,
                                acflag,areaflag : boolean);
  procedure model_fifteen (var p     : par_type;
                            var vpin : array8;
                            var ipin : array8;
                            var qpin : array8;
                            var gmat : array88;
                            var xmat : array88;
                            var nod  : arraynn;
                            var lim  : integer8;
                            var corner,termal : real;
                                infoflag,initflag,dflag,
                                acflag,areaflag : boolean);

 {______________________________________________}

implement

 const
     ver = '1C.02';

 var
     boltz, charge, ctok, eps0, epssil, epsox, pi, ref_temp,
         nom_temp, ref_vt, ref_eg, ref_ni: real;
 { has space been set aside? }
     map_allocated: boolean;
 { points to data (if any) in memory }
     map_array: map_ptr;
 { how big is the map? }
     map_x_actual, map_y_actual: integer;
 { what are we mapping? }
     map_par_num: integer;


 function intnum(i: integer): string_80;
 { -------------------------------------------------------
      Generates the string representation of an integer
      number.  There is no decimal point, and the string
      is of arbitrary length.
  -------------------------------------------------------}

     var
         ix: integer;
         s: string_80;

     begin
         strwrite(s, 1, ix, i);
```

```
        intnum := strltrim(s);
    end (* intnum *);


function realnum(r: real; n, m: integer): string_80;
{----------------------------------------------------------
      Makes a string representation of a real number,
      formatted like the FORTRAN Fn.m style. If a number
      cannot fit in that format, the default Pascal
      style is used.
----------------------------------------------------------}

    var
        ix: integer;
        s: string_80;

    begin
        if (abs(r) < ten_to(n - m)) and (abs(r) > - ten_to(n - m))
        then
            strwrite(s, 1, ix, r: n: m)
        else
            strwrite(s, 1, ix, r);
        realnum := strltrim(s);
    end (* realnum *);


function map_mean(datarray: map_data; nx, ny: integer): real;
{----------------------------------------------------------
    Calculate the mean of data in the map.  This procedure
    ignores the extra zeroes stored in data locations not
    yet explicitly set by the user.
----------------------------------------------------------}

    var
        ix, iy, count: integer;
        sum: real;

    begin
        sum := 0;
        count := 0;
        for ix := 0 to nx do
            for iy := 0 to ny do
                if datarray. setflg[ix, iy]
                then {the data has been set}
                    begin
                        sum := sum + datarray. data[ix, iy];
                        count := count + 1;
                    end (* then *);
        map_mean := sum / count;
    end (* function map_mean *);


function map_standard_deviation(datarray: map_data; nx, ny:
    integer): real;
{----------------------------------------------------------------
```

```
    Calculates the standard deviation of the data in the map,
    also ignoring the unused data in the array.
 --------------------------------------------------------------}

    var
        ix, iy, count: integer;
        s_squared, x_squared, sum_x: real;

    begin
        x_squared := 0;
        sum_x := 0;
        count := 0;
        for ix := 0 to nx do
            for iy := 0 to ny do
                if datarray. setflg[ix, iy]  then
                    begin {the data has been set here}
                        count := count + 1;
                        x_squared := x_squared + sqr(datarray.
                            data[ix, iy]);
                        sum_x := sum_x+datarray.data[ix,iy];
                    end (* then *);
        if count > 1
        then
            s_squared := (count*x_squared-sqr(sum_x))
                            /count/(count-1)
        else
            s_squared := 0.0;
        map_standard_deviation := sqrt(s_squared);
    end (* function map_standard_deviation *);


function round_down(x: real): real;
{----------------------------------------------------
   Rounds down the data; see full definition below.
 ---------------------------------------------------}
     forward;


function round_up(x: real): real;
{----------------------------------------------------------------
     Returns the number x rounded upward to two significant
     digits.
-----------------------------------------------------------------}

    var
        mant, expn, sign: real;

    begin
        if x < 0 then
           round_up := - round_down(-x)
        else
           if x > 0
              then  { then the number isn't zero }
                  begin
                      expn := trunc(log10(x));
                      mant := x / ten_to(expn - 1);
                      mant := trunc(mant + 1) / 10.0;
```

```
                    round_up := mant * ten_to(expn)
                end (* then *)
            else
                round_up := 0.0;
        end (* round_up *);


function round_down(x: real): real;
{-----------------------------------------------------------------
    Returns the number x rounded downward to two significant
    digits.
 -----------------------------------------------------------------}

     var
         mant, expn, sign: real;

     begin
         if x < 0 then
            round_down := - round_up(-x)
         else
            if x > 0
               then  { then the number isn't zero }
                   begin
                       expn := trunc(log10(x));
                       mant := x / ten_to(expn - 1);
                       mant := trunc(mant) / 10.0;
                       round_down := mant * ten_to(expn)
                   end (* then *)
               else
                   round_down := 0.0;
        end (* round_down *);



 procedure compose_name(inname: string_80; var final_name:
     string_80);
 { -----------------------------------------------------------------
     Assembles the filename for getting the map data file, using
     the defaults in the TECAP data base if needed.
  -----------------------------------------------------------------}

     begin
         final_name := '';
         if strpos(':', inname) = 0
         then  {volume not specified, use the default}
             final_name := volume_prefix;
         strappend(final_name, inname);
         if strpos('.', inname) = 0
         then  {extension not specified, add the TECAP standard}
             strappend(final_name, '.M');
     end (* compose_name *);


 procedure store_map_data;
```

```
{---------------------------------------------------------------
   Stores the data in the map in a character format with all
   of the necessary information.  Called from procedure A7,
   "Store/Fetch map"
 --------------------------------------------------------------}

    var
        datafile: text;                               {file for data}
        fname: string_80;                             {file name}
        ix, iy: integer;
        strbool: array [false .. true] of string_8;

    begin
     {these allow printing the literals for booleans}
        strbool[true] := 'TRUE';
        strbool[false] := 'FALSE';
        show_title('Storing Disk Data...');

     { Get file name from user and open it up }
        ask('Enter file name data storage', '', data_file_name,
            data_file_name);
        compose_name(data_file_name, fname);
        rewrite(datafile, fname);

     { Write the file header - will identify map files  }
        write(datafile, 'MEASURED file , TECAP2 : ', ver. 5);
        writeln(datafile, ' {MAP DATA FILE - ', dev. comment, '}');
        writeln(datafile, user.name);

     { Device type }
        writeln(datafile, ord(dev. typ));

     { Various pieces of information }
        writeln(datafile, dev. name);
        writeln(datafile, dev. wafer);
        writeln(datafile, dev. comment);
        writeln(datafile, dev. l, '     ', dev. w);
        writeln(datafile, dev. as, '     ', dev. ad, '     ', dev
            . ps, '     ', dev. pd);
        writeln(datafile, 0.0, 0.0);

     { Mapped model and parameter - the text is for the reader }
        writeln(datafile, activemodel, '     ', model_name[
            activemodel]);
        writeln(datafile, map_par_num, '     ', active_par. name[
            map_par_num]);

     { Map size as set }
        writeln(datafile, map_x_actual, '     ', map_y_actual);

     { Write the data out with flags }
        for ix := 0 to map_x_actual do
            for iy := 0 to map_y_actual do
                writeln(datafile, map_array^. data[ix, iy], ' ',
                    strbool[map_array^. setflg[ix, iy]]);

     { Finish up, all nice and neat }
        writeln(datafile, 'End_of_file');
```

```
        close(datafile, 'SAVE');
    end (* store_map_data *);


procedure fetch_map_data;
{----------------------------------------------------------------
  Loads the data from a (supposedly) map data file into the
  internal database for extension or analysis.  Also called
  from procedure A7 "Store/Fetch map"
 ---------------------------------------------------------------}

    var
        datafile: text;
        fname: string_80;
        ix, iy: integer;
        buffer: string[255];

    begin
        show_title('Retrieving Disk Data...');
     { Get the file name from user and open the file }
        ask('Enter file name containing map data', '',
            data_file_name, data_file_name);
        compose_name(data_file_name, fname);
        reset(datafile, fname);

     { Check to see if the header is there, identifying it
                as a real map data file }
        readln(datafile, buffer);
        if not ((str(buffer, 1, 8) = 'MEASURED') and (str(buffer,
            33, 3) = 'MAP'))
        then {oops, not map data }
            begin
                error(' Not a map data file ');
                pause;
            end (* then *)
        else

     { OK, we'll take it }
            begin

         { Read and store the device parameters }
                readln(datafile,user.name);
                readln(datafile, {dev.typ} ix);
                  case ix of
                      1: dev.typ := nmos;
                      2: dev.typ := pmos;
                      3: dev.typ := npn;
                      4: dev.typ := pnp;
                      5: dev.typ := njfet;
                      6: dev.typ := pjfet;
                      7: dev.typ := diode;
                      8: dev.typ := tube;
                      9: dev.typ := misx;
                  end (* case *);
                readln(datafile, dev. name);
                readln(datafile, dev. wafer);
                readln(datafile, dev. comment);
```

```
            readln(datafile, dev. l, dev. w);
            readln(datafile, dev. as, dev. ad, dev. ps, dev.
                pd);
            readln(datafile);
            readln(datafile, activemodel);
            readln(datafile, map_par_num);
            readln(datafile, map_x_actual, map_y_actual);


      { Read all of the numbers into the correct locations,
          taking care which are really set and which are not }
            for ix := 0 to map_x_actual do
                for iy := 0 to map_y_actual do
                    begin
                        readln(datafile, map_array^. data[ix,
                            iy], buffer);
                        buffer := strltrim(buffer);
                        { is the data previously set? }
                        map_array^. setflg[ix, iy] := (buffer
                            = 'TRUE');
                    end (* for *);

      { Check to see we got the whole file }
          readln(datafile, buffer);
            if buffer <> 'End_of_file'  then
                begin
                    error(' No end of file marker found..');
                    pause;
                end (* then *);
       end (* else *);
end (* fetch_map_data *);


procedure user_init(var m: menu_array);
{----------------------------------------------------------
  Supplied by HP, it loads the menus for the command page.
-----------------------------------------------------------}
    begin
        m[0] := 'A)  Store map data  ';
        m[1] := 'A1) Select map param';
        m[2] := 'A2) Initialize map  ';
        m[3] := 'A3) Print map data  ';
        m[4] := 'A4) Print stat data ';
        m[5] := 'A5) Statistics plot ';
        m[6] := 'A6) Wafer Surf. Plot';
        m[7] := 'A7) Save/Fetch Map  ';
        m[8] := 'A8) Release Prober  ';
        m[9] := '                    ';
        m[10] := 'A10) Set supply vals';
        m[11] := 'A11) Time delay (s) ';
        m[12] := '                    ';
        m[13] := '                    ';
        m[14] := '                    ';
        map_allocated := false;
        map_par_num := 1;
    end (* user_init *);
```

```
procedure a0;
{--------------- Store map data -----------------------------
  Takes the data from the table in the active model and put it
  into the map data base, while also setting the flag so we use
  it later on.
 ---------------------------------------------------------------}

    begin
        if not map_allocated
        then   {no data base to load}
            begin
                error(' Map data space not allocated ');
                pause;
            end (* then *)
        else
            begin
            {Confirm the location and value of the data}
                write('Position ', probe_x - origin_x: 0, ', ',
                    probe_y - origin_y: 0, ' = ');
                writeln('    ', strng(active_par. value[
                    map_par_num]), ' ', active_par. unit[
                    map_par_num]);
                writeln(' ');

             {Set the data and flags now}
                map_array^. setflg[probe_x, probe_y] := true;
                map_array^. data[probe_x, probe_y] := active_par.
                    value[map_par_num];
                wait(2000);
            end (* else *);
    end (* a0 *);


procedure a1;
{------------------ Select map parameter---------------------
   Pick which of the multitude of parameters in the current
   model you want to map out.  Only one to a customer (so far)
--------------------------------------------------------------}
    var
        ip, endnum: integer;
        s: string_80;

    begin
        page;
        show_title('Select Map Parameter');
        with active_par do
            begin
              {Make sure the parameter exists in the model}
                if map_par_num > number  then
                    map_par_num := 1;

              {Show what you've got so far}
                writeln(output, ' Current model is ', title);
                write(output, ' Parameter selected is now ', name
                    [map_par_num]);
                writeln(output, '  (', unit[map_par_num], ')');
                writeln(output, ' Possible parameters are:');
```

```
                { make three columns of parameters, which read
                                    up and down}
                ip := 1;
                if number > 10
                then
                    endnum := 10
                else
                    endnum := number;
                repeat
                    write(output, ip: 3, ' ', name[ip], ' ': 20 -
                        strlen(name[ip]));
                    if ip + 10 <= number  then
                        write(output, ip + 10: 3, ' ', name[ip +
                            10], ' ': 20 - strlen(name[ip + 10]));
                    if ip + 20 <= number  then
                        write(output, ip + 20: 3, ' ', name[ip +
                            20]);
                    ip := ip + 1;
                    writeln(output);
                until ip > endnum;

             { Ask for the new parameter, using the old one as
               the default value }
               writeln(output);
                s := '1..' + intnum(number);
                repeat
                    askinteger(
                      'Enter the number of the mapping parameter'
                        , s, map_par_num, map_par_num);
                until  (map_par_num <= number) and  (map_par_num>0);
            end (* with *);
    end (* a1 *);


procedure a2;
{ Initialize Map Data }
{-------------------------------------------------------------
  This routine allocates the array for the map data.  It uses
   the default size of the arrays established above for
   this work.
 ------------------------------------------------------------}

    var
        answer: string_80;
        ix, jy: integer;

    begin
        page;
        show_title('Initializing Map Data');
        if map_allocated
        then  {somebody's already sleeping in the database}
            begin
                warn('This operation may destroy existing data');
                write(output, ' ');
                ask('OK to destroy map data', 'Y or N', 'N',
                    answer);
                if answer = 'Y'  then {go ahead and destroy}
```

```
                    map_allocated := false;
                end (* then *);
        if not map_allocated
        then  {we need the space set aside}
            begin
                if map_array = nil  then
                    new(map_array);
                writeln(#10,'---- Data space allocated ----',#10);
                map_allocated := true;
                map_x_actual := numb_hor_chip;
                map_y_actual := numb_vert_chip;
               {zero the array}
                for ix := 0 to map_x_size do
                    for jy := 0 to map_y_size do
                        begin
                            map_array^. setflg[ix, jy] := false;
                            map_array^. data[ix, jy] := 0.0;
                        end (* for *);
                pause;
            end (* then *);
    end (* a2 *);



procedure a3;
{-----------------------Print map data----------------------
  Prints out the data in the map, noting the current model,
  parameterm=, and whether each data point it valid by
  location.
 ------------------------------------------------------------}
    var
        s: string_80;
        ix, jy: integer;
        next: integer;

    begin
        page;
        show_title('Print Map Data');
        alpha_only;
        if map_allocated
        then
            begin
                {use the standard TECAP routines}
                print_start;
                print(' ');
                print('Print Stored Map Data');
                print('---------------------');
                print(' ');
              {Show the active model and parameter}
                s := 'Active Model is ';
                strappend(s, active_par. title);
                print(s);
                s := 'Parameter Mapped is ';
                strappend(s, active_par. name[map_par_num]);
                strappend(s, '  (');
                strappend(s, active_par. unit[map_par_num]);
                strappend(s, ')');
                print(s);
```

```
                print(' ');

                print(
                  '---------------------------------------------');
                print(
                  '| X position      Y position          Value |');
                print(
                  '---------------------------------------------');

             { Now print all the data }
                for ix := 0 to map_x_actual do
                    for jy := 0 to map_y_actual do
                        begin
                            s := '';
                            strwrite(s, 1, next, ix - origin_x: 8);
                            strwrite(s, next, next, ' ': 9);
                            strwrite(s, next, next, jy - origin_y
                                : 8);
                            strwrite(s, next, next, ' ': 9);
                            if map_array^. setflg[ix, jy]
                            then {data was set up}
                                strappend(s, strng(map_array^.
                                    data[ix, jy]))
                            else
                                strappend(s, 'Not set');
                            print(s);
                        end (* for *);
                print_end;
            end (* then *)
        else
            begin
                error('No available map data');
                pause;
            end (* else *);
    end (* a3 *);


procedure a4;
{-----------------Print stat data------------------------------
   Print the mean, standard deviation, minimum, and maximum of
   the data, as well as the expected value adjusted for sample
   size.
 --------------------------------------------------------------}

    var
        ix, iy, next, min_x, min_y, max_x, max_y: integer;
        min, max: real;
        s: string_80;

    begin
        show_title('Finding Statistical Data');
        if map_allocated
        then
            begin
                print_start;
                s := 'Parameter Mapped is ';
                strappend(s, active_par. name[map_par_num]);
```

```
  strappend(s, '  (');
  strappend(s, active_par. unit[map_par_num]);
  strappend(s, ')');
  print(s);

  print('Statistical Data for Current Map');
  print('--------------------------------');
  print(' ');

  s := ' Mean                        = ';
  strappend(s, strng(map_mean(map_array^,
      map_x_actual, map_y_actual)));
  print(s);

  s := ' Standard Deviation          = ';
  strappend(s, strng(map_standard_deviation(
      map_array^, map_x_actual, map_y_actual)));
  print(s);

  s := ' Value of parameter          = ';
  strappend(s, strng(map_mean(map_array^,
      map_x_actual, map_y_actual)));
  strappend(s, '  +/-  ');
  strappend(s, strng(1.96 * map_standard_deviation(
      map_array^, map_x_actual, map_y_actual) /
      sqrt((1 + map_x_actual) * (1 + map_y_actual))
      ));
  print(s);
  print(' ');

{Find the min and max, by location}
  min := 1.0e+300;
  max := - min;
  min_x := 1;
  min_y := 1;
  max_x := 1;
  for ix := 0 to map_x_actual do
      for iy := 0 to map_y_actual do
          begin
              if map_array^. setflg[ix, iy]
              then
                  begin
                      if map_array^.data[ix,iy]>max
                      then
                          begin
                              max := map_array^.
                                  data[ix, iy];
                              max_x := ix;
                              max_y := iy;
                          end (* then *);
                      if map_array^.data[ix,iy]<min
                      then
                          begin
                              min := map_array^.
                                  data[ix, iy];
                              min_x := ix;
                              min_y := iy;
                          end (* then *);
```

```
                                end (* then *);
                        end (* for *);

                s := ' Minimum data value          = ';
                strappend(s, strng(min));
                print(s);
                s := '        at location ';
                strappend(s, '(');
                strwrite(s, strlen(s) + 1, next, min_x - origin_x
                    : 0);
                strwrite(s, next, next, ', ');
                strwrite(s, next, next, min_y - origin_y: 0);
                strappend(s, ')');
                print(s);
                s := ' Maximum data value          = ';
                strappend(s, strng(max));
                print(s);
                s := '        at location ';
                strappend(s, '(');
                strwrite(s, strlen(s) + 1, next, max_x - origin_x
                    : 0);
                strwrite(s, next, next, ', ');
                strwrite(s, next, next, max_y - origin_y: 0);
                strappend(s, ')');
                print(s);
                print(' ');
                print_end;
            end (* then *)
        else
            begin
                error(' No Available Map Data to analyze ');
                pause;
            end (* else *);
    end (* a4 *);


procedure a5;
{-------------------Statistical plot--------------------------
  Graph a histogram plot of the data distribution in the map,
  along with a superimposed curve for a normal distribution.
 -------------------------------------------------------------}

    const
        nblocks = 10;

    var
        min_x, max_x, max_height, block_step: real;
        height_data: array [0.. nblocks] of integer;
        mean, st_dev: real;
        count, g, ix, iy: integer;
        x, norm: data_array;
        left, right, top, bottom: real;
        deltax, deltay: real;

    begin
        show_title('Plotting Data Distribution');
        if not map_allocated
```

```
then  {No data to plot}
    begin
        error(' No Available Map Data to Analyze ');
        pause;
    end (* then *)
else
    begin

      { Get mean and standard deviation }
        mean := map_mean(map_array^, map_x_actual,
            map_y_actual);
        st_dev := map_standard_deviation(map_array^,
            map_x_actual, map_y_actual);

     { Set the bar heights to zero to start }
        for ix := 0 to nblocks do
            height_data[ix] := 0;

     { Find the minimum and maximum data (for autoscale)}
        min_x := 1.0e+300;
        max_x := - min_x;
        for ix := 0 to map_x_actual do
            for iy := 0 to map_y_actual do
                if map_array^. setflg[ix, iy]
                then
                    begin
                        if map_array^. data[ix, iy] >
                            max_x
                        then
                            max_x := map_array^. data[ix,
                                iy];
                        if map_array^. data[ix, iy] <
                            min_x
                        then
                            min_x := map_array^. data[ix,
                                iy];
                    end (* then *);
        min_x := round_down(min_x);
        max_x := round_up(max_x);

     { If data is identical (errant) prevent a (-5) error }
        if max_x = min_x
        then
            block_step := 1e-30
        else
            block_step := (max_x - min_x) / (nblocks);

     {Now find the vertical heights of the bars }
        max_height := 0;
        for ix := 0 to map_x_actual do
            for iy := 0 to map_y_actual do
                if map_array^. setflg[ix, iy]
                then
                    begin
                        g := trunc((map_array^. data[ix,
                            iy] - min_x) / block_step);
                        height_data[g] := height_data[g]
                            + 1;
```

```
                    if height_data[g] > trunc(
                        max_height)
                    then
                        max_height := height_data[g];
                end (* then *);
    max_height := max_height + 1;

{ do the actual plotting now }

  {set up the display surface}
    graph_init(plot_output);
    left := min_x - 0.15 * (max_x - min_x);
    right := 0.05 * (max_x - min_x) + max_x;
    top := max_height * 1.15;
    bottom := - 0.35 * (max_height);
    set_window(left, right, bottom, top);

  {Set a frame width}
    deltax := 0.005 * (right - left);
    deltay := 0.005 * (top - bottom);

  {Draw the frame}
    move(min_x, 0.0);
    line(max_x, 0.0);
    line(max_x, max_height);
    line(min_x, max_height);
    line(min_x, 0.0);
    move(min_x - deltax, 0.0 - deltay);
    line(max_x + deltax, 0.0 - deltay);
    line(max_x + deltax, max_height + deltay);
    line(min_x - deltax, max_height + deltay);
    line(min_x - deltax, 0.0 - deltay);

 {label the vertical axis}
    move(left, 0.0);
    set_char_size(0.01 * (right - left), 0.025 * (top
        - bottom));
    gtext(strng(0.0));
    move(left, max_height);
    gtext(strng(max_height));

 { graph label }
    move(min_x + 0.30 * (max_x - min_x), bottom +
        0.14 * (top - bottom));
    set_char_size(0.03 * (right - left), 0.04 * (top
        - bottom));
    gtext(active_par. name[map_par_num]);
    gtext(' (');
    gtext(active_par. unit[map_par_num]);
    gtext(')');

  { Label some important numbers}
    set_char_size(0.015 * (right - left), 0.022 * (
        top - bottom));
    move(min_x + 0.30 * (max_x - min_x), bottom +
        0.08 * (top - bottom));
    gtext('Mean = ' + strng(mean) + ' ' + active_par.
        unit[map_par_num]);
```

```
            set_char_size(0.015 * (right - left), 0.022 * (
                top - bottom));
            move(min_x + 0.30 * (max_x - min_x), bottom +
                0.05 * (top - bottom));
            gtext('Standard Deviation = ' + strng(st_dev));
            set_char_size(0.015 * (right - left), 0.022 * (
                top - bottom));
            move(min_x + 0.30 * (max_x - min_x), bottom +
                0.02 * (top - bottom));
            gtext('Bar width = ' + strng(block_step) + ' ' +
                active_par. unit[map_par_num]);

         { Title Bar }
            set_char_size(0.025 * (right - left), 0.04 * (top
                - bottom));
            move(min_x + 0.30 * (max_x - min_x), 1.025 *
                max_height);
            gtext('Map Data Distribution');
            set_char_size(0.02 * (right - left), 0.023 * (top
                - bottom));
            move(left, 1.1 * max_height);
            gtext('TECAP Ver ' + ver);

         {Label the horizontal axis scale and vertical title}
            set_text_rot(0.0, 1.0);
            move(left + 0.05 * (right - left), 0.30 *
                max_height);
            set_char_size(0.030 * (right - left), 0.04 * (top
                - bottom));
            gtext('Count');
            move(min_x, bottom);
            set_char_size(0.01 * (right - left), 0.023 * (top
                - bottom));
            gtext(strng(min_x));
            move(max_x, bottom);
            gtext(strng(max_x));

         {Draw all of the blocks on the chart NOW}
            for ix := 0 to nblocks - 1 do
                begin
                    move(min_x + (ix) * block_step, 0.0);
                    line(min_x + (ix) * block_step,
                        height_data[ix]);
                    line(min_x + (ix + 1) * block_step,
                        height_data[ix]);
                    line(min_x + (ix + 1) * block_step, 0.0);
                end (* for *);
            move(min_x, 0.0);    { to pick up the pen }

    {draw the normal curve now}
            count := 0;
            for ix := 0 to map_x_actual do
               for iy := 0 to map_y_actual do
                  if map_array^.setflg[ix,iy] then
                      count := count + 1;
            for ix := 1 to 100 do
                begin
                    x[ix] := min_x + (max_x - min_x) * (ix -
```

```
                            1) / 99;
                        norm[ix] := (count * block_step) / (
                            sqrt(2.0 * 3.14159) * st_dev) * exp(-
                            sqr(x[ix] - mean) / 2.0 / sqr(st_dev)
                            );
                    end (* for *);

                move(x[1], norm[1]);
                for ix := 2 to 100 do
                    line(x[ix], norm[ix]);
                graph_term;
            end (* else *);
    end (* a5 *);


procedure a6;
{-----------------------------------------------------------
   Plot the physical distribution of the paramters as a
   Schmoo plot. (Contour plotting was not effective; the
   data varied too quickly for a meaningful picture to
   form.)
 -----------------------------------------------------------}

    const
        ndivs = 10;            {number of "families" of values}

    var
        left, right, bottom, top: real;
        date: string_80;
        mapchar: string_80;
        min_data, max_data, step_data: real;
        g: integer;
        ix, iy: integer;
        shiftx, shifty: real;
        width, height: real;
        thpi, loc, xradius, yradius, step: real;
        center: record
                     x, y: real;
                 end;

    begin
        {characters (in order) for the plotting}
        mapchar := '0123456789X';
        date := form_date;
        show_title('Plot Wafer Map');
        if not map_allocated
        then
            begin
                error(' No map data to plot ');
                pause;
            end (* then *)
        else
            begin

             {Find the extent of the data, for scaling}
                min_data := 1.0e+300;
                max_data := - min_data;
```

```
     for ix := 0 to map_x_actual do
         for iy := 0 to map_y_actual do
             if map_array^. setflg[ix, iy]
             then
                 begin
                     min_data := min(min_data,
                         map_array^. data[ix, iy]);
                     max_data := max(max_data,
                         map_array^. data[ix, iy]);
                 end (* then *);
     min_data := round_down(min_data);
     max_data := round_up(max_data);

  { If data is too close together, prevent a /0 error}
     if abs(max_data - min_data) < 1e-20
     then
         step_data := 1e-30
     else
         step_data := (max_data - min_data) / ndivs;


{ Set up the plotting surface parameters }
     width := 2.0 * (map_x_actual + 1);
     height := 1.5 * (map_y_actual + 1);

     left := - 0.04 * width;
     right := ((map_x_actual + 1)) + 0.50 * width;
     shiftx := 0.01 * width;

     top := (map_y_actual + 1) + 0.16667 * height;
     bottom := - 0.1667 * height;
     shifty := 0.01 * height;

     graph_init(plot_output);
     set_window(left, right, bottom, top);

  { Plot the frame for the map }
     move(- 3*shiftx, - 3*shifty);
     line((map_x_actual + 1) + 3*shiftx, - 3*shifty);
     line((map_x_actual + 1) + 3*shiftx, (map_y_actual +
         1) + 3*shifty);
     line(-3* shiftx, (map_y_actual + 1) + 3*shifty);
     line(-3* shiftx, - 3*shifty);
     move(- 2 * shiftx, - 2 * shifty);
     line((map_x_actual + 1) + 2 * shiftx, - 2 *
         shifty);
     line((map_x_actual + 1) + 2 * shiftx, (
         map_y_actual + 1) + 2 * shifty);
     line(- 2 * shiftx, (map_y_actual + 1) + 2 *
         shifty);
     line(- 2 * shiftx, - 2 * shifty);

  { Label the map }
     set_char_size(0.030 * width, 0.025 * height);
     move(left, top - 0.023 * height);
     gtext('WAFER MAP  - TECAP ver ' + ver);

     set_char_size(0.025 * width, 0.035 * height);
```

```
        move(left, bottom + 0.001 * height);
        gtext('Parameter mapped is ' + active_par. name[
            map_par_num]);
        gtext(' (' + active_par. unit[map_par_num] + ')');

        set_char_size(0.02 * width, 0.02 * height);
        move(left, top - 0.060 * height);
        gtext('Date and Time: ' + date);

        move(left, top - 0.08 * height);
        gtext('Wafer Name: ' + dev. name);
        move(left, top - 0.10 * height);
        gtext('User Name:  ' + user. name);
        move(left, top - 0.12 * height);
        gtext('Type:       ');
        case dev. typ of
            nmos:
                gtext('NMOS');
            pmos:
                gtext('PMOS');
            npn:
                gtext('NPN');
            pnp:
                gtext('PNP');
            njfet:
                gtext('NJFET');
            pjfet:
                gtext('PJFET');
            diode:
                gtext('DIODE');
            tube:
                gtext('TUBE');
            misx:
                gtext('MISX');
        end (* case *);

    {Show the scale chart}
        move((map_x_actual + 1) + 0.05 * width, (
            map_y_actual + 1));
        set_char_size(0.015 * width, 0.020 * height);
        gtext('Blank is unmeasured');
        for ix := 0 to ndivs-1 do
            begin
                move((map_x_actual + 1) + 0.05 * width, (
                    map_y_actual + 1) - 0.035 * (1 + ix)
                    * height);
                gtext(str(mapchar, ix + 1, 1) + ': >');
                gtext(strng(min_data + ix * step_data) +
                    ' ');
                gtext(active_par. unit[map_par_num]);
            end (* for *);

{draw the wafer}
        xradius := 0.5 * (map_x_actual + 1) + 1.5 * shiftx;
        yradius := 0.5 * (map_y_actual + 1) + 1.5 * shifty;
        center. x := 0.5 * (map_x_actual + 1);
        center. y := 0.5 * (map_y_actual + 1);
        step := 0.367 / 4 {radians};
```

```
                thpi := 1.5 * 3.1415926;
                move(center. x + xradius * cos(thpi + step * 4),
                    center. y + yradius * sin(thpi + step * 4));
                loc := thpi + step * 4;
                repeat
                    line(center. x + xradius * cos(loc), center. y
                        + yradius * sin(loc));
                    loc := loc + step;
                until sin(loc) < sin(thpi - step * 4);
                line(center. x + xradius * cos(thpi + step * 4),
                    center. y + yradius * sin(thpi + step * 4));

            {Draw grid outlining chip if requested}
                if grid_flag
                then
                    begin
                        set_line_style(1);
                        for ix := 1 to map_x_actual do
                            begin
                                move(ix, 0);
                                line(ix, (map_y_actual + 1));
                            end (* for *);
                        for iy := 1 to map_y_actual do
                            begin
                                move(0, iy);
                                line((map_x_actual + 1), iy);
                            end (* for *);
                        set_line_style(1);
                    end (* then *);

        {NOW we can plot the map}
                set_char_size(0.4 * width / map_x_actual, 0.6 *
                    height / map_y_actual);
                for ix := 0 to map_x_actual do
                    for iy := 0 to map_y_actual do
                        if map_array^. setflg[ix, iy]
                        then
                            begin
                                g := trunc((map_array^. data[ix,
                                    iy] - min_data) / step_data);
                                move(ix + 0.1 * width /
                                    map_x_actual, iy + 0.15 *
                                    height / map_y_actual);
                                gtext(str(mapchar, g + 1, 1));
                            end (* then *);
                graph_term;
            end (* else *);
    end (* a6 *);


procedure a7;
{-------------------------------------------------------------
   Controlling routine for loading and storing the map data.
   Calls two routines at the top of the file to handle the
   details; this makes sure data goes where it should.
 -------------------------------------------------------------}
```

```
    var
        do_b, ans: string_80;

    begin
        show_title('Disk Data Manager');
        repeat
            ask('Fetch data from or store data to disk', 'F or S'
                , 'S', do_b);
        until (do_b = 'F') or (do_b = 'f') or (do_b = 'S') or (
            do_b = 's');

        if (do_b = 's') or (do_b = 'S')
        then  {We want to store data}
            if not map_allocated
            then
                begin
                    error(' No map data to store ');
                    pause;
                end (* then *)
            else
                store_map_data
        else
            begin
                if map_allocated
                then  {some data is in the way.}
                    begin
                        warn(
                      '  This will overwrite some existing data  ');
                        ask('OK to destroy map data', 'Y or N',
                            'N', ans);
                        if (ans = 'Y') or (ans = 'y')  then
                            fetch_map_data;
                    end (* then *)
                else
                    begin
                        a2;
                        fetch_map_data;
                    end (* else *);
            end (* else *);
    end (* a7 *);


procedure a8;
{---------------------------------------------------------
   Release the prober to front panel control, for loading
   wafers, etc.
 --------------------------------------------------------}
     begin
        local(hpib*100+rk_1032_add);
     end (* a8 *);


procedure a9;  begin end (* a9 *);


procedure a10;
```

```
{--------------------Set Supply Values------------------------
   Will allow the user to set any supply in the 4145 to a value
   desired; useful for long-term voltage applications when used
   with the time delay routine A11.
--------------------------------------------------------------}

     var
         what: string_80;
         which_unit: unit_type;
         value, compl: real;
         mode: string_80;
         src_mode: source_type;

     begin
         page;
         show_title('Setting Supplies');

        {Find out what the user wants, and wait till it's correct}
          repeat
             ask('Set which unit?', 'SMU1,2,3, or 4; VS1 or 2', ''
                 , what);
             which_unit := gnd;
             if (what = 'SMU1') or (what = 'smu1')  then
                 which_unit := smu1;
             if (what = 'SMU2') or (what = 'smu2')  then
                 which_unit := smu2;
             if (what = 'SMU3') or (what = 'smu3')  then
                 which_unit := smu3;
             if (what = 'SMU4') or (what = 'smu4')  then
                 which_unit := smu4;
             if (what = 'VS1') or (what = 'vs1')  then
                 which_unit := vs1;
             if (what = 'VS2') or (what = 'vs2')  then
                 which_unit := vs2;
         until which_unit <> gnd;

      { How big a value? }
         askreal('What bias level?', 'volts or amps', 0.0, value);

      {Just for the SMU's}
         if (which_unit = smu1) or (which_unit = smu2) or (
             which_unit = smu3) or (which_unit = smu4)
         then
             begin
                 askreal('Enter channel compliance', 'amps', 1e-3,
                     compl);
                 ask('Enter source mode', 'V or I', 'V', mode);
                 if ((mode = 'V') or (mode = 'v'))
                 then
                     src_mode := v
                 else
                     src_mode := i;
             end (* then *);

      {Do the change NOW}
         set_bias(which_unit, value, compl, src_mode);
     end (* a10 *);
```

```
procedure a11;
{---------------------------------------------------------------
    Arbitrary time delay.  Value coming in is in seconds and
    may have 1 millisecond resolution.  The screen is kept
    busy, so one may know the system is working at waiting.
 --------------------------------------------------------------}

    var
        s: string_80;
        x: integer;
        z: real;

    begin
        {Wait for a valid number}
         repeat
            ask(
          'Enter time delay in seconds (engineering units valid)'
                , '0..100000 (100k)', '0.0', s);
            z := number(s) * 1000;
            x := trunc(z);
        until (x >= 0) and (x <= maxint);
        write('... Waiting ', s, ' seconds ');
        { Print a dot every two seconds, just to let the
           world know we're still alive}
        while x > 2000 do
            begin
                wait(2000);
                x := x - 2000;
                write(output, '.');
                sound_busy;
            end (* while *);
        wait(x);
    end (* a11 *);


procedure a12;  begin end (* a12 *);
procedure a13;  begin end (* a13 *);
procedure a14;  begin end (* a14 *);


{ The model programs are not listed here because they were
  not a part of the thesis work.  For a complete listing see
  the file "T_USER.TEXT" on the "TCPUS" disk supplied with
  the system update instructions. }

$LIST OFF$

$LIST ON$

end {user module code for thesis}.
```

Vita

Phillip Mark Goldman was born October 25, 1962 in Meadowbrook, Pennsylvania to Robert and Barbara Goldman. He attended a National Science Foundation Summer Science Student Program at Mankato State University in the summer of 1979. He attended Lehigh University from August 1980 until January 1986, during which time he earned a Bachelor of Science in Computer Engineering (June 1984) and a Master of Science in Electrical Engineering (January 1986). During the summer of 1983 he was awarded a Sherman Fairchild Summer Fellowship. He has been elected to Tau Beta Pi, Eta Kappa Nu, and maintains membership in I.E.E.E.