LEGIBILITY NOTICE

A major purpose of the Technical Information Center is to provide the broadest dissemination possible of information contained in DOE's Research and Development Reports to business, industry, the academic community, and federal, state and local governments.

Although a small portion of this report is not reproducible, it is being made available to expedite the availability of information on the research discussed herein.

LA-UR--89-3560

DE90 002415

Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36

Received by OSTI

NOV 06 1989

TITLE **Emulating Conventional Operator Interfaces On Window
Based Workstations**

AUTHOR(S) **Gary P. Carr**

SUBMITTED TO **Proceedings on the International Conference on
Accelerator and Large Experimental Physics Control Systems
Vancouver, British Columbia, Canada
October 30 - November 3, 1989**

DISCLAIMER

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, 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 or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

By acceptance of this article, the publisher recognizes that the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution or to allow others to do so, for U.S. Government purposes.

This Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy.

Los Alamos National Laboratory
Los Alamos, New Mexico 87545

MASTER
DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

Emulating Conventional Operator Interfaces On Window Based Workstations *

Gary P. Carr
MP-6, Mail Stop H852
Los Alamos National Laboratory
Los Alamos, New Mexico 87545

ABSTRACT

This paper explores an approach to initially support the LAMPF and PSR control system on VAX/VMS workstations using DECwindows and VI Corporation DataViews as the operator interface. The PSR control system was recently turned over to MP division and the two control system staffs were merged into one group. One of the goals of this new group is to develop a common work station based operator console and interface which can be used in a single control room controlling both the LINAC and Proton Storage Ring. The new console operator interface will need a high level graphics toolkit for its implementation. During the conversion to the new consoles it will also probably be necessary to write a package to emulate the current operator interfaces at the software level. This paper describes a project to evaluate the appropriateness of VI Corporation's DataViews graphics package for use in the LAMPF control system environment by using it to write an emulation of the LAMPF touch panel interface to a large LAMPF control system application program. A secondary objective of this project was to explore any productivity increases that might be realized by using an object oriented graphics package and graphics editor.

*Work supported by the U.S. Department of Energy

Submitted to the:

1989 INTERNATIONAL CONFERENCE on ACCELERATOR AND LARGE EXPERIMENTAL
PHYSICS CONTROL SYSTEMS

30 October - 3 November 1989

Vancouver, British Columbia, Canada

Sponsored by

TRIUMF, The European Physical Society, and

The IEEE Nuclear and Plasma Sciences Society

1.0 Introduction

The Clinton P. Anderson Meson Physics Facility (LAMPF) is operated by the Los Alamos National Laboratory for the U.S. Department of Energy. The heart of the facility is a 1-kilometer-long linear accelerator (LINAC) designed to provide a proton beam of up to 800 MeV and one milliampere average current. The accelerator is a pulsed machine operating at 120 Hz. The control system for LAMPF has gone through a number of upgrades and modifications (reference [1] describes the latest upgrade). In its current incarnation, it is an "almost" distributed control system based on DEC VAX computers. Generally speaking, VMS systems are used as console computers, and VAXELN systems function as front end computers providing access to all hardware devices. Console and front end computers communicate over Ethernet using a Remote Procedure Call (RPC) protocol. Figure 1 shows the LAMPF control system as it will be at start up in 1990.

The Proton Storage Ring (PSR) came on line at LAMPF in April of 1985. PSR takes beam from LAMPF line D, compresses it, and delivers it to the neutron production targets at the Los Alamos Neutron Scattering Center (LANSCE). The control system for PSR was designed and built by the Accelerator Technology Division (reference [2]). Although the PSR control system is also based on DEC computers, it is a completely different system

from the LAMPF control system. The PSR control system uses a MicroVAX III computer to interface to its operator consoles and maintain a central live database of all devices in the system. Five PDP 11/73 computers running RSX11/S function as front end computers providing access to most hardware devices. The central MicroVAX and PDP 11/73 computers communicate through a Serial CAMAC loop. Figure 2 shows the PSR control system as it will be at start-up in 1990.

In 1986 the support for the PSR system was turned over to MP division, and in 1988 the programming and operations staff for both the LINAC and PSR control systems were merged into a single group, MP-6. The new group now is responsible for two control systems two control rooms, and two very different types of operator consoles using two very different operator interface philosophies. One of the goals for the controls group is to provide a single control room, with a common operator console and operator interface for both the LAMPF and PSR control systems. At this point we assume that the consoles will be based on color graphics workstations, possibly with some auxiliary display devices and hardware (knobs).

2.0 How Do We Get There From Here?

Before we can implement an integrated LAMPF/PSR console we need to define a number of requirements. We hope to use this conference and the workshop on workstations to help us refine the following requirements:

1. Define the hardware requirements (work station) for a common operator console. We are initially assuming that a MicroVAX III class workstation with a 19 inch high resolution color graphics scope is a minimum requirement for a workstation console.

2. Define a common operator interface for application programs on both control systems.
3. Define the software toolkit needed to build (program) the common operator interface, and evaluate currently available tools to determine how well they meet our needs. This requirement interacts with requirement number 2. The available tools inevitably affect what can practically be done with the operator interface. In turn, the operator interface drives the requirements for tools.
4. Devise an implementation plan to move from where we are now to our desired goal of a single control room and common operator console.

3.0 Current Project

The project that I and several other staff members are involved with is to begin work on part three of the above requirements by evaluating graphics toolkits that we might use to build our standard operator interface. The purpose of this initial evaluation is not to make a final decision on what graphics toolkit to use, but to determine what tools are commercially available and find out what they can do for us. In trying to evaluate graphics tool kits, we quickly decided that we are neither smart enough nor knowledgeable enough to evaluate these packages from seeing demonstrations, reading promotional literature, or talking to manufacturers representatives. To evaluate a package you have to actually use it. It is even better if you can use the package in some practical application. The product I am evaluating now is version 6.0x of VI Corporation's DataViews graphics library, DV-Tools, and its graphics editor DV-Draw.

DataViews is a package of high level object oriented graphics utilities. It consists of two parts. Part one is the DV Tools graphics library. Part two is the DV Draw graphics editor. The two utilities are closely integrated. This allows a programmer to use the graphics editor to create user interfaces (views in DataView terminology) and then

manipulate them from a program using the DV-Tools library routines. In addition, anything that DV-Draw can do can also be done by a programmer using the DV-Tools library. DataViews has a large selection of pre-defined graphics objects. They range from simple objects like lines, rectangles, or text, to more complex objects like graphs, input objects, and input techniques. Graphs are dynamic objects that have some input variable (or variables) associated with them. Graphs also have a display formatter associated with them which defines how data will be displayed. A graph might be as simple as a rectangle with an associated variable whose value is displayed in the rectangle, or as complex as a bar or line graph. Higher level graphics objects used for user interaction, like menus or sliders, are defined with a combination of input objects (the graphics object on the screen), input technique objects (a combination of data structures that define how to interact with the input object), and an interaction handler that defines how the object interacts with the user.

As a practical application for the evaluation I decided to write a limited emulation of the LAMPF Touch Panel interface used by the PSR Wire Scanner program (WSPSR). The scope of the emulation was limited enough to be do-able in a couple of weeks while still providing a significant test for the graphics package. Writing this emulation also allowed us to get some feel for the difficulty of emulating our current operator interfaces, and gain experience with object oriented graphics packages, graphics editors, and programming window systems. The hardware used for the evaluation was a VAXstation 3100 with 8 megabytes of memory, and a 19" color graphics scope, running DECWindows on version 5.1 of VMS.

3.1 Evaluation Criteria

Since we do not yet have the specific requirements for the new common operator interface, we also do not have specific requirements for a graphics toolkit to build it. There are, however, some general requirements that will apply to our choice of a graphics toolkit regardless of what specific requirements we may specify for the operator interface. Generally these requirements all reflect the fact that people are our scarcest resource. We can buy hardware or commercial software products much easier than we can hire a programmer.

1. The toolkit chosen for the new operator interface must allow at least an order of magnitude improvement in programming productivity over our current programming methods. The LAMPF and PSR control system have close to 50 man years of effort invested in application programs. If only 20% of that effort were spent on operator interfaces, it would take us 10 man years to re-program all the application programs for a new interface using our old methods. We have to be able to convert all application programs to the new operator interface in a much shorter time frame.
2. The software tools should encourage the use of rapid prototyping to develop new user interfaces. Ideally, users should be able to build a sample user interface without coding.
3. The software toolkit chosen should meet as many of our operator interface requirements as possible. We assume that no toolkit will meet all requirements, which leads to 4.
4. The software toolkit chosen should be extensible. Knowledgeable programmers should be able to easily add new tools to the kit.

5. The software toolkit chosen should be based on open, non-proprietary standards and support distributed computing based on a client server model like the X-Windows standard.
6. Long-term support and maintenance is an important criteria in evaluating any toolkit.
7. Performance and response times are important, but not as important as functionality. We can buy faster, more capable hardware. Hardware is cheap, people are expensive. As an example, adding 16 megabytes of memory to a VAXstation 3100 costs about \$6,000. Even if we did this for 20 workstations, the cost is less than the full cost of one man year of a programmers time. Like any other rule-of-thumb there are limits to how far you can push it. At some point the hardware can become more expensive than the software.

In addition to these general criteria, I evaluated DataViews on its ability to at least match the functionality of the touch panel software used by the PSR wirescanner program.

4.0 PSR WIRESKANER PROGRAM

The Wire Scanners for Proton Storage Ring (WSPSR) program runs from a LAMPF Control System (LCS) console attached to the PSR control computer, and controls the 32 wire scanners used by the PSR control system. The operator interface to the program is through a VT320 based touch panel interface, and a Tektronics 4014 compatible graphics scope used for plotting beam profiles. The actual wire scanner hardware is CAMAC based. It is controlled by a VAXELN MicroVAX II system (PWS) through serial CAMAC (three crates). The code issues commands to PWS and gets data from it using an RPC protocol across Ethernet.

4.1 Touch Panel Hardware/Software

The touch panel hardware was originally based on a VT100 terminal, with a Carrol touch panel system installed. This system mounts an array of infrared beams in front of the VT100 screen and generates an escape sequence to the computer specifying the line and column where the screen is touched. The BTN subroutine library (69 routines, 43 callable by application programs) was written to interface to the touch panel hardware and VT100. The library is called the BTN library because all the routines in it start with the prefix "BTN_". The library provides functions to define, write, and read menus of buttons. The buttons can have labels associated with them. The program using the library can wait for a button to be selected (touched) and read the button selected, or it can be notified of a button select through an Asynchronous System Trap (AST). Normally buttons and their associated text are re-written in reverse video to indicate selection. Text can be written anywhere on the VT100 screen through library routines or ordinary write statements (FORTRAN WRITE(*...)). This year the BTN routines were modified to support a track-ball, or mouse interface as an alternative selection mechanism to the Carrol touch panels. The new "touch panels" are VT320 terminals with a track-ball

interfaced through System Builders ANSI Locators. A reference to a touch panel can refer to either device.

4.2 Wire Scanner Menus

The wire scanner code uses three menu screens on the touch panel it runs from, and a graphics display on a Tektronics 4014 compatible terminal for the wire scanner profiles generated by a scan. Each menu screen has a title, a scrolled area for error messages, a test area to display general information, and a functionally grouped set of sub-menus. The three menu screens are defined as follows:

1. The Main display (figure 3) has two sub-menus:
 1. An action menu section to initiate program actions, such as scanning a set of wire scanners, or selecting other menu screens.
 2. An option menu which sets various flags defining the state of the program.
2. The Pick display (figure 4) has two sub-menus:
 1. A pick menu to select up to six wire scanners for a scan.
 2. A control menu which allows selection of menu screens and some limited set of actions.
3. The Setup display (figure 5) has four sub-menus:
 1. The device menu which allows selection of gain, scale, stroke length, threshold specification, and window size for each wire scanner.
 2. The Miscellaneous menu which allows specification of scan mode, averaging for data and background, beam timing selection, data taking resolution, and centroid and profile width calculation algorithm.

3. The change menu which specifies how data in the other menus is changed. Data values can be increased or decreased when their buttons are selected. The values can increase or decrease "smoothly" (one at a time), or they can "jump" by a value of 10 at a time.
4. The control menu which allows selection of other menu displays and some limited set of actions.

I made no attempt to emulate the Tektronics 4014 graphics display for this study.

4.3 Approach to Emulation

Most of the BTN library involves defining and setting up menus, message areas, etc. I setup the emulating BTN routines to take all the screen definitions from view files (see figure 6 as an example). View files are created by DV-Draw to define a screen and all the graphic objects in it. Typically the objects are named, and a program can reference objects and any values associated with them by name. The calls to the BTN library routines from the wirescanner code remained unchanged in the program, but became null operations as far as defining the button screens was concerned. The routines still returned needed information about menu and button identifiers. Each sub-menu has its own unique menu identifier, and each button within a sub-menu has a unique button identifier. This approach simplified the writing of the emulation BTN routines, and allowed me to work with DV-Draw and interfacing views defined with DV-Draw to a program. I believe this would also be practical for a full emulation of the BTN library.

5.0 Evaluation

I was able to run the wire scanner code using a DataViews emulation of the BTN routines with about two weeks of design and coding work. This was probably about seven or eight working days after subtracting work on other problems, meetings, etc. Before actually

starting work on the emulation, I spent two to three weeks reading DataViews documentation and trying various examples and test programs to try and understand the package. Those weeks of actual work with DataViews were spread out over a period of several months, fit in whenever I had the time to spare from other projects. I should mention that before this project started I had no experience with DataViews, or any other object oriented graphics library. I had no previous programming experience with C.

DV-Draw, the graphics editor, is the easiest tool to use. I found it comparable to most good Macintosh graphics applications in ease of use. The DV-Tools library requires a significant learning curve to fully understand and use its considerable capabilities. I am still climbing the curve. The learning curve problem is compounded by the documentation.

1. Many of the entries in the "DV-Tools Reference Manual" Index refer to the wrong page numbers in the manual .
2. The index for the "DV-Tools Reference Manual" has very few subject, or function entries, most index entries are simply subroutine names. This makes it difficult to find the routines you want to use to perform a particular function.
3. In the "DV-Tools Reference Manual" the documentation for individual subroutines separates the argument calling sequence definitions and the functional explanation of what the subroutines do. The two may be separated by several pages.

4. In the "DV-Tools Reference Manual" there are no examples on how to use subroutines with the subroutine documentation. There is also very little explanation on how subroutines interact with each other with the subroutine documentation. A general overview of how classes of subroutines interact with each other is given in the "DV-Tools User's Guide".
5. The "DV-Tools User's Guide" has a tutorial chapter, but it is empty, it has not been written yet. A good tutorial would significantly help the learning curve problem.
6. The documentation for input objects and input techniques was not detailed enough to allow their convenient use in our applications. DataViews implements complex graphics structures like menus or sliders directly with a combination of input objects, input technique objects, and interaction handlers. For example, when a button on a menu is selected, it is highlighted in a specified way (either background color change, or line thickness change) by its interaction handler. It then remains highlighted until another button is selected. In the limited time I had available for the evaluation, I could not find a way to turn on or off the highlighting of a button at the request of the application program. This is done frequently in the wirescanner program. When an action button, such as "SCAN" is selected, the button is left selected only until the action specified by the button is complete. Then the button is returned to its normal unselected state.

We also bought the Fortran interface for DataViews, and used it for the emulation project. This was a mistake for an initial evaluation. The interface is usable, but trying to learn to use a graphics package written in C by calling it from Fortran is not a good idea. The language differences, mostly in how arguments are called, tend to magnify the documentation and learning curve problem. In addition, DataViews provides an extensive set of C example programs which demonstrate many of the functions of the library. These

examples are essential to really understanding how DataViews works. Only a small percentage of those examples are translated into Fortran examples. One mistake I made early in the project was in not going back to the C examples when I could not find good Fortran examples for what I wanted to do.

I used both the UIS version of DataViews and the DECWindows version of DataViews.

Since Version 5.2 of VAX/VMS will not support UIS, my primary interest was in the DECwindows version. I had some problems with this version of DataViews. For example, when a DataViews program runs under DECWindows, it changes the colors of all the visible parts of other windows in the screen. It also does not properly handle refreshing its window when obscuring windows are moved behind the DataViews window. I am expecting a new release of version 6.0x which is supposed to solve some of the problems. A new major release, 7.0, is also due out soon.

5.1 Order of Magnitude Improvement in Productivity.

My evaluation leads me to believe that in our application DataViews can provide an order of magnitude or better increase in programming productivity for operator interfaces over our current practices. The primary source of this improvement is the ability to use the DV-Draw graphics editor to quickly define the user interface, and then easily integrate it into the program using the DV-Tools library. Having said this, there are several caveats that also need to be clearly stated:

1. The programming staff needs to spend the time needed to climb the learning curve, and become proficient in using the system.
2. A set of graphics extensions needs to be added to the basic DataViews toolkit to support the specific operator interface defined for our common console. In

DataViews terms this would mean adding a library of custom graphics objects, graphs, input objects and input techniques.

3. All of our application programs are written in Fortran. I would recommend not calling low level DataViews routines directly from Fortran. Instead use a set of high level routines with native Fortran argument calling conventions.

5.2 Encourage Rapid Prototyping.

Once the graphics extensions are in place, it should be very easy for us to construct a prototype user interface using DV-Draw. DataViews definitely meets this requirement.

5.3 Already Has Many of Needed Operator Interface tools.

DV-Draw and DV-Tools have a very rich set of functions defined. DataViews meets our requirement here.

5.4 Extensibility.

According to the documentation it should be possible to add graphs types, drawings, input objects and input techniques to the DataViews toolkit. I need to explore this a little further (climb the learning curve a little more). DataViews probably meets this requirement.

5.6 Use Open, Non-Proprietary Standards, X-Windows.

DataViews supports use of X Windows, or DECWindows. The DV Tools library is not based on an open graphics standard, however it is supported on a wide range of computer platforms and operating systems including several versions of Unix. It is important to note that DataViews itself is not a window system. It is built on top of a window system like DECWindows.

5.7 Long-term Support Important.

VI Corporation is an established company with a large customer base. We consider it a "safe" choice. One possible problem for us is that DataViews is oriented towards Unix, and we are a VMS shop. Most of the systems DataViews runs on are Unix systems. The documentation is oriented towards Unix systems. All the file names in the manuals use Unix syntax, for example.

5.8 Performance.

I looked at two test cases. One on the PSR control system computer, a VAX 3600 with 32 megabytes of memory, the other on a VAXstation 3100 with 8 megabytes of memory. In both test cases the processes running the PSR wirescanner program were setup with large WSEXTENT parameters, automatic working set incrementing was enabled, automatic working set decrementing was turned off, and no other programs were active in the system. The performance we are interested in falls into two categories:

1. Response time. How long does it take to get operator input, and perform the useful work requested? This can be anything from how long it takes to change the color of a text or rectangle object to how long it takes to initially run the program. This is our single most important performance criteria.
2. Resource use. How much CPU time and memory does it take to do useful work with DataViews?

The results for a few simple measurements were as follows:

1. Running on the PSR control system, the current production version of the wirescanner program takes about 5 seconds to go from the initial run command, to the the wirescanner program Main menu display, ready to accept input. At this point the program has a working set of 1,237 pages, and has used 1.1 seconds of CPU time. It takes about 3.5 seconds to switch from the Main to the Pick or Setup menus .
2. Running on a VAXstation 3100, the DataViews version of the wirescanner program takes about 12 seconds to go from the initial run command, to the the wirescanner program Main menu display, ready to accept input. At this point the program has a working set of 2,163 pages, and has used 8.0 seconds of CPU time. It takes about 3.0 seconds to switch from the Main to the Pick or Setup menus. Times are about the same for the UIS version of DataViews. These times are probably overstated. All code on the DataViews/DECWindow program was compiled "/debug/noop/check=all", and all libraries were object libraries (no shared libraries). No attempts were made to optimize DataViews, and the use of the emulation package was frequently very inefficient. For example, although the initial menu displays already had all buttons labeled, the the wirescanner program program re wrote all button labels anyway. Regardless of what the problem is, twelve seconds is too long for us to wait for application programs to start. To be usable, the start up time must be at least as fast as the current system.

The start up time for the DataViews version of the wirescanner program is unacceptable to us. The performance of the current program on the VAX 3600 is barely acceptable, we would prefer a startup time of one or two seconds. I believe the problem with the

DataViews version of the wirescanner program is in window/process creation, opening and reading view files, and initializing data structures. Once everything is setup, response time is better than on the current system, although resource use is considerably larger. It takes less time for the DataViews version of the wirescanner program to switch from one menu screen to another, and the response to selecting a button appears faster than the current version running on the 3600. I plan to look at a number of optimization possibilities to improve the initial startup times, and I expect to be able to get startup times down to the five or six second range. The control group has one DECstation 3100, and I expect to receive a demo version of DataViews for the DECstation 3100 within the next couple of weeks. I will then try to get some comparison of the performance of the two systems. We will also have an HP 300 series workstation for about 30 days; if I can get a DataViews distribution for that we can compare it as well. According to technical support, DataViews provides the ability to save a view file defined by DV-Draw in ASCII format, then move it to another type of computer and import it into DV-Draw running on that hardware. I plan to try it on the DECstation, and HP workstations.

5.9 BTN Library Requirements.

DataViews was able to duplicate the functionality of the BTN library routines with one exception. VAX/VMS provides the option of calling an Asynchronous System Trap (AST) on the completion of most low level I/O requests (QIOs). This means that the system will interrupt the calling process, and call a programmer specified subroutine when the I/O completes. The BTN software uses this feature to optionally call a user specified routine when a button is selected. DataViews does not provide an AST interface to its user input routines. AST routines are used by several application programs using the BTN subroutine library, including the wirescanner program. I easily modified the BTN emulation routines to simulate an AST call for the wirescanner program, but this

particular approach might not work in every case. If this function is really needed, I believe we could provide it with some programming effort. DataViews probably meets our requirement here.

6.0 Conclusions

From the view of its functionality, I believe DataViews could be used as the graphics toolkit for the new LAMPF/PSR common operator interface. Using this type of integrator, object oriented graphics library and graphics editor offers the prospect of very large increases in programmer productivity for programming our user interfaces. I am not sure however, that any practical level of optimization will cut the program startup times on the VAXstation 3100 workstation to an acceptable level. If this is the case, then to use DataViews we will need a more powerful VAXstation or switch to one of the more powerful Unix RISC based workstations. This is not just a simple matter of buying more powerful hardware to solve a performance problem. There are no VAX based workstations significantly faster than the VAXstation 3100 available today, although I expect that significantly faster machines will be available within the next year or two. Moving to a Unix based system would also involve major programming effort to convert our control system and application programs to run on a Unix platform. In the long run this might be the best thing to do, but it would be very expensive in the short run. We are still investigating graphics packages other than DataViews, and other workstation hardware as well.

References

- [1] G.P. Carr et al, "The Status of the LAMPF Control System Upgrade," submitted to The 1987 Europhysics Conference on Control Systems for Experimental Physics, Oct 1987.

- [2] P.N. Clout et al, "The PSR Control System," Nuclear Instruments and Methods in Physics Research, A247 pp. 116-121, 1986.

LAMPF Control System

FIGURE 1

Current PSR Control System

FIGURE 2

[illegible][illegible]

LOCAL AND REMOTE	PLAN DATA	FRONT DATA	STORE	SEQUENTIAL MODE
RELOCATE REMOTE	COORD FRAME	LOG POSITIONS	USE WINDOW	USE THRESHOLD

SCAN	SETUP	PICK	STOP
SCAN, REPEAT	REANALYZE DATA		
	CENTER H	TPSLEW	OVERRIDE
RETRACT	CENTER V	PRINT DATABASE	DROP

FIGURE 3

WSFSR--PICK DEVICES

Retracting picked devices
Retract complete

LDWS01	LDWS02	LDWS03	LDWS04	LDWS05	
LDWS06	LDWS07	LDWS08	LDWS09	LDWS10	LDWS11
LDWS12	LDWS13	LDWS14	LDWS15	LDWS16	LDWS17
SRWS01	SRWS02	SRWS03	SRWS04	SRWS05	SRWS06
RDWS01	RDWS02	RDWS03	RDWS04	RDWS05	RDWS06
LDWS01	LDWS02	LDWS03	LDWS04	LDWS05	LDWS06

MODE = FLY BEAM REP = AVG (REQ) = BKG = BI
AUTO 1 SRT0 40 20 (5) 4 40

ALGORITHM WIDTH
Percentile 16

WS	GAIN	SCALE	STROKE	THRESH	WINDO
LDWS01	10	AUTO	50	0	12
LDWS02	10	AUTO	50	0	12
LDWS03	10	AUTO	50	0	12
LDWS04	10	AUTO	50	0	12
LDWS05	10	AUTO	50	0	12
LDWS06	10	AUTO	50	0	12

MAIN SCREEN	SETUP	UNPICK ALL	NEXT BEAM	DROP
----------------	-------	---------------	--------------	------

FIGURE 4

WSPSR--SETUP PARAMETERS

3-0	SCALE	STROBE ON	THRESH COUNTS	WINDOW MM
10 8	1	AUTO	50	0
10 8	1	AUTO	50	0
10 8	1	AUTO	50	0
10 8	10	AUTO	50	0
10 8	10	AUTO	50	0
10 8	10	AUTO	50	0

MODE AUTO	REFLY 1
# AVG (REQ) 20 (5)	# BIGG 4
BEAM SRTO (40)	# BINE 40
ALGORITHM Percentile	PERCENT 16

UP	JUMP
MAIN SCREEN	
PICK	
REANALYZE DATA	
USE THRESHOLD	
USE WINDOW	
DROP	

FIGURE 5

[illegible]

UPDATE END-GROUND	PLOT DATA	PRINT DATA	STORE	SEQUENTIAL MODE
EMULATE REMOTE	DON'T ERASE	LOG POSITIONS	USE WINDOW	USE THRESHOLD

SCAN	SETUP	PICK	STOP
SCAN REPEAT	REANALYZE DATA		
	CENTER H	TPSLEW	OVERRIDE
RETRACT	CENTER V	PRINT DATABASE	DROP

FIGURE 6