SANDIA REPORT

SAND93—0347 • UC—706
Unlimited Release
Printed March 1993

Sensor Based Process Control (SBPC) Laboratories Directed Research and Development (LDRD)

Final Report

John B. Wronosky

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-76DP00789

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
Office of Scientific and Technical Information
PO Box 62
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

Available to the public from
National Technical Information Service
US Department of Commerce
5285 Port Royal Rd
Springfield, VA 22161

NTIS price codes
Printed copy: A04
Microfiche copy: A01

SAND93-0347
Unlimited Release
Printed March 1993

Distribution
UC-706

Sensor Based Process Control (SBPC) Laboratories Directed Research and Development (LDRD) Final Report

John B. Wronosky, 2338

ABSTRACT

This report describes the activities and results of an LDRD entitled Sensor Based Process Control. This research examined the needs of the plating industry for monitor and control capabilities with particular emphasis on water effluent from rinse baths. A personal computer-based monitor and control development system was used as a test bed.

CONTENTS

ACKNOWLEDGEMENTS vi

INTRODUCTION 1

PROJECT PLAN 1

FINDINGS 1
Application Research 1
Facility Tours. 1
Literature Search. 2
Available Sensor Technology. 3
Present Sensor/Interface Development 3
Sandia Sensors. 3

EXPERIMENTAL RESULTS 3
Demonstration System Configuration 3
Development Considerations 4

KNOWLEDGE GAINED 5
How to Maximize Benefits 6

Appendix A - LDRD Proposal 7

Appendix B - Project Plan 11
Project Definition 12
Abstract 12
Goal Statement 12
Project Requirements 14
Project Milestones 14
Work Breakdown Structure 15
Schedule Diagram for SBPC LDRD Project Schedule 16
Budget 16
Project Organization 16
Logistics Support 17
Security 17
Environmental, Safety, and Health 17
Customer Organization Contact Points 17
Nature of Project Reviews 18

CONTENTS (Continued)

Appendix C - Articles Referenced from Literature Search 19

Appendix D - Development System Design 24
Development Resources 25
Major Components 25
Block Diagram 26
Custom Electronics Descriptions 27
Modifications and Connections to SC-2070 Board 27
Potentiostat 28
Potentiostat Stimulus 29
Software Description 29
ASV Data Acquisition Timing Scheme (N.I. AT-MIO-16F) 30
ASV 10 Weight Finite Impulse Response (FIR) Filter 31
ASV Data Smoothing 31
File Descriptions 31
LabWindows Libraries Used 32

Appendix E - Development System Flow Charts 33

Appendix F - Development System Code with Comments 38

FIGURES

1 SBPC Demonstration System 4
2 Development System Data Design 4

ACKNOWLEDGEMENTS

The author thanks Jerry T. Love, Marie C. Kidd, and Kathleen Alam for their investigation for the proposal. Thanks also go to Brad Smith for his assistance in researching industry needs; to Dale Brandt for his work designing and developing the monitor and control system; and to Victoria E. Granstaff, for providing support for the Anodic Stripping Voltammetry sensor technology used with the development system.

INTRODUCTION

A Laboratories Directed Research and Development (LDRD) project was carried out in FY92 by Sandia National Laboratories (Sandia) for the purpose of developing a reconfigurable monitoring system. The system would take advantage of the best features of a suite of sensors to monitor water effluent releases into the environment.

Industry research indicates a need for waste water monitoring and controlling to improve the plating and etching processes, as well as to minimize hazardous discharges in the waste stream. Three plating facilities and one waste recovery facility were visited during the research phase. Operators of these facilities and others that were queried indicated the need for reliable monitoring and control capabilities to meet growing environmental regulations.

The final product of this work would be a system that is adaptable to different etching and plating environments by changing site specific system characteristics. A working demonstration system was constructed utilizing a 386-based portable personal computer (PC), custom and commercial software, commercial- and Sandia-designed sensors, and commercial and custom interface hardware. Preliminary tests indicate that the system could be developed for installation as a waste water monitor. Additional development would include environmental packaging, system integration (control valves, pumps, alarms, etc.) and software for automatic operation and analysis.

PROJECT PLAN

The Project Plan included in Appendix B details the goals and efforts controlling the work for this LDRD. Efforts to develop a demonstration system operating on an electroless plating line were abandoned due to ES&H problems associated with the plating line installation. The Anodic Stripping Voltammetry based sensor and commercial pH, Temperature, and Conductivity sensors were used instead. The ASV represented a challenge to the "configurable monitor" concept in that it requires specific waveform input as stimulus and low amplitude current measurement for analysis. This resulted in the design of a custom hardware and software interface to the computer.

FINDINGS

Application Research

Facility Tours - Four facilities involved in the plating and etching of printed circuit boards were surveyed in order to understand the concerns and desires of the industry. The facilities were: Sandia National Laboratories Printed Wiring Board (PWB) Plating Laboratory (Sandia); Sparton

Technology Inc. (Sparton); Allied Signal, KCD (KCD); and Southern California Chemical Company. Plating and etching operations are carried out at Sandia, Sparton, and KCD. Southern California Chemical is a waste transport, treatment, reclamation, and disposal facility.

Sandia's plating laboratory is a low volume facility operated under more stringent controls than the two industrial facilities. Few, if any, real-time monitors and controls are used at this facility. Chemical monitoring is done primarily by laboratory analysis. Problems stated for not utilizing more real-time monitors and controls include lack of reliable sensors, vulnerability of monitoring hardware to harsh laboratory environments, and proprietary chemistry used in bath make-up.

Sparton's plating facility would be considered small by industry standards. They are particularly conscious of waste-management concerns after being cited for poor practices in the past. They estimate that waste treatment and disposal costs approximately $10,000 per month. Clean-up costs from past mistakes are ongoing and costly.

KCD operates extensive plating facilities to meet both PWB and general plating requirements placed on them by Sandia and the U.S. Department of Energy (DOE). Most chemical monitoring is done by in-house sampling and laboratory analysis. Real-time monitor and control is only being used on an electroless nickel bath. This system has saved $250,000 in two years and cost only $10,000. Real time monitoring is used on an electrolytic copper bath in the PWB facility, but no replenishing system is planned. KCD has a very extensive (and costly) waste treatment facility where all effluent from the plating operations is treated before disposal.

Southern California Chemical Co. receives a large portion of the waste generated by plating and etching industries in the Southwest. They operate a waste treatment and disposal facility that is larger, older, and less pristine than the KCD facility. Almost all of their incoming waste is converted to marketable by-products that are returned to industry at a profit. It is evident that this industry is highly regulated. For example, on-site rain water must be captured and treated as industrial waste. Laboratory analysis is the primary means of monitoring the processes of waste treatment at this facility.

Information and discussion with those involved in plating, etching, and waste management indicates a major interest in reliable and cost-effective monitoring capabilities to meet more stringent environmental regulations. Automatic control is considered secondary, though desirable. These views were further substantiated by the attendees of the Photo-Chemical Machining Institute (PCMI) winter meeting of 1992. At the request of PCMI, Jerry T. Love presented a program entitled, "Sensor-based Process Control Development." Love's program included an overview of sensor and monitor and control development at Sandia. Dr. Dan Doughty, also of Sandia, presented a program entitled, "Development of Chemical Monitors Using Acoustic Wave Devices."

Literature Search - A literature search was conducted for process monitor and control articles relating to plating baths. A bibliography of related articles is included as Appendix C. Many recent articles state the need for real-time monitor and control to meet the increasingly stringent

environmental regulations. Few of these ideas are being pursued for production. Many of the articles usemonitoring techniques that were called unreliable by people at the facilities we visited. Sensors seem to be the main blockade in developing an acceptable system.

Available Sensor Technology - Available monitor and control systems utilize sensors that measure pH, conductivity, and temperature. Contamination of these sensors and related apparatuses appears to be the major reason they are not used more extensively. More than one type of sensor is usually needed to determine the parameters necessary for process control.

Present Sensor/Interface Development

Sandia Sensors - Sandia sensor development is progressing with a strong emphasis on environmental sensing. Sandia sensor technologies that are applicable to monitor and control are Surface Acoustic Wave (SAW) devices, Hydrogen Sensors, Quartz Crystal Microbalances, Tuned Frequency Impedance Probes (TFIPs), and Anodic Stripping Voltammeters (ASVs). We have been involved in a monitor application using a portable PC for the Portable Acoustic Wave Sensor (PAWS) system that uses a SAW device for stack monitoring above spray-cleaning booths and is intended for a variety of air-monitoring applications. This sensing device could be integrated into the demonstration monitor and control system that is being designed for this LDRD. We intend to allow enough flexibility in the system to add any type of sensing device available in the future.

The ASV development is used as a test example for the demonstration system. The ASV can be used to determine the type and concentration of metals in a rinse tank and requires a complex stimulus. Data from these measurements are used to determine metal type (potential correlation) and concentration (peak current). Information obtained using this sensor will be compared to that read from pH, conductivity, and temperature sensors to provide a thorough characterization of the rinse bath. The ASV approach to waste-water monitoring is also under development by Lawrence Livermore National Laboratory in California.

EXPERIMENTAL RESULTS

Demonstration System Configuration

A Dolch portable 386 computer, configured with a National Instruments multi-function data acquisition I/O card, makes up the heart of the waste-water monitor. External circuitry is used to provides signal conditioning for the temperature, conductivity, pH, and ASV sensors. Monitor and control software is written in C using the LabWindows programming environment. Figure 1 is illustrates the demonstration system.

The system provides a menu-driven user interface and graphical data display. Acquired data are saved to user-selectable files for future analysis. The screen output is shown in Figure 2. Additional system description is included in Appendix D; Development System Design.

The initial configuration of the waste monitor provides monitor and alarm capabilities for detection of user defined concentrations of a given metal (e.g., cadmium and lead). The final system was to include more in depth analysis to correlate readings from conductivity, pH, and temperature sensors. The addition of these capabilities is dependant on future funding. Additional software development will be needed to fully realize the capabilities of the ASV sensor for determination of specific metal concentrations. This effort is beyond the scope of this LDRD.

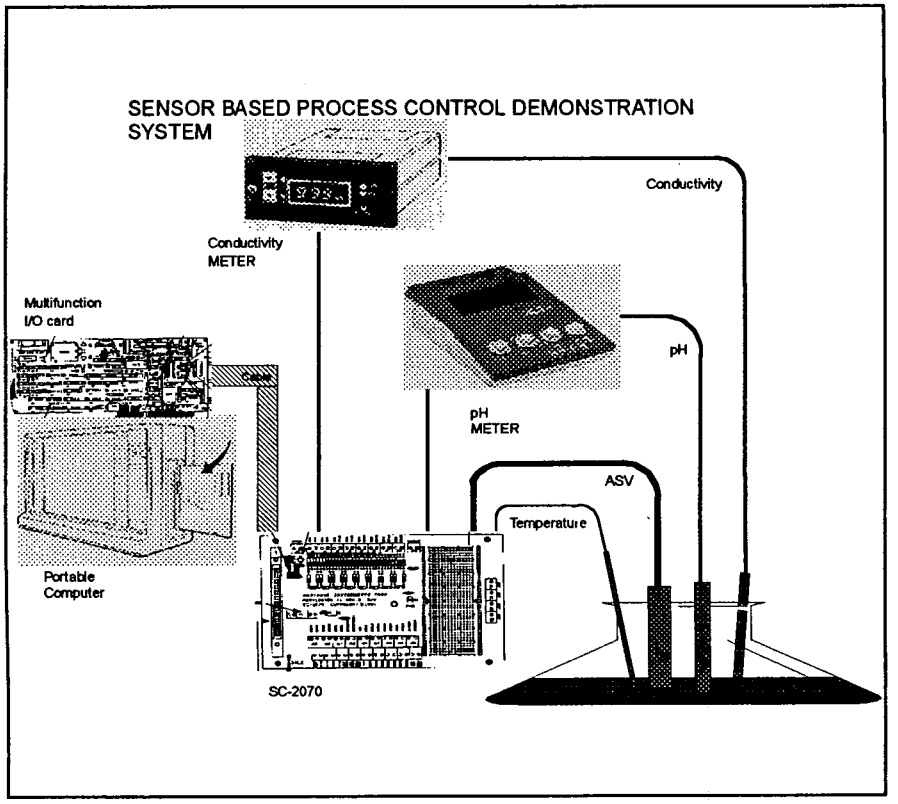


Figure 1. SBPC Demonstration System

Development Considerations

The development of the system exposed several considerations for the design of a configurable monitoring system.

No two sensors types are exactly alike. Each sensor requires a unique interface. This interface can be converted to a common format for transmission to, or interrogation by, the PC. Sensor stimulus can be simple DC voltage, high frequency (100MHz) signals (such as those needed for surface acoustic wave devices used in the PAWS volatile organic monitor system) or complex stepped pulses used for the ASV sensor.

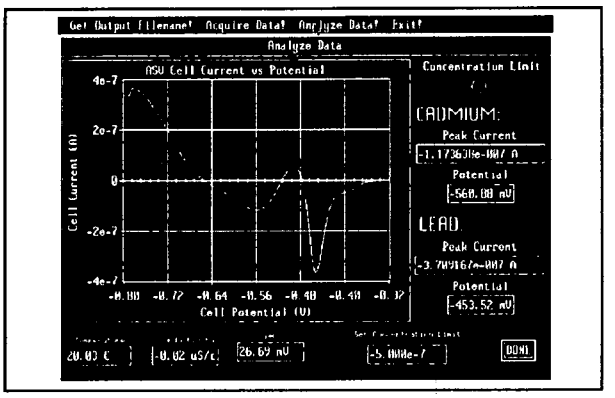


Figure 2. Development System Data Display

Noise considerations are important. Most of the sensors operate at low voltage and current levels (1mv - 500mv). Each of the sensors of the demonstration system utilizes digital software filtering for elimination of 60Hz attenuation error. The ASV sensor monitor software used 10th-order, low-pass, Finite-

Duration Impulse Response (FIR) filtering to eliminate extraneous data. Proper grounding and shielding is important for any computer-based data acquisition system. The computer itself is a noise source.

The unstable nature of the ASV sensor indicatesto the need for automatic calibration capability in a configurable monitor and control system. This can be accomplished by physically moving the sensor to a reference solution or by pumping a reference to the sensor during the calibration period. There may also be ways to compensate for sensor variation during the data analysis. Dept. 1824 is investigating this approach.

An extensive set of requirements must be generated before a truly configurable monitor and control system can be designed to meet the needs of industry. The continuation of this LDRD was to include the formulation of a "straw man" requirements document that would attempt to define a system based on industry needs. Further development of the existing demonstration system will probably be user-specific, based on the input of potential customers.

KNOWLEDGE GAINED

This project provided more extensive knowledge of the aspects of environmental monitoring in the plating industry. Much of the data acquisition knowledge is directly applicable to transportation and security monitoring, as well as to environmental concerns. Sandia weapon programs include a wide range of development and production testing applications that can be developed in an efficient, timely, fashion by using techniques learned during this project.

The demonstration system provides a head start for future environmental sensor support and potential waste monitoring applications. Proposals were submitted for including the monitor and control concept in Sandia's Environmentally Conscious Manufacturing (ECM) initiative. A partnership proposal was submitted by Concurrent Technologies Corporation identified as "Electrochemical Sensing for Rinse Water Monitoring for the Defense Advanced Research Projects Agency (DARPA) Proposal (BAA-92-22)."[1] A Cooperative Research and Development Agreement (CRADA) involving a supplier of plating chemicals and equipment is another area for application of the technology.

---

[1] The SBPC team received a request from Don Bass, Concurrent Technologies Corporation (CTC), 1450 Scalp Avenue, Johnstown, PA. 15901, asking our participation in a proposal entitled, "A Teaming Approach to Provide a Demonstration of Advanced Emission Monitoring, Control, and Reduction Techniques for Bath Plating Operations." The CTC proposal was initiated in response to a Broad Agency Announcment BAA-92-22 from DARPA. If accepted, the SBPC team would provide sensors and computerized monitoring capability for the nickel plating phase of the overall project.

How to Maximize Benefits

Product realization or commercial development has not been a major emphasis of sensor development organizations at Sandia. Much of the major technological improvements sit idle in research reports instead of being developed for use by industry. A Sandia support structure like that used in weapon development programs could improve the realization of state-of-the-art sensor technologies. An "overseer" (systems) department that would observe sensor development, industry needs, and environmental regulations could match up state-of-the-art sensor technology with Sandia "real world" development teams to form partnerships capable of developing practical monitoring and controlling capabilities and systems. Much sensors research is too close to the physics and basic research. Research efforts would benefit from a product realization point of view.

Appendix A

LDRD Proposal

LDRD Proposal

Project Title: Sensor Based Process Control

Principal Investigator: Jerry T. Love, Division 2337, 845-8202

Co-investigators: Marie C. Kidd, Division 2337, 844.1242
Kathleen Alam, Division 6612, 846-0744

Suggested Subcase Manager: W. David Williams, Supervisor, Division 2337

Abstract

Environmental concerns are playing an increasingly important role in manufacturing processes. Hazardous wastes are by-products of companies involved in etching. A reconfigurable monitoring system that takes advantage of the best features of a suite of sensors is needed to monitor water effluent releases into the environment. For optimal performance, these sensors need to be fused into a cooperating sensor array-based system. This system can be adapted to different etching environments by changing only site-specific characteristics of the system. This system will then initiate output signals that will shutdown or alter the components causing the problem and will provide warnings when dangerous hazardous chemical levels are reached.

Work Proposed for FY92

We will develop the demonstration system that will detect various chemicals representative of those released in rinse effluent from etch baths. This demonstration system will be used as a test bed for sensor fusion development, advanced sensor development and to show potential customers our capabilities.

Amount Proposed for Next Year: **217K**

Amount Expended per Fiscal Year: **217K**
1 Year beginning in FY92

Total Estimated Project Cost: **217K**

Problem

In recent years, manufacturing operations of U.S. industry have been increasingly subject to national and local regulations and public concern about the environmental safety and health of their operations. The largest volume source of hazardous waste from the plating and etching industry is contaminated waste water. Therefore, industry has a necessity to minimize and record the amount of hazardous materials which are produced as by- products during their industrial operations.

Objective

This project will develop a demonstration system which can achieve monitoring of water emission levels in a cost efficient manner. The core system will demonstrate the capability to be easily tailored to a wide variety of industrial needs. Thus, lower development cost and schedules could be achieved. Customer dependent activities will focus the sensor fusion system's changes from the core system. The only new development that might need to be conducted for a new customer would pertain to different undeveloped sensors for the chemicals they wish to monitor.

Approach

First, a search of currently available sensors will be conducted. Chromium is the chemical of main concern to this industry. The best of these sensors will be fused into a sensor package with Sandia developed sensors as available.

Second, the hardware subsystem controller will be developed. This will be built from commercially available equipment. For example, the Pro-Log company produces PC based STD Bus systems. The task would be to build an interface card and then put the Pro-Log system parts together in an appropriate configuration. Integration of distributed sensors will also be conducted.

Third, software will be developed utilizing proven Software Engineering methods and tools. This software package will be very generic and flexible to handle the possible uses of the closed-loop monitoring system.

Our final product will be a demonstration system consisting of an array of sensors that through sensor fusion will detect several hazardous chemicals and report their presence.

Potential Contributions

This project will develop a core product that can be used for many other environmental sensing applications with little additional cost and development time. Some examples of broader uses are: any area necessitating environmental sensors, transportation environmental monitors or security monitors.

This project will also lead to new alliances with outside customers since environmental sensor monitoring is a major need in industry. All companies involved in etching are potential customers for this type of technology.

Resources

Jerry Love, the principal investigator, is an SMTS Hardware Project Leader in Division 2337, Electronic Subsystem Division II.

Marie Kidd, a co-investigator, is an SMTS Software Project leader in Division 2337.

Kathleen Alam, a co-investigator, is from Division 6612, Process Optmization and Control Division.

Other resources will come from Departments 1160, 1840, and 2130.

			MANPOWER		DIRECT CHARGES				DIRECT SUPPORT					
ORG	FY	CASE	REG FTE	LC	TRAVEL	PURCHASES	JIT	TRANSFERS	COMPUTING	DESIGN ENG	PROCESS FAB	TEST	TECH COMM	TOTAL COST
2337	92		0.6	84K	3K	12K	3K	0	0	5K	15K	0	0	122K
6612	92		0.2	34K	2K	0	2K	0	0	0	0	0	0	38K
Other	92		0.5	42K	0	0	0	0	0	0	15K	0	0	57K
Total			1.1	160K	5K	12K	5K	0	0	5K	30K	0	0	217K

Milestones

Fiscal Year 1992

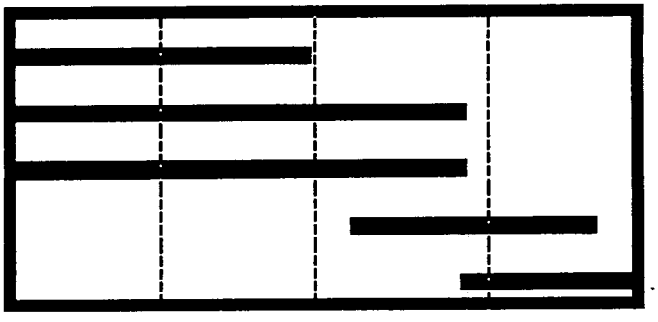
Hardware Development

Software Development

Sensor Development

Assembly

Demo System



Appendix B

Process Control LDRD Project Plan

Process Control LDRD Project Plan

Author: John B. Wronosky

Revision 2, September 1992

Project Definition

This project originated with an LDRD proposal entitled "Sensor Based Process Control." This plan describes the activities that will take place during FY92 and outlines possible direction for continuation in subsequent years. Project operations are being coordinated by Electronic Subsystems Division II, 2337, in partnership with Process Optimization and Control Division, 6612. In March 1992, the partnership expanded to use sensor research from the Anodic Stripping Voltammetry Rinse Bath Monitoring project (Depts. 1824 and 8316).

The following abstract and goal statement summarize the basis for this project.

Abstract

Environmental concerns are playing an increasingly important role in manufacturing processes. Hazardous wastes are by-products of companies involved in etching and plating. A reconfigurable monitoring system that takes advantage of the best features of a suite of sensors is needed to monitor water effluent releases into the environment and to control the manufacturing process to minimize environmental impact. For optimal performance, these sensors need to be fused into a cooperating-sensor-array-based system. This system can be adapted to different environments by changing only site-specific characteristics of the system This system will then initiate output signals that will shutdown or alter the components causing the problem and will provide warnings when dangerous levels of hazardous chemicals are reached.

Goal Statement

The project goal is to develop a configurable demonstration system to monitor a suite of sensors and provide controls to improve product quality and to minimize industrial effluent releases into the environment.

The Process Control LDRD will initially follow a dual path both to develop a plating bath demonstration and to research plating bath technology. The following outline forms a basis for this activity.

Process Control Technology Research

- What has been done?
- What does Industry want?
- What does the NWC need?

- What is the present state of
  - Sensor technology?
  - Process technology?
  - Control technology?

- How can "today" be improved?
- What is feasible in the near future?
  - Prioritize feasibilities

Process Control Demonstration

- Rinse tank monitor and control
- Use cyclic voltammetry sensor for rinse tank monitor.
- Commercial software analysis packages (LabWindows and others)
- Operate long enough to understand system and process.
- Perform demonstration to support research and follow-on LDRD.

Follow-on Activity

- Use demonstration facility to test concepts derived from research path.
  - Sensors
  - Control
  - Process

- Use outside facility (Sparton?) when/if potential technological improvement is theorized.
  - Show application to NWC.

The initial research will be used to define paths for continued activity aimed at improving process control for the printed circuit board (PCB) industry. Three main paths are expected to evolve:

1. Sensor technology - for more accurate monitoring of environmental systems. Waste from plating facilities is a major concern. We expect to show that present sensor technology will not be adequate for future needs.

2. Control technology - to facilitate data gathering and analysis of future chemical and environmental processes. Sensor improvements and requirements for more accurate measurements will require faster, more sensitive hardware and software for monitor and control.

3. Chemical process technology - for improved process, evolving along with monitor and control improvements. New, less environmentally hazardous processes may be capable of equaling or improving the plating quality achieved by present methods.

The demonstration system resulting from this project will be useful in examining each of these paths. New sensors can be developed and monitored. The chemicals can be changed to examine new processes. Monitoring and control of HW and SW can change to accommodate sensor and process improvements. The facility will be designed for flexibility.

Project Requirements

Requirements for completing the first year goals are demonstrating the configurable control system concept and continuing evaluation of sensor and process developments throughout the industry.

Detailed requirements for a configurable control system that can be used for future development of sensors and sensor interface will be formulated during development of the demonstration system. The team will evaluate commercially available software and hardware for use in this application. It is expected that a custom software framework will be used to link an array of commercial interface, display, and analysis software. The hardware platform will be based on a personal computer-like interface but, may use systems designed for industrial applications. Examples are STD products, VME-based systems, and other control-oriented flexible systems. Custom hardware will be kept to a minimum. The best solution may involve the use of different hardware platforms for different applications.

Specific requirements relating to the concept demonstration hardware and software will be documented and used during design and implementation. This will include requirements for implementing the monitor, control, and data manipulation for the Anodic Stripping Voltammetry sensor.

Project Milestones

The following is a list of major milestones toward the first year goals:

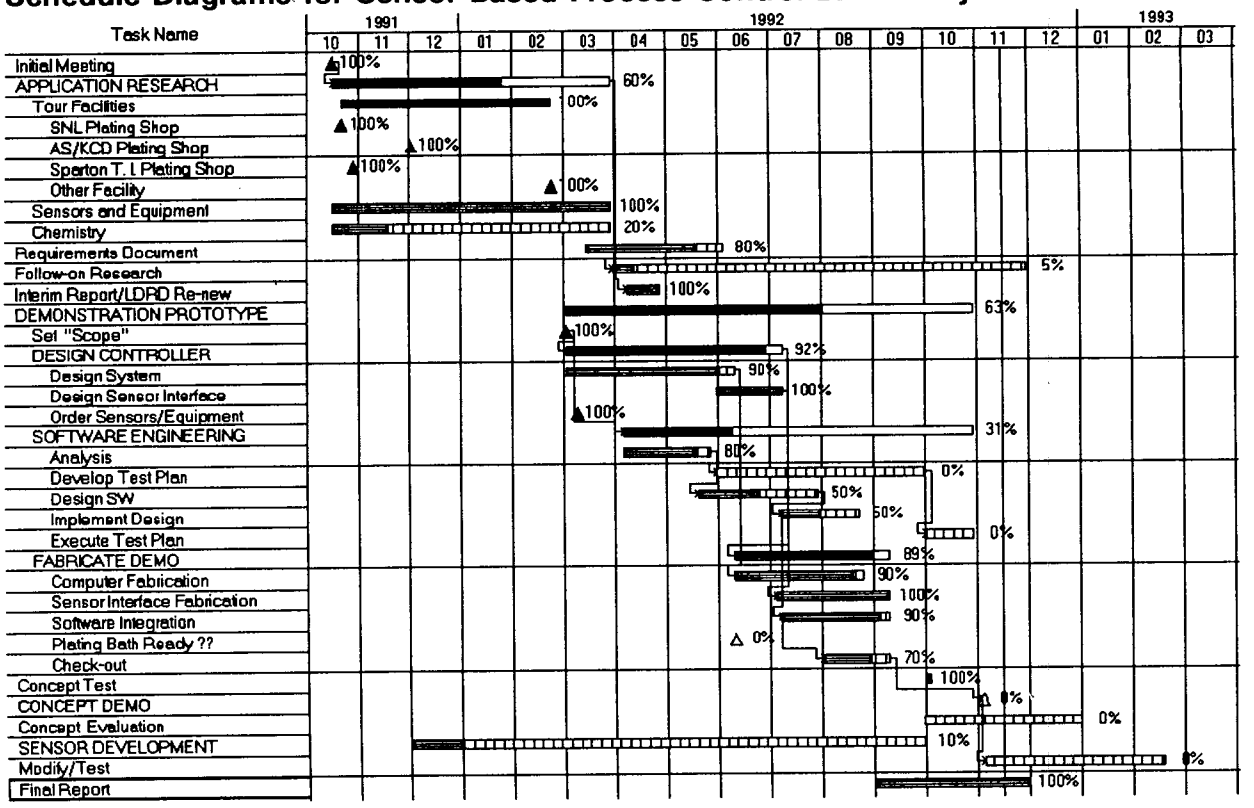
MILESTONE	PLANNED	ACTUAL
1. Conceptual Meeting	10/17/91	10/17/91
2. Compile Demonstration Req.	3/18/92	5/27/92

3. Interim Report (Renewal LDRD Proposal)	5/22/92	4/24/92
4. Concept Demonstration	7/92	11/92
5. Concept Report	9/30/92	12/92

Work Breakdown Structure

Application Research	John B. Wronosky Jerry T. Love Dept. 6612 representative
Requirements	John B. Wronosky
Interim Report	John B. Wronosky Jerry T. Love Dept. 6612 rep.
Demonstration Prototype	John B. Wronosky Dale Brandt
Demonstration Fabrication	Dale J. Brandt
Customer development	Jerry T. Love W. David Williams
First year report	John B. Wronosky
Follow-on Research	John B. Wronosky Jerry T. Love Dept. 6612 rep.
Demo system modification	Dale J. Brandt

Schedule Diagrams for Sensor Based Process Control LDRD Project Schedule



Budget

This project is budgeted for $236K. Department 2337 is allocated $186K and Department 6612 $49K.

Project Organization

<u>Monitor and Control</u>

Dept. 2337, Real Time Monitors & Controllers

Director	Ronald D. Andreas	845-8353
Manager	W. David Williams	844-7659
Project Leader	Jerry T. Love	845-8202
Technical Project Leader	John B. Wronosky	845-8427
Project Engineer-SW	Laney C. Kidd	844-1242
Design Engineer	Dale J. Brandt	844-4710

Plating Bath and Chemistry

Dept. 6612, Process Optimization and Control Department

Director	Joan B. Woodard	845-9917
Manager	Art E. Verardo	845-9614

Anodic Stripping Voltammetry sensor

Dept. 1824, Process Characterization Department

Director	Alton D. Romig, Jr.	844-8358
Manager	Michael R. Keenan	844-2190
Project Engineer	Vicki E. Granstaff	844-8789

Dept. 8316, Materials Technology Department

Director	R. C. Wayne	510-294-3306
Manager	Ron E. Stoltz	510-294-2162

Logistics Support

Additional specialists may be called upon as the need arises. Among these might be specialists in Dept. 2473-3, Plating and Coating Section, to demonstrate the system on an operating plating line.

Security

The process control LDRD is an unclassified project. Proprietary information will be handled appropriately.

Environmental, Safety, and Health

Some of the chemicals that will be used during demonstration of the control system are considered hazardous. All applicable ES&H rules and regulations will be strictly adhered to during this project.

Customer Organization Contact Points

Monitor and Control System

Dept. 2337, Electronic Subsystem Division II

Project Leader	J. T. Love	845-8202
Technical Project Leader	J. B. Wronosky	845-8427

Plating Bath and Chemistry

Dept. 6612, Process Optimization and Control Division

Manager	Art E. Verardo	845-9614

Sensors

Dept. 1824, Process Characterization Division

Project Engineer	Vicki E. Granstaff	844-8789

Dept. 8316, Materials Technology Division

Supervisor	Ron E. Stoltz	294-2162

Nature of Project Reviews

Team meetings will be scheduled regularly to ensure and maintain proper direction in the project. Frequent interaction with customers will also be scheduled to ensure adherence to requirements.

Appendix C

Articles Referenced from Literature Search

Articles Referenced from Literature Search

1. J. C. Uhrmacher, P. E. and P. P. Werschulz, *Automated Control of Electroplating Chemical Process in Printed Wiring Boards - State of The Art,* Proceedings of the 1982 American Control Conference,Vol. 3, pp. 931-935.

   Did what we're doing, but in 1982. Will be good to compare.

2. W.D. Bonivert, J.S. Krafcik, Jr., J. T. Hachman (SNL); J.C. Farmer (LLNL), *Development Report, An In-Situ Sensor for Monitoring Organic Additives in Copper Plating Solutions;* ???

   Developed probe for monitoring brighteners. Uses PC-based monitoring system.

3. John Donaldson and Wally Myers, *Automatic Plating Systems: A Tutorial, Parts I and II,* Metal Finishing, October 1991.

   Speaks of plating shops going out of business if they do not automate.

4. Mamoru Uenoyama and Anthony Revier, *The Recent Trend in Automatic Control Technology for Plating Solutions,* Metal Finishing, May 1991.

5. Gregory K. McMillan, *Understanding Some Basic Truths of pH Measurement,* Monsanto Co. Chemical Engineering Progress, October 1991.

   Says pH measurement is difficult. Recommends using three sensors. Says control is magnitudes more difficult due to pH sensitivity, rangeability, and nonlinearity.

6. Donald L. Snyder, *Electroplating in the 90s,* Metal Finishing, April 1991.

   Mostly pertaining to the automotive industry. Does point out the need to reduce waste.

7. Jay Zacks, *Increasing Plating Thickness Control with On-Line Thickness Monitoring, Micro-ohm Technology,* Metal Finishing, September 1990.

   Lists factors affecting thickness control. Describes a sensor for electroplating for which "the thickness of plating". . . "is inversely proportional to the resistance of the plating." "A microprocessor in the system converts the change in resistance to a reading in micro-inches." Includes a statistical analysis of plating thickness for an electroless nickel process.

8. Greg Martiska, Brinkman Instruments, *On Line Monitoring of Nickel Plating,* Metal Finishing, September 1990.

Coating of steel by the automotive industry using an on-line titro-analyzer. Possibly a good example of what is presently acceptable to industry.

9. Masao Matsuoka and Tadao Hayahi, Universty of Osaka Perfecture, Analysis of Mass Transfer in Electroless Copper Deposition, Metal Finishing, May 1989.

   Focuses on the influence of mass transfer on the reaction kinetics of electroless copper deposition.

10. Nicholas M. Martyack and Bruce McDuffie, University of New York, *Side Reactions in Electroless Copper, Part I- Distillation of Ammonia; Part II- Sparging of Ammonia,* Metal Finishing, October 1990.

    Experiment to detect and remove ammonia formed in a electroless copper bath.

11. John Donaldson, *Computerization of Surface Finishing Systems,* Metal Finishing, March 1990.

    Answers question: Why Computerize? Talks about need to include process control along with equipment control. Lists essential components to computerization. Analyzes cost to computerize a plating operation with $2M sales.

12. Catalog: Electro-chemical Devices, Inc,. Supplied by Brad Buffington, Southwest Controls, Inc., (505) 828-0958.

    Instruments and sensors to measure pH, specific ion, ORP(pE), dissolved oxygen, conductivity, resistivity, temperature. Includes engineering data with index of electrolytes.

13. Raymond H. Clark, *Handbook of Printed Circuit Manufacturing,* Van Nostrand Reinhold Company, 1985, Section 5.

14. Clyde F. Coombs, Jr., *Printed Circuits Handbook, Third Edition,* McGraw Hill, 1988, Chs. 12, 13, and 15.

15. T.R. Matton, P. McSiggen, and S.A. George, Hewlett-Packard, *Printed Circuit Plating Bath Process Control, Part I,* Metal Finishing, February 1985

    Describes UV/visible spectrophotometric determinations using diode array spectrophotometer for precise control of plating solutions. Gold, acid tin, and nickel plating. Describes spectral analysis for these components.

16. T.R. Matton, P. McSiggen, and S.A. George, Hewlett-Packard, *Printed Circuit Plating Bath Process Control, Part II- Bath Analysis Using a Diode Array Spectrophotometer*, Metal Finishing, February 1985.

Acid, copper, tin, gold, catalyst, and nickel examined. This part adds automation to the analysis. Spectrophotometric results are compared to accepted measurement methods to show that a variety of PCB plating baths can be analyzed on a single instrument.

17. T.R. Matton, P. McSiggen, and S.A. George, Hewlett-Packard, *Printed Circuit Plating Bath Process Control, Part III - Automated Bath Analysis,* Metal Finishing, February 1985.

Describes the philosophy and methodology a specific BASIC program should follow for bath analysis within a given facility. Shows the flow used to analyze a given bath by sampling a reference and then the bath in question.

This is a good summary of how spectrophotometry can be used. It looks like the mechanics of this system would be complicated but do-able.

18. Dennis R. Turner, *The Role of Sensors in Automatic Electroplating,* Plating and Surface Finishing, June 1986.

Describes various sensors used in the complete automation of plating facilities. Good overview.

19. Roger A. Smith, *Operators Keep Close Tabs on Plating Process with Touch-screens,* Chiltons Instrument and Control Systems (I&CS), June 1989.

Interesting article. Describes how Touch-screens are used to report monitored data to operators and central computer in a plating line. Graphs can be displayed showing current parameters and history. Operators get more than I/0 feedback, and information can be used to control process.

20. Jerry Murray, *Automated Wet Processing - A Subcontinent of Automation,* Circuits Manufacturing, February 1988.

Article describes wet process control and how it can be done in levels. An entire line doesn't need to be controlled to benefit.

21. M. Eliacin and F. Scaraclino, *Hi-rel Multilayer Plated-through Hole Utilizing the EE-1 Process,* Circuit World, October 1988.

Describes a method of plating through holes for multi-layer boards that replaces electroless copper. Claims higher quality, less waste, and better consistency.

22. Timothy L. Hudson, Assoc. Ed., *The Plating Process Optimizes PCB Performance,* Electronic Packaging and Production, September 1991.

Excellent overview of an electroless copper process.

23. Saad Nakhla and Robert Herberger, *Control of Electroless Copper,* Printed Circuit Fabrication, September 1988.

Describes a monitor and control method that monitors copper concentration, pH, and bath temperature, and recognizes interactions that occur in the bath.

24. Mark Pederson, *Plating Flex Circuits,* PC Fab. April 1991.

Describes the process by which Hughes Aircraft selected a system for automation of their Flex circuit line.

25. S.S. Heberling, K. Haak, S. Carson, and M. Doyle, Dionex Corp., *Analytical Techniques to Monitor and Control Printed Circuit Plating,* Electronic Manufacturing, June 1988.

Describes the application of ion chromatography for determining chemical properties necessary for controlling a wide variety of plating processes.

26. B. Pledger, Technograph & Telegraph Ltd., *Controlling Chemical Processes,* Electronic Production, February 1980.

General discussion of why it is a good idea to control chemical processes.

27. C. B. Castner, Shipley Co., *Automatic Control of Stabilized Electroless Solutions,* Proceedings of PC'81 4th Annual International Printed Circuits Conference, June 2-4, 1981.

Calorimetric in tank probe monitoring system and others discussed.

28. Doug Osadchuk, Anthony R. Revier, and Mamoru Uenoyama,Uyemura International Corp., *Controlling Full-Build Electroless Copper,* Metal Finishing, January 1992.

Discussion of the capabilities and advantages of full-build control.

Appendix D

Development System Design

Development System Design

Development Resources

Computer Platform:	Dolch PAC 386-33C portable.
Software Development:	National Instruments LabWindows version 2.1 interactive programming environment and Microsoft Quick C version 2.5.

Major Components

Computer:	Dolch PAC 386-33C portable Note: Industrial chassis with shielded electronics would provide better noise immunity.
Data Acquisition Card:	National Instruments AT-MIO-16F-5 Multifunction I/O. 12 bit ADC, two 12 bit DACs, 8 Digital I/O3 16 bit Counter/Timers, 16 bit DMA, LabWindows supported.
I/O Adapter Card:	National Instruments SC-2070 general purpose termination breadboard.
Potentiostat:	Custom circuitry for driving ASV
ASV:	Prototype electrode developed by 1824.
Conductivity Sensor:	Omega CDCN-36-EG Electrode with CDCN-36 Conductivity controller. Analog voltage output @ 1mv/least significant digit of readout.
Temperature Sensor:	Omega TMTSS-125U-12 Thermocouple, 12", type "T". driven by SC-2070.
pH sensor:	Fisher Scientific 13-620-17 electrode with Fisher Accumet model 910 pH meter. Recorder output 0-$\pm$700mv, 0.01pH = 1 mv.

Block Diagram

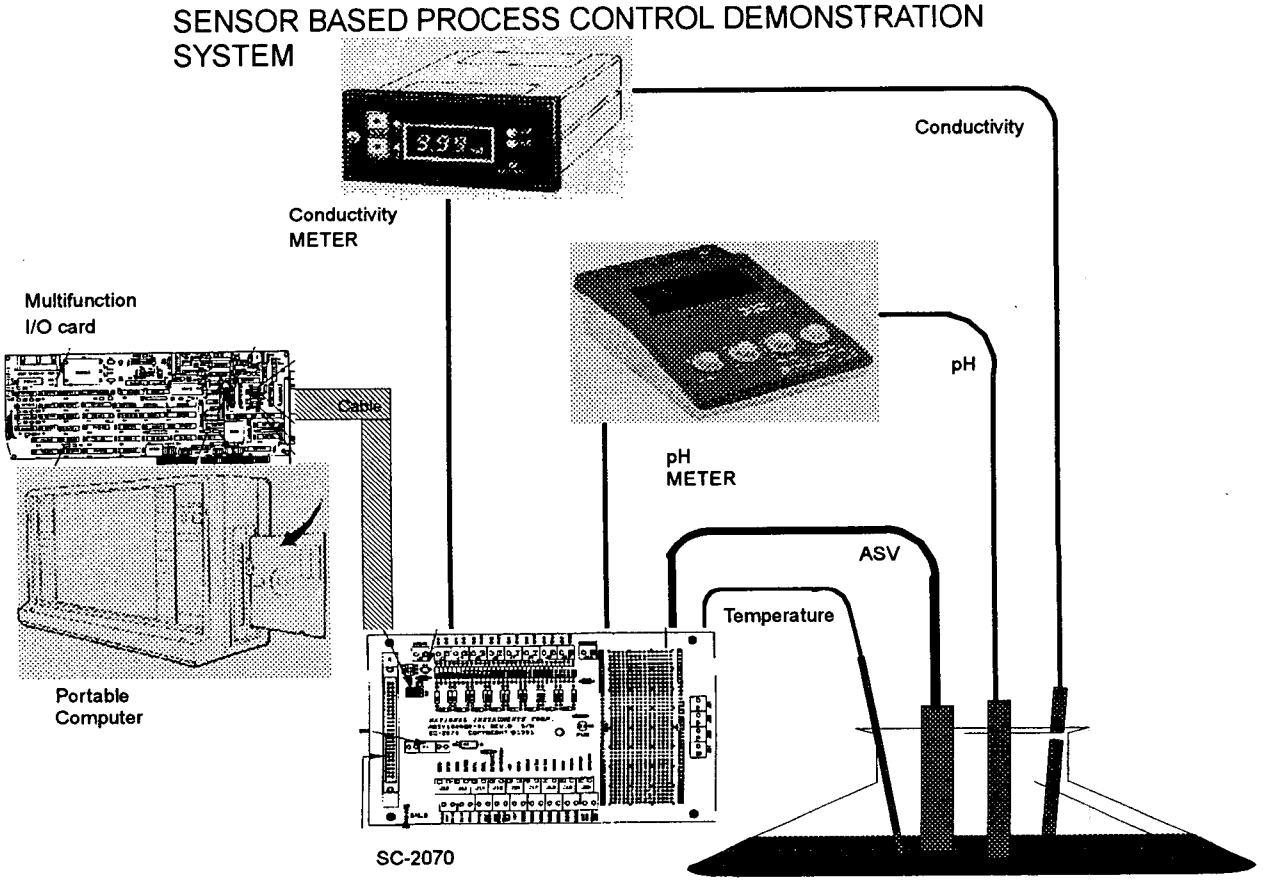


Figure 1. SBPC Demonstration System

Custom Electronics Descriptions

Modifications and Connections to SC-2070 board

The SC-2070 board provides general purpose termination for the multifunction data acquisition card. It allows for custom modification for signal conditioning and special applications. Some modifications are installed on our board. Filtering was added for the temperature, pH, and conductivity inputs. The potentiostat circuit used to drive the ASV is fabricated on the SC-2070 breadboard space.

The thermocouple interface was modified to add a low pass filter for noise reduction. The conductivity and pH inputs were ground referenced to eliminate noise in the data. Additional analysis is needed to determine the exact source of the noise appearing on these signals.

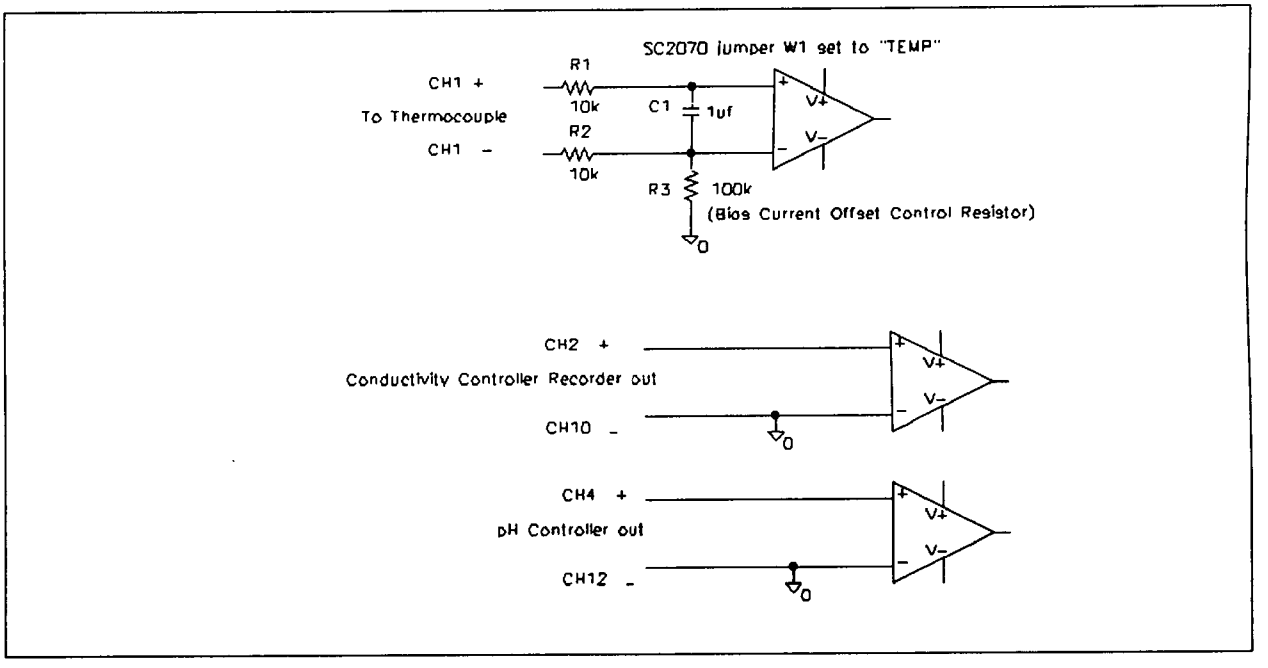


Figure 2. Temperature, Conductivity, and pH Interface to SC2070

Potentiostat

The circuit of Figure 3 is a potentiostat used to drive the ASV cell and to convert the current flow through it to a voltage readable by the A-D input of the Multifunction card. The sign inverter stage is used to convert the negative polarity D-A output from the Multifunction card to positive polarity for input to the drive stage. A negative output polarity was used to facilitate evaluation of another potentiostat design. The sign change could be implemented in software. The drive stage provides a positive potential to the counter electrode. Current flow increases or decreases through the working electrode as a result of changes in the cell impedance. The drive current is controlled by the reference electrode potential. Voltage changes at the reference electrode are coupled to the inverting input of the drive stage such that its output varies the ASV cell current. Current flow through the reference is very small because of the high impedance input of the opamp. The final stage provides current-to-voltage conversion for measuring the current flow through the working electrode. Current flowing into the stage is balanced by current leaving the input through the 1M Ω resistor. The output potential (A/D CH11) is equal to the working electrode current multiplied by the value of R3.. The 0.01 uf capacitor was added to reduce noise at the monitor output.

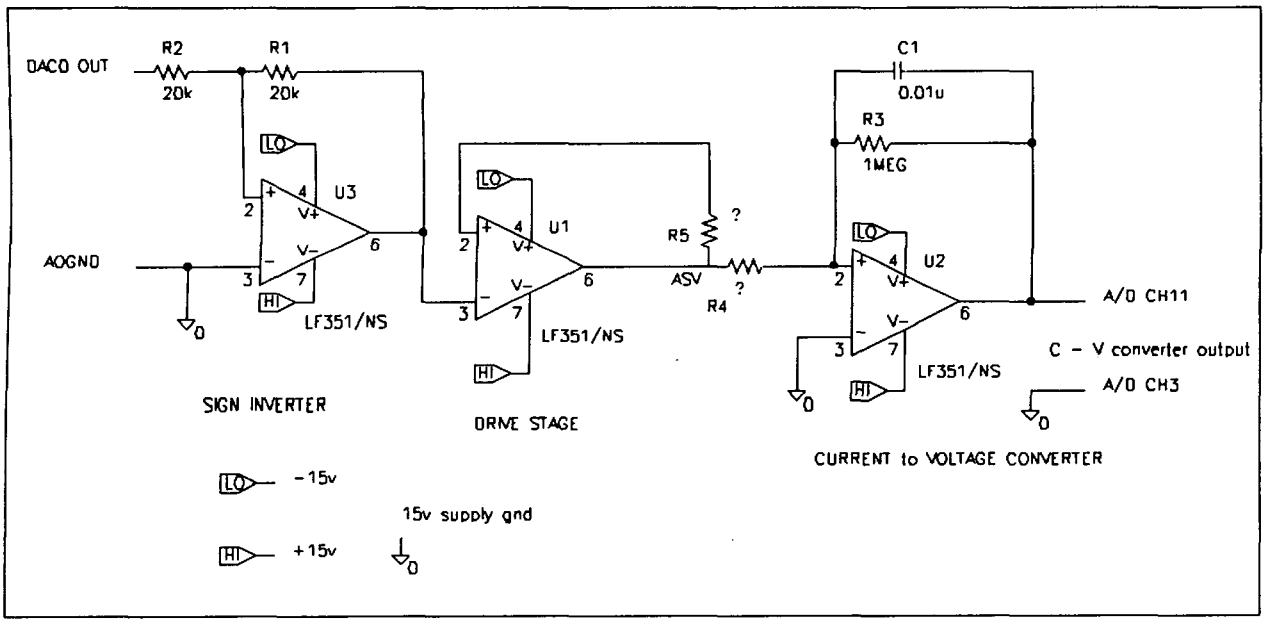


Figure 3. ASV Potentiostat Circuit

Potentiostat Stimulus

The Potentiostat circuit is excited with the Differential Pulse waveform illustrated in Figure 4. For the demonstration system, the pulse amplitude is 50mv, pulse width is 30 ms, pulse period 300 ms, and E step (ΔE) 5mv. The sample width is 17 ms.

Samples are taken during the last half of the pulse as shown in the figure, and just before the rise of each pulse. The difference of the samples associated with a given pulse is used to compute the electrode current for each pulse.

A complete data run functions as follows:

- Clean Time(30 sec at 0v)
- Quiet Time (10 sec at -800mv)
- Background Curve and data acquisition (Differential Pulse waveform beginning at -800 mv and ending at -300 mv)
- Deposition Time (30 sec at -800 mv)
- Quiet Time (10 sec at -800mv)
- Normal Curve and data acquisition (Differential Pulse waveform beginning at -800 mv and ending at -300 mv)
- Final Potential (0 v), and acquisition of Temperature, Conductivity, and pH readings.

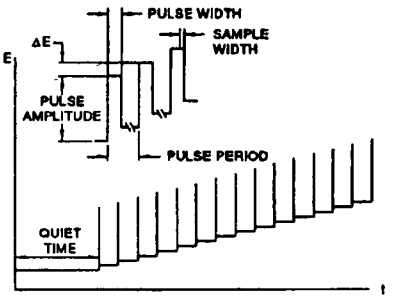


Figure 4. Differential Pulse

The resulting data is the difference between the Normal and Background data curves.

Software Description

The demonstration system software is written in C language using the LabWindows interactive programming environment and compiled and linked with Microsoft Quick C.

The program is menu driven and provides capability for output file control, data acquisition, and data analysis. Figure 5 shows the Main Panel screen after a test. The status box indicates the progress as the test is run. Figure 6 shows the data file selection menu. The output file name can be selected from a list of existing files, or a new name can be entered. The resulting file contains a text list of the electrode current referenced to the input

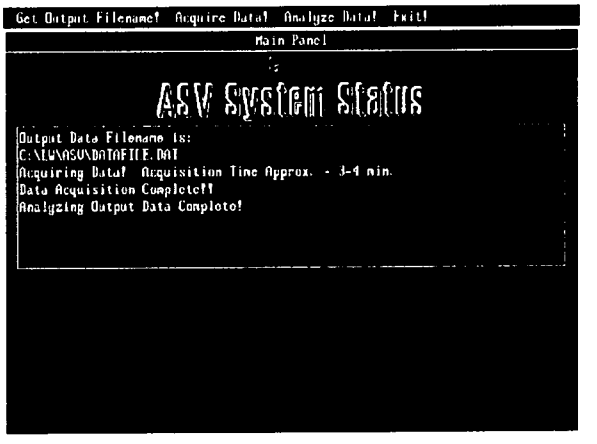


Figure 5. Main Panel

potential. Figure 7 shows the graphical output screen as displayed when the Analyze Data! menu item is selected. The display includes a graph of the ASV cell current vs. potential; peak potential and current for cadmium and lead; solution temperature, conductivity, and pH; the set concentration limit; and a Concentration Limit indicator. The indicator will be activated when the concentration is exceeded. The concentration limit (expressed as peak current) may be set by the user. This screen is dismissed by selecting the Done button.

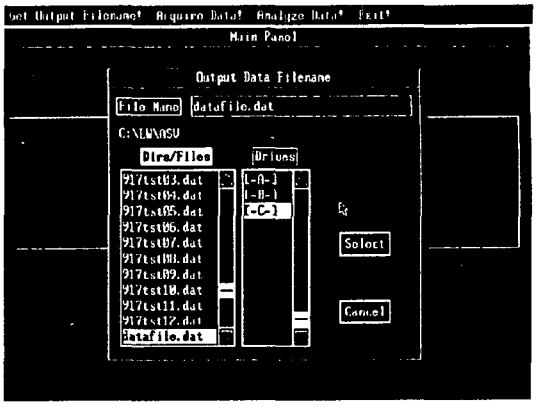


Figure 6. Data File Selection

ASV Data Acquisition Timing Scheme (N.I. AT-MIO-16F)

A "Ramping Step" (Figure 4) signal is generated on an "Analog Out" line (potentiostat control) while taking voltage readings on an "Analog In" line (potentiostat output). One "Analog In" data point is recorded during the negative half and one during the positive half of the ramping step output pulse. Multiple data samples are acquired and averaged for each resultant single recorded data point.

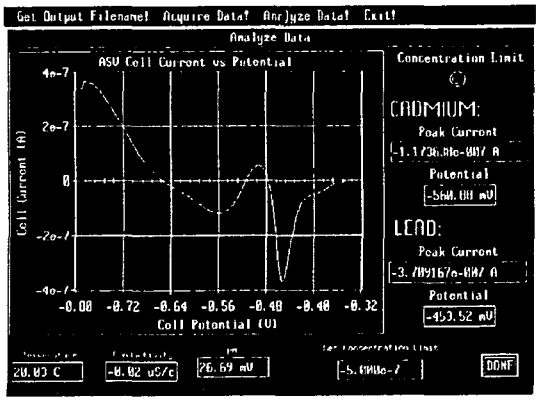


Figure 7. Output display

For a full acquisition window (-800mV to -300mV), a single Data Acquisition is started using the DAQ_Start() function with "Half-Buffering" enabled. Data is acquired into a "Circular Buffer" (described below) of size 100 pts (samples) at a sample interval rate of 170 uS/sample. This allows for a full sample acquisition time of 17 mS (100 samples X 170 uS/sample = 17 mS). The 100 samples acquired during the 17 mS are averaged to generate a final data point. A 17 mS sample window was chosen to average out any 60 Hz noise present on the acquired data.

Data acquired into a "Circular Buffer" means that once the buffer is filled, acquired data will be filled again into the beginning of the buffer overwriting previously acquired data. This begins with a single DAQ_Start() function call and halts with a single DAQ_Clear() function call. "Half-Buffering" allows half of the "Circular Buffer" to be processed while the other half is being acquired. DAQ_DB_HalfReady() can be called to determine when the next half of the circular buffer is ready to be processed. When half of the circular buffer is full, a secondary "Half-Buffer" (1/2 the size of the circular buffer) receives a copy of the recently acquired data via a DAQ_DB_Transfer() function call. Care must be taken to process the "Half-Buffer" before another "Circular Buffer" is filled to prevent losing data.

ASV 10 Weight Finite Impulse Response (FIR) Filter

Each data point resulting from an FIR filter is the result of the sum of a finite set of input samples that have been multiplied by individual coefficients. The coefficients are determined by the desired cutoff frequency and magnitude response. It was determined experimentally that a tenth order filter with 30 Hz cutoff frequency is adequate for this application.

A 10-weight FIR filter is implemented on the acquired ASV data samples. The FIR filter takes the form:

$$H(z) = a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots\ldots\ldots a_k z^{-k} \qquad \text{or} \qquad y(n) = \sum_{i=0}^{k} a_i x(n-i) .$$

where H(z) and y(n) are present filtered data point outputs, "a" is the filter weight coefficient, "z" and "x" are present and previous unfiltered data samples, respectively. The filter weight (or order) is represented by "k" and is 10 for this application.

The 10-weight FIR coefficients used are: $a_0 = 0.0145$; $a_1 = 0.0306$; $a_2 = 0.0725$; $a_3 = 0.1245$; $a_4 = 0.1665$; $a_5 = 0.1826$; $a_6 = 0.1665$; $a_7 = 0.1245$; $a_8 = 0.0725$; $a_9 = 0.0306$; $a_{10} = 0.0145$

ASV Data Smoothing

A five-point, moving average, data smoothing technique is applied to the acquired $I_{forward}$ - $I_{reverse}$ "background" and "normal" ASV data curves. The technique deals with a moving window of five data points at a time. The third data point within the window becomes the average of the remaining four data points. On the first five data points, the first three become the average of the first, second, fourth, and fifth data points. The same is true for the last five data points, the last three points become the average of the outer four points.

First Smooth:	1	~~2~~	~~3~~	4	5	:	1; 2; 3 = (1+2+4+5) / 4
Successive Smoothes:	2	3	~~4~~	5	6	:	4 = (2+3+5+6) / 4
Final Smooth:	96	97	~~98~~	~~99~~	~~100~~	:	98; 99; 100 = (96+97+99+100) / 4

File Descriptions

ACQUIRE.C	Waveform generation and data acquisition code
ASV.APP	LW generated file
ASV.C	The Demonstration Program code
ASV.EXE	The Demonstration Executable code
ASV.H	The Demonstration include file (defines)
ASV.OBJ	The compiled deomonstration code

ASV.UIR	The LabWindows user interface resource file for Demo program
DACAOWRT.C	Analog output test program
DACTEST.APP	LW generated file
DACTEST.C	DAC output test program waveform out from buffer test
DATAFILE.DAT	Example data file
FILETEST.C	Open a file and write formatted data to it test code
FILTREG.DAT	Filtered half buffer data from last run test.
FILTTIME.C	Checks the time required to perform a 10 weight FIR filtering of data.
FILTTIME.EXE	Executable for FILTIME.C
FILTTIME.OBJ	Compiled FILTIME.C
GETCON .C	Tests the acquisition of conductivity reading.
GETPH.C	Tests the acquisition of pH reading.
GETTEM .C	Tests the acquisition of temperature reading.
IF_IR.DAT	Unsmoothed raw forward - reverse data.
S_IF_IR.DAT	Smoothed forward - reverse data.
TIMER.C	Tests the use of an internal timer to determine the passage of time.
UNFILTRG.DAT	Unfiltered half buffer data from last run test.
WAVEFORM.C	Waveform output from buffer test.

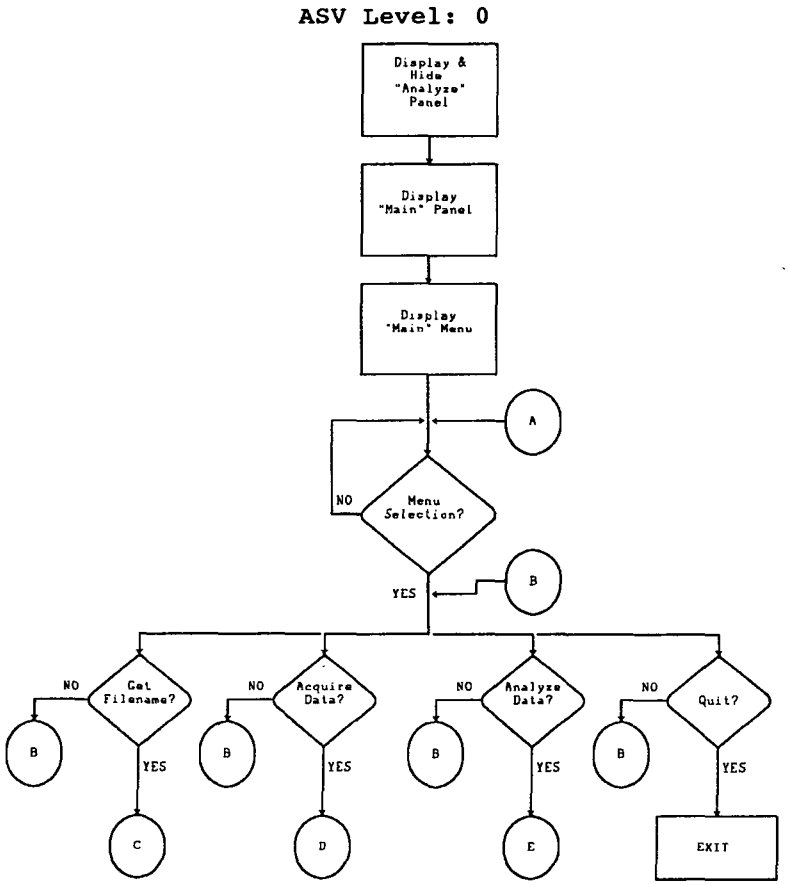
LabWindows Libraries Used

User Interface, Data Acquisition, Formatting and I\O, LabWindows System for C.

Appendix E

Demonstration System Software Flow Charts

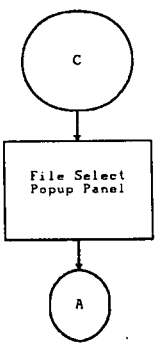
Demonstration System Software Flow Charts



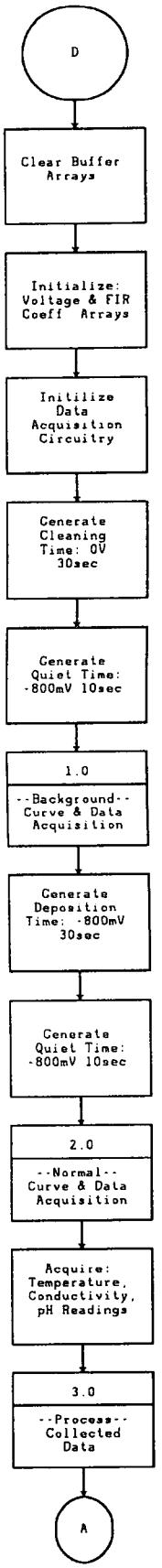
ASV Flow Top Level.

The following flow charts illustrate the operation of the ASV software for the Sensor Based Process Control demonstration System. Chart "ASV Level: 0" shows the top level system control. "ASV Level: C" is the file select path. "ASV Level: D" is the acquire data path. "ASV Level: E" is the analyze and display data path. "ASV Level: D1.0, 2.0, & 3.0" provide detail for "ASV Level: D."

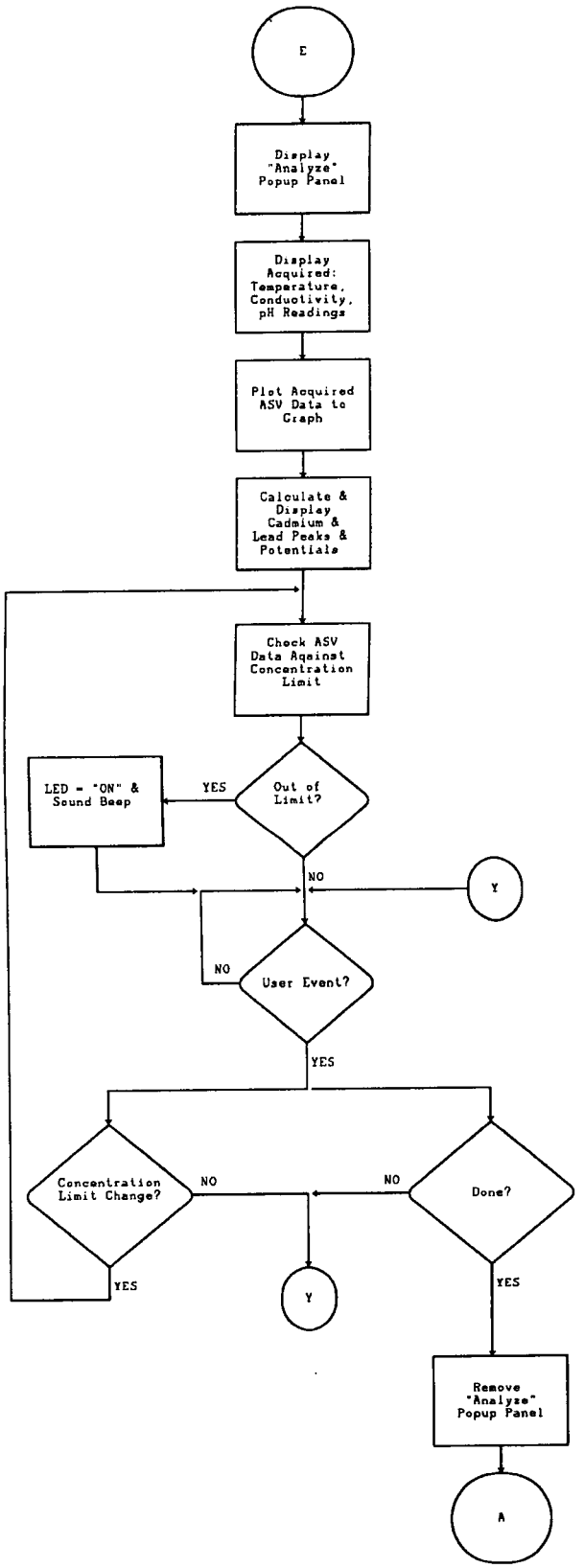
ASV Level: C



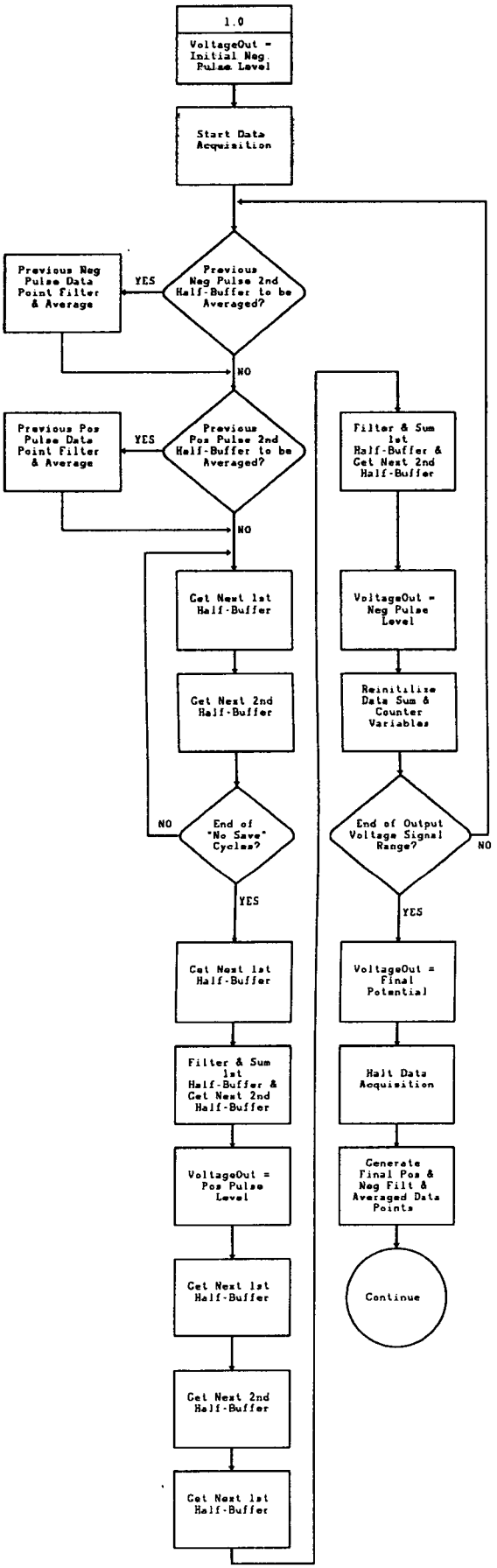
ASV Level: D



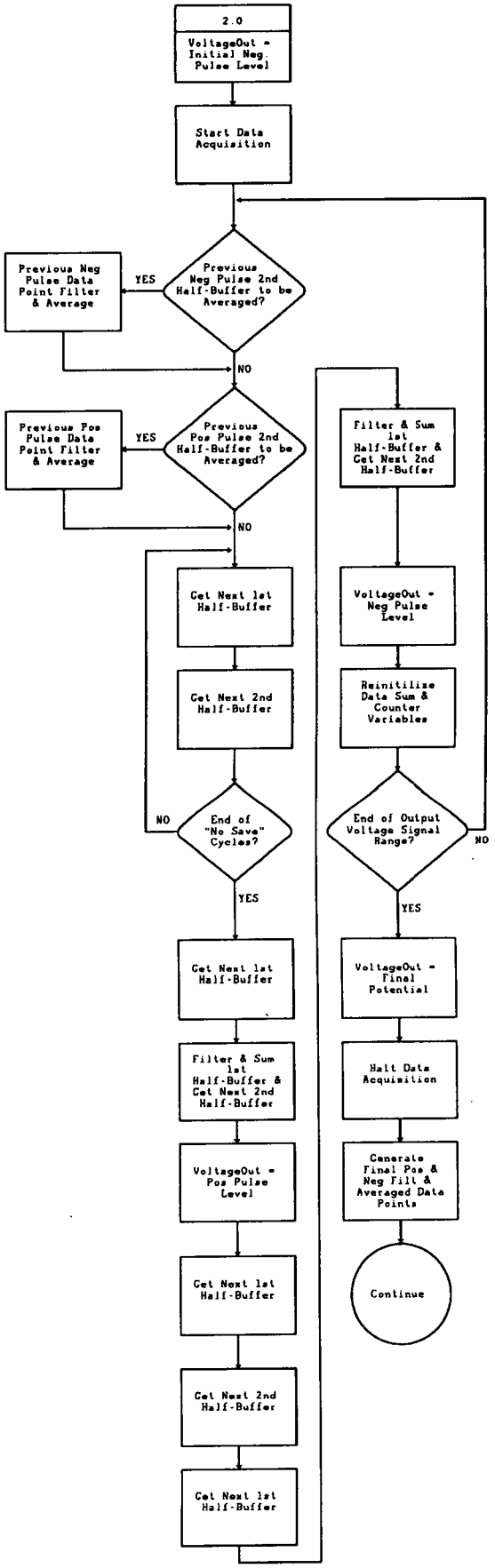
ASV Level: E



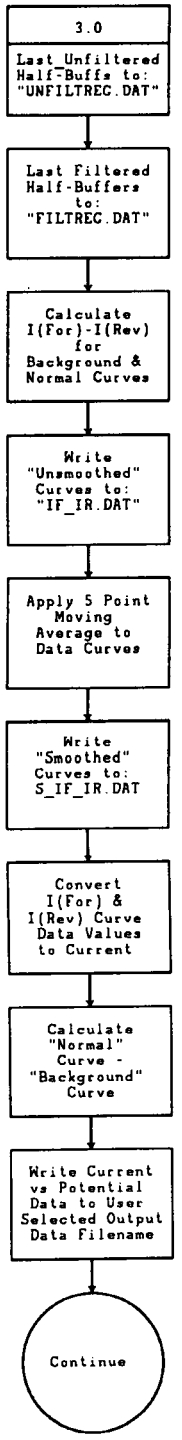
ASV Level: D1.0



ASV Level: D2.0



ASV Level: D3.0



Appendix F

Development System Code with Comments

Development System Code with Comments

```
/*****************************************************/
/* Process Monitor & Control :    ASV          */
/* Author: Dale Brandt    Dept. 2337           */
/* Sandia National Labs       Albuquerque, NM     */
/*****************************************************/

/**/
/*****************************************************************/
/*****              Process Monitor & Control              *****/
/*****                     Main.c                     *****/
/*****************************************************************/

#include "C:\LW\include\lwsystem.h"
#include "C:\LW\include\userint.h"
#include "C:\LW\asv\asv.h"
#include "C:\LW\include\dataacq.h"
#include "C:\LW\include\formatio.h"


void Get_Output_Filename();
void Acquire_Data(double REG_Positive_Data_Buffer[], char
Output_Filename_Buffer[]);
void Analyze_Data();
void Positive_ProcessAvg_Transfer(int Pos_Half_Buffer[], int *Half_Buffer_Ready,
                        double *Pos_Data_Sum, int *Pos_Data_Pt_Count,
                        int *DAQ_Finished, int Save_Flag,
                        double Coeff_Array[], double Filtered_Array[]);
void Negative_ProcessAvg_Transfer(int Neg_Half_Buffer[], int *Half_Buffer_Ready,
                        double *Neg_Data_Sum, int *Neg_Data_Pt_Count,
                        int *DAQ_Finished, int Save_Flag,
                        double Coeff_Array[], double Filtered_Array[]);
void Average_Negative_Data(int Neg_Half_Buffer[], double *Neg_Data_Sum,
            int *Neg_Data_Pt_Count, double Negative_Data_Buffer[],
            int *Neg_Data_Index, double Coeff_Array[],
            double Filtered_Array[]);
void Average_Positive_Data(int Pos_Half_Buffer[], double *Pos_Data_Sum,
            int *Pos_Data_Pt_Count, double Positive_Data_Buffer[],
            int *Pos_Data_Index, double Coeff_Array[],
            double Filtered_Array[]);
```

```
#define YES 1
#define NO 0
#define Initial_Potential -164   /*Initial Operating Window Potential*/
#define Final_Potential -61      /*Final Operating Window Potential*/
#define Cir_Buf_Size 100         /*Circular Buffer Size*/
#define Half_Buf_Size 50         /*Half Buffer Size*/
#define Filter_Weight 10         /*Filter Weight*/
#define Filter_Buf_Size 40       /*Filtered Array Size(HalfBufSize-Filter_Weight)*/
#define Data_Buf_Size 100        /*Final Data Buffer Size*/
#define Voltage_Buf_Size 200     /*Voltage Signal Array Size*/
#define Neg_HalfSig_NoSave_CycleCnt 14 /*# of Reverse V. No Save Cycles*/
#define Measure_Resistor 1000000 /*Current Measure Resistor Value*/
#define Final_Filt_Buf_Element 92  /*Final Filtered If-Ir Data Array*/

int  Continue,                /*Main Panel Display Control Variable*/
     Limit_Flag,              /*Concentration Limit Flag*/
     Main_Menu_Handle,
     Main_Panel_Handle,
     Analyze_Panel_Handle,
     Event_Panel_Or_Menu,
     Event_Control_Or_Item;

 double REG_Positive_Data_Buffer[Data_Buf_Size], /*Data Buffer that holds
                                                  final curve Data*/
       Temperature_Reading,          /*Final Temperature Reading*/
       Conductivity_Reading,         /*Final Conductivity Reading*/
       pH_Reading;                   /*Final pH Reading*/

 char Output_Filename_Buffer[80],      /*Output Filename Buffer*/
     Temperature_Display_Buffer[15],  /*String w/ Temp. Reading for dsply*/
     Conductivity_Display_Buffer[15], /*String w/ Cond. Reading for dsply*/
     pH_Display_Buffer[15];           /*String w/ pH Reading for dsply*/

void main()
{
 /****** Initialize *****/

    /***** Load/Display/Hide Analyze Panel For Future Popup  *****/
 Analyze_Panel_Handle = LoadPanel ("c:\\lw\\asv\\asv.uir", Analyze);
 DisplayPanel (Analyze_Panel_Handle);
 HidePanel (Analyze_Panel_Handle);

    /***** Load/Display Main Menu Bar  *****/
 Main_Menu_Handle = LoadMenuBar ("c:\\lw\\asv\\asv.uir", MainMenu);
```

```
    /***** Load/Display Main Panel *****/
Main_Panel_Handle = LoadPanel ("c:\\lw\\asv\\asv.uir", MainPanel);
DisplayPanel (Main_Panel_Handle);

Limit_Flag = NO;

Continue = YES;
while (Continue = = YES)
 {
    /***** Get User Event From Main Menu or Main Panel *****/
    /***** "Control" from Panel; "Item" from Menu Bar *****/

  GetUserEvent (1, &Event_Panel_Or_Menu, &Event_Control_Or_Item);

    /***** Determine What User Event Occured *****/

  switch (Event_Control_Or_Item)
   {
    case MainMenu_OutFile:
        Get_Output_Filename();
        SetCtrlVal (Main_Panel_Handle, MainPanel_SystemStatus, "Output Data
Filename is:");
        SetCtrlVal (Main_Panel_Handle, MainPanel_SystemStatus,
Output_Filename_Buffer);
        break;

    case MainMenu_Acquire:
        SetCtrlVal (Main_Panel_Handle, MainPanel_SystemStatus, "Acquiring Data!
Acquisition Time Approx. = 3-4 min.");
        Acquire_Data(REG_Positive_Data_Buffer, Output_Filename_Buffer);
        SetCtrlVal (Main_Panel_Handle, MainPanel_SystemStatus, "Data Acquisition
Complete!!");
        break;

    case MainMenu_Analyze:
        Analyze_Data();
        SetCtrlVal (Main_Panel_Handle, MainPanel_SystemStatus, "Analyzing
Output Data Complete!");
        break;

    case MainMenu_Exit:
        Continue = NO;
        break;
```

```
/*****************************************************************/
/*****************************************************************/
/*        Function: Get Output Filename                          */
/*****************************************************************/
/*****************************************************************/

void Get_Output_Filename()
{
 FileSelectPopup ("", "*.DAT", "Output Data Filename", 0, 0, 1,
Output_Filename_Buffer);
}



/*****************************************************************/
/*****************************************************************/
/*          Function: Analyze Data                               */
/*****************************************************************/
/*****************************************************************/
void Analyze_Data()
{
 int y,              /*Generic Index Variable*/
    Popup_Event_ID,  /*User Event ID for "Analyze" Popup Panel*/
    DisplayPopup;    /*Analyze Popup Panel Display Control Variable*/

 double Limit_Value,  /*User Changeable Concentration Limit*/
       Potential,    /*Peak_Value Potential Variable*/
       Peak_Value;   /*Data Curve Peak Value Variable*/

 char Temp_Display_Buffer[20]; /*Temporary String Buffer for Holding
                                Formatted Peak & Potential Data for
                                Display*/
 DisplayPopup = YES;

 /*** Display: Analyze Popup Panel\Temperature, Conductivity, pH\Graph Data**/
 InstallPopup (Analyze_Panel_Handle);
 SetCtrlVal (Analyze_Panel_Handle, Analyze_Temp, Temperature_Display_Buffer);
 SetCtrlVal (Analyze_Panel_Handle, Analyze_Cond, Conductivity_Display_Buffer);
 SetCtrlVal (Analyze_Panel_Handle, Analyze_Ph, pH_Display_Buffer);
 PlotWaveform (Analyze_Panel_Handle, Analyze_ASVGraph,
REG_Positive_Data_Buffer,
           (Data_Buf_Size-1), 4, 1.0, 0.0, -0.800, .00488, 1, 10, 1, 15);
```

```
/***** Determine and Display Cd/Pb Peak and Potentials *****/
            /**** Cd ****/
 Potential = 0;
 Peak_Value = 0;

 for (y = 0; y < (Data_Buf_Size / 2); y++)
  {
   if (REG_Positive_Data_Buffer[y] < Peak_Value)
    {
     Peak_Value = REG_Positive_Data_Buffer[y];
     Potential = y;
    }
  }
       /*** Format & Write Cd Peak & Potential Values ***/
 Fmt(Temp_Display_Buffer, "%s<%f A", Peak_Value);
 SetCtrlVal (Analyze_Panel_Handle, Analyze_Cdpeak, Temp_Display_Buffer);
 Potential = -800 + (Potential * 4.88);
 Fmt(Temp_Display_Buffer, "%s<%f[p2] mV", Potential);
 SetCtrlVal (Analyze_Panel_Handle, Analyze_Cdpot, Temp_Display_Buffer);

            /**** Pb ****/
 Potential = 0;
 Peak_Value = 0;

 for (y = (Data_Buf_Size / 2); y < Data_Buf_Size; y++)
  {
   if (REG_Positive_Data_Buffer[y] < Peak_Value)
    {
     Peak_Value = REG_Positive_Data_Buffer[y];
     Potential = y;
    }
  }
         /*** Format & Write Pb Peak & Potential Values ***/
 Fmt(Temp_Display_Buffer, "%s<%f A", Peak_Value);
 SetCtrlVal (Analyze_Panel_Handle, Analyze_Pbpeak, Temp_Display_Buffer);
 Potential = -800 + (Potential * 4.88);
 Fmt(Temp_Display_Buffer, "%s<%f[p2] mV", Potential);
 SetCtrlVal (Analyze_Panel_Handle, Analyze_Pbpot, Temp_Display_Buffer);

 /***** Initial Check of Default Concentration Limit vs Curve Data *****/
 /***** Checks for initial Out-of-Bounds of Curve Data          *****/

  GetCtrlVal (Analyze_Panel_Handle, Analyze_Limit, &Limit_Value);
  for(y = 0; y < Data_Buf_Size; y++)
   {
```

```
  if(Limit_Value > REG_Positive_Data_Buffer[y])
     Limit_Flag = YES;
 }
if(Limit_Flag = = YES)
 {
  SetCtrlVal (Analyze_Panel_Handle, Analyze_LED, 1);
  beep ();
 }
 else
  SetCtrlVal (Analyze_Panel_Handle, Analyze_LED, 0);
Limit_Flag = NO;

while(DisplayPopup = = YES)
 {
  /***** Get User Event Within the Popup Panel *****/
  GetPopupEvent (1, &Popup_Event_ID);
  /***** Determine which event occured *****/
  switch (Popup_Event_ID)
    {
     case Analyze_Limit:
         GetCtrlVal (Analyze_Panel_Handle, Analyze_Limit, &Limit_Value);
         for(y = 0; y < Data_Buf_Size; y+ +)
          {
           if(Limit_Value > REG_Positive_Data_Buffer[y])
              Limit_Flag = YES;
          }
         if(Limit_Flag = = YES)
          {
           SetCtrlVal (Analyze_Panel_Handle, Analyze_LED, 1);
           beep ();
          }
          else
           SetCtrlVal (Analyze_Panel_Handle, Analyze_LED, 0);
         Limit_Flag = NO;
         break;

     case Analyze_Done:
         DisplayPopup = NO;
         break;
    }
 }
RemovePopup (0); /* Remove Current Popup (Analyze Data Panel) From Screen
*/
}
```

```
/*****************************************************************/
/*****************************************************************/
/*            Function: Acquire Data                             */
/*****************************************************************/
/* This function generates: a "Cleaning" potential of 0 V for 30*/
/* seconds, a"Quiet Time" potential of -800mV for 10 sec.,      */
/* a "Background" curve with data acquisistion from -800mV to   */
/* -300mV, a "Deposition Time" & "Quiet Time" potential of      */
/* -800mV for a total of 40 seconds, a "Sample" or "Regular"    */
/* curve with data acquisition from -800mV to -300mV, a final   */
/* potential of 0 V.                                            */
/*****************************************************************/

void Acquire_Data(double REG_Positive_Data_Buffer[], char
Output_Filename_Buffer[])
{
 int Voltage_Value_Buffer[Voltage_Buf_Size], /*Bin. Step Signal Potentials*/
    DAQ_Buffer[Cir_Buf_Size],          /*Data Acquisition Circular Buff*/
    BG_Neg_Half_Buffer[Half_Buf_Size], /*Background Curve Neg Pulse Half Buff*/
    BG_Pos_Half_Buffer[Half_Buf_Size], /*Background Curve Pos pulse Half Buff*/
    BG_Neg_Data_Pt_Count,              /*Used in Averaging Acquired Data*/
    BG_Pos_Data_Pt_Count,              /*Used in Averaging Acquired Data*/
    BG_Neg_Data_Index,                 /*Index for Data Buffer*/
    BG_Pos_Data_Index,                 /*Index for Data Buffer*/
    REG_Neg_Half_Buffer[Half_Buf_Size], /*Sample Curve Neg Pulse Half Buff*/
    REG_Pos_Half_Buffer[Half_Buf_Size], /*Sample Curve Pos Pulse Half Buff*/
    REG_Neg_Data_Pt_Count,             /*Used in Averaging Acquired Data*/
    REG_Pos_Data_Pt_Count,             /*Used in Averaging Acquired Data*/
    REG_Neg_Data_Index,                /*Index for Data Buffer*/
    REG_Pos_Data_Index,                /*Index for Data Buffer*/
    Half_Buffer_Ready,                 /*Half Buffer Ready to Transfer Var.*/
    Neg_Scan_Cycle,                    /*Neg Pulse Data Acq. Cycle Var.*/
    Voltage_Index,                     /*Index for Voltage Potential Buff*/
    Timebase,                          /*Used to Determine DAQ Sample Rate*/
    SampleInterval,                    /*Used to Determine DAQ Sample Rate*/
    Average_Flag,                      /*Average Data or Not Flag Var*/
    Save_Flag,                         /*Save Data or Not Flag Var*/
    Plating_Voltage,                   /*Start Pos Pulse Step Potential*/
    Stripping_Voltage,                 /*Start Neg Pulse Step Potential*/
    Index_Variable,                    /*Used in Timing Delay Loops*/
    Counter_Overflow,                  /*Counter Timing Overflow Var*/
    Counter_Count,                     /*Counter Count Var*/
    DAQ_Finished,                      /*DAQ Finished Var*/
```

```
    Num_Points_Transfered,          /*Data Acquisition Var*/
    NotUsedStatus,                 /*Unused Status Variable*/
    Unfilt_Half_Data_File,         /*File Handle*/
    Filt_Half_Data_File,           /*File Handle*/
    Minus_Data_File,               /*File Handle*/
    Data_File,                     /*File Handle*/
    Temperature_Buffer[100],       /*Used in Temperature Reading*/
    X;                             /*Generic Index Variable*/

double Coeff_Array[Filter_Weight+1], /*Buffer With FIR Filter Coefficients*/
      Positive_Filtered_Array[Filter_Buf_Size], /*Buff for Filtered Data*/
      Negative_Filtered_Array[Filter_Buf_Size], /*Buff for Filtered Data*/
      BG_Neg_Data_Sum,                  /*Averaging Sum Var*/
      BG_Pos_Data_Sum,                  /*Averaging Sum Var*/
      BG_Negative_Data_Buffer[Data_Buf_Size], /*Acquired Data Buffer*/
      BG_Positive_Data_Buffer[Data_Buf_Size], /*Acquired Data Buffer*/
      REG_Neg_Data_Sum,                 /*Averaging Sum Var*/
      REG_Pos_Data_Sum,                 /*Averaging Sum Var*/
      REG_Negative_Data_Buffer[Data_Buf_Size],/*Acquired Data Buffer*/
      Present_Step_Potential,           /*Step Potential Var*/
      Junction_Temp,   /*Interface Board Screw Terminal Temperature Var*/
      Temp_Voltage,    /*Thermocouple Temperature Reading*/
      Comp_Temp,       /*Linerized Temperature Reading Var*/
      Temp_Sum,        /*Temperature Samples Sum Var*/
      Conductivity,    /*Conductivity Sample Var*/
      Cond_Sum,        /*Averaging Sum Var*/
      pH,              /*pH Sample Var*/
      pH_Sum;          /*pH Sample Sum Var*/


  /* Clear Arrays */
for(X = 0; X < Half_Buf_Size; X++)
 {
  BG_Neg_Half_Buffer[X] = 0;
  BG_Pos_Half_Buffer[X] = 0;
  REG_Neg_Half_Buffer[X] = 0;
  REG_Pos_Half_Buffer[X] = 0;
 }
for(X = 0; X < Data_Buf_Size; X++)
 {
  BG_Negative_Data_Buffer[X] = 0;
  BG_Positive_Data_Buffer[X] = 0;
  REG_Negative_Data_Buffer[X] = 0;
  REG_Positive_Data_Buffer[X] = 0;
```

```
 }
 for(X = 0; X < Filter_Buf_Size; X++)
  {
   Positive_Filtered_Array[X] = 0;
   Negative_Filtered_Array[X] = 0;
  }



/*****************************************************/
/*       Initialize Voltage Buffer              */
/* Initializes Voltage Buffer with "Background" */
/* & "Acquisition" Step Values.                */
/* - Range = -800 mV to -300 mV                */
/* - Voltage values are binary representations  */
/* - i.e.  "1" = 4.88 mV   "-1" = -4.88 mV      */
/*****************************************************/
 Voltage_Index = 0;
 Plating_Voltage = -154;    /*Starting +Pulse Potential -154 = -750 mV*/
 Stripping_Voltage = Initial_Potential;  /*Starting -Pulse Potential -164 = -800
mV*/

 while(Voltage_Index <= (Voltage_Buf_Size - 1))
  {
   Voltage_Value_Buffer[Voltage_Index] = Stripping_Voltage;
   Voltage_Index++;
   Voltage_Value_Buffer[Voltage_Index] = Plating_Voltage;
   Voltage_Index++;
   Plating_Voltage++;
   Stripping_Voltage++;
  }

/* 10 Weight FIR Filter: fc = 30 Hz */
 Coeff_Array[0] = .0145;
 Coeff_Array[1] = .0306;
 Coeff_Array[2] = .0725;
 Coeff_Array[3] = .1245;
 Coeff_Array[4] = .1665;
 Coeff_Array[5] = .1826;
 Coeff_Array[6] = .1665;
 Coeff_Array[7] = .1245;
 Coeff_Array[8] = .0725;
 Coeff_Array[9] = .0306;
 Coeff_Array[10] = .0145;
```

```
/*****************************************************************/
/*                    Initilization                              */
/*****************************************************************/

     /* Data Acquisition Circuitry Initilization */
AI_Config (1, 0, 10, 0);
DAQ_DB_Config (1, 1);        /*Enable Double Buffering*/
DAQ_Clear (1);               /*Clears any existing DAQ processes*/
DAQ_Rate (.00017, 1, &Timebase, &SampleInterval); /*.17 mS/point*/

/*****************************************************************/
/*****************************************************************/
/* Generate "Cleaning Time"  0 V for 30 sec.                     */
/*****************************************************************/
/*****************************************************************/

Counter_Overflow = 0;
Index_Variable = 1;

AO_Write (1, 0, 0);

while(Index_Variable <= 458)    /* 458 * 1 uS * 65535 = 30 sec. */
 {
  CTR_EvCount (1, 5, 1, 0);
  while(Counter_Overflow == 0)
   {
    CTR_EvRead (1, 5, &Counter_Overflow, &Counter_Count);
   }
  Index_Variable++;
  Counter_Overflow = 0;
 }

/*****************************************************************/
/*****************************************************************/
/* Generate "Quiet Time"  -800 mV for 10 sec.                    */
/*****************************************************************/
/*****************************************************************/

Counter_Overflow = 0;
Index_Variable = 1;

AO_Write (1, 0, Initial_Potential);

while(Index_Variable <= 153)    /* 153 * 1uS * 65535 = 10 sec. */
 {
```

```
 CTR_EvCount (1, 5, 1, 0);
 while(Counter_Overflow = = 0)
  {
   CTR_EvRead (1, 5, &Counter_Overflow, &Counter_Count);
  }
 Index_Variable + +;
 Counter_Overflow = 0;
}



/*****************************************************************/
/*****************************************************************/
/* Generate Background Step Signal with Data Acquisition      */
/*****************************************************************/
/*****************************************************************/
Voltage_Index = 0;
BG_Neg_Data_Index = 0;
BG_Pos_Data_Index = 0;
BG_Neg_Data_Sum = 0;
BG_Pos_Data_Sum = 0;
BG_Neg_Data_Pt_Count = 0;
BG_Pos_Data_Pt_Count = 0;
Neg_Scan_Cycle = 0;
Average_Flag = NO;

/* - Write Initial Negative Step Value */
/* - Start Data Acquisition Process    */
AO_Write (1, 0, Voltage_Value_Buffer[Voltage_Index]);
DAQ_Start (1, 3, 1, DAQ_Buffer, Cir_Buf_Size, Timebase, SampleInterval);
Voltage_Index + +;

while(Voltage_Value_Buffer[Voltage_Index] < Final_Potential)
 {
/*****************************************************************/
/* - Negative Half Signal Generation & Data Acquition         */
/* - Process final Average of previous cycle acquired data    */
/*****************************************************************/
 Save_Flag = NO;

 while(Neg_Scan_Cycle < = 14) /*DAQ cycles with NO SAVE for timing*/
  {
   /*TRUE on 2nd and successive cycles. Will process previous Neg step
     cycle last half buffer & generate final average.*/
```

```
  if(Neg_Scan_Cycle = = 0 && Average_Flag = = YES)
   Average_Negative_Data(BG_Neg_Half_Buffer, &BG_Neg_Data_Sum,
                    &BG_Neg_Data_Pt_Count, BG_Negative_Data_Buffer,
                    &BG_Neg_Data_Index, Coeff_Array, Negative_Filtered_Array);

  /*TRUE on 2nd and successive cycles. Will process previous Pos step
   cycle last half buffer & generate final average.*/

  if(Neg_Scan_Cycle = = 1 && Average_Flag = = YES)
   Average_Positive_Data(BG_Pos_Half_Buffer, &BG_Pos_Data_Sum,
                    &BG_Pos_Data_Pt_Count, BG_Positive_Data_Buffer,
                    &BG_Pos_Data_Index, Coeff_Array, Positive_Filtered_Array);

  Negative_ProcessAvg_Transfer(BG_Neg_Half_Buffer, &Half_Buffer_Ready,
                      &BG_Neg_Data_Sum, &BG_Neg_Data_Pt_Count,
                      &DAQ_Finished, Save_Flag, Coeff_Array,
                      Negative_Filtered_Array);

  Negative_ProcessAvg_Transfer(BG_Neg_Half_Buffer, &Half_Buffer_Ready,
                      &BG_Neg_Data_Sum, &BG_Neg_Data_Pt_Count,
                      &DAQ_Finished, Save_Flag, Coeff_Array,
                      Negative_Filtered_Array);
  Neg_Scan_Cycle + +;
 }

 /*Get 1st Half Buffer to be Saved*/
 Negative_ProcessAvg_Transfer(BG_Neg_Half_Buffer, &Half_Buffer_Ready,
                     &BG_Neg_Data_Sum, &BG_Neg_Data_Pt_Count,
                     &DAQ_Finished, Save_Flag, Coeff_Array,
                     Negative_Filtered_Array);
 Save_Flag = YES;

 /*Process 1st Half Buffer. Get 2nd Half Buffer to be Saved*/
 Negative_ProcessAvg_Transfer(BG_Neg_Half_Buffer, &Half_Buffer_Ready,
                     &BG_Neg_Data_Sum, &BG_Neg_Data_Pt_Count,
                     &DAQ_Finished, Save_Flag, Coeff_Array,
                     Negative_Filtered_Array);

/*****************************************************************/
/*          Positive Half Signal Data Acquition          */
/*****************************************************************/
 AO_Write (1, 0, Voltage_Value_Buffer[Voltage_Index]);
 Voltage_Index + +;
 Save_Flag = NO;
```

```
  Positive_ProcessAvg_Transfer(BG_Pos_Half_Buffer, &Half_Buffer_Ready,
                        &BG_Pos_Data_Sum, &BG_Pos_Data_Pt_Count,
                        &DAQ_Finished, Save_Flag, Coeff_Array,
                        Positive_Filtered_Array);

  Positive_ProcessAvg_Transfer(BG_Pos_Half_Buffer, &Half_Buffer_Ready,
                        &BG_Pos_Data_Sum, &BG_Pos_Data_Pt_Count,
                        &DAQ_Finished, Save_Flag, Coeff_Array,
                        Positive_Filtered_Array);

  /*Get 1st Half Buffer to be Saved*/

  Positive_ProcessAvg_Transfer(BG_Pos_Half_Buffer, &Half_Buffer_Ready,
                        &BG_Pos_Data_Sum, &BG_Pos_Data_Pt_Count,
                        &DAQ_Finished, Save_Flag, Coeff_Array,
                        Positive_Filtered_Array);
  Save_Flag = YES;

  /*Process 1st Half Buffer. Get 2nd Half Buffer to be Saved*/

  Positive_ProcessAvg_Transfer(BG_Pos_Half_Buffer, &Half_Buffer_Ready,
                        &BG_Pos_Data_Sum, &BG_Pos_Data_Pt_Count,
                        &DAQ_Finished, Save_Flag, Coeff_Array,
                        Positive_Filtered_Array);
/*****************************************************************/
/* - Write Next Negative Voltage Level                          */
/* - Initilize Variables                                        */
/*****************************************************************/
  AO_Write (1, 0, Voltage_Value_Buffer[Voltage_Index]);
  Voltage_Index++;

  BG_Neg_Data_Sum = 0;
  BG_Pos_Data_Sum = 0;
  BG_Neg_Data_Pt_Count = 0;
  BG_Pos_Data_Pt_Count = 0;
  Neg_Scan_Cycle = 0;
  Average_Flag = YES;
 }/* End of While*/

AO_Write (1, 0, Final_Potential); /*Cell Potential = Final Potential*/
DAQ_Clear (1);                    /*Halt Acquisition*/

/* Filter & Average Final Half Buffers */

Average_Negative_Data(BG_Neg_Half_Buffer, &BG_Neg_Data_Sum,
```

```
                &BG_Neg_Data_Pt_Count, BG_Negative_Data_Buffer,
                &BG_Neg_Data_Index, Coeff_Array, Negative_Filtered_Array);

Average_Positive_Data(BG_Pos_Half_Buffer, &BG_Pos_Data_Sum,
                &BG_Pos_Data_Pt_Count, BG_Positive_Data_Buffer,
                &BG_Pos_Data_Index, Coeff_Array, Positive_Filtered_Array);



/*****************************************************************/
/*****************************************************************/
/* Generate "Deposition Time"  -800 mV for 30 sec.          */
/*       & "Quiet Time"       -800 mV for 10 sec.          */
/* Total Signal Generation  =  -800 mV for 40 sec.          */
/*****************************************************************/
/*****************************************************************/

Counter_Overflow = 0;
Index_Variable = 1;

AO_Write (1, 0, Initial_Potential);

while(Index_Variable <= 610)    /* 610 * 1uS * 65535 = 40 sec. */
 {
  CTR_EvCount (1, 5, 1, 0);
  while(Counter_Overflow == 0)
   {
    CTR_EvRead (1, 5, &Counter_Overflow, &Counter_Count);
   }
  Index_Variable++;
  Counter_Overflow = 0;
 }

/*****************************************************************/
/*****************************************************************/
/* Generate Regular Step Signal with Data Acquisition        */
/*****************************************************************/
/*****************************************************************/
Voltage_Index = 0;
REG_Neg_Data_Index = 0;
REG_Pos_Data_Index = 0;
REG_Neg_Data_Sum = 0;
REG_Pos_Data_Sum = 0;
```

```
REG_Neg_Data_Pt_Count = 0;
REG_Pos_Data_Pt_Count = 0;
Neg_Scan_Cycle = 0;
Average_Flag = NO;

/* - Write Initial Negative Step Value */
/* - Start Data Acquisition Process    */
AO_Write (1, 0, Voltage_Value_Buffer[Voltage_Index]);
DAQ_Start (1, 3, 1, DAQ_Buffer, Cir_Buf_Size, Timebase, SampleInterval);
Voltage_Index++;

while(Voltage_Value_Buffer[Voltage_Index] < Final_Potential)
 {
/*****************************************************************/
/* - Negative Half Signal Generation & Data Acquition        */
/* - Process final Average of previous cycle acquired data    */
/*****************************************************************/
  Save_Flag = NO;
        /*DAQ cycles with NO SAVE for timing*/
  while(Neg_Scan_Cycle <= Neg_HalfSig_NoSave_CycleCnt)
   {
    /*TRUE on 2nd and successive cycles. Will process previous Neg step
      cycle last half buffer & generate final average.*/

    if(Neg_Scan_Cycle == 0 && Average_Flag == YES)
     Average_Negative_Data(REG_Neg_Half_Buffer, &REG_Neg_Data_Sum,
                    &REG_Neg_Data_Pt_Count, REG_Negative_Data_Buffer,
                    &REG_Neg_Data_Index, Coeff_Array, Negative_Filtered_Array);

    /*TRUE on 2nd and successive cycles. Will process previous Pos step
      cycle last half buffer & generate final average.*/

    if(Neg_Scan_Cycle == 1 && Average_Flag == YES)
     Average_Positive_Data(REG_Pos_Half_Buffer, &REG_Pos_Data_Sum,
                    &REG_Pos_Data_Pt_Count, REG_Positive_Data_Buffer,
                    &REG_Pos_Data_Index, Coeff_Array, Negative_Filtered_Array);

    Negative_ProcessAvg_Transfer(REG_Neg_Half_Buffer, &Half_Buffer_Ready,
                        &REG_Neg_Data_Sum, &REG_Neg_Data_Pt_Count,
                        &DAQ_Finished, Save_Flag, Coeff_Array,
                        Negative_Filtered_Array);

    Negative_ProcessAvg_Transfer(REG_Neg_Half_Buffer, &Half_Buffer_Ready,
                        &REG_Neg_Data_Sum, &REG_Neg_Data_Pt_Count,
                        &DAQ_Finished, Save_Flag, Coeff_Array,
```

```
                    Negative_Filtered_Array);
  Neg_Scan_Cycle++;
 }

 /*Get 1st Half Buffer to be Saved*/
 Negative_ProcessAvg_Transfer(REG_Neg_Half_Buffer, &Half_Buffer_Ready,
                    &REG_Neg_Data_Sum, &REG_Neg_Data_Pt_Count,
                    &DAQ_Finished, Save_Flag, Coeff_Array,
                    Negative_Filtered_Array);
 Save_Flag = YES;

 /*Process 1st Half Buffer. Get 2nd Half Buffer to be Saved*/
 Negative_ProcessAvg_Transfer(REG_Neg_Half_Buffer, &Half_Buffer_Ready,
                    &REG_Neg_Data_Sum, &REG_Neg_Data_Pt_Count,
                    &DAQ_Finished, Save_Flag, Coeff_Array,
                    Negative_Filtered_Array);

/*****************************************************************/
/*          Positive Half SignalData Acquition          */
/*****************************************************************/
 AO_Write (1, 0, Voltage_Value_Buffer[Voltage_Index]);
 Voltage_Index++;
 Save_Flag = NO;

 Positive_ProcessAvg_Transfer(REG_Pos_Half_Buffer, &Half_Buffer_Ready,
                    &REG_Pos_Data_Sum, &REG_Pos_Data_Pt_Count,
                    &DAQ_Finished, Save_Flag, Coeff_Array,
                    Positive_Filtered_Array);

 Positive_ProcessAvg_Transfer(REG_Pos_Half_Buffer, &Half_Buffer_Ready,
                    &REG_Pos_Data_Sum, &REG_Pos_Data_Pt_Count,
                    &DAQ_Finished, Save_Flag, Coeff_Array,
                    Positive_Filtered_Array);

 /*Get 1st Half Buffer to be Saved*/

 Positive_ProcessAvg_Transfer(REG_Pos_Half_Buffer, &Half_Buffer_Ready,
                    &REG_Pos_Data_Sum, &REG_Pos_Data_Pt_Count,
                    &DAQ_Finished, Save_Flag, Coeff_Array,
                    Positive_Filtered_Array);
 Save_Flag = YES;

 /*Process 1st Half Buffer. Get 2nd Half Buffer to be Saved*/

 Positive_ProcessAvg_Transfer(REG_Pos_Half_Buffer, &Half_Buffer_Ready,
```

```
                    &REG_Pos_Data_Sum, &REG_Pos_Data_Pt_Count,
                    &DAQ_Finished, Save_Flag, Coeff_Array,
                    Positive_Filtered_Array);
/*****************************************************************/
/* - Write Next Negative Voltage Level                    */
/* - Initilize Variables                          */
/*****************************************************************/
 AO_Write (1, 0, Voltage_Value_Buffer[Voltage_Index]);
 Voltage_Index++;

 REG_Neg_Data_Sum = 0;
 REG_Pos_Data_Sum = 0;
 REG_Neg_Data_Pt_Count = 0;
 REG_Pos_Data_Pt_Count = 0;
 Neg_Scan_Cycle = 0;
 Average_Flag = YES;
}/*End of While*/

AO_Write (1, 0, Final_Potential); /*Cell Potential = Final Potential*/
DAQ_Clear (1);              /*Halt Acquisition*/
AO_Write (1, 0, 0);          /*Final Cell Potential to 0 V*/

/* Filter & Average Final Half Buffers */

Average_Negative_Data(REG_Neg_Half_Buffer, &REG_Neg_Data_Sum,
              &REG_Neg_Data_Pt_Count, REG_Negative_Data_Buffer,
              &REG_Neg_Data_Index, Coeff_Array, Negative_Filtered_Array);

Average_Positive_Data(REG_Pos_Half_Buffer, &REG_Pos_Data_Sum,
              &REG_Pos_Data_Pt_Count, REG_Positive_Data_Buffer,
              &REG_Pos_Data_Index, Coeff_Array, Negative_Filtered_Array);


/*****************************************************************/
/*****************************************************************/
/* Aquire Temperature, Conductivity & pH Measurements        */
/*****************************************************************/
/*****************************************************************/
                  /* Temperature */

 DAQ_DB_Config (1, 0);
 DAQ_Clear (1);

 Temp_Voltage = 0;
 Junction_Temp = 0;
```

```
 Comp_Temp = 0;
 Temp_Sum = 0;

 AI_VRead (1, 0, 10, &Junction_Temp);
 Junction_Temp *= 100;

 DAQ_Op (1, 1, 100, Temperature_Buffer, 100, 5882.353);
 DAQ_Clear (1);

 for(X=0; X<100; X++)
  Temp_Sum += (Temperature_Buffer[X]);
 Temp_Voltage = ((Temp_Sum/100) * .00244)/100;

            /*** Linerize Readings ***/
 Comp_Temp = .100860910 + Temp_Voltage * (25727.94369 +
Temp_Voltage *
          (-767345.8295 + Temp_Voltage * (78025595.81 + (Temp_Voltage
          * -9247486589.0))));
 Temperature_Reading = Comp_Temp + Junction_Temp;



                    /* Conductivity */
 Conductivity = 0;
 Cond_Sum = 0;

  for(X = 0; X < 10; X++)
   {
    AI_VRead (1, 2, 100, &Conductivity);
    Cond_Sum += Conductivity;
   }
  Conductivity = Cond_Sum / 10;
  Conductivity_Reading = Conductivity * 1000.0;

                    /* pH */
 pH = 0;
 pH_Sum = 0;

  for(X=0; X<10; X++)
   {
    AI_VRead (1, 4, 100, &pH);
    pH_Sum += pH;
   }
  pH = pH_Sum / 10;
  pH_Reading = pH * 1000;
```

```
        /* Format Readings to Strings for Display Purposes */
  Fmt (Temperature_Display_Buffer, "%s<%f[p2] C", Temperature_Reading);
  Fmt (Conductivity_Display_Buffer, "%s<%f[p2] uS/cm", Conductivity_Reading);
  Fmt (pH_Display_Buffer, "%s<%f[p2] mV", pH_Reading);


/*****************************************************************/
/*****************************************************************/
/* - Process Collected Data                                      */
/* - Write Background & Regular Data Information to File         */
/*****************************************************************/
/*****************************************************************/

/***                                                          ***/
/*** Write Last Collected UNFILTERED REG_Half_Buffers to file ***/
/***                                                          ***/

Unfilt_Half_Data_File = OpenFile ("\\LW\\ASV\\UNFILTRG.DAT", 0, 0, 1);

for(BG_Neg_Data_Index = 0; BG_Neg_Data_Index < Half_Buf_Size;
BG_Neg_Data_Index++)
 {
  FmtFile (Unfilt_Half_Data_File, "%s<%i[w10]",
REG_Neg_Half_Buffer[BG_Neg_Data_Index]);
  WriteFile (Unfilt_Half_Data_File, "\t", 1);
  FmtFile (Unfilt_Half_Data_File, "%s<%i[w10]",
REG_Pos_Half_Buffer[BG_Neg_Data_Index]);
  WriteFile (Unfilt_Half_Data_File, "\n", 1);
 }
CloseFile (Unfilt_Half_Data_File);


/***                                                          ***/
/***  Write Last Collected FILTERED REG_Half_Buffers to file  ***/
/***                                                          ***/

Filt_Half_Data_File = OpenFile ("\\LW\\ASV\\FILTREG.DAT", 0, 0, 1);

for(BG_Neg_Data_Index = 0; BG_Neg_Data_Index < Filter_Buf_Size;
BG_Neg_Data_Index++)
 {
  FmtFile (Filt_Half_Data_File, "%s<%f[w10]",
Negative_Filtered_Array[BG_Neg_Data_Index]);
  WriteFile (Filt_Half_Data_File, "\t", 1);
  FmtFile (Filt_Half_Data_File, "%s<%f[w10]",
```

```
Positive_Filtered_Array[BG_Neg_Data_Index]);
  WriteFile (Filt_Half_Data_File, "\n", 1);
 }
 CloseFile (Filt_Half_Data_File);



 /***                                                    ***/
 /***    Calculate I(Forward) - I(Reverse) for "Background"    ***/
 /***    & "Regular" Acquisition Curves                        ***/
 /***                                                    ***/
 for(BG_Neg_Data_Index = 0; BG_Neg_Data_Index < Data_Buf_Size;
BG_Neg_Data_Index++)
 {
  BG_Positive_Data_Buffer[BG_Neg_Data_Index] -=
BG_Negative_Data_Buffer[BG_Neg_Data_Index];
  REG_Positive_Data_Buffer[BG_Neg_Data_Index] -=
REG_Negative_Data_Buffer[BG_Neg_Data_Index];
 }

 /***                                                    ***/
 /***     Write "Unsmoothed" I(F) - I(R) curves to file      ***/
 /***                                                    ***/
 Minus_Data_File = OpenFile ("\\lw\\asv\\IF_IR.DAT", 0, 0, 1);

 for(BG_Neg_Data_Index = 0; BG_Neg_Data_Index < Data_Buf_Size;
BG_Neg_Data_Index++)
 {
  FmtFile (Minus_Data_File, "%s<%f",
BG_Positive_Data_Buffer[BG_Neg_Data_Index]);
  WriteFile (Minus_Data_File, "\t", 1);
  FmtFile (Minus_Data_File, "%s<%f",
REG_Positive_Data_Buffer[BG_Neg_Data_Index]);
  WriteFile (Minus_Data_File, "\n", 1);
 }
 CloseFile (Minus_Data_File);

 /***                                                    ***/
 /***        Data Smoothing: 5 Point Moving Average         ***/
 /***                                                    ***/
 for(BG_Neg_Data_Index = 2; BG_Neg_Data_Index <= (Final_Filt_Buf_Element-2);
    BG_Neg_Data_Index++)
 {
  BG_Positive_Data_Buffer[BG_Neg_Data_Index] =
(BG_Positive_Data_Buffer[BG_Neg_Data_Index-2]
```

```
+BG_Positive_Data_Buffer[BG_Neg_Data_Index-1]+BG_Positive_Data_Buffer[BG_N
eg_Data_Index+1]
    +BG_Positive_Data_Buffer[BG_Neg_Data_Index+2]) / 4;

  REG_Positive_Data_Buffer[BG_Neg_Data_Index] =
(REG_Positive_Data_Buffer[BG_Neg_Data_Index-2]

+REG_Positive_Data_Buffer[BG_Neg_Data_Index-1]+REG_Positive_Data_Buffer[BG
_Neg_Data_Index+1]
    +REG_Positive_Data_Buffer[BG_Neg_Data_Index+2]) / 4;

  if(BG_Neg_Data_Index = = 2)
   {

BG_Positive_Data_Buffer[BG_Neg_Data_Index-1]=BG_Positive_Data_Buffer[BG_Neg
_Data_Index];

BG_Positive_Data_Buffer[BG_Neg_Data_Index-2]=BG_Positive_Data_Buffer[BG_Neg
_Data_Index];


REG_Positive_Data_Buffer[BG_Neg_Data_Index-1]=REG_Positive_Data_Buffer[BG_
Neg_Data_Index];

REG_Positive_Data_Buffer[BG_Neg_Data_Index-2]=REG_Positive_Data_Buffer[BG_
Neg_Data_Index];
   }

  if(BG_Neg_Data_Index = = (Final_Filt_Buf_Element-2))
   {

BG_Positive_Data_Buffer[BG_Neg_Data_Index+1]=BG_Positive_Data_Buffer[BG_N
eg_Data_Index];

BG_Positive_Data_Buffer[BG_Neg_Data_Index+2]=BG_Positive_Data_Buffer[BG_N
eg_Data_Index];


REG_Positive_Data_Buffer[BG_Neg_Data_Index+1]=REG_Positive_Data_Buffer[BG
_Neg_Data_Index];

REG_Positive_Data_Buffer[BG_Neg_Data_Index+2]=REG_Positive_Data_Buffer[BG
_Neg_Data_Index];
   }
```

```
 }

 /***                                                  ***/
 /***      Write "Smoothed" I(F) - I(R) curves to file        ***/
 /***                                                  ***/

 Minus_Data_File = OpenFile ("\\lw\\asv\\S_IF_IR.DAT", 0, 0, 1);

 for(BG_Neg_Data_Index = 0; BG_Neg_Data_Index < Data_Buf_Size;
BG_Neg_Data_Index++)
  {
   FmtFile (Minus_Data_File, "%s<%f",
BG_Positive_Data_Buffer[BG_Neg_Data_Index]);
   WriteFile (Minus_Data_File, "\t", 1);
   FmtFile (Minus_Data_File, "%s<%f",
REG_Positive_Data_Buffer[BG_Neg_Data_Index]);
   WriteFile (Minus_Data_File, "\n", 1);
  }
 CloseFile (Minus_Data_File);

 /***                                                  ***/
 /*** Convert I(Forward) - I(Reverse) Curves to Current Values ***/
 /***                                                  ***/

 for(BG_Neg_Data_Index = 0; BG_Neg_Data_Index < Data_Buf_Size;
BG_Neg_Data_Index++)
  {
   BG_Positive_Data_Buffer[BG_Neg_Data_Index] *= (.00244 / Measure_Resistor);
   REG_Positive_Data_Buffer[BG_Neg_Data_Index] *= (.00244 /
Measure_Resistor);
  }

 /***                                                  ***/
 /***       Calculate "Regular" - "Background" Curve        ***/
 /***                                                  ***/

 for(BG_Neg_Data_Index = 0; BG_Neg_Data_Index < Data_Buf_Size;
BG_Neg_Data_Index++)
  {
   REG_Positive_Data_Buffer[BG_Neg_Data_Index] -=
BG_Positive_Data_Buffer[BG_Neg_Data_Index];
  }

 /***                                                  ***/
 /***    Write "REG - BG" Curve vs. Step Potential to File    ***/
```

```
/***                                                    ***/

 Present_Step_Potential = -.8;
 Data_File = OpenFile (Output_Filename_Buffer, 0, 0, 1);

 for(BG_Neg_Data_Index = 0; BG_Neg_Data_Index < Data_Buf_Size;
BG_Neg_Data_Index++)
  {
   FmtFile (Data_File, "%s<%f[w10]", Present_Step_Potential);
   WriteFile (Data_File, "\t", 1);
   FmtFile (Data_File, "%s<%f", REG_Positive_Data_Buffer[BG_Neg_Data_Index]);
   WriteFile (Data_File, "\n", 1);
   Present_Step_Potential += .00488;
  }
 CloseFile (Data_File);

}



/*****************************************************************/
/*****************************************************************/
/*        Function: Negative_ProcessAvg_Transfer                 */
/*                                                               */
/* Process previously acquired Neg_Half_Buffer; Wait for next     */
/* Half_Buffer; Transfer when ready to Neg_Half_Buffer            */
/* Upon Return: Neg_Data_Sum & Neg_Data_Pt_Count are updated      */
/*            Neg_Half_Buffer is full of acquired data            */
/*****************************************************************/
/*****************************************************************/


void Negative_ProcessAvg_Transfer(int Neg_Half_Buffer[], int *Half_Buffer_Ready,
                          double *Neg_Data_Sum, int *Neg_Data_Pt_Count,
                          int *DAQ_Finished, int Save_Flag,
                          double Coeff_Array[], double Filtered_Array[])
 {
  int NotUsedStatus,
     Half_Buffer_Index,
     Coeff_Index,
     Num_Points_Transfered;

  *Half_Buffer_Ready = NO;
```

```
 if(Save_Flag = = YES)
  {
   /* 10 weight FIR Filter */

   for(Half_Buffer_Index = Filter_Weight; Half_Buffer_Index < Half_Buf_Size;
Half_Buffer_Index+ +)
    {
     Filtered_Array[Half_Buffer_Index - Filter_Weight] = 0;
     for(Coeff_Index = 0; Coeff_Index < = Filter_Weight ; Coeff_Index+ +)
      {
       Filtered_Array[Half_Buffer_Index - Filter_Weight] + =
((double)Neg_Half_Buffer[Half_Buffer_Index
                                        - Coeff_Index] * Coeff_Array[Coeff_Index]);
      }
    }

  /* Sum Filtered_Array Elements */

   for(Half_Buffer_Index = 0; Half_Buffer_Index < = (Filter_Buf_Size - 1);
                                        Half_Buffer_Index+ +)
    {
     *Neg_Data_Sum + = Filtered_Array[Half_Buffer_Index];
     (*Neg_Data_Pt_Count)+ +;
    }
  }

  /* Get next Half_Buffer */

 while(*Half_Buffer_Ready = = NO)
  {
   DAQ_DB_HalfReady (1, Half_Buffer_Ready, DAQ_Finished);
  }
 DAQ_DB_Transfer (1, Neg_Half_Buffer, &Num_Points_Transfered,
&NotUsedStatus);
 }



/*****************************************************************/
/*****************************************************************/
/*        Function: Positive_ProcessAvg_Transfer                 */
/*                                                               */
/* Process previously acquired Pos_Half_Buffer if a SAVE cycle; Wait */
/* for next Half_Buffer; Transfer when ready to Pos_Half_Buffer      */
```

```
/* Upon Return: Pos_Data_Sum & Pos_Data_Pt_Count are updated        */
/*            Pos_Half_Buffer is full of acquired data            */
/*****************************************************************/
/*****************************************************************/

void Positive_ProcessAvg_Transfer(int Pos_Half_Buffer[], int *Half_Buffer_Ready,
                          double *Pos_Data_Sum, int *Pos_Data_Pt_Count,
                          int *DAQ_Finished, int Save_Flag,
                          double Coeff_Array[], double Filtered_Array[])
{
 int NotUsedStatus,
     Half_Buffer_Index,
     Coeff_Index,
     Num_Points_Transfered;

 *Half_Buffer_Ready = NO;

 if(Save_Flag = = YES)
  {
   /* 10 weight FIR Filter */

   for(Half_Buffer_Index = Filter_Weight; Half_Buffer_Index < Half_Buf_Size;
Half_Buffer_Index+ +)
    {
     Filtered_Array[Half_Buffer_Index - Filter_Weight] = 0;
     for(Coeff_Index = 0; Coeff_Index < = Filter_Weight; Coeff_Index+ +)
      {
       Filtered_Array[Half_Buffer_Index - Filter_Weight] + =
((double)Pos_Half_Buffer[Half_Buffer_Index
                                  - Coeff_Index] * Coeff_Array[Coeff_Index]);
      }
    }


  /* Sum Filtered_Array Elements */

   for(Half_Buffer_Index = 0; Half_Buffer_Index < = (Filter_Buf_Size - 1);
                                        Half_Buffer_Index+ +)
    {
     *Pos_Data_Sum + = Filtered_Array[Half_Buffer_Index];
     (*Pos_Data_Pt_Count)+ +;
    }
  }

  /* Get next Half_Buffer */
```

```
 while(*Half_Buffer_Ready = = NO)
  {
   DAQ_DB_HalfReady (1, Half_Buffer_Ready, DAQ_Finished);
  }
 DAQ_DB_Transfer (1, Pos_Half_Buffer, &Num_Points_Transfered,
&NotUsedStatus);
 }



/*****************************************************************/
/*****************************************************************/
/*        Function: Average_Negative_Data                        */
/*                                                               */
/* Process final Neg_Half_Buffer & Pos_Half_Buffer then calculate  */
/* overall average for each into a single saved data point for each */
/*****************************************************************/
/*****************************************************************/


void Average_Negative_Data(int Neg_Half_Buffer[], double *Neg_Data_Sum,
            int *Neg_Data_Pt_Count, double Negative_Data_Buffer[],
            int *Neg_Data_Index, double Coeff_Array[],
            double Filtered_Array[])
{
 int Half_Buffer_Index,
    Coeff_Index;

 double Sum_Average;

/* 10 weight FIR Filter */

 for(Half_Buffer_Index = Filter_Weight; Half_Buffer_Index < Half_Buf_Size;
Half_Buffer_Index++)
  {
   Filtered_Array[Half_Buffer_Index - Filter_Weight] = 0;
   for(Coeff_Index = 0; Coeff_Index <= Filter_Weight; Coeff_Index++)
    {
     Filtered_Array[Half_Buffer_Index - Filter_Weight] += 
((double)Neg_Half_Buffer[Half_Buffer_Index
                                - Coeff_Index] * Coeff_Array[Coeff_Index]);
    }
  }
/* Sum Filtered_Array Elements */
```

```
 for(Half_Buffer_Index = 0; Half_Buffer_Index <= (Filter_Buf_Size - 1);
                                        Half_Buffer_Index++)
  {
   *Neg_Data_Sum += Filtered_Array[Half_Buffer_Index];
   (*Neg_Data_Pt_Count)++;
  }

 Sum_Average = *Neg_Data_Sum / (double)*Neg_Data_Pt_Count;
 Negative_Data_Buffer[*Neg_Data_Index] = Sum_Average;

 (*Neg_Data_Index)++;
}


 /*****************************************************************/
 /*****************************************************************/
 /*        Function: Average_Positive_Data                        */
 /*                                                  */
 /* Process final Neg_Half_Buffer & Pos_Half_Buffer then calculate   */
 /* overall average for each into a single saved data point for each */
 /*****************************************************************/
 /*****************************************************************/


void Average_Positive_Data(int Pos_Half_Buffer[], double *Pos_Data_Sum,
             int *Pos_Data_Pt_Count, double Positive_Data_Buffer[],
             int *Pos_Data_Index, double Coeff_Array[],
             double Filtered_Array[])

{
 int Half_Buffer_Index,
    Coeff_Index;

 double Sum_Average;

/* 10 weight FIR Filter */

 for(Half_Buffer_Index = Filter_Weight; Half_Buffer_Index < Half_Buf_Size;
Half_Buffer_Index++)
  {
   Filtered_Array[Half_Buffer_Index - Filter_Weight] = 0;
   for(Coeff_Index = 0; Coeff_Index <= Filter_Weight; Coeff_Index++)
    {
     Filtered_Array[Half_Buffer_Index - Filter_Weight] +=
((double)Pos_Half_Buffer[Half_Buffer_Index
```

```
                                        - Coeff_Index] * Coeff_Array[Coeff_Index]);
    }
  }

/* Sum Filtered_Array Elements */

 for(Half_Buffer_Index = 0; Half_Buffer_Index <= (Filter_Buf_Size - 1);
                                        Half_Buffer_Index++)
 {
  *Pos_Data_Sum += Filtered_Array[Half_Buffer_Index];
  (*Pos_Data_Pt_Count)++;
 }

 Sum_Average = *Pos_Data_Sum / (double)*Pos_Data_Pt_Count;
 Positive_Data_Buffer[*Pos_Data_Index] = Sum_Average;

 (*Pos_Data_Index)++;
}
```

Distribution List for Sensor Based Process Control LDRD Final Report

1000	P. A. Fleury
1010	P. L. Mattern
1800	A. D. Romig
1824	M. R. Keenan
1824	V. E. Granstaff
2000	H. W. Schmitt
2300	R. D. Andreas
2304	W. D. Williams (10)
2337	G. R. Laguna
2337	J. T. Love
2337	D. J. Brandt
2338	S. M. Kohler
2338	J. B. Wronosky (5)
2600	J. H. Stichman
2665	D. H. Schroeder
2900	W. E. Alzheimer
2901	T. J. Young
6000	D. L. Hartley
6600	J. B. Woodard
6611	B. Granoff
6612	A. E. Verardo
8000	J. C. Crawford
8701	R. E. Stoltz
7141	Technical Library (5)
7151	Technical Publications
7613-2	Document Processing for DOE/OSTI (10)
8523-2	Central Technical Files

THIS PAGE INTENTIONALLY LEFT BLANK