NASA-CR-178065
19860013848

RESEARCH TRIANGLE INSTITUTE

NASA Contractor Report 178065

RTI/3052/00-01F

Display System Software For The Integration Of An ADAGE 3000 Programmable Display Generator Into The Solid Modeling Package C. A. D. Software

R. J. Montoya and Harold H. Lane, Jr.

Research Triangle Institute
Research Triangle Park, N. C. 27709

Contract NAS1-17890

March 1986

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

FOREWORD

This report was prepared by the Center for Systems Engineering of the Electronics and Systems Unit, Research Triangle Institute, Research Triangle Park, North Carolina, under contract NAS1-17890. The work was administered by the Systems and Analysis Branch (SAB) of the Space Systems Division of the Langley Research Center of the National Aeronautics and Space Administration.

A software system that integrates an ADAGE 3000 Programmable Display Generator into a C.A.D. software package known as the Solid Modeling Program is described. The software system was designed, implemented, and tested at the Research Triangle Institute and later installed and demonstrated at the Systems and Analysis Branch's display system laboratory.

The Solid Modeling Program (SMP) is an interactive program that provides the capability to model complex solid objects through the composition of primitive geometric entities. In addition, SMP provides extensive facilities for model editing and display. SMP was developed at Langley Research Center by Computer Sciences Corporation.

The ADAGE 3000 Programmable Display Generator (PDG) is a sophisticated, color, raster scan, programmable display generator with local intelligence provided by a 32-bit bit-slice, bipolar microprocessor (BPS). The modularity of the system architecture and the width and speed of the system bus allow for additional co-processors in the system. These co-processors combine to provide efficient operations on and rendering of graphics entities.

The resulting software system takes advantage of the graphics capabilities of the ADAGE 3000 PDG in the operation of SMP by distributing its processing modules between the host and the PDG. Initially, the target host computer was a PRIME 850, which was later substituted with a VAX-11/785. Two versions of the software system were developed, a phase I version and a phase II version. In the phase I version the ADAGE 3000 is used as a frame buffer. In the phase II version SMP was partitioned, and some of its functions were implemented in the ADAGE 3000 by means of ADAGE's SOLID 3000 software package.

In the performance of the tasks described in this report, the Research Triangle Institute has worked closely with personnel of the NASA Langley Research Center. Ms. Dariene DeRyder and Ms. Cheryl Allen of SAB have provided valuable technical guidance and coordination throughout the project. Mr. Kenneth H. Jones and Mr. Donald P. Randall of Computer Sciences Corporation have provided helpful insight into SMP.

The work described in this report was performed by personnel of the Control and Display Systems Section of the Research Triangle Institute. Personnel that conducted the work for the tasks described in this report were Mr. R. Jorge Montoya, Mr. Harold H. Lane, Jr., and Mr. Timothy L. Turner. Mr. Montoya served as the Project Leader, and Dr. James G. Haidt, Director of the Center for Systems Engineering, served as Project Manager. Messrs. Montoya and Lane authored the report.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

LIST OF ILLUSTRATIONS

LIST OF TABLES

LIST OF ABBREVIATIONS, ACRONYMS, AND SYMBOLS

ACD	Analysis and Computation Division
AGG4	Advanced Graphics Generator
AMD	Advanced Micro Devices
ASCII	American Standard Code for Information Interchange
BPS	Bipolar Processor Set
BYU	Brigham Young University
CAD	Computer Aided Design
CMP	Color Map
CSE	Center for Systems Engineering
CIG	Computer Image Generation
CRT	Cathode Ray Tube
CRS	Cursor
CSC	Computer Sciences Corporation
DEC	Digital Equipment Corporation
DMA	Direct Memory Access
DR	Dynamic RAM
EADI	Electronic Attitude Director Indicator
FORTRAN	Formula Translator
FBC	Frame Buffer Controller
FSS	FORTRAN Support Subroutines
GIA	Gary Intermediate Assembler
HIRES	High resolution
Hz	Hertz
IDL	IKONAS Display Language
IF	Interface
ICROSS	Intermetrics Cross Assembler
IKASM	IKONAS Assembler

LIST OF ABBREVIATIONS, ACRONYMS, AND SYMBOLS

(Continued)

ISC	(Graphics Device)
I/O	Input/Output
KW	Kilowords
K	Thousand
LaRC	Langley Research Center
LUVO	Look Up and Video Output
LORES	Low Resolution
MA1024/D	Multiplier Accumulator with Perspective Divide
Mbits	Megabits
MEGA	Million
MCM	Microcode Memory
MPC	Multi Peripheral Controller
NASA	National Aeronautics and Space Administration
NTSC	National Television Standards Commission
OTV	Orbital Transfer Vehicle
PDG	Programmable Display Generator
PIXEL	Picture Element
PIO	Programmed Input/Output
RAM	Random Access Memory
RTI	Research Triangle Institute
SAB	Systems and Analysis Branch
SMP	Solid Modeling Program
SR	Static RAM
TCS	Terminal Control System
TV	Television

LIST OF ABBREVIATIONS, ACRONYMS, AND SYMBOLS

(Concluded.)

TM	Trademark
VLSI	Very Large Scale Integration
XBC	Cross Bar Channel
XBS	Cross Bar Switch

1.0 INTRODUCTION

The practice of engineering has come to depend more and more on computer aided design (CAD) software packages. These packages provide the designer with the ability to synthesize concepts efficiently and, depending on the sophistication of the package, analyze and view the results fairly quickly.

Perhaps one of the most CAD-dependent engineering endeavors is that branch of aerospace engineering that deals with the design and analysis of large space vehicles and structures such as the space station. To address the needs of this application, a research and development effort was started in October, 1984, to enhance and improve the interactive capabilities and response time of an existing CAD program (SMP) used by researchers of the Systems and Analysis Branch of the NASA/Langley Research Center.

The purpose of this report is to document the work performed under this contract.

1.1 Research Background

The Systems and Analysis Branch of the NASA/Langley Research Center supports its research on proposed space station configurations with a variety of software tools available on a variety of display systems. One such tool is provided by the Solid Modeling Program (SMP), a sophisticated CAD software system. SMP is an interactive program capable of modeling complex solid objects through the composition of

primitive geometric entities (Ref. 1). The outputs of SMP (wire-framed or shaded solid models transformed and resolved for visibility) are sent to a display terminal for subsequent display. SMP also provides extensive facilities for model editing.

SMP operates in a variety of display systems including the one consisting of a PRIME 850 host computer and a variety of graphic output devices such as the Tektronix 401X, the AED 512 (or 767), and the ISC 8001. In this general configuration, the host computer provides the interactive and operational environment for SMP, and the graphics display devices serve as display buffers and interfaces to the display medium (CRT). A conceptual block diagram of this display system is illustrated in Figure 1-1.

About two years ago, SAB purchased an ADAGE 3000 raster scan programmable display generator to enhance the graphics generating capabilities of the branch. The ADAGE 3000 PDG, with its local intelligence and architectural capacity for modular expansion including special purpose co-processors, has the capability of performing some of the graphics functions then being performed by SMP in the host computer. This capability formed the genesis for the research and development effort described in this report.

1.2 Research Scope And Goals

The overall objective of the work described in this report has been to exploit the graphics capabilities of the ADAGE 3000 PDG in the operation of SMP by distributing SMP's processing modules between the host computer and the PDG. The goal was the development of a graphics software system which would allow the use of SMP in an optimally partitioned display system. The resulting graphics system software

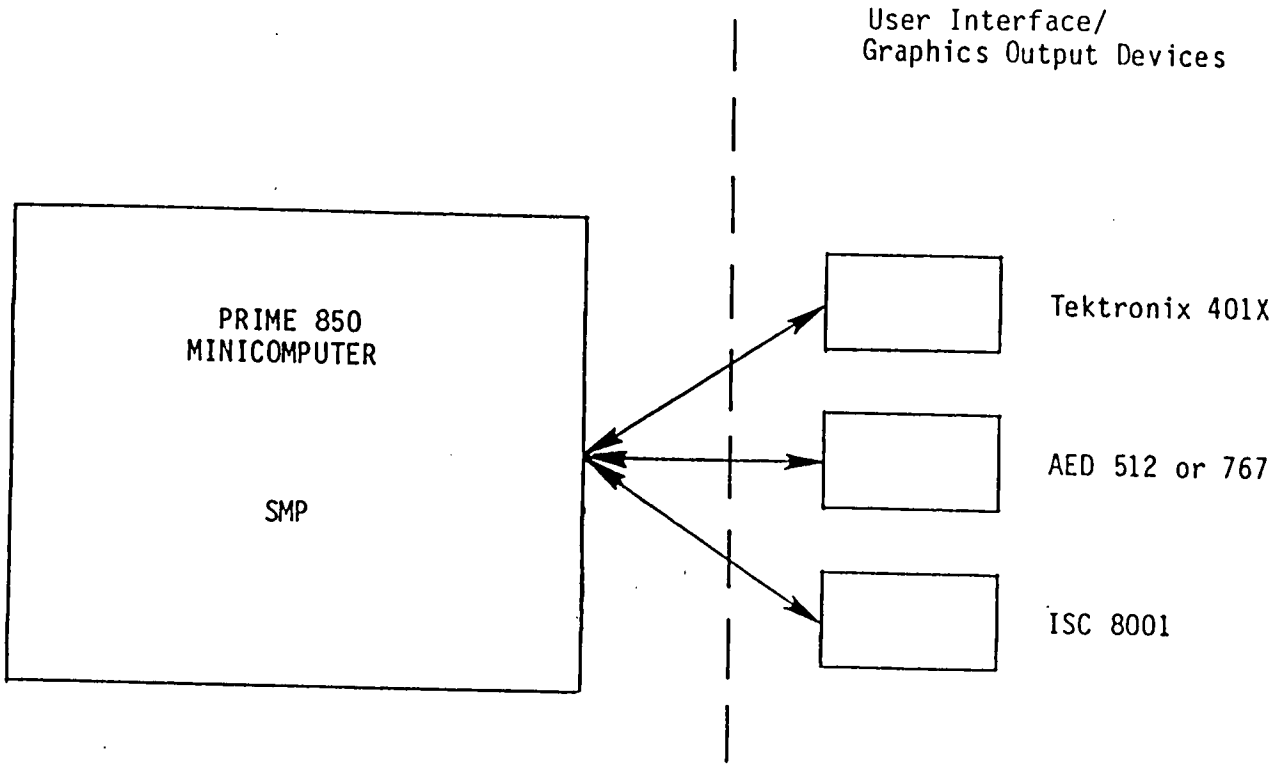


Figure 1-1. Conceptual Block Diagram of SAB's Original Display System.

would provide a capability for rapidly creating solid 3-D models of proposed space station configurations and studying the effect of space station's orientation and orbit position on its intended operations.

The scope of work in this contract included the following:

1) Provide consultation as needed to LaRC personnel in the area of PRIME 850/ADAGE 3000 interface development.

2) Integrate ADAGE 3000 PDG into existing LaRC solid modeling software.
3) Restructure LaRC solid modeling software to exploit the power of the ADAGE 3000 PDG to generate high-speed renderings of complex, 3-D models and to manipulate these models.
4) Assist in the implementation of the software modifications on the NASA/LaRC PRIME 850 minicomputer.
5) Document the software modifications.

The specific tasks to be performed under this contract included the following:

1) Initial integration of ADAGE 3000 PDG into LaRC Solid Modeling Software System to include:

   a) Assistance to LaRC personnel in the development of the host (PRIME)/PDG interface.
   b) Integration of the ADAGE 3000 PDG capabilities into the existing software system such that the PDG is capable of displaying images generated by the host. This task included the porting of SMP

to RTI's VAX-11/750, the writing of FORTRAN subroutines to send images to the ADAGE 3000 PDG, and the installation of the resulting software system in the NASA PRIME 850.

2) Restructure of LaRC Solid Modeling Software to use features of the ADAGE 3000 PDG to include:

   a) Incorporation of the high-speed graphics features of the ADAGE 3000 PDG including distributed processing between the PRIME 850 host and the ADAGE 3000 PDG. This task included the evaluation of alternate partitioning schemes for host-PDG interaction, evaluations of available languages for the PDG, design of the PDG software architecture and host-PDG interface, and specification of PDG routines required for this task. The PDG routines shall include the following features: coordinate transformations, high-speed vector and character generation, high-speed grey scale/color fill, and high speed window/zoom.
   b) Implementation of the PDG-based rendering software.

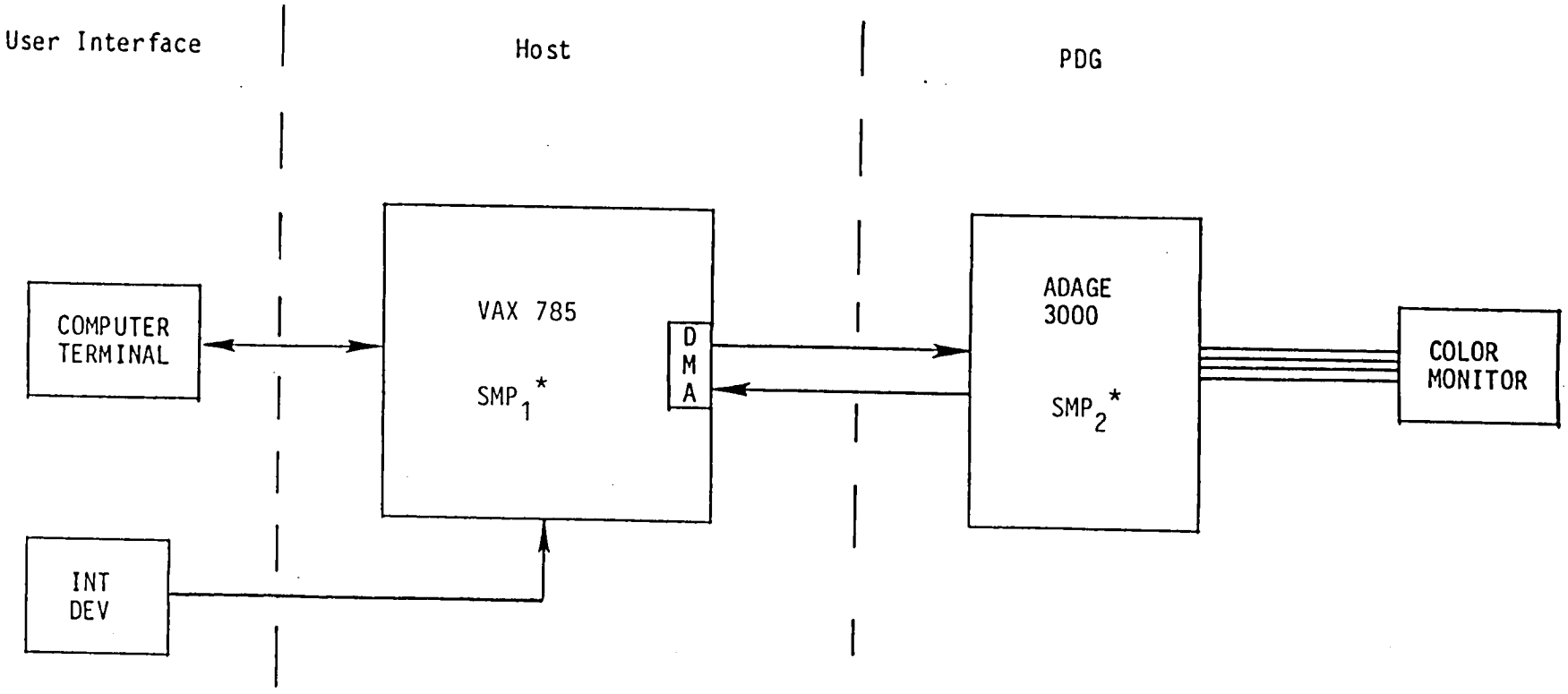
3) Document the implementation and use of the PDG-rendering software including:

   a) Discussion of the partitioning schemes and PDG languages considered and evaluated.
   b) Description of the data flow between SMP and the ADAGE 3000 PDG.
   c) Overall logic flow of the resulting graphics software system.
   d) Detailed instructions on program usage and operation.
   e) Listings of source code.

As can be seen in the tasks above, originally, the display system was specified as that consisting of a PRIME 850 host computer and the ADAGE 3000 PDG. Later on, the host changed to a DEC VAX-11/785 computer. The work described in this report was developed for the VAX-ADAGE display system. A conceptual block diagram of the target display system is illustrated in Figure 1-2. The graphics software system was developed at RTI's Digital Graphics Laboratory and later installed in SAB's display system.

1.3 Organization Of The Report

The organization of this report is as follows. The hardware and software components of the target display system are described in Section 2.0. Next, Section 3.0 describes the design and implementation of the SMP/ADAGE 3000 integration. Section 4.0 presents a users' guide to interactions with the integrated SMP/ADAGE 3000 software system. Section 5.0 presents a discussion of technical issues which need to be addressed prior to the implementation of further enhancements to the SMP/ADAGE PDG graphics software system. Section 6.0 presents a summary of accomplishments, and Section 7.0 presents a summary of conclusions and recommendations. The report concludes with nine appendices. Appendix A includes the static calling structure for the SMP/ADAGE software system. Appendix B provides a list of the RTI-modified SMP subroutines. Appendix C includes listings of the software modules developed for the integration of SMP with the ADAGE 3000. Appendix D includes a list of source code modules with the subroutines contained in each. Appendix E presents a VAX/VMS compilation command file indicating the compiler options required by each module. Appendix F presents a list of include files and VMS logical symbols required during the compilation, linking, and execution processes. Appendix G includes



*SMP_1 + SMP_2 > SMP, where SMP_1 contains user interactions and modeling software, and SMP_2 contains modeling or display software. The implication is that the performance of the partitioned SMP system (i.e., SMP_1 and SMP_2) will be better than the orginal SMP.

Figure 1-2. Conceptual Block Diagram of SAB's Target Display System.

the SMP link file which shows which object modules and libraries should be linked to create the executable file for SMP. Appendix H describes subroutine STRMDL. Appendix I includes references.

2.0 SYSTEMS AND ANALYSIS BRANCH DISPLAY SYSTEM

The display system at SAB consists of a DEC VAX-11/785, an ADAGE 3000 PDG, SMP, and the software system developed by RTI. This section presents a detailed description of this display system with particular emphasis on the configuration and operation of the ADAGE 3000 PDG.

2.1 Hardware Configuration

The hardware complement of SAB's display system consists of a DEC VAX-11/785 host computer and an ADAGE RDS-3000 color, raster scan programmable display generator. The function of the host computer is to provide the operational and interactive environment for SMP. The function of the PDG is to provide the efficient rendering of the objects defined through SMP in the host.

2.1.1 The Host Computer.- The VAX-11/785 consists of an 11/785 CPU, a floating point accelerator (FP785), and 6 megabytes of ECC MOS memory. The Unibus adapter supports two RA-81 456-megabyte disks, a 9-track 1600 BPI tape drive, 8 terminals, a line printer, and a DECNET interface to other DEC computers. The parallel DMA interface to the ADAGE 3000 PDG resides in the Unibus adapter. The configuration of the VAX-11/785 is illustrated in Figure 2-1.

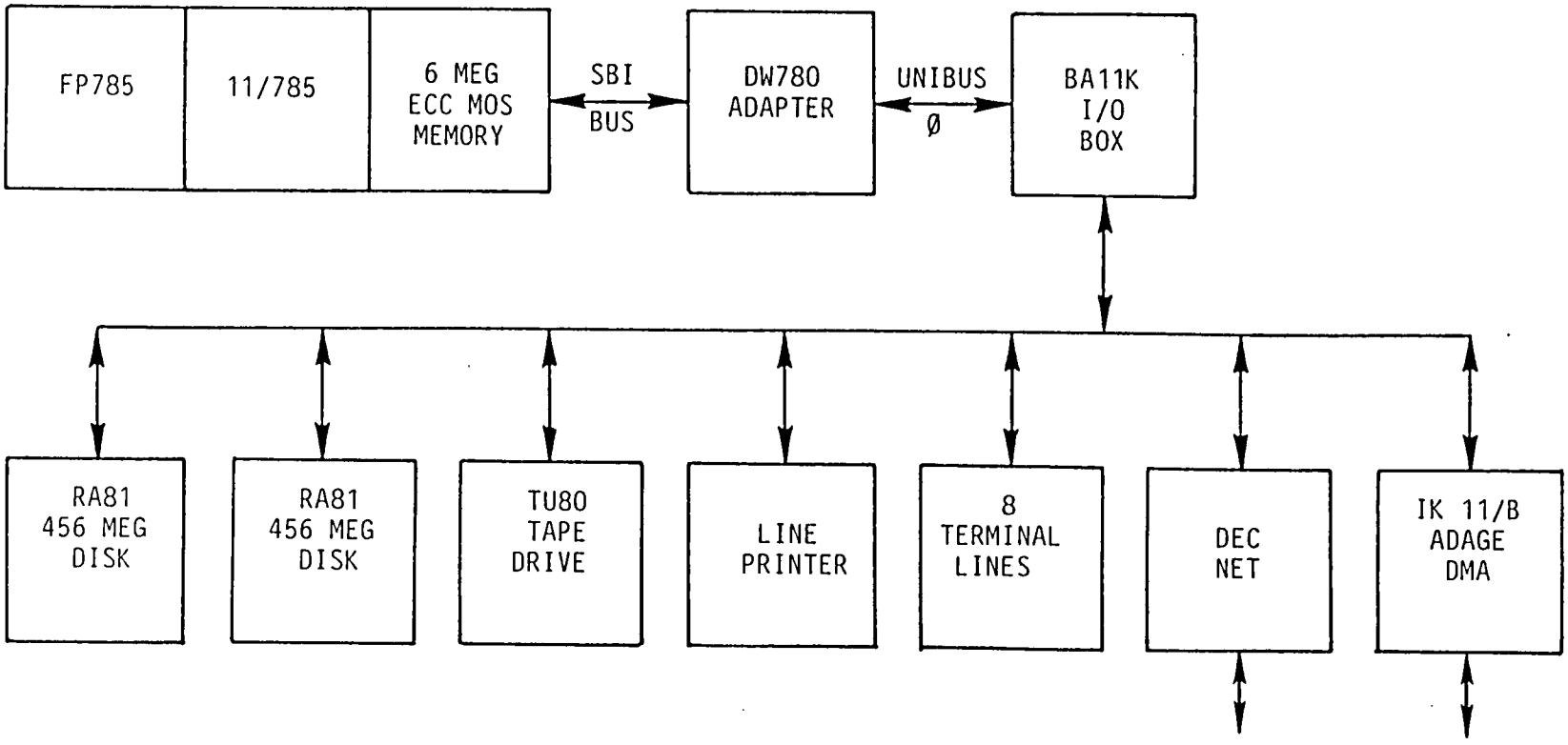


Figure 2-1. SAB's VAX-11/785 Configuration.

2.1.2 The Programmable Display Generator.- The ADAGE 3000 is a high performance, modular, raster scan, color display generator with a 32-bit bipolar microprocessor built around AMD's 2903 bit slice microprocessor and the AMD AM2911 microprocessor sequencer (Refs. 2 and 3). Individual modules plug into a 32-bit wide data bus having a 100 nsec. cycle time. The PDG has a large address space supported by a 24-bit address bus also having a 100-nsec cycle time. This bus-oriented architecture and large address space allow the installation and control of additional processing and storage modules, such as image memory to support increased pixel depth (more colors) at a given resolution within the limitations of the system.

Figure 2-2 illustrates a PDG architecture which is representative of the SAB's ADAGE 3000 PDG. The figure shows the co-processing modules (BPS, MA1024, and AGG4) and their associated memories (microcode MCM4 and data (SR8)) in its lower portion. The figure also shows the digital video stream (FBC, XBS, and LUVO's) in its upper portion. The display or image memories shown in the middle portion of the figure provide the interface between the processing modules and the video stream. Each processing module, including the host computer, can write to the image memories without intervention of the local processor. Interactions between the host computer and the PDG occur through the interface (IF) module which shares a board with the frame buffer controller.

Also shown in the lower portion of the figure is the multiperipheral controller (MPC). This Motorola 68000-based module supports a variety of interactive graphics devices (joystick, push buttons, mouse, etc.). This module, with an appropriate memory complement and software, also provides the potential for an onboard host in the ADAGE 3000 PDG, i.e., a potential for using the ADAGE 3000 PDG as a workstation.

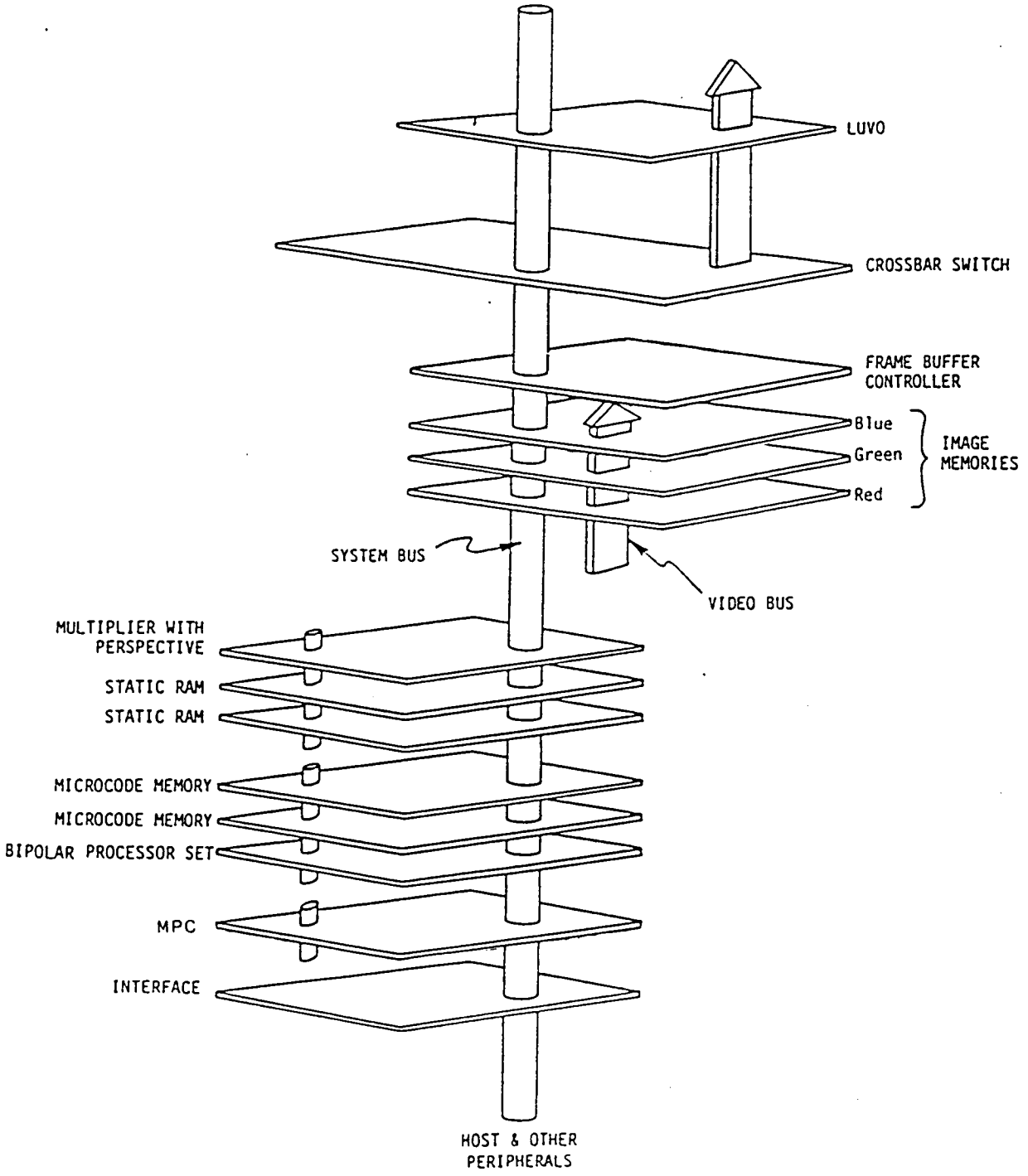


Figure 2-2. SAB's ADAGE RDS-3000 PDG Architecture.

2.1.2.1 The interface module: The interface module (IF) is a two card set (one IK11B in the host and one FBC/IF in the PDG) that provides the coordination necessary for fast, efficient parallel DMA or PIO data communications. This is a two way link that allows the host to transfer instructions and data to the PDG as well as to read back status and data from the PDG.

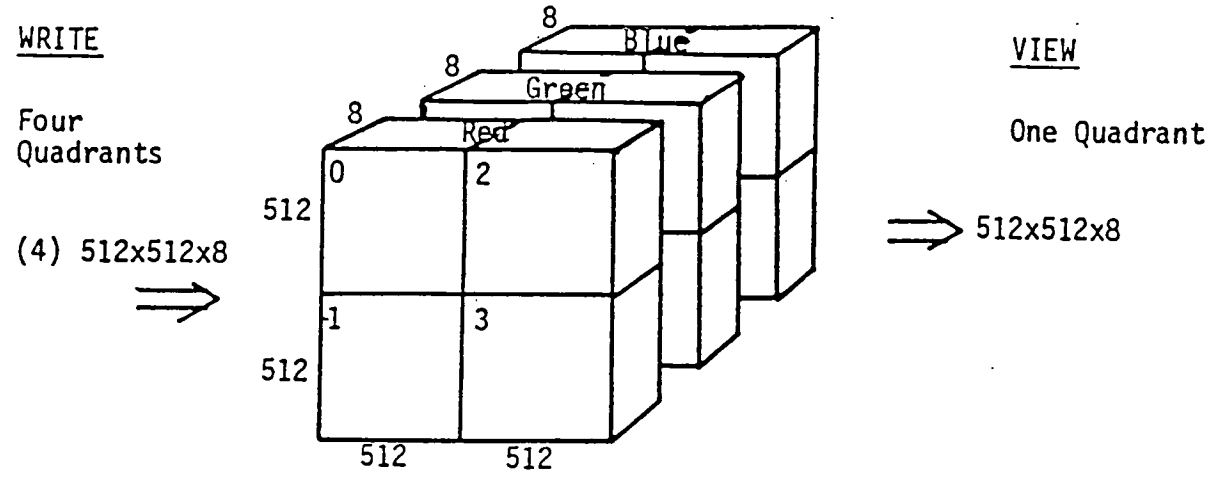
2.1.2.2 The PDG processor: The ADAGE 3000 main processor, the BPS, is implemented using AMD's AM2903 bipolar, bit slice processor and the AM2911 bit slice microprogram sequencer. The processor operates on 32-bit words and is controlled by a microprogram consisting of a sequence of 64-bit wide microcode words. The BPS operates in conjunction with at least 4K of microcode memory (MCM4) into which the microcode is downloaded from the host and with at least 8K of static RAM (SR8) memory into which graphics application programs and data are downloaded by the host. Both the MCM4 and the SR8 are dual ported. The processor communicates with the MCM4 through dedicated data and address busses. Each microinstruction executes in 200 nsec. The combination of the wide horizontal microcode word (64 bits) used by the processor and the parallel data path implementation allow the execution of fairly complex instructions in one microcode cycle.

2.1.2.3 The multiplier/accumulator: The multiplier/accumulator with perspective divide option, the MA1024/D, is a microprogrammable hardware multiplier and accumulate module based on a TRW VLSI chip. The MA1024/D has an on board 1024x16 memory for storage of coefficients which allows for the efficient calculation of isometrics and perspective, 3D, point transformations. The multiplier is ported to both the ADAGE bus and to at least one static RAM memory. The operation of the module is controlled by its own microprogram. Typical operation multiplies an array of coefficients taken from the coefficient memory with another array obtained from the working (SR8) memory. Results or accumulation of results can be read through the ADAGE bus or placed in the SR8. The

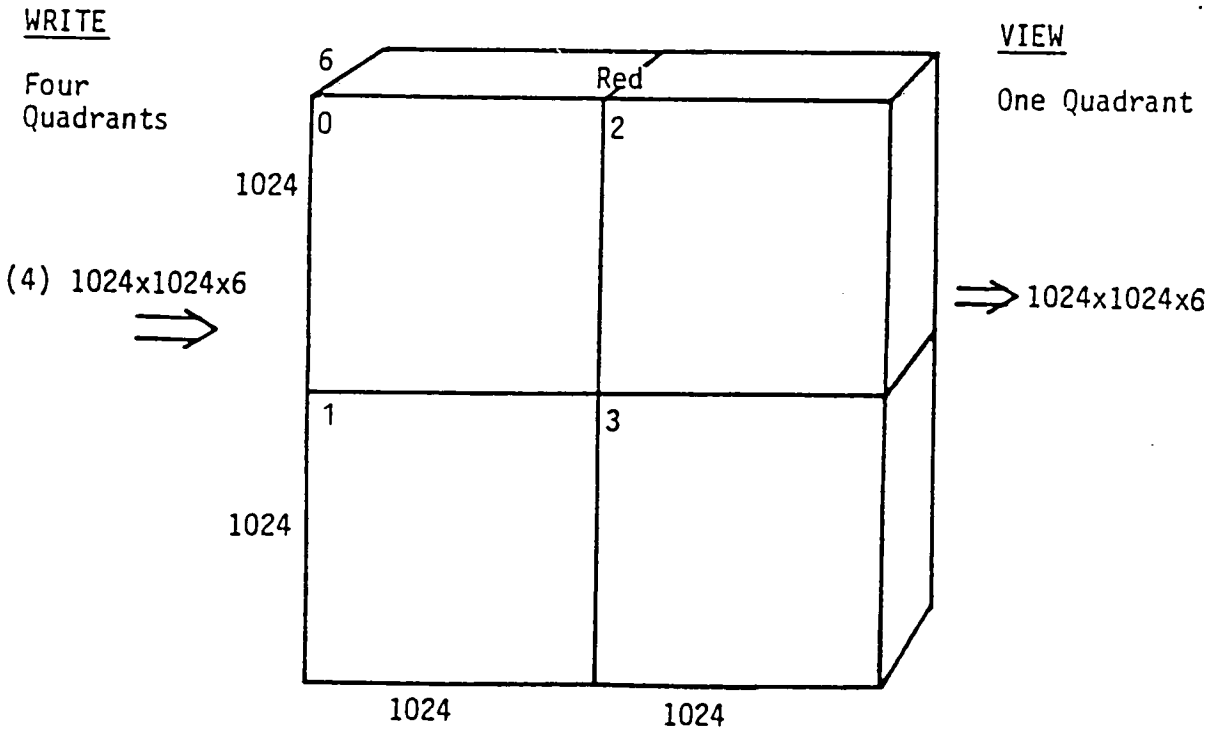
perspective option provides for the computation of "W" and its reciprocal, allowing perspective division to be performed. Three dimensional point transformations with and without perspective are performed in less than 4 and 6 microseconds, respectively. Even faster transformation rates can be achieved with additional multiplier modules.

2.1.2.4 The advanced graphics generator: The advanced graphics generator , the AGG4, is a 16-bit processor for the ADAGE 3000 implemented using the AMD AM2901 bit slice processor and the AM2910 microprogram sequencer. The AGG4 can process and write a string of 32 pixels (in high resolution mode) and a string of 16 pixels (in low resolution mode) in one bus cycle. The main use of this module is for fast rendering of flat shaded polygons. Another important use of this module is for the fast generation of character strings.

2.1.2.5 The frame buffer: The frame buffer or display image memory, the GM256, is a dual-ported, 200 nsec cycle time dynamic RAM into which the images are usually written on a pixel by pixel basis. One port of the memory communicates with the ADAGE data bus and the other with the digital video stream. However, addressing takes place in both instances through the ADAGE bus. The storage capacity for each GM256 is 8 Mbits (four 512x512 8-bit images or four 1024x1024 2-bit images). The viewable capacity of each GM256 is 2 Mbits (one 512x512 8-bit image or one 1024x1024 2-bit image). Frame buffer memories are grouped to provide the pixel depth needed in the particular application. Color variability in the image is a function of the pixel depth as well as the contents of the color lookup table (LUVO). The LUVO is discussed in more detail in a later section. The configuration of the display memory in SAB's PDG is illustrated in Figure 2-3. The figure shows that each GM256 provides essentially four quadrants for selective storage of image data and one quadrant for selective viewing. This expanded storage capacity (over the GM-64 memories) allows image buffering in depth or z, in x, and in y. The choice of buffering scheme depends on how many bits



(A) Medium Resolution: 512x512xN, N = 1→24



(B) High Resolution: 1024x1024xM, M = 1→6

Figure 2-3. SAB's ADAGE RDS-3000 Display Memory Configuration.

per pixel (how many colors) are needed for each application program. The memory can be organized in bit planes and can be configured in a variety of formats. Which bit planes get written into during a write cycle is controlled by write mask registers whose values can be changed dynamically under program control. Which bit planes contribute to an image during the image refresh cycle is a function of the dynamic settings of the crossbar switch (XBS). In addition, the GM256 memory has a shade register which allows the AGG4 or the BPS to perform efficient mask mode writes.

An image in the PDG consists of a set of three image boards (red, green, and blue) with an optional (alpha) board which can be used for overlays and other functions. The red, green, and blue components of each image are at the same x and y addresses but differ in z (bit depth). The write mask register is used to write to the z location (by depth) of each board. In low resolution mode (512x512) each board contributes eight bits in z whereas in high resolution mode (1024x1024) each board contributes two bits in z. Since the PDG is expandable, a system can have many image memories. Each image has a distinct address with the first set of four cards (including an alpha card) corresponding to image 0, the next set to 1, and so on. The ADAGE 3000's system capacity is four sets of four full color, 512x512 resolution images. The same cards can also be addressed as one full color (plus alpha), 1024x1024 resolution image.

2.1.2.6 The digital video stream: The digital video stream consists of the frame buffer controller (FBC), the crossbar switch (XBS), and the lookup table and video output module (LUVO).

The FBC is a programmable module that controls the reading of image data from the frame buffer memories and the sending of this data to the LUVO either directly or through the XBS. The FBC also generates all

video system timing. The FBC contains eight 32-bit wide registers which can be written into either by the user's program in the PDG or by the host computer. These registers control the scan standard at which the PDG operates, the x/y window and viewport, cursor position, and image erasure. The FBC accesses all cards of an image simultaneously through their video output ports and outputs the resulting serial video stream to the crossbar switch on a pixel by pixel basis.

The crossbar switch (XBS/F) is a programmable module which allows the arbitrary mapping of image memory bits to color map input bits. The mapping is controlled by registers which can be set dynamically by the application program. The register settings allow connecting an output bit to any one of the input bits. There are thirty-five input bits and thirty-two output bits. On the input side, the extra three bits correspond to paging and cursor on/off bits added to the video stream by the frame buffer controller. These bits are control bits for the LUVO. The other thirty-two bits correspond to pixel depth information including the eight bits of the overlay channel. The XBS is used for multiple buffering of displays and for routing pixels of individual displays to each of several possible LUVO's in the system.

An additional feature of the XBS is the hardware fill option. This is a special digital circuit in the XBS which latches on when a single pixel in a specific bit plane is detected and remains on until another single pixel in the same bit plane is detected. Thus, this feature allows the flat shading of large areas of the screen at video rates at the cost of writing only a few pixels, i.e., the writing of the boundary of the area to be shaded.

The implementation of this option is as follows. The first eight input bits (bits 0-7) of the XBS are also connected to the last eight input bits (bits 24-31) of the XBS. When a user wishes to apply

hardware fill to an object in the display, he/she must do two things: draw the boundary of the object into one or more of the first eight bits of the display memory (RED card) and connect the particular output bit of the XBS to which that bit is going to contribute to the corresponding input bit (24-31) of the XBS in which the outline of the object was drawn. Thus for example, if one wishes to fill the runway of an EADI display, one can draw the outline of the runway onto bit plane 1 of the image memory and program the XBS to connect its output bit five (arbitrary for this example but color map dependent in a given application) to its input bit twenty-four. The result is that when the left edge of the runway is encountered the hardware fill turns itself on and remains on until the right edge of the runway passes through the circuit. Appropriate loading of the color lookup table is necessary to insure that when the particular bit is on, the desired color is obtained. This option is used for large area fill at video rates. (It should be noted that systems with this option do not have a pixel path between the alpha card in the image memory and the corresponding input bits into the XBS).

The color lookup table and video output module (LUVO/24) is a combination memory and digital-to-analog conversion module. The lookup table portion of the module is a high -speed 24-bit memory with 1024 locations in which a pallete of colors is stored. The memory is dual ported with one port connected to the video bus and the other to the system bus. The colors are defined by the user as combinations of red, green, and blue values. Each component can be defined to an 8-bit resolution. The colors can be downloaded from the host and placed at a specific address in the color map. The value of each pixel is used to index into the lookup table. The particular values of red, green, and blue stored at this location are then used as the inputs of the corresponding D/A converters. The lookup table can be operated in two modes, full color and pseudo color. In full color mode, the value of the pixel arriving from the red, green, and blue cards are treated as three independent addresses into the red, green, and blue color maps.

In pseudo color mode, the value of the specified pseudo color channel pixel is replicated to the inputs of the other two non-specified channels. Therefore in pseudo color, all three color locations are at the same address whereas in full color mode, all three color locations can be at different addresses.

The video output module is a three-channel digital to analog converter which receives red, green, and blue data from the lookup table and converts them to RS-170 or RS-343A analog video signals. The eight-bit DAC's can operate at 40 MHz to support the required pixel data rate of 1024x1024 displays.

The elements of the SAB's ADAGE 3000 PDG are summarized in Table 2-1.

2.2 Software Configuration

The software environment of SAB's display system is distributed between the host and the PDG. Host based software subsystems include SMP, SOLID 3000, the FORTRAN-based interface routines developed by RTI, and the FORTRAN runtime environment provided by the VMS operating system. PDG-based software subsystems are the SOLID-3000 supporting microcode and the graphics display lists developed by RTI. The VAX-11/785 operates under the VMS Version 4.2 operating system. The ADAGE 3000 PDG operates under SOLID 3000/16M Version 1.0. Also available in SAB's display system is an IDL2 compiler for the ADAGE. However, IDL2 is not used in this application.

Table 2-1. SAB's ADAGE 3000 Configuration

MODULE	QTY	DESCRIPTION
CB-24	1	Twenty-four slot card cage with power supply
IF/FBC	1	DMA interface and Frame Buffer Controller
BPS-256	1	Bipolar microprocessor and sequencer
MCM4	2	Microcode Memory (4 KW, 64 bits per word)
MA1024/D	1	Multiplier/Accumulator with Perspective divide option
SR8	2	Static Ram memory (8 KW, 32 bits per word)
GM256	3	Graphic Memories (512x512x8 or 1024x1024x2 each)
XBS-34/F	1	Cross-Bar Switch with Hardware Fill Option
LUVO-24/HS	1	Look-up Table and Video Ouput module
MPC/32	1	Multifunction Peripheral Controller
JS3	1	Joystick

2.2.1 Host-based Software.- The function of the host-based software is to enable the user to build new models and assemblies, support editing of existing models and assemblies, and support interactions with the PDG. The software suite in the host consists of SMP, SOLID 3000, the RTI-developed interfaces between SMP and the PDG software, and the FORTRAN runtime system.

2.2.1.1 Solid Modeling Program: The Solid Modeling Program developed by Computer Sciences Corporation for NASA's Systems and Analysis Branch is a comprehensive software system that allows a user to create and modify complex three-dimensional models built up from three-dimensional "primitive" parts. SMP has both keyboard-based and graphic-based model editors as well as a variety of rendering methods to allow users to observe the results of their modeling. The completed model can be stored in several formats, of which most are compatible with other graphics-related software.

A solid model generated by SMP (Ref. 1) is the result of an aggregate of geometric modeling primitives available to the user. These modeling primitives fall into five categories: basic, swept, Boolean, external, and assemblies. For a description of each category see pages 6 through 34 of reference 1. The basic modeling primitives currently available in SMP are: "boxes," "cones," "spheres," "paraboloids," "tori," and "trusses." Each of these are completely defined through dimension and construction parameters. The user can generate additional shapes by varying the construction attributes of certain of the primitives.

According to reference 1, "the SMP software is structured as a hierarchy with each level being associated with a set of program commands. The system is menu and command driven with online help facility available at each level." The highest level of user interaction

is the command level which allows the user to input an existing model (READ), output a solid model (WRITE), modify the solid model by performing basic editing operation on its primitives (EDIT), display the solid model (DISPLAY), and perform limited mass property analysis and dimensioning on the solid model (MISCELLANEOUS). A command level may contain one or more command sublevels. A description of the available sublevels and their relationships is presented in Table 4 of the cited reference.

In addition to the geometry editor available as a command level, SMP provides the user with a graphics editor as a sublevel of the DISPLAY command which "allows manipulation of the model parts through direct interaction between the two dimensional (2-D) projection of the model on the screen and a graphics input device." (Ref. 1).

2.2.1.2 SOLID 3000: SOLID 3000 (Ref. 4) is a package of FORTRAN-callable microcode routines developed by ADAGE, Inc. for the ADAGE 3000 PDG. The microcode routines generate line and shaded (flat and smooth) surface display of polygonal and mesh-structured solid modeling data. The package also provides FORTRAN and microcode routines to support communications between the host and the ADAGE 3000 PDG. SOLID 3000 microcode routines are generated by the ICROSS microcode compiler.

The SOLID 3000 system depends heavily on FORTRAN applications programs to set display attributes (such as color, light, and viewport), and to send model data (such as a mesh of x, y, and z coordinates or a string of text) to the SOLID 3000 microcode through the FORTRAN interface routines. According to reference 4 since "SOLID 3000 performs z-buffered display processing, data and attributes can be sent from the host in any order and images can be created with any amount or mixture of data, number of viewports, light sources, etc."

There are two versions of SOLID 3000, SOLID 16M AND SOLID 64K which support full and false color images, respectively. SOLID 16M requires 24 bits of image memory (red, green, and blue boards) which provide over 16 million colors whereas SOLID 64K requires 16 bits of image memory (9 bits of hue and 7 bits of intensity) which provide over 64 K colors. A 16-bit z-buffer is used by both configurations. In addition, SOLID 16M has an 8-bit coverage buffer for image enhancements such as anti-aliasing, translucency, and texture mapping.

SOLID 16M supports wireframe as well as flat and smooth (Gouraud and Phong) shaded image rendering with/without anti-aliasing as well as with other image enhancing features such as realistic translucency and texture mapping. In addition, the latest version of SOLID 3000 can support transformations, picking, instancing, cutting, and animation playback.

SOLID 3000 uses a left handed coordinate system (+x to the right, +y up, and +z into the screen) with the origin at the center of the screen. SOLID 16M supports full color and pseudo color display capabilities. Another, very useful, feature of SOLID 3000 is its use of different BPS sender ID's for image writes and z-buffer writes. This allows the application programmer to protect selected bit planes in the image.

A detailed description of these features as well as the FORTRAN interface routines available in SOLID is found in reference 4.

2.2.1.3 RTI Interface routines: Because SMP was developed to operate in a display system that contained a variety of graphics rendering devices (display terminals), its design emphasized the use of device independent routines for the functions of SMP and a standard interface to the display terminals. This allowed almost all the code in SMP to use an identical subroutine interface to the graphics rendering device (terminals), no matter which device was selected by the user. In some instances, this standard interface used the Terminal Control System (TCS) subset of the Tektronix PLOT10 software package.

As will be seen in the next section, RTI integrated SMP with the ADAGE 3000 in two phases. In the Phase I version (the ADAGE used as a frame buffer) RTI made use of this device independent scheme, essentially adding another device to the SMP software system. However, a Plot10-like set of subroutines was not available for the RDS-3000. Therefore, to implement the interface to the device independent functions (such as erase, draw line, draw cursor, etc.), RTI wrote a set of FORTRAN subroutines which implemented the ADAGE interface routines needed by SMP. These subroutines are: ADDRWS, ADLINE, ADMOVA, ADMOVS, ADVUPR, ADWIND, CLIP2D, ENDPT_CODE, and LOGIC_INTERSECT.

In the Phase II version (the ADAGE used as a PDG) RTI interfaced SMP to the SOLID-3000 subroutine package in the host computer. The SOLID subroutines then communicate with the ADAGE PDG. The RTI-developed interface routines are: GET_SLDCOLR, SLDSMOOTH, SODRWA, SODRWS, SOMOVA, SOMOVS, SEND_2_SOLID, and SLDZWIND.

In addition, certain general purpose interface subroutines were needed to control the frame buffer controller, load the cursor registers, and load the color map. These subroutines are: FBC, CURSLD, and CMAPLD.

Listings of the RTI-developed subroutines are included in Appendix C.

2.2.2 PDG-based Software.- The two items that together constitute the software in the ADAGE RDS-3000 PDG are the ADAGE-supplied microcode and the instructions which the application program (SMP/SOLID system) sends to the ADAGE.

2.2.2.1 ADAGE-supplied microcode: The ADAGE-supplied microcode support routines reside in two MCM4 microcode memory modules which together have a capacity of 8KW of 64-bit horizontal microcode words. These routines are invoked when instructions and data, sent by the host-resident portion of SMP, call for a particular type of rendering to be done. Control is then passed to the microcode routine and the object is rendered into the image memory. The microcode used in the ADAGE is unmodified.

2.2.2.2 Instructions and data sent to the ADAGE at runtime: These instructions and data are created by applications program calls to the SOLID 3000 FORTRAN library at runtime. When the call occurs during the running of the applications program a display list is created, initialized, built, and sent to the ADAGE SR8 memory module. These instructions and data then cause the image to be rendered in the manner described in the preceding section.

3.0 SMP/ADAGE 3000 PDG INTEGRATION

As stated in the introduction, the overall objective of the work described in this report has been to incorporate the sophisticated graphics generation capabilities of the ADAGE 3000 PDG into the operation of SMP. To accomplish this objective it was recognized early in the design process that the functional modules of SMP would have to be distributed between the host computer and the ADAGE PDG. The resulting graphics software system would make full use of the resources of the host and the PDG to improve user/SMP interactions in the model rendering area. This would provide the capability for rapidly creating solid 3-D models and viewing them in different orientations. This would also allow the users to evaluate more options in the configuration of a model. Thus, for example, the software system could be used to evaluate proposed space station configurations and studying the effect of space station's orientation and orbit position on its intended operations.

3.1 Overview

This section provides a general description of the SMP/ADAGE 3000 PDG integration. The method of integration evolved from issues relating to how the host and the PDG can interact as well as from the comprehensiveness of the software interfaces required to implement the access of the ADAGE 3000 rendering functions by the SMP modules.

3.1.1 VAX/ADAGE 3000 Interactions. - From a hardware standpoint, interactions between the host computer and the PDG occur through a two-card set, one located in the host and the other in the PDG. In the host side of the interface, the module is an IK11/B ADAGE interface card which is located in the VAX-11/785 Unibus(TM). In the PDG side of the interface, the module is an IF interface card which shares the same board with the frame buffer controller (FBC) module and which is located in the ADAGE system bus. The IK11/B and the IF perform the necessary hardware protocol to support DMA reads and writes from and to the ADAGE 3000. The IK11/B can select any address within the ADAGE address space. Read and write requests are arbitrated by the main ADAGE bus arbitration logic on a manufacturer-set priority basis. This technique avoids bus contention problems during host/PDG interactions.

3.2 Design

The basic design goal was to make the rendering capabilities of the ADAGE PDG fully available to SMP while maintaining, to the largest possible extent, SMP's basic software architecture and user interface. Therefore, design issues revolved primarily around what language and what level of software interface to use to implement the SMP/ADAGE PDG communications and interactions.

SMP is a comprehensive and sophisticated software system structured hierarchically with many levels of commands, menus, and submenus (Ref. 1 and Appendix A). On the other hand, because the ADAGE 3000 PDG is a very powerful and flexible graphics system, application programs developed for its use tend to be complex (Refs. 4 and 5). Because of these considerations, the integration of SMP with the ADAGE PDG was done in two phases. In phase I, the ADAGE 3000 would be used solely as a frame buffer. In phase II, the ADAGE 3000 would be used as a graphics computer. This approach would allow RTI to acquire extensive

operational experience with SMP (during phase I) and to use this experience as well as its experience with the ADAGE graphics languages to keep the complexity of the implementation of the interface in phase II to a minimum.

The phase I configuration would only use the frame buffer and video stream portion of the ADAGE 3000 PDG, without using the local processing capabilities of the PDG. In this configuration, SMP would perform its traditional role of supporting the definition and display of models. In particular, the display function would be done by SMP's graphics rendering algorithms in the same way as they are done for the other graphics devices that SMP supports. In order to display the SMP-generated images, this approach would require that only the lowest level routines in SMP's device independent code (primarily those in source file DIDDGS.FOR) be modified to include pixel writes to the ADAGE frame buffer. This configuration also would require that SMP configure the elements of the ADAGE 3000 PDG's digital video stream, i.e., the frame buffer controller, the cross bar switch, channel cross bar, and the color look up tables. Additional subroutines would be needed to accomplish these controlling functions.

The phase II configuration would use the full capabilities of the ADAGE 3000 PDG. In this configuration, SMP would also perform its traditional role with respect to model generation but its display functions would be distributed between itself and the PDG. User interactions would be preserved in SMP and any additional interaction resulting from the distributed display functions would also be maintained within SMP. Software interfaces would be needed to match the data and command conventions in SMP with those in the application programs written in the PDG graphics language to render the images defined through SMP. Examples of data that would be passed between SMP and the PDG application programs running in the BPS high-speed processor are endpoints and shades of lines and vertices, shades and normals of

polygons, hidden surface removal coverage data, transformation coefficients, etc. It should be noted that this configuration would also require the implementation of subroutines to control the configuration of the PDG's video stream modules.

3.3 Implementation

The original version of SMP used in SAB's display system (PRIME 850 interconnected to a variety of display devices) was written in ANSII-77 FORTRAN with PRIME extensions to FORTRAN and PRIMOS operating system calls. RTI modified this version of SMP and installed it in its display system (VAX-11/750 operating under the VMS operating system and interconnected with an ADAGE 3000 PDG). Since the original SAB's ADAGE-based target display system was to be hosted in the PRIME, the project plan initially called for the development work to be done in RTI's display system and for porting the resulting display system software to SAB's PRIME/ADAGE display system. This plan would allow RTI to develop the software system while SAB personnel, with some consultation with RTI, developed the PRIME/ADAGE interface microcode. In the Spring of 1985 it was learned that SAB's VAX-11/785 would be available earlier than expected. Consequently, the target display system became a VAX/ADAGE one. This development made the initial adaptation of SMP to RTI's VAX a very significant endeavor and also made the porting of the resulting software system to SAB a more straightforward process. The concerns shifted from dealing with significant differences between operating systems (VMS versus PRIMOS) to dealing with differences between different versions of the same operating system (VMS Vers. 3.4 at RTI versus VMS Vers. 4.2 at SAB). It should be noted that, in parallel with the RTI effort, SAB modified and enhanced the original version of SMP to operate in the VAX environment. However, since this version did not become available to RTI until late into the contract, it was decided to proceed with the version of SMP originally adapted to the VAX by RTI.

3.3.1 Phase I Integration. - Phase I integration consisted of implementing a "Tektronix's Plot10-like" interface to the ADAGE. This needed to be done since ADAGE does not provide an interface of this nature for its RDS 3000. Initially, this appeared to be a straightforward task. However, this involved implementing many non-trivial functions such as two dimensional object clipping, windowing, viewporting, etc. These are features supported by all of the display devices with which SMP interacts. The phase I implementation does not support the drawing of text on the graphics screen (in contrast to other SMP/display device implementations). This feature was not implemented because drawing characters on on a pixel by pixel basis from the host into the ADAGE's frame buffer would be an extremely time consuming operation. Text prompts and menus appear on the user's computer terminal.

To implement the Phase I configuration several of SMP's "device independent" support routines were modified. These include: GDINIT, CMPRES, DDVTOS, DICOLR, DICTAB, DICURS, DIDRWS, DIDRWW, DIDSHS, DIDSHW, DIDUMP, DIERAS, DIMOVS, DIMOVW, DIPAUS, DISCTE, DIVUPR, DIWIND, LOGO, TITLE, and WPXLA.

When the ADAGE frame buffer is selected as the display device, the phase I software is invoked as follows:

1) GDINIT initializes the VAX/ADAGE communications channel, the frame buffer controller to 512 X 512 60 Hz repeat field, the crossbar switch to straight through with overlay defeat, the channel crossbar switch to pseudocolor off the red memory card, the write and erase masks to select all image bits, the color map to the standard SMP color map, the cursor register to display a tracking cross (when enabled), and erases the

screen.

2) CMPRES sets the computation resolution for the ADAGE to 512 X 512.

3) DDVTOS converts virtual coordinates [0, 1] into ADAGE screen coordinates.

4) DICOLR sets the current foreground color.

5) DICTAB sends the selected color table to the ADAGE color map.

6) DICURS prompts for and obtains an X, Y specification from the user. See section 4.1 for a more complete description of the user interface to the cursor routine.

In the subroutines which implement line drawing, modifications took place to the following routines:

7) DIDRWS draws into the ADAGE frame buffer memory using the virtual coordinate system [0, 1].

8) DIDRWW draws into the ADAGE frame buffer memory using world (model) coordinates.

9) DIDSHS draws dashed lines into the ADAGE frame buffer memory using the virtual coordinate system [0, 1].

10) DIDSHW draws dashed lines into the ADAGE frame buffer memory using world (model) coordinates.

11) DIMOVS performs a move by setting the "current location" using the virtual coordinate system [0, 1].

12) DIMOV performs a move by setting the "current location" using world (model) coordinates.

13) DIDUMP sends the contents of the display command buffer to the ADAGE display.

14) DIERAS clears the ADAGE memory. It is used to erase whatever display has been previously written into memory.

15) DIPAUS prompts the user for a <CR> in order to be able to continue program execution.

16) DISCTE sets the value of the "current color" and also sends this value to the specified ADAGE color map location.

17) DIVUPR defines a rectangular screen-coordinate viewport from user-specified virtual coordinates [0, 1].

18) DIWIND defines a rectangular window in the world (model) coordinate system.

19) LOGO writes the logo "SMP/LARC" in the lower right corner of the frame buffer.

20) TITLE writes a title page when SMP is initially run containing "SMP" in large characters and credits for software authorship.

21) WPXLA is a utility routine for writing a rectangular array of pixels to the display.

In addition, the phase I configuration required the implementation of the following group of FORTRAN-based, ADAGE support routines: ADDRWA, ADDRWS, ADLINE, ADMOVA, ADMOVS, ADVUPR, ADWIND, CLIP2D, ENDPT_CODE, LOGIC_INTERSECT, CMPALD, CURSLD, FBC and XBS.

Brief descriptions of these routines follow:

1) ADDRWA draws solid or dashed lines in floating point world coordinates. Windowing, viewporting and clipping are performed.

2) ADDRWS draws solid or dashed lines in integer screen coordinates. Clipping is performed.

3) ADLINE performs line drawing by writing pixels to the ADAGE frame buffer using a Bresenham-like algorithm.

4) ADMOVA performs a logical "move" in floating point world coordinates. Windowing and viewporting are performed.

5) ADMOVS performs a logical "move" in integer screen coordinates.

6) ADVUPR defines a rectangular screen-coordinate viewport from user-specified screen coordinates.

7) ADWIND defines a rectangular window in world coordinates.

8) CLIP2D performs the Sutherland-Cohen 2D line clipping algorithm.

9) ENDPT_CODE assigns clipping codes to end points. It is used by the Sutherland-Cohen 2D line clipping algorithm.

10) LOGIC_INTERSECT logically intersects the clipping codes of two points. It is used by the Sutherland-Cohen 2D line clipping algorithm.

11) CMPALD loads a color map obtained from a disk file into the ADAGE color look up table.

12) CURSLD loads a cursor definition (a tracking cross) from a disk file into the ADAGE cursor registers.

13) FBC loads user-specified data into the ADAGE frame buffer controller registers.

14) XBS loads user-specified data into the ADAGE crossbar switch registers.

The phase I implementation of the SMP/ADAGE integration kept the changes to the code in a few tightly-controlled routines. Therefore, the impact on the user's interactions with SMP is minimal as will be seen in the USER'S GUIDE section of this report. The chief difference to the user is that the text of the menus appears on the terminal that the user is logged onto, while the graphics appear on the ADAGE monitor.

Implementation of the phase I configuration of the SMP/ADAGE integration served its intended purpose well. It provided enhanced raster graphics display capabilities to SMP and improved display generation speed. It also served to familiarize RTI with the general internal structure of SMP.

3.3.2 Phase II Integration. - The phase II integration places the rendering of text, lines, flat-shaded surfaces, and smooth-shaded surfaces in the ADAGE. The high-speed 32-bit microprogrammed BPS-32 processor's tightly-coupled relationship to the ADAGE'S image memory allows it to render images at a much greater speed than host-drawn pixels. This speedup is the order of 10 to 100 times the rate that the VAX 11/785 can draw the pixels over the DMA interface. The speedup factor is a function of what particular objects are being viewed, what view of the objects is being rendered, whether the viewed objects are requested to be rendered as lines, flat-shaded surfaces, or smooth-shaded surfaces, and whether anti-aliasing has been requested. This speedup in rendering time should greatly enhance the user-SMP interaction rate.

In order for the ADAGE PDG to perform the functions identified above at a significant speedup, it must be programmed very efficiently using one of the graphics language available for the PDG. In addition to the programming task itself, this approach requires a matching of the the data structures describing the model in SMP with those assumed by the graphics language in the PDG. Therefore, it is also required to locate the data structures which are accessed by SMP and properly format them for SOLID 3000 so that the model description can be sent to the ADAGE for rendering. Consequently, the implementation of the phase II configuration was significantly more complex than that of the phase I configuration. In phase I the interface to SMP was clearly defined in isolated sections of code. On the other hand, in phase II the interface to SMP required detailed knowledge of the internal data structures of SMP so that the proper information could be located, formatted and sent to the ADAGE for rendering.

Several candidate graphics languages are available for the task of rendering objects in the ADAGE PDG. These included the Ikonas Display Language (IDL), GIA, ICROSS, FSS, and SOLID 3000. Each of these

languages and their associated requirements for interfacing the ADAGE with SMP has its advantages and disadvantages.

IDL is an assembler-like language which produces opcodes and data which are interpreted by a microcoded dispatcher running in the ADAGE PDG. While this system can draw lines and polygons at a high rate of speed, it leaves the formatting, sending, hand-shaking and timing of the SMP/ADAGE interaction entirely to the applications programmer. Furthermore, the language itself is not very flexible, requiring the user to resort to writing awkward code to accomplish many simple tasks. Also, IDL does not have intrinsically built in support for such necessary graphics tasks as clipping or smooth shading polygons.

GIA and ICROSS are similar C-like languages which generate microcode for the ADAGE BPS-32 processor using cross compilers running in the host computer. While these languages can generate images at acceptable rates on the ADAGE, they both lack a library of basic functions to perform basic graphics functions. Basically, users are given access to the entire ADAGE 3000 PDG internal address, register, and data space and from there on they are on their own. Formatting and managing data in the host are also left up to the user as in the case of IDL. It should be noted that RTI has recently completed the development of a comprehensive library of ICROSS/GIA-based graphics routines. This package, called RAP, for Real-time Animation Package, provides the user with a high level interface to the rendering functions of the ADAGE 3000 PDG. Unfortunately, RAP was not available in time for this project.

FSS and SOLID 3000 are both FORTRAN-based libraries of graphhics routines callable by the user's application program. FSS uses IDL-like code and microcode in the ADAGE PDG; SOLID 3000 uses only microcode in the ADAGE PDG. This microcode was written specifically for rendering high quality images in the ADAGE.

Although FSS can perform anti-aliasing, monitor gamma correction, and z-buffer wire frame, flat and smooth shaded images, it has some drawbacks which prevented its selection as the graphics language for this application. A major drawback of FSS with respect to this application is that its smooth shading method assumes that the ADAGE PDG video stream is configured in pseudocolor mode. This means that only 512 different distinct colors can be used in smooth shading all of the objects appearing on the screen at any one time. Another drawback of FSS is that the FORTRAN library interface is quite intricate. This may in turn reduce the host processing speed because of the large number of subroutine and function calls needed to specify what data and opcodes should be sent to the ADAGE. Also, with such a detailed interface the potential exists for errors to remain hidden in the developed code.

SOLID 3000 is an ADAGE supported graphics rendering package intended to provide high quality image rendering. The package is designed to be called by an application program using a FORTRAN library call interface. The package supports full color (16 million shades) line drawing, flat shading of polygons, and smooth shading (Gouraud and Phong) of polygons with user access to the lighting model parameters. Additional features of the package include anti-aliasing, displaying of PDG-resident display lists, z-buffered rendering, and monitor gamma correction. A potentially important feature of SOLID 3000 is its ability to overlap the PDG's rendering processing with the host's processing. This could be used in the future to further enhance the speed of the SMP/ADAGE software system.

SOLID 3000 offers an additional important feature for this application. The user's application interface to SOLID 3000 is less complex than to FSS. Two or three function calls can render a complicated object after a similar number of initialization calls.

Based on the review of the graphics languages identified above and the requirements of the application, RTI recommended SOLID 3000 as the language of choice for the integration of the ADAGE PDG rendering capabilities into SMP. Without question, SOLID 3000 provides the best collection of features required to render images in the ADAGE. The results obtained in both image quality and speed of rendering these images substantiates the choice of SOLID 3000 as the method to use for this application. Side by side speed comparisons of rendering the same parts using both the ADAGE/SOLID 3000 and an AED display terminal showed a clear speed advantage using the ADAGE.

Thus, in the Phase II version of SMP (the ADAGE used as a PDG) RTI interfaced SMP to the SOLID-3000 subroutine package in the host computer. The SOLID subroutines then provide the communication with the ADAGE PDG. The RTI-developed interface routines are: GET_SLD_COLR, SLDSMOOTH, SODRWA, SODRWS, SOMOVA, SOMOVS, SEND_2_SOLID, and SLDZWIND.

Brief descriptions of these routines follow:

1) GET_SLD_COLR converts SMP color numbers 0-7 into the equivalent colors in the SOLID 3000 full color representation.

2) SLDSMOOTH smooth shades one part. This is done by calculating normals at all vertices in the part and then sending the vertices' coordinates and normals to the ADAGE along with a request to perform smooth shading of the part.

3) SODRWA draws solid lines in floating point world coordinates. Windowing, viewporting and clipping are performed.

4) SODRWS draws solid lines in integer screen coordinates. Clipping is performed.

5) SOMOVA performs a logical "move" in floating point world coordinates. Windowing and viewporting are performed.

6) SOMOVS performs a logical "move" in integer screen coordinates.

7) SEND_2_SOLID sends one face of a polygon to the ADAGE 3000. The normal to this face is calculated and also sent to the ADAGE. Windowing and viewporting are performed.

8) SLDZWIND stores Z-window specifications in a FORTRAN COMMON area in world coordinates.

The RTI-modified SMP subroutines are identified in Appendix B. With the exception of subroutine STRMDL, the modifications to these routines are relatively minor and essentially consist of calls to SOLID 3000 subroutines. The meaning and use of the SOLID 3000 routines are documented in the SOLID 3000 Reference manual (Ref. 4). The RTI-modified and RTI-added code has been carefully documented in SMP's FORTRAN source with bracketed comments, "CSLD:" and "CSLDEND" for SMP Phase II; "CADG:" and "CADGEND" for SMP Phase I code changes. Modifications to subroutine STRMDL are described in Appendix H.

4 USER'S GUIDE

This chapter describes the differences in the user's interactions with SMP between the original version of SMP and the two versions which render models on the ADAGE 3000. The differences are minimal in scope. Therefore, an experienced user of SMP should have no difficulty in learning to use the new features in the new versions. The changes that affect the user most are the prompts and responses in the area of rendering flat and smooth shaded objects in the phase II (ADAGE/SOLID) version of SMP.

4.1 Changes To SMP Common To Both Phase I And Phase II

This section details changes that affect the user's interactions with SMP and which are common to both Phase I (ADAGE 3000 PDG used as a frame buffer) and Phase II (ADAGE 3000 PDG used as a graphics computer).

A key difference between the original version of SMP and both ADAGE versions of SMP is that when a character string is to be "printed" on the display it appears on the user's terminal instead. This was done because implementing the drawing of characters on the graphics display pixel by pixel could slow user interaction rates with SMP to an unacceptable rate.

Another difference is found in the graphics editor, in that, for the zoom function, and mass property routines, the user must specify locations on the screen. The method of specification is dependent upon the particular graphics display used, although some type of user control of a cursor is the most often used method. The ADAGE/SMP cursor location function assumes no particular user interaction device. However, the user interaction function has been simulated using the terminal keyboard. The user is prompted for an X, Y pair of screen coordinates and the cursor is then moved to that location on the screen. The user can continue to specify X, Y pairs until the cursor is located in the desired location. The user then indicates that this is the location to be accepted by SMP by typing the character cntl-Z. This action causes SMP to save the last X, Y pair as the selected screen location and also to exit the loop which prompts the user for further X, Y pairs.

This method of screen location specification is to be replaced with a more natural method of user interaction, such as a data tablet, a joystick, or a mouse, when it is procured, installed, and interfaced by SAB personnel. The purpose of including the present method of interaction using the keyboard was to be able to exercise SMP's graphics editor, zoom, and mass property code. An example of this method of screen location specification follows:

```
ENTER CURSOR X, Y LOCATION (^Z TO CONFIRM):
SCREEN IS 0-511 IN X AND Y.  Y IS UPRIGHT
100,200                                  <---[user's response]
[the cursor is moved to (100, 200) on the graphics display]

ENTER CURSOR X, Y LOCATION (^Z TO CONFIRM):
SCREEN IS 0-511 IN X AND Y.  Y IS UPRIGHT
150,200                                  <---[user's response]
```

[the cursor is moved to (150, 200) on the graphics display]

```
ENTER CURSOR X, Y LOCATION (^Z TO CONFIRM):
SCREEN IS 0-511 IN X AND Y.  Y IS UPRIGHT
cntl-Z                                  <---[user's response]
```

The location (150, 200) is then stored as the user's location specification.

4.2 Phase I

The differences in user interaction between the original SMP and the Phase I version of SMP are minimal. This is due to the fact that the Phase I version of SMP performs all of the calculations for rendering and shading for raster devices in the host computer as it did in the original version. The interface between the ADAGE and SMP is restricted to being in the so-called "device independent" routines. These routines are organized in such a way as to make clear what code needed to be modified to support a particular display construct, such as erase, draw line, etc. After these basic constructs had been provided, the ADAGE then presented the same standardized interface to SMP.

The following sections deal with the exact differences that users will see when using the new versions of SMP.

If in response to the prompt:

```
ENTER DEVICE CODE
   1 = TEKTRONIX 401X
   2 = AED 512
```

3 = AED 767
4 = ISC 8001R
5 = ADAGE 3000
6 = ADAGE 3000/SOLID

the user selects

5 - ADAGE 3000

a new message appears stating:

ADAGE RDS-3000 FRAME BUFFER INTERFACE
DEVELOPED BY

RESEARCH TRIANGLE INSTITUTE
RESEARCH TRIANGLE PARK, NC

4.3 Phase II

The Phase II version of SMP/ADAGE has more changes than the Phase I version. They are concentrated mainly in the area relating to flat and smooth shading of models.

If in response to the prompt:

ENTER DEVICE CODE
1 = TEKTRONIX 401X
2 = AED 512
3 = AED 767

4 = ISC 8001R
5 = ADAGE 3000
6 = ADAGE 3000/SOLID

the user responds with:

6 - ADAGE 3000/SOLID

device 6 is selected and a new message (emphasizing that the ADAGE is a Programmable Display Generator) appears stating:

ADAGE RDS-3000 PDG INTERFACE
DEVELOPED BY

RESEARCH TRIANGLE INSTITUTE
RESEARCH TRIANGLE PARK, NC

Two new prompts are issued in both the display command's "menu" command and in the alpha editor's "print part definition and display part" command. The detailed new user interactions are shown in Figures 4-1 and 4-2.

The prompts to the user in regard to a display monitor's gamma in Figures 4-1 and 4-2 represent a new capability. A display monitor's gamma is a specification of the relationship between the brightness of an area on the screen and the voltage needed at the input of the monitor to obtain that brightness. The monitor's gamma is used in calculating the brightness of the pixels used to draw an anti-aliased line and thus affects the appearance of anti-aliased lines and edges of polygons in the image on the display monitor. Typically, high resolution, RGB color monitors have a gamma near 1.8. To obtain the correct gamma value for a particular monitor, the user should try different values near 1.8 while

```
$ R SMP
         ENTER DEVICE CODE
            1 = TEKTRONIX 401X
            2 = AED 512
            3 = AED 767
            4 = ISC 8001R
            5 = ADAGE 3000
            6 = ADAGE 3000/SOLID
6
            .
            .       <-------  A part is either read or created here
            .

         ENTER COMMAND
           R - READ A PARTS OR GEOMETRY FILE
           E - EDIT A PARTS FILE
           D - DISPLAY A GEOMETRY FILE
           P - EVALUATE MASS PROPERTIES
           W - WRITE A PARTS OR GEOMETRY FILE
           H - HELP (DOCUMENTATION)
           Q - QUIT
           X - EXIT
D
          ENTER DISPLAY COMMAND>
         R - RESET VIEWING OPTIONS
         M - VIEWING OPTION MENU
         D - DRAW MODEL
         T - DISPLAY FOUR VIEWS OF MODEL
         Z - ZOOM
         S - HIDDEN SURFACE
         G - GRAPHICS EDITOR
         H - HELP (DOCUMENTATION)
         Q - EXIT DISPLAY
         X - EXIT PROGRAM
M
```

Figure 4-1. New User Prompts in MENU Command

```
        DESIGNATE THE "PARTS" TO DISPLAY
        "*" INDICATES ALL PARTS
        "," IS DELIMITER BETWEEN PARTS
        "-" INDICATES A RANGE OF PARTS
*
         ENTER "Y" TO CHANGE THE
           VIEWING TRANSFORMATION
N
         ENTER "Y" TO CHANGE THE
           VIEWING OPTIONS
Y
         ENTER BACK FACE CULL OPTION
          -1 - NO CULL
           0 - HIDE BACK FACES
           1 - DASH BACK FACES
0
        ENTER "Y" FOR PART LABELS
           OR "N"    FOR NO PART LABELS
Y
        ENTER "Y" FOR ELEMENT SHRINKING
        OR "N"    FOR NO ELEMENT SHRINKING
N
         ENTER "Y" FOR ANTI-ALIASING
Y
         ENTER MONITOR GAMMA (1.8 TYPICAL):
2.0
```

New prompts in SMP Phase II

Figure 4-1. Concluded.

```
$ R SMP
          ENTER DEVICE CODE
             1 = TEKTRONIX 401X
             2 = AED 512
             3 = AED 767
             4 = ISC 8001R
             5 = ADAGE 3000
             6 = ADAGE 3000/SOLID
6
         ENTER COMMAND
           R - READ A PARTS OR GEOMETRY FILE
           E - EDIT A PARTS FILE
           D - DISPLAY A GEOMETRY FILE
           P - EVALUATE MASS PROPERTIES
           W - WRITE A PARTS OR GEOMETRY FILE
           H - HELP (DOCUMENTATION)
           Q - QUIT
           X - EXIT
E
         ENTER EDITOR COMMAND>
                      A - ADD PART
                      M - MODIFY PART
                      D - DELETE PART
                      P - PRINT PARTS
                      C - COPY AN EXISTING PART
                      R - RESTORE A DELETED PART
                      H - HELP (DOCUMENTATION)
                      Q - EXIT EDITOR
                      X - EXIT PROGRAM
A
         ENTER PART NAME - BOX(B),  SPHERE(S),  CONE(C),
         PARABOLOID(P), TORUS(T), TRANSLATIONAL-SWEEP(N),
         ROTATIONAL-SWEEP(R), ASSEMBLY(A)
         BOOLEAN(E), HELP(H), OR QUIT(Q)
B
         ENTER PART DESCRIPTION - 80 CHARACTERS MAXIMUM
B
         ENTER LENGTH(X AXIS), HEIGHT(Y AXIS),AND WIDTH(Z AXIS)(ALL > 0)
1 1 1
         ENTER INTEGER COLOR CODE (1..7)
1
```

Figure 4-2. New User Prompts in Alpha Editor

```
        ENTER:   1 PRINT PART DEFINITION
                 2 PRINT PART DEFINITION AND DISPLAY PART
                 3 CHANGE PART DESCRIPTION
                 4 CHANGE PART SPECIFICATION
                 5 CHANGE PART TRANSFORMATION
                 6 DETERMINE ORDER INDEPENDENT ROTATION ANGLES
                 7 CHANGE PART COLOR
                 8 TO RETURN TO EDITOR (PART OK)
2
        DESIGNATE THE "PARTS" TO DISPLAY
        "*" INDICATES ALL PARTS
        "," IS DELIMITER BETWEEN PARTS
        "-" INDICATES A RANGE OF PARTS
*
         ENTER "Y" TO CHANGE THE
           VIEWING TRANSFORMATION
N
         ENTER "Y" TO CHANGE THE
           VIEWING OPTIONS
Y
         ENTER BACK FACE CULL OPTION
          -1 - NO CULL
           0 - HIDE BACK FACES
           1 - DASH BACK FACES
0
        ENTER "Y" FOR PART LABELS
           OR "N"    FOR NO PART LABELS
Y
        ENTER "Y" FOR ELEMENT SHRINKING
        OR "N"    FOR NO ELEMENT SHRINKING
N
         ENTER "Y" FOR ANTI-ALIASING
Y
         ENTER MONITOR GAMMA (1.8 TYPICAL):
2.0
```

New prompts in SMP Phase II

Figure 4-2. Concluded.

observing nearly horizontal or vertical lines. When these lines lose their "rope-like," "knotted" appearance, the gamma value is correctly set.

In the area of hidden surface display, the new user prompts and appropriate form of responses is given in Figure 4-3. Note that when either the HIDDEN LINE or BOTH option is selected a new message appears stating that hidden surface removal will be performed instead. This is because ADAGE's SOLID 3000 package does not support hidden line removal, a display technique which was often the only type of display that could show depth information on vector or storage-tube displays. Raster displays can effectively display hidden surfaces so the hidden line removal function is not as important for these displays.

Also note that, as in the original SMP, there are two SHADING OPTIONS, flat and smooth (Gouraud) shading. However, within the smooth option, there are new prompts asociated with the lighting model, antialiasing, and monitor gamma correction. The lighting model used by the SOLID 3000 system consists of a fixed light source whose characteristics are defined by the sum of ambient (AMB), diffused (DIF), and reflected (REF) components. Each of these components can be specified in the range from zero (0) to 32767 which corresponds in the lighting model to a fractional value between zero (0) and one (1). The zero value represents black whereas one represents the full intensity of each part's color. In addition to brightness, the reflected component of the light has an exponent specification which determines the visual characteristics (appearance) of highlights on the parts. Although there is no range specified for this parameter, it is observed that small values (<5) produce images with large, diffused highlights whereas large values (>15) produce images with small, concentrated highlights. While this description provides a basic understanding of the lighting model

```
$ R SMP
          ENTER DEVICE CODE
             1 = TEKTRONIX 401X
             2 = AED 512
             3 = AED 767
             4 = ISC 8001R
             5 = ADAGE 3000
             6 = ADAGE 3000/SOLID
6
             .
             .        <-------  A part is either read or created here
             .

          ENTER COMMAND
            R - READ A PARTS OR GEOMETRY FILE
            E - EDIT A PARTS FILE
            D - DISPLAY A GEOMETRY FILE
            P - EVALUATE MASS PROPERTIES
            W - WRITE A PARTS OR GEOMETRY FILE
            H - HELP (DOCUMENTATION)
            Q - QUIT
            X - EXIT
D
           ENTER DISPLAY COMMAND>
          R - RESET VIEWING OPTIONS
          M - VIEWING OPTION MENU
          D - DRAW MODEL
          T - DISPLAY FOUR VIEWS OF MODEL
          Z - ZOOM
          S - HIDDEN SURFACE
          G - GRAPHICS EDITOR
          H - HELP (DOCUMENTATION)
          Q - EXIT DISPLAY
          X - EXIT PROGRAM
S
           ENTER  S - ELIMINATE HIDDEN SURFACE
                  L - ELIMINATE HIDDEN LINES
                  B - BOTH (SEPARATE VIEWS)
                  R - RETURN
L                    <----------- [S or B give the same series of prompts]
```

Figure 4-3. New User Prompts in Hidden Surface Display

```
        DEVICE 6: S - ELIMINATE HIDDEN SURFACE USED <-- [This message is
        ENTER "Y" TO OVERRIDE DEFAULT                    given when L or
        IMAGE DISPLAY OPTIONS                            B is answered]
N
        ENTER SHADING OPTION
           0 - FLAT ELEMENT SHADING
           1 - SMOOTH ELEMENT SHADING
1
        ENTER Y TO CHANGE SHADING PARAMETERS
Y

         ENTER AMBIENT, DIFFUSE, REFLECTED VALUES OF LIGHT:
         (THE SUM OF AMBIENT, DIFFUSE, & REFLECTED SHOULD BE < 32767)
         (AMB=6000, DIF=22000, & REF=4000 IS GOOD)
6000,22000,4000
         ENTER EXPONENT FOR REFLECTED LIGHT (20 IS GOOD):
20
         ANTI-ALIASING? (DEFAULT-NONE)(Y OR N):       New prompts in
Y                                                     SMP Phase II
         ENTER MONITOR GAMMA (1.8 IS TYPICAL):
2.0
         PHONG SHADING? (DEFAULT-GOURAUD)(Y OR N):
Y
         ENTER BACK FACE CULL OPTION
           N - NO BACK FACE CULL
           Y - PERFORM BACK FACE CULL
           R - RETURN
Y

         ENTER "Y" FOR ANTI-ALIASING
Y                                                     New prompts in
         ENTER MONITOR GAMMA (1.8 TYPICAL):           SMP Phase II
2.0
```

Figure 4-3. Concluded.

used by SOLID 3000, it is noted that perception of the effect of a lighting model is very subjective. Thus, to obtain the desired effect, the user might need to iterate on the lighting model parameters. The SOLID 3000 reference manual (Ref. 4) provides additional information about the lighting model.

In addition, under phase II, the smooth shading option supports Phong shading as well as the original Gouraud shading. These shading methods basically differ in that in Gouraud shading the intensities are linearly interpolated between values at the edges of the polygons being rendered whereas in Phong shading the surface normals are interpolated across the visible span of the polygon being rendered (Ref. 6). As a result, Phong shading produces more realistic images than Gouraud shading because highlights of the objects are more faithfully reproduced. However, the rendering time is longer for Phong shading than for Gouraud shading in this application.

Another area where a change is seen by the user can occur in the area where the user is given the option to store an image on disk. A response of "1" answered to the prompt:

ENTER DISPLAY/DISK IMAGE OPTION

gives the message:

DEVICE 6: IMAGE NOT IN STORABLE FORMAT

The SMP Manual states that storing the image in file SMGG.IMG "...is most applicable to generating a shaded image for post processing while working from a non-raster display device." The ADAGE RDS 3000 is a raster device so this option is not as important as it is to a non-raster display.

Another minor difference between the original version of SMP and the Phase II ADAGE version is as follows.

If in response to the prompt:

ENTER BACK FACE CULL OPTION

the user selects

DASH BACK FACES

the back faces will be drawn as solid lines since ADAGE's rendering package currently does not directly support dashed lines. The dashed line feature could be simulated but with an overhead that could substantially slow the draw time of models using this feature.

It should be noted that there are two features on the PRIME/PRIMOS version of SMP which are not implemented in the versions of SMP which RTI ported to the VAX/VMS environment at SAB. These are the interrupt capability and the HELP facility. However, the lack of these features does not affect the ability to render images in either of the two versions of SMP developed by RTI.

5.0 ADDITIONAL TECHNICAL ISSUES

This chapter addresses additional technical issues which have the potential for further improving the performance of the SMP/ADAGE PDG display software system in specific areas. These include performing local transformations in the PDG, adding the capability for model animation, and improving the access time to read the temporary geometry files during image generation.

5.1 Local Transformations

In a typical application, an SMP user develops a model, renders it in wireframe form, and transforms it several times until he/she is satisfied with the result. It is only then that the model is shaded using one of the available shading schemes in SMP. The process of rendering a transformed, wireframe model consists of retrieving the model definition from disk where it exists in a geometry file format and performing transformations on the data twice, once for setting clipping box boundaries and scaling and once for model viewing. The first set of transformations determines the greatest extent of the transformed model in X, Y, and Z for setting clipping box boundaries and scaling parameters. The second set of transformations actually transform the model's points for scaling and viewing. These transformations cannot take place simultaneously - they must be done in sequence in order to have the clipping and scaling information available during the second set of transformations.

In the software system developed by RTI, transformations are performed by SMP in FORTRAN in the VAX-11/785. It has been suggested that performing transformations in the ADAGE PDG would provide additional gains in image rendering speed over what has already been achieved.

There are two ways in which transformations can be achieved in the PDG: splitting the two sets of transformations between the host and the PDG and performing both sets of transformations in the ADAGE PDG. The former can be done using existing facilities in SOLID 3000 and no additional hardware. The latter requires modifications to the SOLID 3000 microcode system residing in the ADAGE's MCM4 memories and additional space in MCM4 memory to store the user written microcode routines. (If this is a problem, additional MCM4 memories can be purchased). In addition, this method would require enough storage space in the PDG to accomodate the entire model. This storage space can be provided by an SP-256 memory. The control and configuration of the SP-256 and the model size that it can accomodate are discussed in the next section.

Although both approaches can be implemented with varying degrees of technical effort, RTI feels that other associated areas need to be addressed prior to the implementation of local transformations in the PDG. This is due to the fact that the portion of the rendering time associated with disk I/O is the dominant factor of the rendering time. In fact, the potential speed improvement attainable with local transformations could be insignificant compared to the time spent doing I/O to retrieve parts in geometry file format. For example, it has been observed during the development effort that the time spent doing disk accesses slows the rendering of "wire frame" line drawings. This is in contrast to the rendering of shaded images in which the time spent rendering is usually longer than the time spent in reading the model from the disk.

Because performing local transformations in the PDG shows uncertain gains and the need for additional hardware and microcode development, RTI recommends that this area is not addressed until the disk I/O area is dealt with.

5.2 Model Animation

Model animation in real time would provide the SMP user with a new capability to facilitate his/her ability to analyze the three dimensional nature of the resulting models. The resources required to perform model animation include additional hardware in the PDG and in the host computer, additional software in the host, and an expanded user interface in SMP.

Model animation would involve putting an entire model into the ADAGE, leaving it there and repeatedly sending transformation information to the ADAGE. This means that the model would not have to be read from its direct access disk file, or sent to the ADAGE each time a different view of the model was drawn. In fact, a continuous smooth rotation of relatively complex models should be attainable using the techniques of double buffering, mask mode erasing of image memory, and drawing of aliased wire-frame image. An additional issue here is choosing how the user would specify the model's rotation. This could make use of a joystick, "toothpick"-style joystick, data tablet, or other "valuator" interaction device.

In order to store an entire model's data in the ADAGE additional memory would be required. It is in this area that ADAGE's SP-256 memory could be useful. The intended use of the SP-256 memory board is to allow users the ability to store large display lists in the ADAGE 3000 display system. An SP-256 is essentially a DR-256 image memory board

which is accessed in word mode rather than pixel mode. One SP-256 memory can hold approximately 16,000 wire frame polygons or 11,000 flat or smooth shaded polygons. For comparison purposes, it is noted that the space station, shuttle, OTV, and solar engine data provided to RTI by SAB, which contains 6400 polygons, can be accomodated easily by one SP-256, leaving space for 4600 additional polygons. Larger models can be accomodated with additional SP-256's.

The SOLID 3000 rendering package can make use of the SP-256 to store large display lists. However, there are certain restrictions in its use because of the way the SOLID 3000's internal stack pointer is implemented. The stack pointer is a 24-bit address, but the high 8 bits remain unchanged during the execution of the program. Thus, the low order 16 bits can only address a 64K portion of the 256K. Also, it is desirable to implement SOLID's stack at the highest address in memory and build "downward" as values are pushed onto the stack, while display lists start at a low memory address and build "upward" so that the most efficient use of the memory can be made.

However, the standard distribution of SOLID 3000 has its stack pointer pointed at the highest address in the SR-8 memory (which is a required component of a system which is to run SOLID 3000). Thus, when an SP-256 is used in place of the SR-8 memory, (by setting its address to be the same as an SR-8) only one quarter of the memory (64K) can normally be used. This situation can be remedied by compiling the starting location of the stack as being the highest address of the SP-256 and letting display lists build upward from the lowest address in the SP-256. Using this recompilation would allow access to the entire SP-256 memory address space.

Using an SP-256 in the manner just described has a drawback; the SP-256 has a memory cycle time of 200 nsec which is half the speed of the SR-8 memory. So both the stack and display lists would be in this slower memory. While this may be acceptable, a partial solution to this problem would be to use both an SP-256 and an SR-8 in the system at the same time. The stack could be implemented in the SR-8, and the display lists would use the SP-256. The stack would then make use of the higher speed memory. The SP-256 would need to have its location in address space changed so that it did not overlap the SR-8's address space in this configuration. Display lists would not be sent to the lowest address in SR-8 memory (which is the default software configuration) but specifically sent to the lowest address in the SP-256 using the SOLID 3000 function DSLOAD instead of the function DSSEND which is the usual method used to send display lists to the ADAGE. Also, a DSTART is required to be used after DSLOADing the display list to the ADAGE 3000.

Additional software development in the host would be required to implement model animation in real time. This software would have to support requests by the user to perform model animation, specifications as to how the model is to be animated, requests to terminate the animation, and requests to re-scale the model based on the last viewing transformation used during model animation.

This last capability is needed to insure that the model is scaled correctly to completely fill the screen. During model animation, corners of the model may occassionally be clipped at the edges of the display. This could result because, in contrast to single image rendering, the model would not be rescaled each time it is transformed and rendered.

There is a possibility to eliminate the need for the last two requirements identified above by implementing support for interactive zooming. This approach would require an additional control in the interactive device, prescaling the model data stored in the ADAGE to allow both expansion and shrinking of the image, and altering the scaling factors of the transformations.

5.3 Temporary Geometry Format Files

Improving the time required to read temporary geometry format files from disk has been identified as a precursor to the implementation of local transformations in the ADAGE PDG. This is a function which uses a relatively large amount of time while doing image rendering. Whenever a user defines a part in the SMP editor or reads an existing parts file, SMP writes to disk a temporary, geometry-format file corresponding to the parts. This file is read from disk whenever the user displays a model or a part. Note that because of the way model animation is proposed, disk read time would not be a factor in the model animation update rate.

There are several approaches that can be investigated to improve the performance of interacting with the geometry data file. One approach is to create large arrays internal to the SMP program and storing in them the data normally stored in disk in the temporary geometry file. Since these arrays would make SMP larger than the physical memory in the VAX, the virtual memory feature of the operating system would be used to read the geometry file information into physical memory as it is needed.

Another approach is to use data structures for the geometry file which are different from the one currently used, e.g., hashed storage.

Yet another possibility is the placing of the geometry file data in an ADAGE SP-256 memory. This would require the implementation of specialized data structures.

5.4 Summary

In summary, the various technical issues dealt with in this section are attainable with various degrees of technical effort and hardware investment.

Model animation depends primarily on having enough memory to hold the entire model in the ADAGE display list memory area. The advisability of using local transformations in the ADAGE PDG depends on the relative merits of splitting the location of the existing SMP transformations compared to the gains that might be obtained in rendering speed. The various methods of speeding up the extensive disk I/O for temporary geometry files depend on the availability of enough physical memory in the VAX-11/785. Thus, the capabilities of SMP can be expanded to provide new features and even faster rendering speeds.

6.0 SUMMARY OF ACCOMPLISHMENTS

The Research Triangle Institute, working closely with SAB personnel, have accomplished the following work:

1) Converted the PRIME/PRIMOS version of SMP to a VAX/VMS version.

2) Validated the converted VAX/VMS version of SMP using assembly files furnished by SAB.

During the enhancement of the VAX/VMS version of SMP to support the ADAGE RDS-3000 display the following work was performed:

3) Added code to SMP to load the ADAGE color look-up table.

4) Added code to SMP to initialize the ADAGE frame buffer controller to 512 X 512 60 Hz, noninterlaced.

5) Added code to SMP to initialize the ADAGE crossbar switch to straight through configuration.

6) Added code to SMP to initialize the ADAGE crossbar switch to defeat overlay option.

7) Added code to SMP to initialize the ADAGE channel crossbar switch to full color.

8) Added code to SMP to initialize the ADAGE cursor registers with a default cursor, a tracking cross.

9) Modified RTI's copy of MOVIE BYU to accomodate the size of the parts created by the assembly file furnished by SAB in order to test SMP's MOVIE-format write capability.

10) Added code to SMP to initialize the ADAGE image memory read and write masks.

11) Added code to SMP to perform windowing and viewporting.

12) Added code to SMP to perform clipping to the screen viewing region.

13) Added code to SMP to perform drawing of dashed lines (with the ADAGE used as a frame buffer)

14) Modified code in SMP to remove the drawing of boxes around the 4 views display (as well as in other formats of displays).

15) Installed the Phase I version of SMP/ADAGE 3000 PDG in SAB's VAX11/785.

16) Added code to SMP to load the color map in the ADAGE with the values required by the SOLID 3000 system.

17) Added code to SMP to perform SOLID 3000 line drawing.

18) Added code to SMP to perform SOLID 3000 flat shading.

19) Designed and implemented a linear time algorithm to create normals at each corner of each polygon as required by SOLID 3000 for smooth shading.

20) Added code to SMP to perform SOLID 3000 Gouraud smooth shading.

21) Added code to SMP to perform SOLID 3000 Phong smooth shading.

22) Added code to SMP to perform SOLID 3000 hidden surface removal using the ADAGE internal Z-buffering.

23) Provided assistance to SAB personnel in their work of hosting the ADAGE 3000 in the VAX-11/785.

24) Added code to SMP to perform anti-aliasing.

25) Added code to SMP to perform monitor gamma correction.

26) Installed the Phase II version of SMP/ADAGE 3000 PDG (SOLID 3000) in SAB's VAX11/785.

27) Documented the above work.

7.0 SUMMARY OF CONCLUSIONS AND RECOMMENDATIONS

The research conducted under this project has led to the following conclusions:

1) ADAGE's SOLID 3000 rendering package was determined to be the best tool to interface SAB's SMP program to its ADAGE RDS-3000 display.

2) SOLID 3000's function SPOLY was determined to be the method of choice to enter SMP's data structures into SOLID 3000.

3) The ADAGE RDS 3000 is viable graphic display to use to support enhanced speed of rendering by the solid modeling program SMP. Reasonable improvements in rendering speed were obtained with the phase I configuration and very significant improvements in rendering speed were obtained with the phase II configuration of the display system software.

4) SMP is a more valuable research tool to the personnel of SAB now that the ADAGE RDS 3000 can be used as a display.

5) SMP/SOLID 3000 can provide enhanced types of rendering such as anti-aliasing, monitor gamma correction, and higher quality smooth shading of both the Gouraud and Phong type.

The following recommendations emerge from the work performed under this contract:

6) Exploit the capabilities provided by having a large model's display list stored in the ADAGE in an SP-256 memory card. This could provide animation of the models.

7) Investigate the possible speed gain which might be obtained by storing the data normally stored in the temporary geometry disk file in large arrays internal to the SMP program. The operating system's optimized disk paging I/O would then be used to read the geometry file information into memory as it is needed using the page faulting mechanism.

8) Explore the possibility of using other data structures (e.g. hashed storage, etc.) for storing the geometry file data.

9) Explore the possibility of holding all geometry file data in the ADAGE SP-256. Specialized data structures would be necessary to do this.

10) Acquire an interactive device capable of specifying X and Y (and possibly Z) information for interaction with SMP.

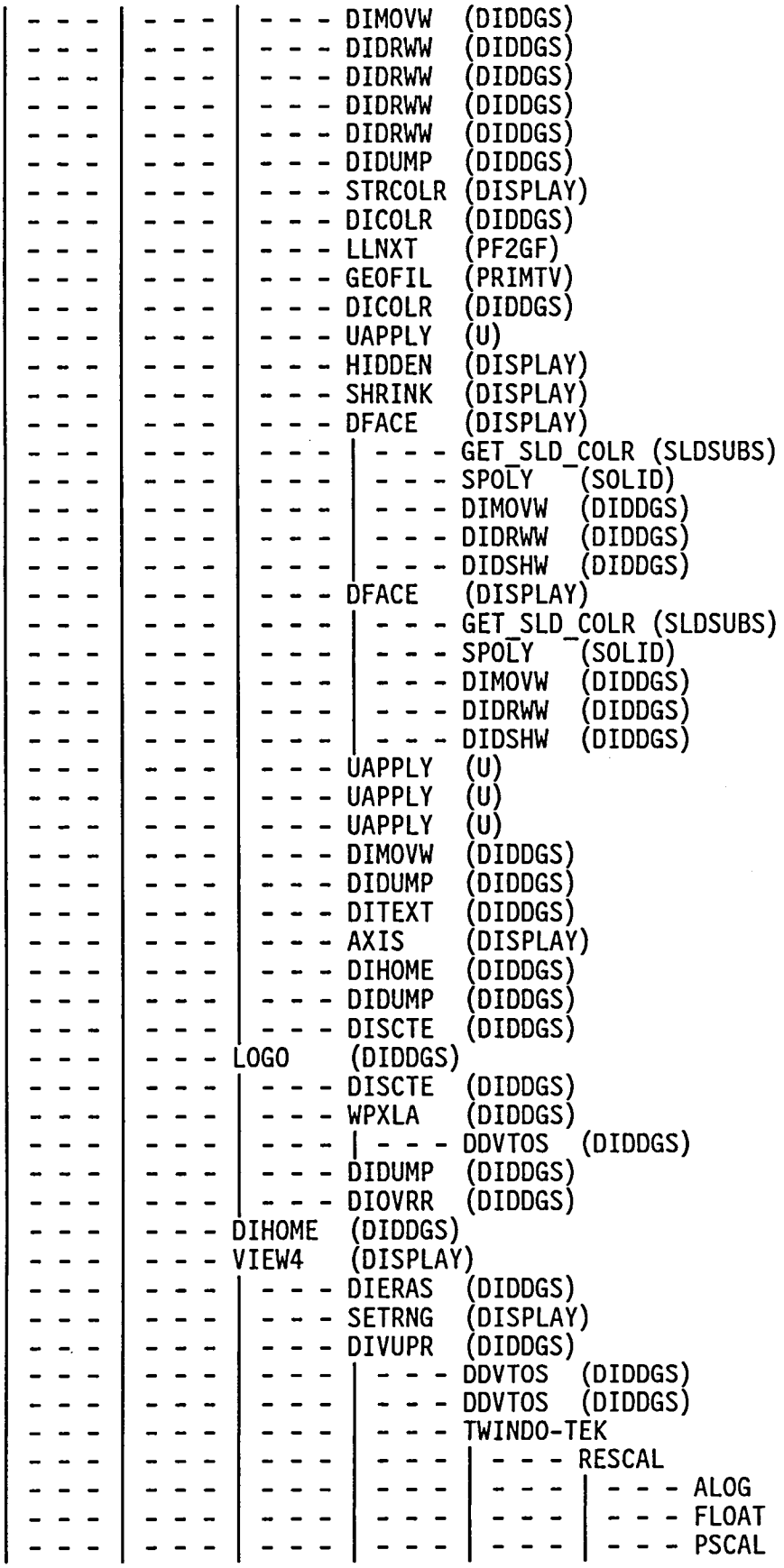
APPENDIX A

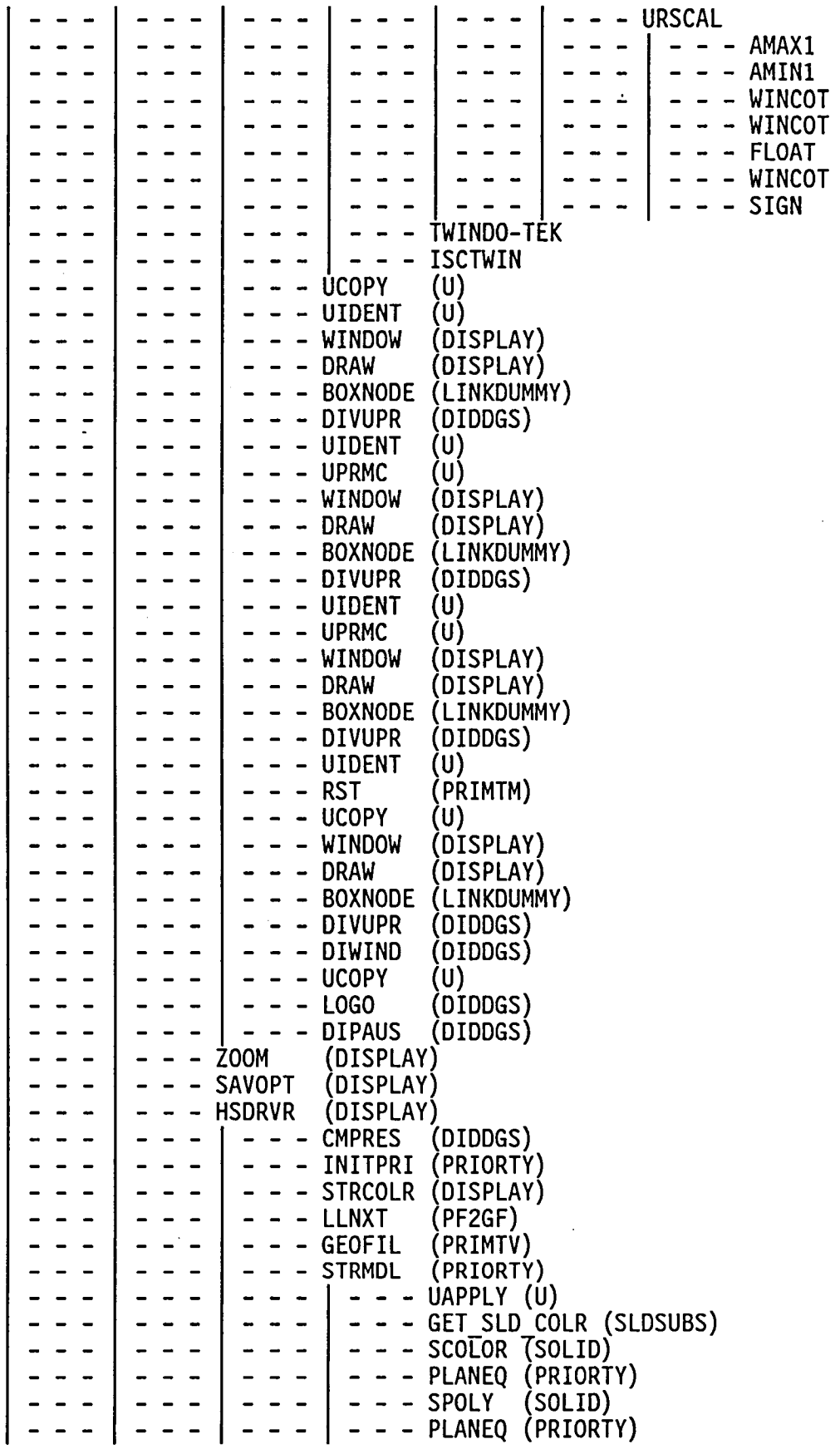
PARTIAL STATIC CALLING STRUCTURE FOR SMP/ADAGE 3000

FORMAT: | - - - | - - - ... Routine-Name (Filename)

The number of "| - - - " patterns indicates the depth of the calling structure.

```
SMP      (SMPMISC)
| - - - SINITT  (SMPMISC)
| - - - | - - - GDINIT  (DIDDGS)
| - - - | - - - TITLE   (DIDDGS)
| - - - | - - - GINITT  (SMPMISC)
| - - - OPENGEO (PRIMTV)
| - - - DISCTE  (DIDDGS)
| - - - DIERAS  (DIDDGS)
| - - - SMPDOC  (SMPMISC)
| - - - DIERAS  (DIDDGS)
| - - - SMPDOC  (SMPMISC)
| - - - DISPLAY (DISPLAY)
| - - - | - - - SAVOPT  (DISPLAY)
| - - - | - - - DIERAS  (DIDDGS)
| - - - | - - - DISCTE  (DIDDGS)
| - - - | - - - RSTOPT  (DISPLAY)
| - - - | - - - RSETTV  (DISPLAY)
| - - - | - - - SAVOPT  (DISPLAY)
| - - - | - - - DMENU   (DISPLAY)
| - - - | - - - | - - - DIDUMP  (DIDDGS).
| - - - | - - - | - - - SPP     (DISPLAY)
| - - - | - - - | - - - RST     (PRIMTV)
| - - - | - - - | - - - DIDUMP  (DIDDGS).
| - - - | - - - | - - - SETRNG  (DISPLAY)
| - - - | - - - | - - - WINDOW  (DISPLAY)
| - - - | - - - SAVOPT  (DISPLAY).
| - - - | - - - DIERAS  (DIDDGS).
| - - - | - - - DRAW    (DISPLAY)
| - - - | - - - | - - - DICOLR  (DIDDGS)
```


DIMOVW (DIDDGS)
DIDRWW (DIDDGS)
DIDRWW (DIDDGS)
DIDRWW (DIDDGS)
DIDRWW (DIDDGS)
DIDUMP (DIDDGS)
STRCOLR (DISPLAY)
DICOLR (DIDDGS)
LLNXT (PF2GF)
GEOFIL (PRIMTV)
DICOLR (DIDDGS)
UAPPLY (U)
HIDDEN (DISPLAY)
SHRINK (DISPLAY)
DFACE (DISPLAY)
GET_SLD_COLR (SLDSUBS)
SPOLY (SOLID)
DIMOVW (DIDDGS)
DIDRWW (DIDDGS)
DIDSHW (DIDDGS)
DFACE (DISPLAY)
GET_SLD_COLR (SLDSUBS)
SPOLY (SOLID)
DIMOVW (DIDDGS)
DIDRWW (DIDDGS)
DIDSHW (DIDDGS)
UAPPLY (U)
UAPPLY (U)
UAPPLY (U)
DIMOVW (DIDDGS)
DIDUMP (DIDDGS)
DITEXT (DIDDGS)
AXIS (DISPLAY)
DIHOME (DIDDGS)
DIDUMP (DIDDGS)
DISCTE (DIDDGS)
LOGO (DIDDGS)
DISCTE (DIDDGS)
WPXLA (DIDDGS)
DDVTOS (DIDDGS)
DIDUMP (DIDDGS)
DIOVRR (DIDDGS)
DIHOME (DIDDGS)
VIEW4 (DISPLAY)
DIERAS (DIDDGS)
SETRNG (DISPLAY)
DIVUPR (DIDDGS)
DDVTOS (DIDDGS)
DDVTOS (DIDDGS)
TWINDO-TEK
RESCAL
ALOG
FLOAT
PSCAL


URSCAL
AMAX1
AMIN1
WINCOT
WINCOT
FLOAT
WINCOT
SIGN
TWINDO-TEK
ISCTWIN
UCOPY (U)
UIDENT (U)
WINDOW (DISPLAY)
DRAW (DISPLAY)
BOXNODE (LINKDUMMY)
DIVUPR (DIDDGS)
UIDENT (U)
UPRMC (U)
WINDOW (DISPLAY)
DRAW (DISPLAY)
BOXNODE (LINKDUMMY)
DIVUPR (DIDDGS)
UIDENT (U)
UPRMC (U)
WINDOW (DISPLAY)
DRAW (DISPLAY)
BOXNODE (LINKDUMMY)
DIVUPR (DIDDGS)
UIDENT (U)
RST (PRIMTM)
UCOPY (U)
WINDOW (DISPLAY)
DRAW (DISPLAY)
BOXNODE (LINKDUMMY)
DIVUPR (DIDDGS)
DIWIND (DIDDGS)
UCOPY (U)
LOGO (DIDDGS)
DIPAUS (DIDDGS)
ZOOM (DISPLAY)
SAVOPT (DISPLAY)
HSDRVR (DISPLAY)
CMPRES (DIDDGS)
INITPRI (PRIORTY)
STRCOLR (DISPLAY)
LLNXT (PF2GF)
GEOFIL (PRIMTV)
STRMDL (PRIORTY)
UAPPLY (U)
GET_SLD_COLR (SLDSUBS)
SCOLOR (SOLID)
PLANEQ (PRIORTY)
SPOLY (SOLID)
PLANEQ (PRIORTY)

```
| - - - | - - - | - - - HIDSURF (PRIORTY)
| - - - | - - - | - - - | - - - PRIORTY (PRIORTY)
| - - - | - - - | - - - | - - - SPECIAL (PRIORTY)
| - - - | - - - | - - - | - - - DRAWHS  (SCANLINE)
| - - - | - - - | - - - | - - - | - - - DDSTOV  (DIDDGS)
| - - - | - - - | - - - | - - - | - - - DDSTOV  (DIDDGS)
| - - - | - - - | - - - | - - - | - - - DIVUPR  (DIDDGS)
| - - - | - - - | - - - | - - - | - - - DIWIND  (DIDDGS)
| - - - | - - - | - - - | - - - | - - - DICTAB  (DIDDGS)
| - - - | - - - | - - - | - - - | - - - DICTAB  (DIDDGS)
| - - - | - - - | - - - | - - - | - - - SHADER  (SCANLINE)
| - - - | - - - | - - - | - - - | - - - | - - - CENTER (SCANLINE)
| - - - | - - - | - - - | - - - | - - - | - - - PLANEQ (PRIORTY)
| - - - | - - - | - - - | - - - | - - - INTENS  (SCANLINE)
| - - - | - - - | - - - | - - - | - - - | - - - CHKNOD (SCANLINE)
| - - - | - - - | - - - | - - - | - - - | - - - NORAV  (SCANLINE)
| - - - | - - - | - - - | - - - | - - - | - - - NORAV  (SCANLINE)
| - - - | - - - | - - - | - - - | - - - | - - - | - - - PLANEQ (PRIORTY)
| - - - | - - - | - - - | - - - | - - - OPEN    ('SMGG.IMG')
| - - - | - - - | - - - | - - - | - - - VTOS    (SCANLINE)
| - - - | - - - | - - - | - - - | - - - INITAET (SCANLINE)
| - - - | - - - | - - - | - - - | - - - INITSET (SCANLINE)
| - - - | - - - | - - - | - - - | - - - SCNFIL  (SCANLINE)
| - - - | - - - | - - - | - - - | - - - | - - - SCNINT  (SCANLINE)
| - - - | - - - | - - - | - - - | - - - | - - - | - - - EMAXY   (SCANLINE)
| - - - | - - - | - - - | - - - | - - - | - - - | - - - EMAXY   (SCANLINE)
| - - - | - - - | - - - | - - - | - - - | - - - SETSET  (SCANLINE)
| - - - | - - - | - - - | - - - | - - - | - - - SMOOTH  (SCANLINE)
| - - - | - - - | - - - | - - - | - - - DSPLASL (SCANLINE)
| - - - | - - - | - - - | - - - | - - - | - - - DIDUMP  (DIDDGS)
| - - - | - - - | - - - | - - - | - - - | - - - SL2D    (SCANLINE)
| - - - | - - - | - - - | - - - | - - - | - - - DDSTOV  (DIDDGS)
| - - - | - - - | - - - | - - - | - - - | - - - WPXLA   (DIDDGS)
| - - - | - - - | - - - | - - - | - - - | - - - WPXLA   (DIDDGS)
| - - - | - - - | - - - | - - - | - - - SETAET  (SCANLINE)
| - - - | - - - | - - - | - - - | - - - DSPLASL (SCANLINE)
| - - - | - - - | - - - | - - - | - - - | - - - DIDUMP  (DIDDGS)
| - - - | - - - | - - - | - - - | - - - | - - - SL2D    (SCANLINE)
| - - - | - - - | - - - | - - - | - - - | - - - DDSTOV  (DIDDGS)
| - - - | - - - | - - - | - - - | - - - | - - - WPXLA   (DIDDGS)
| - - - | - - - | - - - | - - - | - - - | - - - WPXLA   (DIDDGS)
| - - - | - - - | - - - | - - - | - - - LOGO    (DIDDGS)
| - - - | - - - | - - - | - - - | - - - DIPAUSE (DIDDGS)
| - - - | - - - | - - - | - - - | - - - RSTOPT  (DISPLAY)
| - - - | - - - | - - - | - - - | - - - DIOVRR  (DIDDGS)
| - - - | - - - | - - - | - - - | - - - DICTAB  (DIDDGS)
| - - - | - - - | - - - | - - - | - - - DICOLR  (DIDDGS)
| - - - | - - - | - - - DIHOME  (DIDDGS)
| - - - | - - - | - - - DIDUMP  (DIDDGS)
| - - - | - - - GRAPHIC (GRAPHICS)
| - - - | - - - SMPDOC  (SMPMISC)
| - - - | - - - DIERAS  (DIDDGS)
| - - - | - - - SMPDOC  (SMPMISC)
| - - - | - - - OPENGEO (PRIMTV)
```

```
| - - - | - - - EXIT    (FORTRAN EXIT)
| - - - AEDIT   (ALPHA)
| - - - | - - - DISCTE  (DIDDGS)
| - - - | - - - DIERAS  (DIDDGS)
| - - - | - - - EDADD   (ALPHA)
| - - - | - - - EDCHNG  (ALPHA)
| - - - | - - - | - - - EDDRAW  (ALPHA)
| - - - | - - - EDDEL   (ALPHA)
| - - - | - - - DIERAS  (DIDDGS)
| - - - | - - - EDPRNT  (ALPHA)
| - - - | - - - DIPAUS  (DIDDGS)
| - - - | - - - SMPDOC  (SMPMISC)
| - - - | - - - DIERAS  (DIDDGS)
| - - - | - - - SMPDOC  (SMPMISC)
| - - - | - - - SMPDOC  (SMPMISC)
| - - - | - - - SMPDOC  (SMPMISC)
| - - - | - - - SMPDOC  (SMPMISC)
| - - - | - - - SMPDOC  (SMPMISC)
| - - - | - - - SMPDOC  (SMPMISC)
| - - - | - - - SMPDOC  (SMPMISC)
| - - - | - - - SMPDOC  (SMPMISC)
| - - - | - - - EDCOPY  (ALPHA)
| - - - | - - - OPENGEO (PRIMTV)
| - - - | - - - EXIT    (FORTRAN EXIT)
| - - - | - - - EDREST  (ALPHA)
| - - - READGM  (SMPMISC)
| - - - MASPROP (MASPROP)
| - - - OPENGEO (PRIMTV)
| - - - EXIT    (FORTRAN EXIT)
| - - - SAVEGM  (SMPMISC)
| - - - OPENGEO (PRIMTV)
| - - - EXIT    (FORTRAN EXIT)
```

APPENDIX B

LIST OF SMP'S MODIFIED SUBROUTINES

```
SUBROUTINE AEDIT
SUBROUTINE ANGCALC (A,B,C,ISTAT)
SUBROUTINE AXIS
SUBROUTINE BREAKIT (IPNTR)
SUBROUTINE CHNGALL (ID,IOPER,NCOMPNU,PCOMPNU)
SUBROUTINE CLEANN
SUBROUTINE CMPRES (IDISK,ICHANG,IMIN,IMAX)
SUBROUTINE COLMOV(IASSEM1)
SUBROUTINE DDVTOS (XV,YV,IXS,IYS)
SUBROUTINE DDWTOS (XW,YW,IXS,IYS)
SUBROUTINE DEGREES(X,Y,XC,YC,THETA)
SUBROUTINE DFACE(N,X,IDASH)
SUBROUTINE DICHRSZ (NUM)
SUBROUTINE DICOLR (NUM)
SUBROUTINE DICTAB (IOPT)
SUBROUTINE DICURS (ICH,XW,YW)
SUBROUTINE DIDRWS (XV,YV)
SUBROUTINE DIDRWW (XW,YW)
SUBROUTINE DIDSHS (XV,YV,IDASH)
SUBROUTINE DIDSHW (XW,YW,IDASH)
SUBROUTINE DIDUMP
SUBROUTINE DIERAS
SUBROUTINE DIMOVS (XV,YV)
SUBROUTINE DIMOVW (XW,YW)
SUBROUTINE DIPAUS
SUBROUTINE DISCTE (NCTE,IR,IG,IB)
SUBROUTINE DISPLAY
SUBROUTINE DISTCK(XPT,YPT,XNEW,YNEW,IVW,INRNG)
SUBROUTINE DITEXT (LENS,CHARS)
SUBROUTINE DIVUPR (XMINV,XMAXV,YMINV,YMAXV)
SUBROUTINE DIWIND (XMINW,XMAXW,YMINW,YMAXW)
SUBROUTINE DMENU
SUBROUTINE DRAW
SUBROUTINE DRAWHS
SUBROUTINE EDADD
SUBROUTINE EDCHNG (ID,IOPT)
```

```
SUBROUTINE EDCOPY (ID,IOPT)
SUBROUTINE EDDEL (ID,IOPT)
SUBROUTINE EDDRAW (IPART)
SUBROUTINE EDPRNT(FIRST,LAST)
SUBROUTINE EDREST (ID,IOPT)
SUBROUTINE EXTRN (NPART,TP,INR)
SUBROUTINE GDINIT (IGD,IBAUD)
SUBROUTINE GEOFIL (OPTION,NPART)
SUBROUTINE GRAPHIC
SUBROUTINE HEADR(NAME,NUMHD,STATHD,TRANSHD)
SUBROUTINE HSDRVR
SUBROUTINE LOGO
SUBROUTINE MASPROP
SUBROUTINE OPENGEO(IOPT)
SUBROUTINE OPENW(DESC,ICODE)
SUBROUTINE PCFFIO (IVW,ILINES)
SUBROUTINE PRINIT(IIBAUD)
SUBROUTINE RDGEO(IREAD)
SUBROUTINE RDMOV (IREAD)
SUBROUTINE RDRNEW(IREAD)
SUBROUTINE READER(IREAD)
SUBROUTINE RSETTV
SUBROUTINE RST(VECTOR,ARRAY,NPART)
SUBROUTINE SAVGEO
SUBROUTINE SAVMODL
SUBROUTINE SINITT
SUBROUTINE SMPDOC (INDEX,ICODE)
SUBROUTINE STRMDL (NMAX,NN,X,N,IF,T,IPC,INERR)
SUBROUTINE TITLE
SUBROUTINE USETAU (NAME, VAL)
SUBROUTINE VIEW4(IFLAG)
SUBROUTINE WINDOW
SUBROUTINE WPXLA (XV,YV,MAXR,NR,NC,IPIXEL)
SUBROUTINE ZOOM
```

APPENDIX C

LISTINGS OF SOFTWARE MODULES DEVELOPED FOR SMP/ADAGE INTEGRATION

```
     C       FILENAME:        ADSUBS.FOR
C
C*********************************************************************
        SUBROUTINE ADDRWA(XW, YW, IDASH     )
C
C       DRAW A LINE USING FLOATING WORLD COORDS.  THE OTHER POINT TO
C       DRAW FROM IS (XCUR, YCUR) IN COMMON /ADAGE/.
C
        IMPLICIT NONE
C
        COMMON /ADVUPR/ IXMINS, IXMAXS, IYMINS,  IYMAXS,
     .                  IXVSIZ, IYVSIZ, IXVCNTR, IYVCNTR
        INTEGER         IXMINS, IXMAXS, IYMINS,  IYMAXS,
     .                  IXVSIZ, IYVSIZ, IXVCNTR, IYVCNTR
        COMMON /ADWIND/ XMINW,  XMAXW,  YMINW,   YMAXW,
     .                  XWSIZE, YWSIZE, XWCNTR,  YWCNTR
        REAL            XMINW,  XMAXW,  YMINW,   YMAXW,
     .                  XWSIZE, YWSIZE, XWCNTR,  YWCNTR
C
        INTEGER*2 IX, IY
        INTEGER*4 IDASH
        REAL      XW, YW
C
C       PERFORM WINDOWING & VIEWPORTING:
        IX = (((XW-XWCNTR) / XWSIZE) + 0.5 ) * IXVSIZ + IXMINS
        IY = (((YW-YWCNTR) / YWSIZE) + 0.5 ) * IYVSIZ + IYMINS
        C
        CALL CLIP2D(IX, IY, 0, 511, 0, 511, IDASH     )
        RETURN
        END
```

```
C*************************************************************************
      SUBROUTINE ADDRWS(IXS, IYS, IDASH     )
C
C     DRAW A LINE USING INTEGER SEREEN COORDS.  THE OTHER POINT TO
C     DRAW FROM IS (XCUR, YCUR) IN COMMON /ADAGE/.
C
      IMPLICIT NONE
C
      COMMON /ADVUPR/ IXMINS, IXMAXS, IYMINS,  IYMAXS,
     .                IXVSIZ, IYVSIZ, IXVCNTR, IYVCNTR
      INTEGER         IXMINS, IXMAXS, IYMINS,  IYMAXS,
     .                IXVSIZ, IYVSIZ, IXVCNTR, IYVCNTR
C
      INTEGER*2 IXS, IYS, IX, IY
      INTEGER*4 IDASH
C
C     PERFORM WINDOWING & VIEWPORTING:
C     IX = (IXS * IXVSIZ / 512.0) + IXVCNTR
C     IY = (IYS * IYVSIZ / 512.0) + IYVCNTR
C
      CALL CLIP2D(IXS, IYS, 0, 511, 0, 511, IDASH     )
      RETURN
      END
```

```
C*********************************************************************
      SUBROUTINE ADLINE(X1, Y1, IDASH     )
C
C     ROUTINE TO DRAW A LINE ON THE ADAGE 3000 IN SCREEN COORDINATES
C     THIS ROUTINE USES SINGLE PIXEL WRITES & A BRESSENHAM-LIKE ALGORITHM
C
      IMPLICIT NONE
C
      COMMON /ADAGE/ CURCOLR, ICURX, ICURY
      INTEGER*4      CURCOLR
      INTEGER*2                ICURX, ICURY
C
      INTEGER*2 X1, Y1
      INTEGER*2 DELX,DELY,XCHNG,YCHNG,I,XA,YA,XB,YB,ISTOP
      REAL*4 REM, SLOPE
C
C     IDASH = 0 ---> DO NOT DRAW DASHED LINES
C             1 ---> DRAW DASHED LINES (4 PIX ON; 4 OFF)
      INTEGER*4 IDASH
C
      INTEGER*2 IABS, TEST, MOD
      LOGICAL*4 DRAW
      REAL      FLOAT
C
      DRAW = .TRUE.
      XA=ICURX
      YA=ICURY
      XB=X1
      YB=Y1
C
      DELX = XB - XA
      DELY = YB - YA
      REM = 0.5
C
      XCHNG = 1
      IF (DELX .LT. 0) XCHNG = -1
      YCHNG = 1
      IF (DELY .LT. 0) YCHNG = -1
C
      IF (IABS(DELX) .GT. IABS(DELY) ) THEN
         IF (DELX .EQ. 0) THEN
            SLOPE = 100000.0
         ELSE
            SLOPE = (FLOAT(IABS(DELY))) / (FLOAT(IABS(DELX)))
         END IF
         ISTOP = IABS(DELX)
C
C        IF DASHED LINE SET WHETHER TO DRAW LINE OR BLANK:
         IF (IDASH .EQ. 1) THEN
            TEST = MOD(XA, 8)
            IF (TEST .GE. 0   .AND.   TEST .LE. 3) THEN
               DRAW = .TRUE.
            ELSE
               DRAW = .FALSE.
```

```
            END IF
         END IF
         IF(DRAW)   CALL IKPWR(0, XA, 511-YA, 1, CURCOLR)
C
         DO 20 I = 1, ISTOP
            REM = REM + SLOPE
            IF (REM .GE. 1.0) THEN
               YA = YA + YCHNG
               REM = REM - 1.0
            ENDIF
            XA = XA + XCHNG
C
C           IF DASHED LINE SET WHETHER TO DRAW LINE OR BLANK:
            IF (IDASH .EQ. 1) THEN
               TEST = MOD(XA, 8)
               IF (TEST .GE. 0   .AND.   TEST .LE. 3) THEN
                  DRAW = .TRUE.
               ELSE
                  DRAW = .FALSE.
               END IF
            END IF
            IF (DRAW)   CALL IKPWR(0, XA, 511-YA, 1, CURCOLR)
 20      CONTINUE
      ELSE
         IF (DELY .EQ. 0) THEN
            SLOPE = 100000.0
         ELSE
            SLOPE = (FLOAT(IABS(DELX))) / (FLOAT(IABS(DELY)))
         END IF
         ISTOP = IABS(DELY)
C
C        IF DASHED LINE SET WHETHER TO DRAW LINE OF BLANK:
         IF (IDASH .EQ. 1) THEN
            TEST = MOD(YA, 8)
            IF (TEST .GE. 0   .AND.   TEST .LE. 3) THEN
               DRAW = .TRUE.
            ELSE
               DRAW = .FALSE.
            END IF
         END IF
         IF (DRAW)   CALL IKPWR(0, XA, 511-YA, 1, CURCOLR)
C
         DO 50 I = 1, ISTOP
            REM = REM + SLOPE
            IF (REM .GT. 1.0) THEN
               XA = XA + XCHNG
               REM = REM - 1.0
            ENDIF
            YA = YA + YCHNG
C
C           IF DASHED LINE SET WHETHER TO DRAW LINE OF BLANK:
            IF (IDASH .EQ. 1) THEN
               TEST = MOD(YA, 8)
               IF (TEST .GE. 0   .AND.   TEST .LE. 3) THEN
```

```
                  DRAW = .TRUE.
               ELSE
                  DRAW = .FALSE.
               END IF
            END IF
            IF (DRAW)   CALL IKPWR(0, XA, 511-YA, 1, CURCOLR)
 50       CONTINUE
       ENDIF
C
C      SET DRAW TO POINT AS THE CURRENT POINT:
       ICURX = X1
       ICURY = Y1
       RETURN
       END
```

```
C*************************************************************************
      SUBROUTINE ADMOVA(XW, YW     )
C
C     MOVE USING FLOATING WORLD COORDS.
C
      IMPLICIT NONE
C
      COMMON /ADAGE/ CURCOLR, ICURX, ICURY
      INTEGER*4      CURCOLR
      INTEGER*2                ICURX, ICURY
C
      COMMON /ADVUPR/ IXMINS, IXMAXS, IYMINS,  IYMAXS,
     .                IXVSIZ, IYVSIZ, IXVCNTR, IYVCNTR
      INTEGER         IXMINS, IXMAXS, IYMINS,  IYMAXS,
     .                IXVSIZ, IYVSIZ, IXVCNTR, IYVCNTR
      COMMON /ADWIND/ XMINW,  XMAXW,  YMINW,   YMAXW,
     .                XWSIZE, YWSIZE, XWCNTR,  YWCNTR
      REAL            XMINW,  XMAXW,  YMINW,   YMAXW,
     .                XWSIZE, YWSIZE, XWCNTR,  YWCNTR
C
      INTEGER*2 IX, IY
      REAL      XW, YW
C
C     PERFORM WINDOWING & VIEWPORTING:
      IX = (((XW-XWCNTR) / XWSIZE) + 0.5 ) * IXVSIZ + IXMINS
      IY = (((YW-YWCNTR) / YWSIZE) + 0.5 ) * IYVSIZ + IYMINS
C
C     SET THE PASSED IN POINT AS THE CURRENT POINT:
      ICURX = IX
      ICURY = IY
      RETURN
      END
```

```
C*************************************************************************
      SUBROUTINE ADMOVS(IXS, IYS     )
C
C     MOVE USING INTEGER SEREEN COORDS.
C
      IMPLICIT NONE
C
      COMMON /ADAGE/ CURCOLR, ICURX, ICURY
      INTEGER*4      CURCOLR
      INTEGER*2                ICURX, ICURY
C
      COMMON /ADVUPR/ IXMINS, IXMAXS, IYMINS,  IYMAXS,
     .                IXVSIZ, IYVSIZ, IXVCNTR, IYVCNTR
      INTEGER         IXMINS, IXMAXS, IYMINS,  IYMAXS,
     .                IXVSIZ, IYVSIZ, IXVCNTR, IYVCNTR
C
      INTEGER*2 IXS, IYS, IX, IY
C
C     PERFORM WINDOWING & VIEWPORTING:
C     IX = (IXS * IXVSIZ / 512.0) + IXVCNTR
C     IY = (IYS * IYVSIZ / 512.0) + IYVCNTR
C
      ICURX = IXS
      ICURY = IYS
      RETURN
      END
```

```
C*************************************************************************
      SUBROUTINE ADVUPR(IXMIN, IXMAX, IYMIN, IYMAX     )
C
C     SAVE VIEWPORT SPECS IN COMMON IN SCREEN COORDINATES:
C
C
      IMPLICIT NONE
C
      COMMON /ADVUPR/ IXMINS, IXMAXS, IYMINS,  IYMAXS,
     .                IXVSIZ, IYVSIZ, IXVCNTR, IYVCNTR
      INTEGER         IXMINS, IXMAXS, IYMINS,  IYMAXS,
     .                IXVSIZ, IYVSIZ, IXVCNTR, IYVCNTR
C
      INTEGER  IXMIN, IXMAX, IYMIN, IYMAX
C
      IXMINS = IXMIN
      IXMAXS = IXMAX
      IYMINS = IYMIN
      IYMAXS = IYMAX
      IXVSIZ = IXMAX - IXMIN
      IYVSIZ = IYMAX - IYMIN
      IXVCNTR = (IXMAX + IXMIN) / 2
      IYVCNTR = (IYMAX + IYMIN) / 2
      RETURN
      END
```

```
C*********************************************************************
      SUBROUTINE ADWIND(XMIN, XMAX, YMIN, YMAX     )
C
C     SAVE WINDOW SPECS IN COMMON IN WORLD COORDINATES:
C
      COMMON /ADWIND/ XMINW,  XMAXW,  YMINW,   YMAXW,
     .                XWSIZE, YWSIZE, XWCNTR,  YWCNTR
      REAL            XMINW,  XMAXW,  YMINW,   YMAXW,
     .                XWSIZE, YWSIZE, XWCNTR,  YWCNTR
C
      REAL    XMIN, XMAX, YMIN, YMAX
C
      XMINW = XMIN
      XMAXW = XMAX
      YMINW = YMIN
      YMAXW = YMAX
      XWSIZE = XMAXW - XMINW
      YWSIZE = YMAXW - YMINW
      XWCNTR = (XMINW + XMAXW) / 2.0
      YWCNTR = (YMINW + YMAXW) / 2.0
      RETURN
      END
```

```
      SUBROUTINE CLIP2D(P2X, P2Y, XMIN, XMAX, YMIN, YMAX, IDASH     )
C
C     SUTHERLAND-COHEN 2D LINE CLIPPER (AFTER ROGERS 1985)
C
      IMPLICIT NONE
C
      COMMON /ADAGE/ CURCOLR, ICURX, ICURY
      INTEGER*4      CURCOLR
      INTEGER*2                ICURX, ICURY
C
      COMMON /CLIP_ALIGN/ CLIPPING_BOX(4)
      INTEGER*2 CLIPPING_BOX
C
      INTEGER*4
     . IDASH        ! DASHED LINE FLAG (1=DASHED, 0=SOLID)
C
      INTEGER*2
     . DELTA_X,     ! P2X - P1X
     . DELTA_Y,     ! P2Y - P1Y
     . I,           ! DO LOOP INDEX 1-4
     . INTERSECT,! RESULT OF ANDING P1 & P2 CLIP CODES
     . INTERSECT_X, ! X-VALUE OF INTERSECTION WITH BOT OR TOP CLIPPING BOX
     . INTERSECT_Y, ! Y-VALUE OF INTERSECTION WITH L   OR R   CLIPPING BOX
     . NO /0/,      ! LINE NOT VISIBLE
     . P1CODES(4),  ! P1'S CLIPPING CODES
     . P2CODES(4),  ! P2'S CLIPPING CODES
     . P1X,         ! X & Y COORDINATES OF BEGINNING OF LINE
     . P1Y,         !
     . P2X,         ! X & Y COORDINATES OF END OF LINE
     . P2Y,         !
     . PARTIAL /-1/,! PARTIALLY VISIBLE LINE
     . SUM1,        ! SUM OF P1'S CLIP CODES
     . SUM2,        ! SUM OF P2'S CLIP CODES
     . TEMP,        ! ALLOWS SWAPPING OF CLIPPING CODES
     . TEMPX,       ! ALLOWS SWAPPING OF P1 & P2
     . TEMPY,       ! ALLOWS SWAPPING OF P1 & P2
     . VISIBLE,     ! LINE VISIBILITY (YES, NO, OR, PARTIAL)
     . XMIN,        ! LEFT   EDGE OF CLIPPING WINDOW
     . XMAX,        ! RIGHT  EDGE OF CLIPPING WINDOW
     . YMIN,        ! BOTTOM EDGE OF CLIPPING WINDOW
     . YMAX,        ! TOP    EDGE OF CLIPPING WINDOW
     . YES /1/      ! LINE COMPLETELY VISIBLE
C
      LOGICAL*4
     . HORIZONTAL,  ! .TRUE. IF HORIZONTAL LINE
     . VERTICAL     ! .TRUE. IF VERTICAL LINE
C
      REAL*4
     . SLOPE        ! DELTA_X / DELTA_Y
C
```

```
C*********************************************************************
C
C       INIT:
C
C       GET P1 FROM COMMON /ADAGE/
        P1X = ICURX
        P1Y = ICURY
C
C       DETECT VERTICAL LINE CASE:
        VERTICAL = .FALSE.
        DELTA_X = P2X - P1X
        IF (DELTA_X .EQ. 0) THEN
           VERTICAL = .TRUE.
        ELSE
           SLOPE = (P2Y - P1Y) / FLOAT(DELTA_X)
        END IF
C
C       DETECT HORIZONTAL LINE CASE:
        HORIZONTAL = .FALSE.
        DELTA_Y = P2Y - P1Y
        IF (DELTA_Y .EQ. 0)   HORIZONTAL = .TRUE.
C
C       LOAD ARRAY WITH CLIPPING WIMDOW SPECIFICATIONS:
        CLIPPING_BOX(1) = XMIN
        CLIPPING_BOX(2) = XMAX
        CLIPPING_BOX(3) = YMIN
        CLIPPING_BOX(4) = YMAX
```

```
C
C*********************************************************************
C
C       MAIN:
C
C       FOR EACH CLIP BOX EDGE, CALC INTERSECTIONS WITH LINE:
        CALL ENDPT_CODE(P1X, P1Y, CLIPPING_BOX, P1CODES, SUM1 )
        CALL ENDPT_CODE(P2X, P2Y, CLIPPING_BOX, P2CODES, SUM2 )
C
C       DO WHILE ONE OR MORE LINE ENDPOINTS OUTSIDE WINDOW:
        DO WHILE (SUM1 .NE. 0   .OR.   SUM2 .NE. 0)
C
C          TRIVIALLY REJECT LINES WITH BOTH POINTS ON
C          THE SAME SIDE OF A CLIPPING BOX SIDE:
           CALL LOGIC_INTERSECT(P1CODES, P2CODES,    INTERSECT)
           IF (INTERSECT .NE. 0)   RETURN
C
C          INSURE P1 IS OUTSIDE THE CLIPPING BOX:
           IF (SUM1 .EQ. 0) THEN
C
C             SWAP END POINTS & CLIP CODES:
              TEMPX = P1X
              TEMPY = P1Y
              P1X = P2X
              P1Y = P2Y
              P2X = TEMPX
              P2Y = TEMPY
C
              DO 33 I = 1, 4
                 TEMP = P1CODES(I)
                 P1CODES(I) = P2CODES(I)
                 P2CODES(I) = TEMP
 33           CONTINUE
           END IF
C
C          TEST P1 AGAINST THE 4 CLIPPING BOX EDGES:
           IF (P1CODES(1) .NE. 0 .AND. .NOT. VERTICAL)THEN
C
C             P1 TO LEFT OF WINDOW & LINE NOT VERTICAL
              P1Y = SLOPE * (CLIPPING_BOX(1) - P1X) + P1Y
              P1X = CLIPPING_BOX(1)
           ELSE IF (P1CODES(2) .NE. 0 .AND. .NOT. VERTICAL)THEN
C
C             P1 TO RIGHT OF WINDOW & LINE NOT VERTICAL
              P1Y = SLOPE * (CLIPPING_BOX(2) - P1X) + P1Y
              P1X = CLIPPING_BOX(2)
           ELSEIF(P1CODES(3) .NE. 0 .AND. .NOT. HORIZONTAL)THEN
C
C             P1 BELOW WINDOW & LINE NOT HORIZONTAL
              IF (VERTICAL) THEN
                 P1Y = CLIPPING_BOX(3)
              ELSE
                 P1X = (1/SLOPE)*(CLIPPING_BOX(3)-P1Y)+P1X
                 P1Y = CLIPPING_BOX(3)
```

```
            END IF
         ELSEIF(P1CODES(4) .NE. 0 .AND. .NOT. HORIZONTAL)THEN
C
C           P1 ABOVE WINDOW & LINE NOT HORIZONTAL
            IF (VERTICAL) THEN
               P1Y = CLIPPING_BOX(4)
            ELSE
               P1X = (1/SLOPE)*(CLIPPING_BOX(4)-P1Y)+P1X
               P1Y = CLIPPING_BOX(4)
            END IF
         END IF
         CALL ENDPT_CODE(P1X, P1Y, CLIPPING_BOX, P1CODES, SUM1 )
         CALL ENDPT_CODE(P2X, P2Y, CLIPPING_BOX, P2CODES, SUM2 )
      END DO
C
C     DRAW LINE:
      CALL ADMOVS(P1X, P1Y     )
      CALL ADLINE(P2X, P2Y, IDASH     )
      RETURN
      END
```

```
      SUBROUTINE ENDPT_CODE (PX, PY, CLIPPING_BOX,     PCODES, SUM)
C
C     ASSIGNS END POINT CODES FOR A SINGLE POINT:
C
      IMPLICIT NONE
C
      INTEGER*2
     . CLIPPING_BOX(4),    ! LEFT, RIGHT, BOTTOM & TOP OF CLIPPING WINDOW
     . I,                  ! DO LOOP INDEX 1-4
     . PCODES(4),          ! 1-4 ARE LEFT, RIGHT, BOTTOM & TOP ENDPOINT
     .                     !    CLIP CODES (0 OR 1):
     .                     !    PCODES(1) = 1 IF POINT IS LEFT OF WINDOW
     .                     !      "    (2) = 1 IF POINT IS RIGHT OF WINDOW
     .                     !      "    (3) = 1 IF POINT IS BELOW WINDOW
     .                     !      "    (4) = 1 IF POINT IS ABOVE WINDOW
     .                     !    PCODES(N) = 0 OTHERWISE
     . PX,                 ! X COMPONENT OF POINT P
     . PY,                 ! Y COMPONENT OF POINT P
     . SUM                 ! SUM OF CLIPPING CODES
C
C*********************************************************************
C
C     MAIN:
C
C     CALCULATE THE FOUR ENDPOINT CODES FOR THE POINT P:
      IF (PX .LT. CLIPPING_BOX(1) ) THEN
         PCODES(1) = 1
      ELSE
         PCODES(1) = 0
      END IF
C
      IF (PX .GT. CLIPPING_BOX(2) ) THEN
         PCODES(2) = 1
      ELSE
         PCODES(2) = 0
      END IF
C
      IF (PY .LT. CLIPPING_BOX(3) ) THEN
         PCODES(3) = 1
      ELSE
         PCODES(3) = 0
      END IF
C
      IF (PY .GT. CLIPPING_BOX(4) ) THEN
         PCODES(4) = 1
      ELSE
         PCODES(4) = 0
      END IF
C
C     FORM SUM OF CODES:
      SUM = 0
      DO 100 I = 1, 4
 100  SUM = SUM + PCODES(I)
      RETURN
```

END

```
      SUBROUTINE LOGIC_INTERSECT(P1CODES, P2CODES,  INTERSECT)
C
C     LOGICALLY ANDS THE CLIPPING CODES OF P1 & P2
C
      IMPLICIT NONE
C
      INTEGER*2
     . I,                  ! DO LOOP INDEX 1-4
     . INTERSECT,          ! SUM OF BITS FOR LOGICAL INTERSECTION
     .                     !    0=NO COMMON AREAS
     .                     !    0#BOTH ENDS OF LINE IN THE
     .                     !      SAME AREA
     . P1CODES(4),         ! P1'S END POINT CODES
     . P2CODES(4)          ! P2'S END POINT CODES
C
C*************************************************************************
C
C     MAIN:
C
      INTERSECT = 0
      DO 100 I = 1, 4
 100  INTERSECT = INTERSECT + (P1CODES(I) + P2CODES(I) ) / 2
      RETURN
      END
```

```
C*********************************************************************
C  FILENAME:     [CSE.TLT.SMP]CMAPLD.FOR
C  PROGRAMMER:   HAROLD LANE, JR.;  STEVE HUFFMAN
C
C  VERSION:      1.1     REMOVED PROMPT FOR FILE NAME. ALWAYS
C                        LOADS SMP.CMP.
C
C  EXTERNAL CALLS (FILENAME):
C  IKBWR(DMAXXX), IKBWT(DMAXXX), JOIN32(ALUXXX)
C
C  FUNCTION:     LOADS IKONAS COLOR MAP FROM DISK FILE.
C                THE COLOR MAP DATA IS PRECEDED BY A RECORD CONTAINING
C                THE NUMMBER OF COLORS TO BE LOADED.
C                THE FORMAT OF THE FILE IS 3I10 FOR BLUE, GREEN, & RED.
C                IN THAT ORDER.
C
C*************************************************************
      SUBROUTINE CMAPLD
C*************************************************************
C
C
 COMMON /SYSIO/  TTYIN, TTYOUT, HELPIN, PIXFIL
 INTEGER*2       TTYIN, TTYOUT, HELPIN, PIXFIL
C
      INTEGER IMAP ( 3 ), STRTAD
C        LUVO(4)
      INTEGER * 4 CLRMAP ( 1024 ), CMPADR
 BYTE FNAME ( 40 ),  DFLTFN( 20 )
      DATA FNAME ( 40 ) / 0 /
C     DATA LUVO / 0, 4096, 8192, 12288 /
 DATA DFLTFN/'S', 'M', 'P', '$', 'S', 'C', 'R',
     *   'A', 'T', 'C', 'H', ':',
     *   'S', 'M', 'P', '.', 'C', 'M', 'P', 0/
C
```

```
C*****************************************************************
C        MAIN PROG:
C*****************************************************************
C
C******
 TTYOUT = 1
C        PROMPT FOR AND READ FILE NAME OF COLOR MAP TO LOAD:
C100  WRITE ( TTYOUT, 105)
C105      FORMAT(' ENTER COLOR MAP FILENAME:', /
C    *           ' (RETURN FOR DEFAULT: GREY.CMP)' )
C         READ(TTYIN,1000,END=9999,ERR=100)NAMLEN,(FNAME(I),I=1,NAMLEN)
C1000 FORMAT ( Q, 40A1 )
C         FNAME(NAMLEN + 1) = 0
 NAMLEN = 0
C
C******  DEFAULT COLOR MAP:
 IF (.NOT.(NAMLEN .EQ. 0)) GO TO 1080
         DO 1050 I=1,19
                 FNAME(I) = DFLTFN(I)
 1050            CONTINUE
         NAMLEN = 19
 1080    CONTINUE
 FNAME( NAMLEN + 1) = 0
C
C******
C     WRITE (TTYOUT, * ) 'ENTER LUVO NUMBER (0-3):'
C     READ (TTYIN, 1100 ) NLUVO
C
C****** OPEN, READ, & LOAD COLOR MAP FILE:
      OPEN ( UNIT=PIXFIL, NAME=FNAME, TYPE='OLD', IOSTAT=IOST,
     $ FORM = 'FORMATTED', READONLY, ERR=30000 )
C
C        READ STARTING LOCATION WITHIN COLOR MAP:
 READ(PIXFIL, 25)STRTAD
 25      FORMAT(I10)
C
C        READ BLUE, GREEN, RED COLOR DEFNS UNTIL EOF:
 I = 1
 2000    CONTINUE
       READ (PIXFIL, 1100, END=20000 ) ( IMAP ( J ), J = 1, 3 )
1100           FORMAT ( 3I10 )
       CLRMAP ( I ) = 0
C
C              MERGE R, G, B VALUES TO REQD IKONAS FORMAT:
       DO 20002  J = 1, 3
                 CLRMAP(I) = ISHFT(CLRMAP(I),8)+IMAP(J)
                 CLRMAP(I)=ISHFT(CLRMAP(I),2)
20002            CONTINUE
       I = I + 1
       GO TO 2000
20000    CONTINUE
      CLOSE ( UNIT = PIXFIL )
 N = I - 1
C
```

```
C******
      CALL JOIN32 ( CMPADR, 131, STRTAD )
C     CALL JOIN32 ( CMPADR, 131, LUVO(NLUVO + 1) )
      CALL IKBWR ( 16, CMPADR, N, CLRMAP )
      CALL IKBWT
      RETURN
C
C        FILE OPEN FAILURE:
30000    CONTINUE
 WRITE(TTYOUT,*) '(CMAPLD) UNABLE TO OPEN REQUESTED FILE.',
     *   ' FORTRAN I/O STATUS = ', IOST
 RETURN
C
C        END OF FILE ON FILE NAME READ;   RETURN:
 9999    CONTINUE
 RETURN
      END
```

```
C*********************************************************************
C  FILENAME:      [SMP]CURSLD.FOR
C  PROGRAMMER:    HAROLD LANE, JR.
C
C  EXTERNAL CALLS (FILENAME):
C                 IKBWT(DMAXXX), IKBWR(DMAXXX), JOIN32(ALUXXX)
C
C  FUNCTION:      THIS ROUTINE READS AN ASCII FORMATTED FILE OF A
C                 CURSOR DEFINITION OF 32 COLS & 32 ROWS & LOADS
C                 IT INTO THE CURSOR DEFINITION REGITERS. THE
C                 ROUTINE INTERPRETS ASTERISKS *S AS BINARY ONES
C                 AND ANY OTHER CHARACTER AS BINARY ZEROS. THE
C                 ROUTINE PROVIDES THE BIT MANIPULATION, PACKING,
C                 ADDRESSING, & WRITING TO AN RDS-3000 FRAME
C                 BUFFER CONTROLLER #0 TO EFFECT A LOADING OF A
C                 PREDEFINED CURSOR.
C
C*************************************************************
      SUBROUTINE CURSLD
C*************************************************************
C
C
 COMMON /SYSIO/  TTYIN, TTYOUT, HELPIN, PIXFIL
 INTEGER*2       TTYIN, TTYOUT, HELPIN, PIXFIL
C
 BYTE FNAME ( 40 ),  DFLTFN( 26 )
 BYTE CRSCHR(32,32), CRSBIT(32,32)
 INTEGER*2 ROW, COL, WDMODE, CURSOR(2,256), CURS(256)
 INTEGER*2 CRSADR(2)
 DATA DFLTFN/'S', 'M', 'P', '$', 'S', 'C', 'R',
     *   'A', 'T', 'C', 'H', ':', 'T', 'R', 'A', 'C', 'K',
     *   'C', 'R', 'O', 'S', '.', 'C', 'R', 'S', 0/
C
```

```
C*************************************************************
C       MAIN PROG:
C*************************************************************
C
C
C       WORD WRITE BUS FUNCTION CODE:
 WDMODE = 0
C
C       CURSOR ADDRESS IN FRAME BUFFER CONTROLLER:
 CALL JOIN32(CRSADR, 192, 256)
C
C******
C       PROMPT FOR AND READ FILE NAME OF CURSOR DEFN TO LOAD:
C55CC   WRITE ( TTYOUT, 60)
C60CC    FORMAT(' ENTER CURSOR FILE NAME:' /
CCCCC*           ' (RETURN FOR DEFAULT: RTI.CRS)' )
CCCCC    READ(TTYIN,1000,END=9999,ERR=55)NAMLEN,(FNAME(I),I=1,NAMLEN)
C1000    FORMAT ( Q, 40A1 )
CCCCC    FNAME(NAMLEN + 1) = 0
C
C******  DEFAULT CURSOR:
CCCCC    IF (.NOT.(NAMLEN .EQ. 0)) GO TO 1080
         DO 1050 I=1,25
                 FNAME(I) = DFLTFN(I)
 1050            CONTINUE
         NAMLEN = 25
C1080C   CONTINUE
 FNAME( NAMLEN + 1) = 0
C
C******
C
C****** OPEN, READ, & LOAD CURSOR FILE:
 OPEN( UNIT=19, NAME=FNAME, TYPE='OLD', IOSTAT=IOST,
     $   FORM = 'FORMATTED', ERR=2000, READONLY )
 DO 100  COL = 1, 32
         READ(19, 105) (CRSCHR(COL, ROW), ROW=1, 32)
 105             FORMAT(32(32A1) )
 100     CONTINUE
 CLOSE(UNIT=19)
D        TYPE *, '(CURSLD) ASCII CURSOR FILE READ IN.'
C
C        CONVERT CHARACTERS TO 1 OR 0:
 DO 200 COL = 1, 32
         DO 150 ROW = 1, 32
                 CRSBIT(COL, ROW) = 0
                 IF(CRSCHR(COL,ROW).EQ.'*') CRSBIT(COL,ROW)=1
 150             CONTINUE
 200     CONTINUE
C
C        PACK THE 32 X 32 BITS 4 PER NIBBLE IN 256 WORDS AS
C        REQUIRED BY THE HARDWARE.  N. B. THE BITS HAVE TO
C        BE PACKED IN 'BACKWARDS':
 N = 1
 DO 300 COL = 1, 32
```

```
          DO 250 ROW = 1, 29, 4
                  CURS(N)=8 * CRSBIT(COL, ROW+3) +
     *                           4 * CRSBIT(COL, ROW+2) +
     *                           2 * CRSBIT(COL, ROW+1) +
     *                           1 * CRSBIT(COL, ROW  )
                  CALL JOIN32(CURSOR(1,N), 0, CURS(N) )
                  N = N + 1
 250              CONTINUE
 300      CONTINUE
C
C         SEND CURSOR TO IKONAS:
D         WRITE(1,305) WDMODE, CRSADR, (CURSOR(1,I), I=1,256)
D305      FORMAT(' (CURSLD) CURSOR WDMODE, CRSADR, DATA:',
D     *  / I6, 2014 // 32(1X, 803 /) / )
 CALL IKBWR(WDMODE, CRSADR, 256, CURSOR)
 CALL IKBWT
      RETURN
C
C         FILE OPEN FAILURE:
 2000     CONTINUE
 TYPE *, '(CURSLD) UNABLE TO OPEN REQUESTED FILE.',
     *    '  FORTRAN I/O STATUS = ', IOST
 RETURN
C
C         EOF ON READ TO TERMINAL:
 9999     CONTINUE
 RETURN
      END
```

```
C*********************************************************************
C  FILENAME:     [CSE.HHL.NTEC]FBC.FOR
C  PROGRAMMER:   HAROLD LANE, JR.
C  VERSION:      1.0
C
C  EXTERNAL CALLS (FILENAME):
C                IKADDR, IKBWR(DMAXXX), JOIN32(ALUXXX), IKBWT(DMAXXX)
C
C  FUNCTION:     FBC IS A GENERAL PURPOSE ROUTINE TO MANAGE
C        THE FRAME BUFFER CONTROLLER.  IT ALLOWS ONE TO SET
C        THE FBC TO CERTAIN DEFAULT CONDITIONS. IT ALLOWS
C        INDIVIDUAL FIELDS IN THE REGISTERS TO BE SET TO
C        NEW VALUES WITHOUT DISTURBING VALUES OF OTHER
C        FIELDS IN THE SAME REGISTER. ONE CAN ALSO READ
C        BACK THE VALUE OF INDIVIDUAL FIELDS IN REGISTERS.
C        THIS ROUTINE IS DERIVED FROM A FRAME BUFFER
C        WRITE ROUTINE DEVELOPED BY HENRY RICH OF IKONAS.
C        THIS VERSION IS SUBSTANTIALLY MODIFIED IN THAT IT HAS NO
C        I/O TO THE TERMINAL, THAT IT IS A SUBROUTINE (NOT
C        A MAIN PROGRAM), AND THAT IT CAN HAVE FIELDS FROM
C        REGISTERS READ BACK FROM IT. ERROR CHECKING HAS
C        BEEN IMPROVED AND THE TYPE OF THE PARTICULAR
C        ERROR IS PASSED BACK TO THE CALLING ROUTINE.
C
C*************************************************************
 SUBROUTINE FBC(IMODE, IFBNO, IWRITE, IINVBL, JINVAL, IERR)
C*************************************************************
C
C        FBC CALLING PARAMETER USAGE:
C
C        IMODE:  0:  SET A FIELD IN A REG
C                1:  SET FBC TO 512 X 512 30 HZ
C                2:  SET FBC TO 512 X 512 60 HZ
C                3:  SET FBC TO 1024 X 1024 30 HZ
C                4:  FAKE READING BACK A FIELD VALUE FROM FBC
C        IFBNO:  0:  FBC #0
C                1:  FBC #1 (SUPPORT FOR MULTIPLE FBC'S)
C        IWRITE: 0:  INHIBIT WRITING REGISTER VALUES TO DISPLAY
C                1:  WRITE REGISTER VALUES TO DISPLAY
C        IINVBL:     # OF LOGICAL VARIABLE (FIELD) TO BE SET (1-23)
C        JINVAL:     VALUE TO SET LOGICAL VARIABLE TO; OR VALUE
C                    RETURNED WHEN IN READ BACK MODE(IMODE=4)
C        IERR:   0:  NO ERROR
C                1:  BAD IFBNO (0-1 ALLOWED)
C                2:  BAD IMODE # (0-4 ALLOWED)
C                3:  BAD IWRITE # (0-1 ALLOWED)
C                4:  VALUE TO SET REG TO OUT OF RANGE
C                5:  ATTEMPT TO READ AN UNINITIALIZED VARIABLE
C                6:  ATTEMPT TO SEND AN UNINITIALIZED VARIABLE
C
C         USAGE OF ARRAYS:
C         IVALUE - THE LOGICAL SETTINGS FOR THE FBHC
C         IREGS - THE PHYSICAL FBHC REGISTERS, WHICH CONSIST OF
C            LOGICAL VARIABLES OR-ED TOGETHER
```

```
C          IMAX - THE MAX. VALUE FOR A VARIABLE
C          IMIN - THE MIN. VALUE FOR A VARIABLE
C          IVBL - FOR EACH LOGICAL VARIABLE, THE NUMBER OF THE
C             PHYSICAL VARIABLE IN WHICH IT IS TO BE PUT
C          ISHIFT - FOR EACH LOGICAL VARIABLE, THE SHIFT FACTOR BY
C              WHICH IT IS TO BE MULTIPLIED BEFORE BEING PUT INTO ITS
C              PHYSICAL VARIABLE
C          IDEFAL - DEFAULT SETTING FOR LOGICAL VARIABLES, DEPENDING ON
C              USER'S CONFIGURATION
C
C          IREG32 - 32-BIT FBHC REGISTERS
C
C        MEANINGS OF LOGICAL VARIABLES (REGISTER FIELDS):
C                1,2     X-, Y- VIEWPORTS, RESPECTIVELY
C                3,4     X-, Y- SIZES, RESP
C                5,6     X-, Y- WINDOWS, RESP
C                7,8     X-, Y- ZOOMS, RESP
C                9       HORIZONTAL TIME
C                10      # OF LINES/FRAME
C                11      CURSOR ON
C                12      1024 MODE
C                13      AUTO CLEAR
C                14      EXTERNAL SYNC
C                15      COLOR MAP PAGE
C                16      RS-343
C                17      REPEAT FIELD
C                18      PIXEL CLOCK
C                19,20   X-, Y- CURSOR LOCATIONS, RESP
C                21      EXTERNAL PIXEL CLOCK
C                22      PRG SYN
C                23      DRV BPCK
C
```

```
C*************************************************************
C        TYPE DECLARATIONS & INITIALIZATIONS:
C*************************************************************
C
C UNITS.COMMON
      COMMON /UNITS/ IOUT,IN,IU,IUEX
C
 INTEGER*4 IREG32(8),IKADDR,IKCURA
 DIMENSION IVALUE(23,2),IDEFAL(23,3),IREGS(16),IMAX(23),IMIN(23)
 DIMENSION IVBL(23),ISCALE(23)
 INTEGER*2 UNINIT
C
C        # OF LOGICAL & PHYSICAL VARIABLES:
 DATA NVALUE/23/, NREGS/16/
C
C        INITIALIZATION FOR FBC DEFAULT SETTINGS & MIN, MAX, & SHIFTS:
 DATA IDEFAL/0,32,511,511,4095,4067,0,0,300,560,0,0,0,0,0,0,0,45,
     * 0,32,0,0,1,
     * 0,72,511,1023,0,4061,0,0,144,1167,0,0,0,0,0,1,1,19,0,32,0,0,1,
     * 0,64,1023,1023,0,4033,0,0,144,1166,0,1,0,0,0,1,0,20,0,64,0,0,1/
 DATA IMAX/1023,1023,1023,1023,4095,4095,255,255,
     *  4095,4095,1,1,1,1,3,1,1,63,1023,1023,1,1,1/
 DATA IMIN/0,0,0,0,0,0,0,0,100,1,0,0,0,0,0,0,0,0,0,0,0,0,0/
 DATA IVBL/1,2,3,4,5,6,7,8,9,10,11,11,11,11,11,11,11,
     *  12,13,14,12,12,12/
 DATA ISCALE/1,1,1,1,1,1,1,1,1,1,4,8,32,64,128,512,1024,
     *  1,1,1,64,128,256/
 DATA IVALUE/46*-1/
 DATA IFBALO/0/
C
```

```
C*************************************************************
C        MAIN PROG:
C*************************************************************
C
C        SAVE RUN RESET STATUS OF PROCESSOR & FORCE DMA AUTO-INCREMENT:
C        CALL IKBGEI(ICWT)
C        ICWT=IOR(IAND(ICWT,1536),320)
C        CALL IKBSEI(ICWT)
C
C        SET DEFAULT RETURN CODE:
 IERR = 0
        IINVAL = JINVAL
D        WRITE(IOUT, 20) IMODE, IFBNO, IWRITE, IINVBL, JINVAL
D20      FORMAT(' (FBC) IMODE, IFBNO, IWRITE, IINVBL, JINVAL:' / 5I5)
C
C        CHECK TO MAKE SURE THERE IS A VALID FRAME BUFFER NUMBER:
 IF(.NOT.(IFBNO .LT. 0 .OR. IFBNO .GT. 1))GOTO40
         IERR = 1
         RETURN
 40      CONTINUE
C
C        CHECK FOR VALID MODE #:
 IF (.NOT.(IMODE .LT. 0 .OR. IMODE .GT. 4))GOTO 50
         IERR = 2
         RETURN
 50      CONTINUE
C
C        CHECK FOR VALID IWRITE VALUE:
 IF (.NOT.(IWRITE .LT. 0 .OR. IWRITE .GT. 1)) GO TO 55
         IERR = 3
         RETURN
 55      CONTINUE
C
C        SET IFBCNO FOR ARRAY USE,ALSO SET FBC HI ADDR
 IFBAHI=192+IFBNO*8
 IFBCNO=IFBNO+1
 CALL JOIN32(IKADDR,IFBAHI,IFBALO)
C
C        SEE IF A FIELD IN A REGISTER IS BEING SET:
 IF (.NOT.(IMODE .EQ. 0))   GO TO 60
C
C            CHECK FOR VALID VBL #
     IF (.NOT.(IINVBL .GT. NVALUE .OR. IINVBL .LT. 1))GOTO 54
         IERR = 3
         RETURN
54           CONTINUE
C
C            VBL # IS VALID.  BUT IS THE VALUE?
     IF (.NOT.(IINVAL .LT. IMIN(IINVBL) .OR.
     *   IINVAL .GT. IMAX(IINVBL)))GOTO 58
C
C                THE VALUE WAS OUT OF RANGE; RETURN WITH ERROR:
         IERR = 4
         RETURN
```

```
58          CONTINUE
C
C           VALUE WAS VALID;  SET IT INTO ARRAY:
     IVALUE(IINVBL,IFBCNO) = IINVAL
C
C           GO WRITE OUT REGISTERS TO FBC IF REQUESTED TO DO SO:
     GO TO 250
 60      CONTINUE
C
C        SEE IF A REGISTER FIELD IS BEING READ BACK:
 IF (.NOT.(IMODE .EQ. 4)) GO TO 90
C
C                ALLOW REGISTER FIELD TO BE READ BACK:
         IINVAL = IVALUE(IINVBL, IFBCNO)
         IF (.NOT.(IINVAL .EQ. -1))   GO TO 80
C
C                        ATTEMPT TO READ AN UNINITIALIZED VARIABLE:
                 IERR = 5
                 RETURN
 80              CONTINUE
         JINVAL = IINVAL
         RETURN
 90      CONTINUE
C
C        IMODE IS 1, 2, OR 3. INITIALIZE DEFAULTS FOR
C        THE LOGICAL VARIABLES:
 DO 100 I = 1,NVALUE
         IVALUE(I,IFBCNO) = IDEFAL(I,IMODE)
100      CONTINUE
C
C        SET FBC ADDRESS
200      CONTINUE
C
C        NOW COMPUTE THE PHYSICAL VARIABLES FROM
C        THE LOGICAL VARIABLES.
C        FIRST, CLEAR ALL PHYSICAL VBLS TO 0.
 250     CONTINUE
 DO 300 I = 1,NREGS
         IREGS(I) = 0
300      CONTINUE
C
C        NOW, FOR EACH LOGICAL VARIABLE, ADD IN THE VALUE TO THE CORRECT
C        PHYSICAL VARIABLE
 UNINIT = 0
 DO 310 I = 1,NVALUE
         IF (IVALUE(I, IFBCNO) .EQ. -1)   UNINIT = 1
         IREGS(IVBL(I)) = IREGS(IVBL(I))+IVALUE(I,IFBCNO)*ISCALE(I)
310      CONTINUE
C
C        CONVERT FROM 16-BIT VALUES TO IKONAS 32-BIT VALUES
 NREG32 = NREGS/2
 DO 320 I = 1,NREG32
         CALL JOIN32(IREG32(I),IREGS(I*2),IREGS(I*2-1))
320      CONTINUE
```

```
C
D       WRITE(IOUT,350)(I,IVALUE(I,IFBCNO),I=1,NVALUE),(I,IREGS(I),I=1,16)
D350    FORMAT('(FBC) IVALUES:' / 23(/ ,1X, I3, 07) //
D    *   ' IREGS:' / 16(/ ,1X, I3, 07)  )
 IF (.NOT.(IWRITE .EQ. 1)) GO TO 340
C
C               SEE IF ANY UNINITIALIZED DATA; IF SO DON'T SEND IT:
        IF (.NOT.(UNINIT .EQ. 1)) GO TO 335
D                       WRITE(IOUT, 330)
D330                    FORMAT('0(FBC) ATTEMPT TO WRITE UNINITIALIZED DATA'/
D    *                   ' TO FRAME BUFFER CONTROLLER. DATA NOT SENT' )
                IERR = 6
                RETURN
 335            CONTINUE
C
C               THE IKONAS FBHC WORDS ARE COMPUTED. SEND EM IF SO COMMANDED:
D               WRITE(IOUT, 337) IKADDR, NREG32, IREG32
D337            FORMAT(' (FBC) AT IKBWR. IKADDR, NREG32, IREG32(1-8) ',
D    *          012, I12 / 8(1X, 012 /)  )
                CALL IKBWR(0,IKADDR,NREG32,IREG32)
        CALL IKBWT
 340    CONTINUE
 RETURN
 END
```

```
C*********************************************************************
 SUBROUTINE GET_SLD_COLR(CURCOLR,     CLRA, CLRB, CLRC)
C
C        THIS IS THE   SOLIDM   VERSION COLOR CONVERSION ROUTINE
C        (SMP COLORS 0-7 CONVERT TO SOLID 0 < CLRA,-B,-C < 255 AS
C        APPROPRIATE) (CLRA=RED, CLRB=GREEN, CLRC=BLUE)
C
 IMPLICIT NONE
 INTEGER*4 CURCOLR
 INTEGER*2 CLRA, CLRB, CLRC
C       ------------------------------------------------------
 IF     (CURCOLR .EQ. 0)  THEN   ! BLACK
         CLRA = 0
         CLRB = 0
         CLRC = 0
 ELSE IF(CURCOLR .EQ. 1)  THEN   ! RED
         CLRA = 255
         CLRB = 0
         CLRC = 0
 ELSE IF(CURCOLR .EQ. 2)  THEN   ! GREEN
         CLRA = 0
         CLRB = 255
         CLRC = 0
 ELSE IF(CURCOLR .EQ. 3)  THEN   ! YELLOW
         CLRA = 255
         CLRB = 255
         CLRC = 0
 ELSE IF(CURCOLR .EQ. 4)  THEN   ! BLUE
         CLRA = 0
         CLRB = 0
         CLRC = 255
 ELSE IF(CURCOLR .EQ. 5)  THEN   ! MAGENTA
         CLRA = 255
         CLRB = 0
         CLRC = 255
 ELSE IF(CURCOLR .EQ. 6)  THEN   ! CYAN
         CLRA = 0
         CLRB = 255
         CLRC = 255
 ELSE IF(CURCOLR .EQ. 7)  THEN   ! WHITE
         CLRA = 255
         CLRB = 255
         CLRC = 255
 ELSE                            ! DEFAULT IS WHITE
         CLRA = 255
         CLRB = 255
         CLRC = 255
 END IF
 RETURN
 END
```

```
C*********************************************************************
C      SUBROUTINE SLDSMOOTH(NN, N, IF     )
C
C           SMOOTH SHADES ONE PART USING ADAGE'S SOLID-3000 PACKAGE.
C*         ROUTINE SENDS ALL FACES AND NORMALS IN ONE PART TO SOLID.
C           THE PARTS ARE PASSED TO THIS ROUTINE IN COMMON HS.CMN
C
C     NN          = NPOINT (# OF VERTICES (NODES) IN THIS PART)
C     N           = NFACE  (# OF FACES IN THIS PART)
C     IF(5,MAXFPP)= IFACE  (LIST OF FACES (3 OR 4 VERTEX POLYGONS) )
C
C     NPF         = # OF NODES PER FACE IN THIS FACE
C
 INCLUDE 'PARTS.PRM'
        INCLUDE 'DEVICE.CMN'
 INCLUDE 'HS.CMN'
C
      INTEGER*2 ON, OFF, NRMLST, DUMMY
      INTEGER*2 ICOMP, I2NPF, NORMS(4, 5)
C
      REAL*4    XT(4,5), NORM
      INTEGER*2 NNODES, NFACES, NODE, FACE_NO, NPF, INDEX, COUNT(MAXNPP)
      INTEGER*4 IF(5,MAXFPP)
      REAL*4    NORMX(MAXFAC), NORMY(MAXFAC), NORMZ(MAXFAC)
      REAL*4    SUMX(MAXNPP),  SUMY(MAXNPP),  SUMZ(MAXNPP)
      REAL*4    NORMXCOMP,     NORMYCOMP,     NORMZCOMP
C
      DATA ON/1/, OFF/0/
```

```
C-----------------------------------------------------------------------
C
C        INITIALIZE FOR NORMAL CALCULATION:
 NNODES = NN
 NFACES = N
C
 DO 10 NODE = 1, NNODES
    COUNT(NODE) = 0
    SUMX(NODE)  = 0.0
    SUMY(NODE)  = 0.0
    SUMZ(NODE)  = 0.0
 10      CONTINUE
C
C     FOR EACH FACE, CALCULATE AND ACCUMULATE ITS NORMAL:
      II = NF
      DO 30  FACE_NO = 1, NFACES
         NPF = IF(1,FACE_NO)
C
C           FORM THIS FACE'S NORMAL:
C
C           1ST, COPY THE FACE POINTERS INTO FACE(J,II) FOR PLANEQ
C           IGNORING POINT OR LINE ELEMENTS:
            IF (NPF .GE. 3)  THEN
               II = II+1
               FACE(1,II) = NPF
C
               DO 14  J=2,NPF+1
                  FACE(J,II) = IF(J,FACE_NO) + NG
   14          CONTINUE
C
C              LET PLANEQ COMPUTE THE EQUATION OF THE PLANE:
               CALL PLANEQ (II,A,B,C,D)
C
C              NORMALIZE EQN OF THE PLANE OF THIS FACE TO GET ITS NORMAL
C              AND CONVERT IT TO A 16-BIT SIGNED FRACTION FOR SOLID-3000:
               NORM = SQRT(A*A + B*B + C*C     ) / 32767.0
               A = A / NORM
               B = B / NORM
               C = C / NORM
C
C              THE NORMALS ARE INVERTED BECAUSE SMP HAS A
C              RIGHT-HANDED COORDINATE SYSTEM AND
C              SOLID-3000 HAS A LEFT-HANDED COORDINATE SYSTEM:
               NORMXCOMP = -A
               NORMYCOMP = -B
               NORMZCOMP = -C
C
C               FOR EACH NODE (POINT) IN THIS FACE:
C               ADD THIS FACE'S NORMAL COMPONENTS INTO THE NORM ACCUMULATOR:
        DO 11 J = 2, NPF + 1
           NODE = IF(J, FACE_NO)
           COUNT(NODE) = COUNT(NODE) + 1
           SUMX(NODE) = SUMX(NODE) + NORMXCOMP
           SUMY(NODE) = SUMY(NODE) + NORMYCOMP
```

```
            SUMZ(NODE) = SUMZ(NODE) + NORMZCOMP
 11             CONTINUE
      END IF
   30 CONTINUE
C
C         CALCULATE THE AVERAGE NORMAL FOR EACH NODE (POINT; VERTEX):
 DO 700 NODE = 1, NNODES
     NODE_CNT = COUNT(NODE)
     NORMX(NODE+NG) = NINT(SUMX(NODE) / NODE_CNT)
     NORMY(NODE+NG) = NINT(SUMY(NODE) / NODE_CNT)
     NORMZ(NODE+NG) = NINT(SUMZ(NODE) / NODE_CNT)
 700     CONTINUE
C
C         GIVE ONE FACE'S NODES (POINTS) AND AVERAGE NORMALS TO SOLID-3000
C         ONE FACE AT A TIME AS REQUIRED BY SOLID'S SPOLY ROUTINE:
 II = NF
 DO 800 FACE_NO = 1, NFACES
     NPF = IF(1, FACE_NO)
     DO 750 J = 2, NPF + 1
C
C              COPY FACE'S POINTS TO XT(4,5) SO SEND_2_SOLID CAN GET THEM:
               JM1 = J - 1
               INDEX = IF(J, FACE_NO) + NG
               XT(1,JM1) = XG(INDEX)
               XT(2,JM1) = YG(INDEX)
               XT(3,JM1) = ZG(INDEX)
C
C               PUT FACE'S NORMS INTO NORMS(4,N) FOR SEND_2_SOLID:
        NORMS(1, JM1) = NORMX(INDEX)
        NORMS(2, JM1) = NORMY(INDEX)
        NORMS(3, JM1) = NORMZ(INDEX)
 750        CONTINUE
C
C           GIVE FACE'S POINTS & NORMALS TO SOLID-3000:
            I2NPF = NPF
            CALL SEND_2_SOLID(I2NPF, XT, NORMS, IDASH     )
 800     CONTINUE   ! CLOSE FACE LOOP
C
C     UPDATE NODE AND FACE COUNTERS
      NG = NG+NN
      NF = II
C
      RETURN
      END
```

```
C       FILENAME:       SLDSUBS.FOR
C
C*********************************************************************
        SUBROUTINE SODRWA(XW, YW, IDASH     )
C
C       DRAW A LINE USING FLOATING WORLD COORDS.  THE OTHER POINT TO
C       DRAW FROM IS (XCUR, YCUR) IN COMMON /ADAGE/.
C
        IMPLICIT NONE
C
        COMMON /ADAGE/ CURCOLR, ICURX, ICURY
        INTEGER*4      CURCOLR
        INTEGER*2                ICURX, ICURY
C
        COMMON /ADVUPR/ IXMINS, IXMAXS, IYMINS,  IYMAXS,
     .                  IXVSIZ, IYVSIZ, IXVCNTR, IYVCNTR
        INTEGER         IXMINS, IXMAXS, IYMINS,  IYMAXS,
     .                  IXVSIZ, IYVSIZ, IXVCNTR, IYVCNTR
        COMMON /ADWIND/ XMINW,  XMAXW,  YMINW,   YMAXW,
     .                  XWSIZE, YWSIZE, XWCNTR,  YWCNTR
        REAL            XMINW,  XMAXW,  YMINW,   YMAXW,
     .                  XWSIZE, YWSIZE, XWCNTR,  YWCNTR
C
        INTEGER*2 IX, IY
        INTEGER*4 IDASH
        REAL      XW, YW
C
 INTEGER*2 SCOLOR, SPOLY         ! CALLED FUNCTIONS NAMES
 INTEGER*2 INTEN, COLOR, DUMMY_COLOR, DUMMY, ICOMP
 INTEGER*2 SET_LINE_COLOR/4/
C
 INTEGER*2 NO_VERTICES, VERTEX_LIST(4, 2)
 DATA NO_VERTICES /2/,  VERTEX_LIST /8*0/
C       ------------------------------------------------------
C
C       PERFORM WINDOWING & VIEWPORTING:
        IX = (((XW-XWCNTR) / XWSIZE) + 0.5 ) * IXVSIZ + IXMINS - 255
        IY = (((YW-YWCNTR) / YWSIZE) + 0.5 ) * IYVSIZ + IYMINS - 255
C
C        CONVERT CURRENT COLOR TO SOLID'S FORMAT;  SEND IT TO SOLID:
 CALL GET_SLD_COLR(CURCOLR,     INTEN, COLOR, DUMMY_COLOR)
 ICOMP = SCOLOR(INTEN, COLOR, DUMMY_COLOR     )
C
C        PUT LINE ENDPOINTS INTO FORMAT REQUIRED BY SOLID:
 VERTEX_LIST(1,1) = ICURX
 VERTEX_LIST(2,1) = ICURY
 VERTEX_LIST(1,2) = IX
 VERTEX_LIST(2,2) = IY
C
C        DRAW LINE BY PRETENDING IT IS A POLYGON:
 ICOMP = SPOLY(SET_LINE_COLOR, INTEN, COLOR, DUMMY_COLOR,
     .   DUMMY, NO_VERTICES, VERTEX_LIST, DUMMY     )
C
C        SET LAST POINT DRAWN TO AS THE 'CURRENT' POINT:
```

```
      ICURX = IX
      ICURY = IY
C
      RETURN
      END
```

```
C*********************************************************************
        SUBROUTINE SODRWS(IXS, IYS, IDASH     )
C
C       DRAW A LINE USING INTEGER SEREEN COORDS.  THE OTHER POINT TO
C       DRAW FROM IS (ICURX, ICURY) IN COMMON /ADAGE/.
C
        IMPLICIT NONE
C
        COMMON /ADAGE/ CURCOLR, ICURX, ICURY
        INTEGER*4      CURCOLR
        INTEGER*2                ICURX, ICURY
C
        COMMON /ADVUPR/ IXMINS, IXMAXS, IYMINS,  IYMAXS,
     .                  IXVSIZ, IYVSIZ, IXVCNTR, IYVCNTR
        INTEGER         IXMINS, IXMAXS, IYMINS,  IYMAXS,
     .                  IXVSIZ, IYVSIZ, IXVCNTR, IYVCNTR
C
        INTEGER*2 IXS, IYS
        INTEGER*4 IDASH
 INTEGER*2 SCOLOR, SPOLY          ! CALLED FUNCTIONS NAMES
 INTEGER*2 INTEN, COLOR, DUMMY_COLOR, DUMMY, ICOMP
 INTEGER*2 SET_LINE_COLOR /4/
C
 INTEGER*2 NO_VERTICES, VERTEX_LIST(4, 2)
 DATA NO_VERTICES /2/,  VERTEX_LIST /8*0/
C       ------------------------------------------------------
C
C        CONVERT CURRENT COLOR TO SOLID'S FORMAT;  SEND IT TO SOLID:
 CALL GET_SLD_COLR(CURCOLR,     INTEN, COLOR, DUMMY_COLOR)
 ICOMP = SCOLOR(INTEN, COLOR, DUMMY_COLOR     )
C
C        OFFSET THE POINT TO AGREE WITH SOLID'S COORDINATE SYSTEM:
 IXS = IXS - 255
 IYS = IYS - 255
C
C        PUT LINE ENDPOINTS INTO FORMAT REQUIRED BY SOLID:
 VERTEX_LIST(1,1) = ICURX
 VERTEX_LIST(2,1) = ICURY
 VERTEX_LIST(1,2) = IXS
 VERTEX_LIST(2,2) = IYS
C
C        DRAW LINE BY PRETENDING IT IS A POLYGON:
 ICOMP = SPOLY(SET_LINE_COLOR, INTEN, COLOR, DUMMY_COLOR,
     .   DUMMY, NO_VERTICES, VERTEX_LIST, DUMMY     )
C
C        SET LAST POINT DRAWN TO AS THE 'CURRENT' POINT:
        ICURX = IXS
        ICURY = IYS
        RETURN
        END
```

```
C*********************************************************************
      SUBROUTINE SOMOVA(XW, YW     )
C
C       MOVE USING FLOATING WORLD COORDS.
C
        IMPLICIT NONE
C
        COMMON /ADAGE/ CURCOLR, ICURX, ICURY
        INTEGER*4      CURCOLR
        INTEGER*2                ICURX, ICURY
C
        COMMON /ADVUPR/ IXMINS, IXMAXS, IYMINS,  IYMAXS,
     .                  IXVSIZ, IYVSIZ, IXVCNTR, IYVCNTR
        INTEGER         IXMINS, IXMAXS, IYMINS,  IYMAXS,
     .                  IXVSIZ, IYVSIZ, IXVCNTR, IYVCNTR
        COMMON /ADWIND/ XMINW,  XMAXW,  YMINW,   YMAXW,
     .                  XWSIZE, YWSIZE, XWCNTR,  YWCNTR
        REAL            XMINW,  XMAXW,  YMINW,   YMAXW,
     .                  XWSIZE, YWSIZE, XWCNTR,  YWCNTR
C
        INTEGER*2 IX, IY
        REAL      XW, YW
C
C       PERFORM WINDOWING & VIEWPORTING:
        IX = (((XW-XWCNTR) / XWSIZE) + 0.5 ) * IXVSIZ + IXMINS - 255
        IY = (((YW-YWCNTR) / YWSIZE) + 0.5 ) * IYVSIZ + IYMINS - 255
C
C       SET THE PASSED IN POINT AS THE CURRENT POINT:
        ICURX = IX
        ICURY = IY
        RETURN
        END
```

```
C*********************************************************************
      SUBROUTINE SOMOVS(IXS, IYS     )
C
C     MOVE USING INTEGER SEREEN COORDS.
C
      IMPLICIT NONE
C
      COMMON /ADAGE/ CURCOLR, ICURX, ICURY
      INTEGER*4      CURCOLR
      INTEGER*2                ICURX, ICURY
C
      COMMON /ADVUPR/ IXMINS, IXMAXS, IYMINS,  IYMAXS,
     .                IXVSIZ, IYVSIZ, IXVCNTR, IYVCNTR
      INTEGER         IXMINS, IXMAXS, IYMINS,  IYMAXS,
     .                IXVSIZ, IYVSIZ, IXVCNTR, IYVCNTR
C
      INTEGER*2 IXS, IYS, IX, IY
C
      ICURX = IXS - 255
      ICURY = IYS - 255
      RETURN
      END
```

```
C*********************************************************************
      SUBROUTINE SEND_2_SOLID(N, X, NORMS, IDASH     )
C
C     DEFINE ONE FACE OF A POLYGON. PUT A NORMAL
C     AT THE FIRST VERTEX IN EACH FACE.  IDASH IS NOT USED CURRENTLY.
C     SEND POLYGON AND NORMAL TO SOLID-3000.
C
C     N.B. CONTAINS DEVICE DEPENDENT CODE
C
      INCLUDE 'PARTS.PRM'
      INCLUDE 'DEVICE.CMN'
      INCLUDE 'ADAGE.CMN'
C
        COMMON /ADVUPR/ IXMINS, IXMAXS, IYMINS,  IYMAXS,
     .                  IXVSIZ, IYVSIZ, IXVCNTR, IYVCNTR
        INTEGER         IXMINS, IXMAXS, IYMINS,  IYMAXS,
     .                  IXVSIZ, IYVSIZ, IXVCNTR, IYVCNTR
        COMMON /ADWIND/ XMINW,  XMAXW,  YMINW,   YMAXW,
     .                  XWSIZE, YWSIZE, XWCNTR,  YWCNTR
        REAL            XMINW,  XMAXW,  YMINW,   YMAXW,
     .                  XWSIZE, YWSIZE, XWCNTR,  YWCNTR
C
        COMMON /SOZWND/ ZMINSLDW, ZMAXSLDW, ZWSIZW, ZWCNTRW
        REAL            ZMINSLDW, ZMAXSLDW, ZWSIZW, ZWCNTRW
C
      REAL*4 X(4,5)
C
C     VRTXLST(X|Y|Z|W, # OF VERTICES):
      INTEGER*2 VRTXLST(4, 5)
      INTEGER*2 NORMS(4, 5)
C
      INTEGER*2 ICOMP, SPOLY, DUMMY/0/, N
      INTEGER*2 DEFNORM/1/
C
C---------------------------------------------------------------------
C
C
C       PUT POLYGON DEFINITION INTO FORMAT REQUIRED BY SOLID 3000:
C
C       PERFORM WINDOWING & VIEWPORTING:
        IX = ((X(1,1)-XWCNTR) / XWSIZE) * IXVSIZ + XMINS
        IY = ((X(2,1)-YWCNTR) / YWSIZE) * IYVSIZ + YMINS
C
C        Z IS "WINDOWED" AND "VIEWPORTED" TO MAKE THE BEST USE OF
C        SOLID-3000'S Z-BUFFER ALGORITHM:
        IZ = ((X(3,1)-ZWCNTRW) / ZWSIZW) * 32767.0
C
        VRTXLST(1,1) = IX
        VRTXLST(2,1) = IY
C
C        TRY INVERTING THE Z VALUE OF POINTS TO SEE IF THAT WILL MAKE
C        THE PIX MATCH THE HIDDEN LINE RENDERED PIX IN THE PIXEL
C        STUFFING VERSION OF SMP:
        VRTXLST(3,1) = IZ
```

```
C
C        THIS TRICKY LITTLE BIT OF CODE COPIES THE FIRST
C        VERTEX AND ADDS IT AT THE END OF THE VERTEX LIST
C        TO CLOSE THE POLYGON:                -HHL
C
C
C         NO IT DOESN'T;  IT MERELY SHIFTS THE POINTS AROUND THE FACE BY
C         ONE INDEX.   I DON'T KNOW WHY.      -HHL
C
C


  DO 5 I = 1, N
            IF (I .NE. N) THEN
               IP1 = I + 1
            ELSE
               IP1 = 1
            END IF
C
C           PERFORM WINDOWING & VIEWPORTING:
            IX = ((X(1,IP1)-XWCNTR) / XWSIZE) * IXVSIZ + XMINS
            IY = ((X(2,IP1)-YWCNTR) / YWSIZE) * IYVSIZ + YMINS
C
C            Z IS "WINDOWED" AND "VIEWPORTED" TO MAKE THE BEST USE OF
C            SOLID-3000'S Z-BUFFER ALGORITHM:
            IZ = ((X(3,IP1)-ZWCNTRW) / ZWSIZW) * 32767.0
C
            VRTXLST(1,IP1) = IX
            VRTXLST(2,IP1) = IY
C
C            TRY INVERTING THE Z VALUE OF POINTS TO SEE IF THAT WILL MAKE
C            THE PIX MATCH THE HIDDEN LINE RENDERED PIX IN THE PIXEL
C            STUFFING VERSION OF SMP:
            VRTXLST(3,IP1) = -IZ
 5       CONTINUE
C
C        GIVE POLYGON DEFINITION TO SOLID SYSTEM (IT IS NOT SENT
C        TO THE DISPLAY UNTIL A DIDUMP CALL IS MADE):
        ICOMP = SPOLY(DEFNORM, DUMMY, DUMMY, DUMMY,
     .  DUMMY, N, VRTXLST, NORMS     )
C
      RETURN
      END
```

```
C*********************************************************************
        SUBROUTINE SLDZWIND(ZMINSLD, ZMAXSLD     )
C
C       SAVE Z-"WINDOW" SPECS IN COMMON IN WORLD COORDINATES:
C
 IMPLICIT NONE
C
        COMMON /SOZWND/ ZMINSLDW, ZMAXSLDW, ZWSIZW, ZWCNTRW
        REAL            ZMINSLDW, ZMAXSLDW, ZWSIZW, ZWCNTRW
C
        REAL             ZMINSLD,  ZMAXSLD
C
C---------------------------------------------------------------------
C
 ZMINSLDW = ZMINSLD
 ZMAXSLDW = ZMAXSLD
 ZWSIZW  = ZMAXSLDW - ZMINSLDW
 ZWCNTRW  = (ZMAXSLDW + ZMINSLDW) / 2.0
        RETURN
        END
```

NOTE: Although this subroutine was not entirely written by RTI,
it is so heavily modified that it should be included here.

```
C*********************************************************************
      SUBROUTINE STRMDL (NMAX,NN,X,N,IF,T,IPC,INERR)
C
C*          ROUTINE TO STORE ALL MODEL NODES AND FACES IN ONE PART
CSLD:
C           N.B. DEVECE DEPENDENT CODE
C
C     NMAX        = MAXNPP (MAX NODES PER PART PARAMETER IN PARTS.PRM
C                   ACTUALLY SHOULD NOT BE PASSED)
C     NN          = NPOINT (# OF VERTICES (NODES) IN THIS PART)
C     X(3,4)      = XKNOT  (VERTEX (NODE) LIST)
C     N           = NFACE  (# OF FACES IN THIS PART)
C     IF(5,MAXFPP)= IFACE  (LIST OF FACES (3 OR 4 VERTEX POLYGONS) )
C     T(4,4)      = TRANSFORMATION ARRAY TO RE-ORIENT PART FOR VIEWING)
C     IPC         = THIS PARTS COLOR. (SET BY SUBR. STRCOLR)
C     INERR       = ARRAY OVERFLOW INDICATOR = 0,1, OR 2. (RETURNED)
C
C     NPF         = # OF NODES PER FACE IN THIS FACE
C
 INCLUDE 'PARTS.PRM'
 INCLUDE 'PARTS.CMN'
        INCLUDE 'DEVICE.CMN'
CSLDEND
 INCLUDE 'HS.CMN'
C
      COMMON /TMNMX/ TXMIN,TXMAX,TYMIN,TYMAX,TZMIN,TZMAX
      COMMON /HSOPT/ SORL,BFCULL,ISMOTH,DIF,EN,DORD
      CHARACTER*1 SORL,BFCULL
      COMMON /OBSRVR/ OX,OY,OZ
CSLD: DIMENSION X(NMAX,3),IF(5,1),T(4,4)
CSLD:
      DIMENSION X(NMAX,3),IF(5,MAXFPP),T(4,4)
      INTEGER*2 COLORA, COLORB, COLORC, ON, NORMS(4,5), NRMLST, DUMMY
      INTEGER*2 ICOMP, I2NPF, OFF, LIGHT1, SCOLOR
      INTEGER*2 XLITVEC, YLITVEC, ZLITVEC, VECTOR, IBFINV
      INTEGER*2 AMBLIT, DIFUSE, RFLCTD, EXPONT, WEIGHT, WHITE
      REAL*4    XT(4,5), NORM
      CHARACTER*1 ANS
      DATA NRMLST/1/, ON/1/, OFF/0/
      DATA XLITVEC/100/, YLITVEC/100/, ZLITVEC/-500/, VECTOR/0/
C
```

```
C-----------------------------------------------------------------------
CSLDEND
C
C     CHECK FOR NODE NUMBER AND FACE NUMBER LIMITS
      INERR = 0
      IF ((NG+NN) .GT. MAXNOD)  THEN
         INERR = 1
         GO TO 999
      END IF
      IF ((NF+N ) .GT. MAXFAC)  THEN
         INERR = 2
         GO TO 999
      END IF
C
C     TRANSFORM AND STORE NODES
CSLD:
      IF (IDEV .EQ. 6)   THEN
         DO 10  I=1,NN
           NODENO = I + NG
           CALL UAPPLY (X(I,1),X(I,2),X(I,3), T,
     .     XG(NODENO),YG(NODENO),ZG(NODENO))
           TXMAX = MAX (TXMAX , XG(NODENO))
           TXMIN = MIN (TXMIN , XG(NODENO))
           TYMAX = MAX (TYMAX , YG(NODENO))
           TYMIN = MIN (TYMIN , YG(NODENO))
           TZMAX = MAX (TZMAX , ZG(NODENO))
           TZMIN = MIN (TZMIN , ZG(NODENO))
 10      CONTINUE
      ELSE
CSLDEND
 DO 11 I=1, NN
    II = I+NG
    CALL UAPPLY (X(I,1),X(I,2),X(I,3), T, XG(II),YG(II),ZG(II))
    TXMAX = MAX (TXMAX , XG(II))
    TXMIN = MIN (TXMIN , XG(II))
    TYMAX = MAX (TYMAX , YG(II))
    TYMIN = MIN (TYMIN , YG(II))
    TZMAX = MAX (TZMAX , ZG(II))
    TZMIN = MIN (TZMIN , ZG(II))
 11      CONTINUE
CSLD:
      END IF
CSLDEND
CSLD 10 CONTINUE
C
C     DEFINE OBSERVER POSITION
CRG   OZ = TZMAX + 0.1*ABS(TZMAX-TZMIN)
      OZ = TZMAX + 2.0* ABS(TZMAX-TZMIN)
CSLD:
      IF (IDEV .EQ. 6)   THEN
C
C        GIVE COLOR OF PART TO SOLID-3000:
         CALL GET_SLD_COLR(IPC,    COLORA, COLORB, COLORC)
         ICOMP = SCOLOR(COLORA, COLORB, COLORC     )
```

```
C
C        IF REQUESTED, DO SMOOTH SHADING:
         IF (ISMOTH .EQ. 1) THEN
     CALL SLDSMOOTH(NN, N, IF     )
     RETURN
         END IF
      END IF
CSLDEND
C
C    STORE FACES AND COLOR
      II = NF
      DO 30  I=1,N
CSLD:
         NPF = IF(1,I)
C
C        FOR EACH FACE, GIVE FACE DATA, COLOR, & NORMALS TO SOLID:
         IF (IDEV .EQ. 6)   THEN
C
C           FORM THIS FACE'S NORMAL (REQUIRED BY SOLID-3000):
C
C           1ST, COPY THE FACE POINTERS INTO FACE(J,II) FOR PLANEQ
C           IGNORING POINT OR LINE ELEMENTS:
            IF (NPF .GE. 3)  THEN
               II = II+1
               FACE(1,II) = NPF
C
               DO 14  J=2,NPF+1
                  FACE(J,II) = IF(J,I) + NG
   14          CONTINUE
C
C              LET PLANEQ COMPUTE THE EQUATION OF THE PLANE:
               CALL PLANEQ (II,A,B,C,D)
C
C              NORMALIZE EQN OF THE PLANE OF THIS FACE TO GET ITS NORMAL
C              AND CONVERT IT TO A 16-BIT SIGNED FRACTION FOR SOLID-3000:
               NORM = SQRT(A*A + B*B + C*C     ) / 32767.0
C
C               INSURE THAT THERE WILL BE NO FLOATING DIVIDE BY ZERO:
         IF (NORM .EQ. 0.0) THEN
            A = 32767.0
            B = 32767.0
            C = 32767.0
         ELSE
            A = A / NORM
            B = B / NORM
            C = C / NORM
         END IF
C
C              SOLID-3000 ONLY REQUIRES 1 NORM FOR EACH FACE:
C              THE NORMALS ARE INVERTED BECAUSE SMP HAS A
C              RIGHT-HANDED COORDINATE SYSTEM AND
C              SOLID-3000 HAS A LEFT-HANDED COORDINATE SYSTEM:
C
C               INSURE NO OVERFLOW WHEN CONVERTING FROM REAL*4 TO I*2:
```

```
         IF (A .GT. 32767.0) THEN
            NORMS(1,1) = -32767
         ELSE
            NORMS(1,1) = -NINT(A)
         END IF
C
         IF (B .GT. 32767.0) THEN
            NORMS(2,1) = -32767
         ELSE
            NORMS(2,1) = -NINT(B)
         END IF
C
         IF (C .GT. 32767.0) THEN
            NORMS(3,1) = -32767
         ELSE
            NORMS(3,1) = -NINT(C)
         END IF
C
C              COPY FACE'S POINTS TO XT(4,5) SO SEND_2_SOLID CAN GET THEM:
               DO 12 J = 2, NPF + 1
                  JM1 = J - 1
                  INDEX = IF(J, I) + NG
                  XT(1,JM1) = XG(INDEX)
                  XT(2,JM1) = YG(INDEX)
                  XT(3,JM1) = ZG(INDEX)
 12            CONTINUE
C
C              GIVE FACE'S POINTS & NORMALS TO SOLID-3000:
               I2NPF = NPF
               CALL SEND_2_SOLID(I2NPF, XT, NORMS, IDASH     )
            END IF
          ELSE
CSLDEND
            NPF = IF(1,I)
C
C           IGNORE POINT OR LINE ELEMENTS
            IF (NPF .LT. 3)  GO TO 30
            II = II+1
            FACE(1,II) = NPF
            IFC(II) = IPC
            DO 20  J=2,NPF+1
               FACE(J,II) = IF(J,I) + NG
   20       CONTINUE
C
C           "CULL" THE BACK-FACES BY TAKING DOT PRODUCT OF PLANE
C           NORMAL WITH VECTOR IN DIRECTION OF OBSERVER
            IF (BFCULL .EQ. 'Y')  THEN
C
C              COMPUTE THE PLANE EQUATION OF THE GIVEN FACE
               CALL PLANEQ (II,A,B,C,D)
               DOTPR = A*OX + B*OY + C*OZ
               IF (DOTPR .LE. 0.0)  THEN
                  PFLAG(II) = .FALSE.
                  II = II-1
```

```
                  END IF
               END IF
CSLD:
            END IF
CSLDEND
   30 CONTINUE
C
C     UPDATE NODE AND FACE COUNTERS
      NG = NG+NN
      NF = II
C
  999 CONTINUE
C
      RETURN
      END
```

NOTE: Although this subroutine was not entirely written by RTI,
it is so heavily modified that it should be included here.

```
C*********************************************************************
      SUBROUTINE HSDRVR
C
C          N.B. CONTAINS DEVICE DEPENDENT CODE
C
C*         DRIVER FOR THE HIDDEN SURFACE (LINE) REMOVAL ROUTINES.
C*         DISPLAY AND SHADING OPTIONS ARE SPECIFIED BY THE USER
C*         IN THIS ROUTINE.
C
 INCLUDE 'PARTS.PRM'
CSLD:
        INCLUDE 'DEVICE.CMN'
 INCLUDE 'ADAGE$SOLID:SPARMS.INC/NOLIST'
CENDSLD
 INCLUDE 'PARTS.CMN'
 INCLUDE 'LL.CMN'

CHARACTER*80 LINE
      LOGICAL LSHRINK,PLABEL
      COMMON /DISOPT/ LHIDE,LSHRINK,SFAC,PLABEL,LSELECT(MAXPART),LINE
     +              , IDSDW(MAXPART)
      COMMON /TW/ T,W,S
      DIMENSION T(4,4),W(4),S(4)
      COMMON /HSOPT/ SORL,BFCULL,ISMOTH,DIF,EN,DORD
      CHARACTER*1 SORL,BFCULL
      COMMON /SCROPT/  IMIN,IMAX,ICTABL
      CHARACTER*1 IANS
      DIMENSION XT(3,4)
      DIMENSION ORG(3)
      CHARACTER*4 OPTION
      DIMENSION NREC(4)
CSLD:
      COMMON /ANTIALIAS/ AALIAS_REQ
      CHARACTER*1 AALIAS_REQ
      INTEGER*2 FULCOL/17/
      INTEGER*2 ON/1/, OFF/0/, ICOMP
      INTEGER*2 AMBLIT, DIFUSE, RFLCTD,
     .     EXPONT, WEIGHT, WHITE, LIGHT1
      INTEGER*2 COLORA, COLORB, COLORC, DUMMY
      INTEGER*2 XLITVEC, YLITVEC, ZLITVEC, VECTOR
      DATA XLITVEC/100/, YLITVEC/100/, ZLITVEC/-500/, VECTOR/0/
      DATA AMBLIT/6000/, DIFUSE/22000/, RFLCTD/4000/,
     .     EXPONT/20/, WEIGHT/32767/, WHITE/7/, LIGHT1/1/
CSLDEND
      DATA ISMOTH,EN,DIF /0,1.0,0.30/
```

```
C-----------------------------------------------------------------------
C
C     READ HIDDEN SURFACE INPUT OPTIONS
      WRITE (IOUT,2002)
 2002 FORMAT (' ENTER  S - ELIMINATE HIDDEN SURFACE',/,
     +        '        L - ELIMINATE HIDDEN LINES',/
     +        '        B - BOTH (SEPARATE VIEWS)',/,
     +        '        R - RETURN')
      READ (IN,'(A)',END=1,ERR=1)  SORL
    1 IF (SORL.EQ.'R') RETURN
      IF (SORL.NE.'S' .AND. SORL.NE.'L')  SORL='B'
C
CSLD:
      IF (IDEV .EQ. 6) THEN
  IF (SORL .EQ. 'B'   .OR.   SORL .EQ. 'L') THEN
     WRITE(IOUT, 2001)
 2001        FORMAT( 'DEVICE 6: S - ELIMINATE HIDDEN SURFACE USED' /)
     SORL = 'S'
  END IF
      ENDIF
CSLDEND
C     THE NEXT THREE OPTIONS ARE "SPECIAL PURPOSE" ONLY
C     ALLOW THE USER TO BYPASS THESE AND USE DEFAULT VALUES
  120 WRITE (IOUT,2018)
 2018 FORMAT ('ENTER "Y" TO OVERRIDE DEFAULT',/,
     +        'IMAGE DISPLAY OPTIONS')
      IANS = 'N'
      READ (IN,'(A)',ERR=120,END=130)  IANS
C     IF RESPONSE IS "NEGATIVE" USE DEFAULTS
  130 IF (IANS .NE. 'Y')  THEN
         DORD = 0
         IANS = 'N'
         ICTABL = 2
         ICHANG = 0
         GO TO 199
      END IF
C
C
C     IF HIDDEN SURFACES HAVE BEEN REQUESTED, DETERMINE WHETHER
C     IMAGE SHOULD BE DISPLAYED OR STORED ON DISK
      DORD = 0
      IF (SORL.NE.'L')  THEN
  150    WRITE (IOUT,2013)
 2013    FORMAT ('ENTER DISPLAY/DISK IMAGE OPTION',/
     1           '   0 - IMAGE OUTPUT TO DISPLAY',/,
CADGSLD 2           '   1 - IMAGE STORED ON FILE "SMP.IMG"')
CADGSLD     READ (IN,'(I1)',END=160,ERR=150)  DORD
CADGSLD:
     2           '   1 - IMAGE STORED ON FILE "SMGG.IMG"')
         READ (IN,'(F1.0)',END=160,ERR=150)  DORD
CADGSLDEND
  160    IF (DORD.NE.0 .AND. DORD.NE.1)  THEN
            DORD = 0
         END IF
```

```
      END IF
C
CSLD:
      IF (IDEV .EQ. 6   .AND.   DORD .EQ. 1) THEN
   WRITE(IOUT, 2014)
 2014      FORMAT('DEVICE 6: IMAGE NOT IN STORABLE FORMAT' / )
   DORD = 0
      END IF
C
      IF (IDEV .NE. 6) THEN
CSLDEND
  170     WRITE (IOUT,2015)
 2015     FORMAT ('ENTER RESOLUTION CHANGE OPTION',/,
     1         '   N - DEFAULT TO DEVICE LIMITS',/,
     2         '   Y - REDUCE FOR FASTER DISPLAY')
           ICHANG = 0
           READ (IN,'(A)',ERR=170,END=180)  IANS
  180      IF (IANS .EQ. 'Y')  THEN
              ICHANG = 1
  190         WRITE (IOUT,2016)
 2016         FORMAT ('ENTER NEW RESOLUTION (SQUARE ONLY)',/,
     1            '   RESOLUTION FOLLOWED BY STARTING PIXEL')
              READ (IN,*,ERR=190,END=195)  IMAX,IMIN
  195         ITEMP = IMIN
              IMIN = MIN (ABS(ITEMP),ABS(IMAX))
              IMAX = IMIN + MAX(ABS(ITEMP),ABS(IMAX))
              IF (IMIN .EQ. IMAX)  IANS='N'
           END IF
C
  197     WRITE (IOUT,2017)
 2017     FORMAT ('ENTER COLOR TABLE OPTION',/,
     1         '      0 - USE ASSIGNED PART COLORS',/,
     2         '      1 - OVER-RIDE WITH GREY SCALE')
           READ (IN,'(I1)',ERR=197,END=198)  ITEMP
  198      IF (ITEMP .EQ. 1)  THEN
              ICTABL = 1
           ELSE
              ICTABL = 2
           END IF
C
C          SET "COMPUTATION RESOLUTION"
  199      CALL CMPRES (DORD,ICHANG,IMIN,IMAX)
CSLD:
      END IF
CSLDEND
CRG
      IF (SORL.NE.'L')  THEN
 200    WRITE(IOUT,2004)
 2004   FORMAT('ENTER SHADING OPTION',/,
     +            '   0 - FLAT ELEMENT SHADING',/,
     +            '   1 - SMOOTH ELEMENT SHADING')
        READ(IN,'(I1)',END = 200,ERR=200) ISMOTH
        IF (ISMOTH.GT.1) ISMOTH = 1
         IF (ISMOTH.LT.0) ISMOTH = 0
```

```
C
C       CHANGE SHADING PARAMETERS?
250     WRITE(IOUT,2005)
 2005   FORMAT('ENTER Y TO CHANGE SHADING PARAMETERS')
        READ(IN,'(A)',ERR=250,END = 500) IANS
        IF (IANS .EQ.'Y') THEN
CSLD:
          IF (IDEV .EQ. 6) THEN
C
     WRITE(IOUT,*)'ENTER AMBIENT, DIFFUSE, REFLECTED VALUES OF',
     .       ' LIGHT:'
     WRITE(IOUT,*)'(THE SUM OF AMBIENT, DIFFUSE, & REFLECTED',
     .      ' SHOULD BE < 32767)'
     WRITE(IOUT,*)'(AMB=6000, DIF=22000, & REF=4000 IS GOOD)'
            READ(IN,19,END=17,ERR=17) AMBLIT, DIFUSE, RFLCTD
 19         FORMAT(3I6)
 17         CONTINUE
C
     WRITE(IOUT,*)'ENTER EXPONENT FOR REFLECTED LIGHT',
     .       ' (20 IS GOOD):'
     READ(IN,*,END=21,ERR=21) EXPONT
     IF (EXPONT .EQ. 0.0) THEN
         WRITE(IOUT,*) 'EXPONENT MUST NOT BE ZERO'
         GO TO 17
     END IF
 21         CONTINUE
C
     ICOMP = SLINUM(LIGHT1, OFF, OFF, OFF     )
     CALL GET_SLD_COLR(WHITE,     COLORA, COLORB, COLORC)
     CALL SLIGHT(LIGHT1, XLITVEC, YLITVEC, ZLITVEC, VECTOR,
     .      COLORA, COLORB, COLORC, AMBLIT, DIFUSE, RFLCTD,
     .      EXPONT, WEIGHT     )
C
     WRITE(IOUT, *) 'ANTI-ALIASING? (DEFAULT-NONE)(Y OR N):'
     READ(IN,50,END=22,ERR=22) AALIAS_REQ
 50         FORMAT(A1)
 22         CONTINUE
C
     IF (AALIAS_REQ .EQ. 'Y') THEN
         ICOMP = SALIAS(ON)
         ICOMP = SQDRAW(OFF)
C
C               MONITOR GAMMA?
         WRITE(IOUT, *) 'ENTER MONITOR GAMMA (1.8 IS TYPICAL):'
         READ(IN, '(F5.3)', END=557, ERR=557) GAMMA
 557             CALL SLUVO(FULCOL, GAMMA)

     ELSE
         ICOMP = SALIAS(OFF)
         ICOMP = SQDRAW(ON)
     END IF
C
C            DON'T ASK PHONG/GOURAUD QUESTION IF DOING FLAT SHADING:
     IF (ISMOTH .EQ. 1) THEN
```

```
C
        WRITE(IOUT, *) 'PHONG SHADING? (DEFAULT-GOURAUD)(Y OR N):'
        IANS = 'N'
        READ(IN,50,END=23,ERR=23) IANS
 23             CONTINUE
C
        IF (IANS .EQ. 'Y') THEN
           ICOMP = SPHONG(ON)
        ELSE
           ICOMP = SPHONG(OFF)
        END IF
            END IF
  ELSE
C
C             DEVICE IS NOT 6:
CSLDEND
  300       WRITE(IOUT,2006)
CHHL:
CHHL 2006   FORMAT('ENTER DIFFUSED LIGHT VALUE')
 2006       FORMAT('ENTER DIFFUSED LIGHT VALUE (0.0-1.0)')
CHHLEND
            READ(IN,*,END = 300,ERR=300) DIF
            IF (DIF .GT.1.0.OR.DIF.LT.0.0) THEN
              WRITE(IOUT,2008)
 2008         FORMAT('WARNING- VALUE RANGE (0.0-1.0)')
              GO TO 300
            ENDIF
 400        WRITE(IOUT,2007)
 2007       FORMAT('ENTER REGULAR LIGHT EXPONENT')
            READ(IN,*,END=400,ERR=400) EN
 500        CONTINUE
          END IF
        ENDIF
      ENDIF
      WRITE (IOUT,2003)
 2003 FORMAT (' ENTER BACK FACE CULL OPTION',/,
     +        '   N - NO BACK FACE CULL',/,
     +        '   Y - PERFORM BACK FACE CULL',/,
     +        '   R - RETURN')
      READ (IN,'(A)',END=2,ERR=2)  BFCULL
    2 IF (BFCULL .EQ. 'R') RETURN
      IF (BFCULL .NE. 'Y')  BFCULL = 'N'
C
CSLD:
      IF (IDEV .EQ. 6) THEN
  ICOMP = SSHADE(ON)
  ICOMP = SQDRAW(OFF)
C
         IF (BFCULL .EQ. 'Y') THEN
     ICOMP = SBFCUL(ON)
CDEBUG:
CCCC          WRITE(IOUT, '( ''(HSDRVR)SBFCUL(ON) - ON='', I1)'), ON
         ELSE
     ICOMP = SBFCUL(OFF)
```

```
CDEBUG:
CCCC          WRITE(IOUT, '( ''(HSDRVR)SBFCUL(OFF) - OFF='', I1)'), OFF
         END IF
C
  IF (ISMOTH .EQ. 1) THEN
     ICOMP = SSFLAT(OFF)
  ELSE
     ICOMP = SSFLAT(ON)
  END IF
C
C          SEND ALL DRAWING ATTRIBUTES TO THE ADAGE-3000:
  CALL DIDUMP
      END IF
CSLDEND
C
      OPTION = 'READ'
C
C     INITIALIZE FOR HIDDEN SURFACE REMOVAL
      CALL DIERAS
      CALL INITPRI
C
C     LOOP THRU FOR THE NUMBER OF PARTS
      DO 100  NP=1,NUMPART
C
C
C        SKIP IF PART IS NOT TO BE DISPLAYED
         IF (LSELECT(NP) .NE. 0)  GO TO 100
C
         IERR = 0
         IF (PSTAT(1,NP) .EQ. 0.0) THEN
            IC = 1
            CALL STRCOLR (NP,IERR)
         ELSE
            IC = 0
            NFCOLR = INT(PSTAT(1,NP))
         END IF
C
         LCN = LLPNTR(NP)
   30    CALL LLNXT(LCN,NREC,LNN)
         CALL GEOFIL(OPTION,NREC(3))
C
C
         IF (IC .NE. 0 .AND. IERR .EQ. 0) THEN
            NFCOLR = NCMPCOL(IC)
            IC = IC + 1
         END IF
C
C        STORE MODEL FOR HIDDEN SURFACE REMOVAL:
         CALL STRMDL (MAXNPP,NPOINT,XKNOT,NFACE,IFACE,T,NFCOLR,
     +              INERR)
         IF (IERR .NE. 0)  THEN
            IF (INERR .EQ. 1)  THEN
               WRITE (IOUT,2011)  MAXNOD,NP
 2011          FORMAT (' ERROR - MAXIMUM NODE NUMBER LIMIT (',I5,
```

```
     +         ') EXEEEDED',/,'PART ',I5,' IGNORED')
            END IF
            IF (INERR .EQ. 2)  THEN
               WRITE (IOUT,2012)  MAXFAC,NP
 2012          FORMAT (' ERROR - MAXIMUM FACE NUMBER LIMIT (',I5,
     +         ') EXEEEDED',/,'PART ',I5,' IGNORED')
            END IF
         END IF
CSLD:
C
C        SEND THE PART TO THE DISPLAY:
         IF (IDEV .EQ. 6) CALL DIDUMP
CSLDEND
C
         IF (LNN .NE. 0) THEN
            LCN = LNN
            GO TO 30
         END IF
  100 CONTINUE
C
C     PERFORM HIDDEN SURFACE REMOVAL
CSLD  CALL HIDSRF
CSLD:
      IF (IDEV .EQ. 6) THEN
C
C        ANTI-ALIAS THE IMAGE (IF REQUESTED):
  IF (AALIAS_REQ .EQ. 'Y')  ICOMP = SFILT(0)
      ELSE
  CALL HIDSRF
      END IF
CENDSLD
C
C     PERFORM HOUSEKEEPING
CSLD:
C     RESTORE LINE DRAWING CAPABILITY:
      IF (IDEV .EQ. 6) THEN
         ICOMP = SSHADE(OFF)
C
C          SET ANTI-ALIASING AND QUICK-DRAW ON/OFF IF THEY WERE ASKED FOR:
     IF (AALIAS_REQ .EQ. 'Y') THEN
        ICOMP = SALIAS(ON)
        ICOMP = SQDRAW(OFF)
     ELSE
        ICOMP = SALIAS(OFF)
        ICOMP = SQDRAW(ON)
     END IF
      END IF
CSLDEND
      CALL DIHOME
      CALL DIDUMP
CADGSLD: REWIND IU
C
      RETURN
      END
```

```
C  FILENAME:      XBS.FTN
C  PROGRAMMER:    HAROLD LANE, JR.
C  VERSION:       1.0
C
C  EXTERNAL CALLS (FILENAME):
C                 NONE
C
C  FUNCTION:      XBS ALLOWS BOTH THE CROSS BAR SWITCH AND THE
C        CHANNEL CROSS BAR IN THE LUVO-24 TO BE SET.  A READ BACK
C        FUNCTION IS SIMULATED, BECAUSE THE ACTUAL REGISTERS IN
C        THE XBS AND CHANNEL XBS ARE NOT ABLE TO BE READ BACK TO
C        THE HOST.  EXTENSIVE VALIDATION IS DONE ON THE ARGUMENTS
C        PASSED TO XBS TO INSURE THAT REASONABLE VALUES ARE SENT
C        TO THE HARDWARE.  AN OPTION IS AVAILABLE TO SET THE
C        CROSSBARS TO A STRAIGHT THROUGH FULL COLOR DEFAULT.
C        ALSO, VALUES CAN BE SET IN THE MASTER XBS ARRAY IN COMMON
C        WITH OR WITHOUT THE ARRAY BEING SENT TO THE IKONAS DISPLAY.
C        AN ATTEMPT TO SEND ANY UNINITIALIZED DATA TO THE DISPLAY
C        IS TRAPPED AND REPORTED. UNINITIALIZED CROSSBAR BITS ARE
C        READ BACK AS THE VALUE -1.  BETWEEN 1 AND 35 BITS CAN
C        BE SET AT THE SAME TIME.  THE INTEGER*4 DATA TO BE SET
C        INTO THE CROSS BARS IS PASSED IN (OR BACK OUT) IN THE
C        ARRAY IODATA WHICH SHOULD BE DIMENSIONED WITH 35 ELEMENTS
C        IN THE CALLING PROGRAM TO BE ABLE TO RECEIVE THE MAXIMUM
C        AMOUNT OF DATA RETURNED.  THE ENTIRE XBS AND THE CHANNEL
C        XBS ARE ALL SENT WHENEVER THE WRITE MODE IS SPECIFIED.
C
```

```
C*************************************************************
 SUBROUTINE XBS(N, STRTAD, IODATA, IMODE,     IERR)
C*************************************************************
C
C        XBS CALLING PARAMETER USAGE:
C
C        N:      # OF OUTPUT BITS TO BE SET IN THE XBS OR CHANNEL XBS
C        STRTAD: 0-ORIGIN BIT # OF XBS TO BE SET. 34 TO SET CHANNEL XBS
C        IODATA: 35 ELEMENT I*4 ARRAY TO PASS DATA IN OR OUT
C                FOR XBS ADDRESSES 0-33, VALUES OF 0-33,63 ARE ALLOWED.
C                VALUE OF 63 FORCES XBS OUTPUT BIT TO 0 VALUE
C                FOR CHAN XBS ADDR (34), VALUES OF 0,21,36,& 42 ALLOWED
C                0:  RED PSEUDOCOLOR
C                21: GREEN PSEUDOCOLOR
C                36: FULL COLOR
C                42: BLUE PSEUDOCOLOR
C        IMODE:  0:  READ BACK N (1<N<35) XBS OUTPUT BIT SETTINGS
C                1:  SET ONE OR MORE VALUES INTO XBS ARRAY. NO DATA SENT
C                2:  SET 1 OR MORE VALUES IN XBS ARRAY. ALL DATA SENT
C                3:  SET XBS TO DEFAULT STRAIGHT THRU 32 BIT FULL COLOR
C                    THE DATA IS SENT TO THE IKONAS
C        IERR:   0:  NO ERROR
C                1:  # OF WORDS TO BE WRITTEN TO XBS BAD. 1-35 ALLOWED
C                2:  START ADDR OF XBS BIT(S) TO SET BAD. 0-34 ALLOWED
C                3:  END ADDR OF LAST XBS BIT TO SET BAD. 34 MAX ALLOWED
C                4:  VALUE OF IMODE BAD. 0-3 ALLOWED
C                5:  CHANNEL XBAR VALUE TO SET BAD. 0,21,36,& 42 ALLOWED
C                6:  XBS VALUE TO BE SET IS BAD. 0-33 OR 63 ALLOWED
C
C*************************************************************
C        TYPE DECLARATIONS:
C*************************************************************
C
C
 COMMON /SYSIO/  TTYIN, TTYOUT, HELPIN, PIXFIL
 INTEGER*2       TTYIN, TTYOUT, HELPIN, PIXFIL
C
 INTEGER*4 XBDATA(35)
 INTEGER*2 N, STRTAD, ENDAD, TNDAD, IMODE, IERR
C
 INTEGER*4 IODATA(35), DATA, DEFALT(35), XBSADR, CXBADR
 INTEGER*4 ADDR
C
C        INITIALLY, DATA IS SET TO THE UNINITIALIZED DATA VALUE:
 DATAXBDATA /35*-1/
 DATA DEFALT/0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,
     *           19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,   36/
C
```

```
C*****************************************************************
C        MAIN PROG:
C*****************************************************************
C
 IERR = 0

C
C        FORM 32-BIT CROSSBAR SWITCH ADDR - 302\0 OCTAL:
 CALL JOIN32(XBSADR, 194, 0)
C
C        FORM 32-BIT CHANNEL CROSSBAR SWITCH ADDR - 203\2000:
 CALL JOIN32(CXBADR, 131, 1024)
C
C        TEST FOR VALID # OF WORDS TO WRITE:
 IF (.NOT.(N .LT. 1 .OR. N .GT. 35)) GO TO 100
         WRITE(TTYOUT, 50) N
 50              FORMAT('0(XBS) # OF WORDS TO SEND TO XBS:', I6, ' BAD.',
     *           ' 1-35 ALLOWED')
         IERR = 1
         RETURN
 100     CONTINUE
C
C        VALIDATE STARTING ADDRESS OF XBS BIT TO SET:
 IF (.NOT.(STRTAD .LT. 0 .OR. STRTAD .GT. 34)) GO TO 200
         WRITE(TTYOUT, 150) STRTAD
 150             FORMAT('0(XBS) STARTING ADDRESS OF XBS:' I6,
     *           ' BAD. 0-34 ALLOWED')
         IERR = 2
         RETURN
 200     CONTINUE
C
C        VALIDATE ENDING ADDRESS OF XBS BIT TO SET:
 ENDAD = STRTAD + N - 1
 IF (.NOT.(ENDAD .GT. 34)) GO TO 300
         WRITE(TTYOUT, 250) ENDAD
 250             FORMAT('0(XBS) ENDING ADDRESS OF XBS:', I6,
     *           ' BAD. A MAX OF 34 IS ALLOWED')
         IERR = 3
         RETURN
 300     CONTINUE
C
C        VALIDATE IMODE PARM:
 IF (.NOT.(IMODE .LT. 0 .OR. IMODE .GT. 3))GO TO 400
         WRITE(TTYOUT, 350) IMODE
 350             FORMAT('0(XBS) IMODE:', I6, ' BAD. 0-3 ALLOWED')
         IERR = 4
         RETURN
 400     CONTINUE
C*****************************************************************
C
C        IS THIS A REQUEST TO READ BACK DATA?
 IF (.NOT.(IMODE .EQ. 0)) GO TO 425
C
C                YES, IT IS. ALLOW DATA TO BE PASSED BACK:
```

```
        DO 415 ADDR = STRTAD, ENDAD
                IODATA(ADDR-STRTAD+1) = XBDATA(ADDR+1)
 415            CONTINUE
        RETURN
 425    CONTINUE
C*************************************************************
C
C       SEE IF THIS IS REQUEST TO WRITE DATA INTO THE MASTER XBS ARRAY:
 IF (.NOT.(IMODE .EQ. 1   .OR.   IMODE .EQ. 2)) GO TO 625
C
C               YES, COPY DATA INTO ARRAY; REPORT & RETURN IF BAD DATA:
        TNDAD = ENDAD
        IF (.NOT.(ENDAD .EQ. 34)) GO TO 430
C
C                       VALIDATE CHANNEL XBS SETTING VALUE:
                DATA = IODATA(ENDAD - STRTAD + 1)
                IF (.NOT.(DATA .NE. 0  .AND. DATA .NE. 21
    *                   .AND. DATA .NE. 36 .AND. DATA .NE. 42))GOTO 420
C
C                               INVALID CHANNEL XBS DATA VALUE; REPORT:
                        WRITE(TTYOUT, 417) DATA
 417                            FORMAT('0(XBS) CHANNEL XBAR DATA:', I11,
    *                           ' BAD. 0,21,36,42 ALLOWED')
                        IERR = 5
                        RETURN
 420                    CONTINUE
C
C                       SET CHANNEL CROSS BAR:
                XBDATA(35) = DATA
                TNDAD = 33
                IF (N .EQ. 1)   GO TO 600
 430            CONTINUE
C
C               SET XBS DATA INTO ARRAY; REPORT BAD DATA & RETURN:
        DO 600 ADDR = STRTAD, TNDAD
                DATA = IODATA(ADDR - STRTAD + 1)
                IF ((DATA.GE.0.AND.DATA.LE.34).OR.DATA.EQ.63)GOTO500
C
C                               XBS INPUT DATA BAD:
                        WRITE(TTYOUT, 450)ADDR, DATA
 450                            FORMAT('0(XBS) DATA AT ADDRESS:', I6,
    *                           ' HAS BAD VALUE:', I11, '. 0-34,63 ALLOWED')
                        IERR = 6
                        RETURN
 500                    CONTINUE
C
C                       DATA IN RANGE; PUT INTO MASTER ARRAY:
                XBDATA(ADDR + 1) = DATA
 600            CONTINUE
        IF (IMODE .EQ. 1)   RETURN
 625    CONTINUE
C*************************************************************
C
C       SEE IF DEFAULT XBS SETTINGS WANTED:
```

```
 IF (.NOT.(IMODE .EQ. 3)) GO TO 640
         DO 630 ADDR = 1, 35
                 XBDATA(ADDR) = DEFALT(ADDR)
 630             CONTINUE
 640     CONTINUE
C*************************************************************
C
C        SEE IF DATA IS TO BE WRITTEN TO IKONAS:
 IF (.NOT.(IMODE .EQ. 2   .OR.   IMODE .EQ. 3)) GO TO 800
C
C                YES, IT IS. INSURE ALL DATA TO BE SENT IS INITIALIZED:
         DO 750 ADDR = 0, 34
                 IF (.NOT.(XBDATA(ADDR+1) .EQ. -1)) GO TO 700
C
C                                UNINITIALIZED DATA; REPORT & RETURN:
                         WRITE(TTYOUT, 650) ADDR
 650                             FORMAT('0(XBS) UNINITIALIZED DATA'
     *                           ' ASKED TO BE SENT TO XBS BIT:', I6 /)
                         IERR = 7
                         RETURN
 700                     CONTINUE
 750             CONTINUE
C
C                ALL DATA IS INITIALIZED; SEND IT:
         CALL IKBWR(0, XBSADR, 34, XBDATA)
         CALL IKBWT
         CALL IKBWR(0, CXBADR,  1, XBDATA(35) )
         CALL IKBWT
         RETURN
 800     CONTINUE
C*************************************************************
C
C        YOU ARN'T SUPPOSED TO BE ABLE TO GET HERE. IF SO, ERROR:
 WRITE(TTYOUT, 850)
 850     FORMAT('0(XBS) INTERNAL PROGRAM ERROR. NO DATA SENT TO IKONAS')
 RETURN
 END
```

APPENDIX D

SUBROUTINES LISTED BY SOURCE CODE MODULE

```
******************************
ADSUBS.FOR

        SUBROUTINE ADDRWA(XW, YW, IDASH     )
        SUBROUTINE ADDRWS(IXS, IYS, IDASH     )
        SUBROUTINE ADLINE(X1, Y1, IDASH     )
        SUBROUTINE ADMOVA(XW, YW     )
        SUBROUTINE ADMOVS(IXS, IYS     )
        SUBROUTINE ADVUPR(IXMIN, IXMAX, IYMIN, IYMAX     )
        SUBROUTINE ADWIND(XMIN, XMAX, YMIN, YMAX     )

******************************
ALPHA.FOR

      SUBROUTINE AEDIT
      SUBRQUTINE EDADD
      SUBROUTINE EDPRNT(FIRST,LAST)
      SUBROUTINE HEADR(NAME,NUMHD,STATHD,TRANSHD)
      SUBROUTINE TFORM
      SUBROUTINE EDCHNG (ID,IOPT)
      SUBROUTINE CHNGALL (ID,IOPER,NCOMPNU,PCOMPNU)
      SUBROUTINE CKASCO (IDC,IDP,MCOMP,ACOMP,INDX,IERR)
      SUBROUTINE CYCLIC (IDP,INC,NCOMPNU,PCOMPNU,IERR)
      SUBROUTINE EDDEL (ID,IOPT)
      SUBROUTINE EDREST (ID,IOPT)
      SUBROUTINE EDCOPY (ID,IOPT)
      SUBROUTINE EDDRAW (IPART)
      SUBROUTINE ANGCALC (A,B,C,ISTAT)
      SUBROUTINE CKID(ID,ICODE,INDX)
      SUBROUTINE EXIMPFN (NUMP,NSS2,NSTACK)
      SUBROUTINE CHEKATT (ID)
```

```
*******************************
CLIP.FOR

        SUBROUTINE CLIP2D(P2X, P2Y, XMIN, XMAX, YMIN, YMAX, IDASH     )
        SUBROUTINE ENDPT_CODE (PX, PY, CLIPPING_BOX,     PCODES, SUM)
        SUBROUTINE LOGIC_INTERSECT(P1CODES, P2CODES,  INTERSECT)

*******************************
CMAPLD.FOR

      SUBROUTINE CMAPLD

*******************************
CTABLE.FOR

      SUBROUTINE CTABLE (IOPT)

*******************************
CURSLD.FOR

      SUBROUTINE CURSLD

*******************************
DICTAB.FOR

      SUBROUTINE DICTAB (IOPT)

*******************************
DIDDGS.FOR

      SUBROUTINE GDINIT (IGD,IBAUD)
      SUBROUTINE PRINIT(IIBAUD)
      SUBROUTINE CMPRES (IDISK,ICHANG,IMIN,IMAX)
      SUBROUTINE DDSTOV (IXS,IYS,XV,YV)
      SUBROUTINE DDVTOS (XV,YV,IXS,IYS)
      SUBROUTINE DDWTOS (XW,YW,IXS,IYS)
      SUBROUTINE DICHRSZ (NUM)
      SUBROUTINE DICOLR (NUM)
      SUBROUTINE DICTAB (IOPT)
      SUBROUTINE DICURS (ICH,XW,YW)
      SUBROUTINE DIDRWS (XV,YV)
      SUBROUTINE DIDRWW (XW,YW)
      SUBROUTINE DIDSHS (XV,YV,IDASH)
      SUBROUTINE DIDSHW (XW,YW,IDASH)
      SUBROUTINE DIDUMP
      SUBROUTINE DIERAS
      SUBROUTINE DIHOME
      SUBROUTINE DIINQD (JDEV)
      SUBROUTINE DIMOVS (XV,YV)
      SUBROUTINE DIMOVW (XW,YW)
      SUBROUTINE DIOVRR (ITORA)
      SUBROUTINE DIPAUS
      SUBROUTINE DISCTE (NCTE,IR,IG,IB)
      SUBROUTINE DITEXT (LENS,CHARS)
```

```
      SUBROUTINE DIVUPR (XMINV,XMAXV,YMINV,YMAXV)
      SUBROUTINE DIWIND (XMINW,XMAXW,YMINW,YMAXW     )
      SUBROUTINE LOGO
      SUBROUTINE TITLE
      SUBROUTINE PCFFIO (IVW,ILINES)
      SUBROUTINE WPXLA (XV,YV,MAXR,NR,NC,IPIXEL)

*****************************
DISPLAY.FOR

      SUBROUTINE DISPLAY
      SUBROUTINE DMENU
      SUBROUTINE SPP (LINE,LSELECT,ICODE)
      SUBROUTINE DRAW
      SUBROUTINE HSDRVR
      SUBROUTINE RSETTV
      SUBROUTINE VIEW4(IFLAG)
      SUBROUTINE ZOOM
      SUBROUTINE WINDOW
      SUBROUTINE SETRNG
      SUBROUTINE AXIS
      SUBROUTINE HIDDEN (N,X,ORG,IFLAG)
      SUBROUTINE DFACE(N,X,IDASH)
      SUBROUTINE SHRINK (F,N,X)
      SUBROUTINE SAVOPT
      SUBROUTINE RSTOPT
      SUBROUTINE STRCOLR (NP,IERR)

*****************************
FBC.FOR

        SUBROUTINE FBC(IMODE, IFBNO, IWRITE, IINVBL, JINVAL, IERR)

*****************************
GETCOLORM.FOR

        SUBROUTINE GET_SLD_COLR(CURCOLR,     CLRA, CLRB, CLRC)

*****************************
GRAPHICS.FOR15

      SUBROUTINE GRAPHIC
      SUBROUTINE PICVIEW(IVW,ICODE)
      SUBROUTINE SUCES(ICNT,IVW,ITRY,CHOICE,ISAVPT,ISAVID,ISEND)
      SUBROUTINE FNDPRT(ISAVPT,NVW)
      SUBROUTINE FNDPNT(NPART,IPT,NVW)
      SUBROUTINE PICKPNT(X1,Y1,IX,IY,NVW,IPART,IPT,ICODE)
      SUBROUTINE SQUARE (X,Y,Z)
      SUBROUTINE CROSS (X,Y,Z)
      SUBROUTINE DISPNT
      SUBROUTINE TRANPRT
      SUBROUTINE MOVPRT
      SUBROUTINE MOVPNT
      SUBROUTINE EXORNEW(X1,Y1,Z1,IX,IY,IVW,IPART,IPT,IPICK,STRING,
```

```
      SUBROUTINE DISTCK(XPT,YPT,XNEW,YNEW,IVW,INRNG)
      SUBROUTINE MODPNT
      SUBROUTINE ADDPNT
      SUBROUTINE DELPT
      SUBROUTINE ADDFACE
      SUBROUTINE DELFACE
      SUBROUTINE CAPIT
      SUBROUTINE WRTGEO(NPART)
      SUBROUTINE ROTPRT
      SUBROUTINE DEGREES(X,Y,XC,YC,THETA)
      SUBROUTINE GETPNT(X,Y,IX,IY,IPT)
      SUBROUTINE LODGEO(NPART,NP)

*******************************
HSDRVR.FOR

      SUBROUTINE HSDRVR

*******************************
MASPROP.FOR

      SUBROUTINE MASPROP
      SUBROUTINE VOLUME
      SUBROUTINE DIMEN
      SUBROUTINE DIMPART
      SUBROUTINE DIMALL
      SUBROUTINE DIMNODE

*******************************
PF2GF.FOR

      SUBROUTINE PRT
      SUBROUTINE INITLL
      SUBROUTINE BUILDPF (NUMP,IOPER,NCOMPNU,PCOMPNU)
      SUBROUTINE ASTRAN(NUMP)
      SUBROUTINE PADD (NUMP,NCOMPNU,PCOMPNU)
      SUBROUTINE PCADD (NUMP,MCOMP,MCNUM)
      SUBROUTINE PPADD (NUMP,NAMP,NUMC,NER,LCNC)
      SUBROUTINE PMOD (NUMP,NCOMPNU,PCOMPNU)
      SUBROUTINE PPMOD (NUMP,NAMP)
      SUBROUTINE PDEL (NUMP)
      SUBROUTINE EXIMPF (NUMP)
      SUBROUTINE LLNXT (LCN,NREC,LNN)
      SUBROUTINE LLADD (LSN,NREC)
      SUBROUTINE LLDEL (LSN,LCN,INDX)
      SUBROUTINE LLREP (NUMP,NUMC,LCN)
      SUBROUTINE LLINS (LSN,LCN,NRECC)

*******************************
PRIMTV.FOR

      SUBROUTINE BUILDGF(NUMP,IER,INR)
      SUBROUTINE EXTRN (NPART,TP,INR)
      SUBROUTINE CUBE(NPART,L,H,W,TP,INR)
```

```
      SUBROUTINE CONE (NPART,RU,RL,H,NF,DVANGL,TP,INR)
      SUBROUTINE SPHERE (NPART,R,NH,NV,DVANGL,TP,INR)
      SUBROUTINE PARAB (NPART,R,P,NH,NV,DVANGL,TP,INR)
      SUBROUTINE TORUS(NPART,RI,RO,NV,NH,STANG,TP,INR)
      SUBROUTINE TRANSWP (NPART,PNTORG1,PNTORG2,PNTORG3,
      SUBROUTINE CROSSV (X,I,J)
      FUNCTION LNGTHV (X,Y,Z)
      SUBROUTINE ROTASWP (NPART,NS,DVANGL,NDCAP,XYCAP1,XYCAP2,TP,INR)
      SUBROUTINE ASSEM (NPART,TP,IER,INR)
      SUBROUTINE ASSMRNG(NPART)
      SUBROUTINE BOOLEN (NPART,INR)
      SUBROUTINE PRTRNG(NPART)
      SUBROUTINE RST(VECTOR,ARRAY,NPART)
      SUBROUTINE GEOFIL (OPTION,NPART)
      SUBROUTINE OPENGEO(IOPT)
      SUBROUTINE MOVORG(T,NPART)
      SUBROUTINE MVORGAS(T,NPART)

******************************
PRIORTY.FOR

      SUBROUTINE INITPRI
      SUBROUTINE STRMDL (NMAX,NN,X,N,IF,T,IPC,INERR)
      SUBROUTINE HIDSRF
      SUBROUTINE PRIORTY (ISTART)
      SUBROUTINE FOREST(NUM,DATIN,DATOUT,INDEX,ICOMP)
      SUBROUTINE SPECIAL
      SUBROUTINE MINIMAX (II,JJ,IFLAG)
      SUBROUTINE MNMX (N,X,XMAX,XMIN)
      SUBROUTINE PLANEQ (II,A,B,C,D)
      SUBROUTINE PDEEPQ (II,JJ,IALLPT,IFLAG)
      SUBROUTINE OVERLAP (II,JJ,IFLAG)
      SUBROUTINE CONTANP (X,Y,JJ,IFLAG)
      SUBROUTINE EDGMNMX (I1,I2,J1,J2,IFLAG)
      SUBROUTINE NTRSCT (A1,B1,C1,A2,B2,C2,EPS,XO,YO,IFLAG)
      SUBROUTINE SAMPLN (II,JJ,IFLAG)
      SUBROUTINE SLICEP (II,JJ)
      SUBROUTINE MIDDIV (II)
      SUBROUTINE PENTRAT (II,JJ)
      SUBROUTINE RESORT (N)
      SUBROUTINE PPLAN (II,JJ,IFLAG)

******************************
SCANLINE.FOR

      SUBROUTINE DRAWHS
      SUBROUTINE CENTER (II,CM,DIST)
      SUBROUTINE SCNFIL(IFAC,NVPF,XYI,ICOLOR,ISCAN,IBUFF)
      SUBROUTINE VTOS (X,Y,IFLAG)
      SUBROUTINE SCNINT (N,X,Y,ISCAN,XI,IE,HFLAG)
      SUBROUTINE EMAXY(ISTART,IEDGE,Y,N,ICOUNT,ILAST,INDX)
      SUBROUTINE DSPLASL (N,LINE,ISCAN)
      SUBROUTINE INTENS (RINT,CLR)
      SUBROUTINE NORAV (NODE,NNODE,XNORM)
```

```
      SUBROUTINE SMOOTH (IEDGE,NVPF,Y,RI,ISCAN,AI,BI)
      SUBROUTINE SHADER (COLOR)
      SUBROUTINE CHKNOD(NODE,NODLIST,ICOUNT)
      SUBROUTINE SL2D (N,LINE)
      SUBROUTINE INITAET
      SUBROUTINE INITSET
      SUBROUTINE SETSET(SCAN,IFAC,XINT,IEDGE,HFLAG,ICOLOR)
      SUBROUTINE NEARHOR(SCAN,J,NSET)
      SUBROUTINE SETHOR(HFLAG,XINT,ICOLOR)
      SUBROUTINE NXTHOR(NHL)
      SUBROUTINE SETAET(ISCAN)
      SUBROUTINE DRWHOR(ISCAN)
      SUBROUTINE DRWEDG(I,NAET,J,NSET,SCAN)
      SUBROUTINE CINTSCT(X1,Y1,X2,Y2,NSET,CINTX,CINTY,SCAN)
      SUBROUTINE SETINT(X1,Y1,X2,Y2,J,NSET,CINTX,CINTY,SCAN,DIST)
      SUBROUTINE GETINT(X1,Y1,X2,Y2,XF1,YF1,XF2,YF2,X,Y)
      SUBROUTINE DRWLIN(N,I)

******************************
SETOP.FOR

      SUBROUTINE SETOP (NPART,IA,IB,SOP,LLEGAL)
      SUBROUTINE SOLBOX (XYZA,XYZB,XYZS,INTFLG)
      SUBROUTINE PCENTR (MAXN,N,XYZ,C)
      SUBROUTINE FACBOX (NVPF,IFACE,MAXN,XYZ,XYZMM)
      SUBROUTINE EDGBOX (X1,Y1,Z1,X2,Y2,Z2,XYZMM)
      SUBROUTINE EIFP (X1,Y1,Z1,X2,Y2,Z2,A,B,C,D,CTR,INTFLG)
      SUBROUTINE EPINT (X1,Y1,Z1,X2,Y2,Z2,A,B,C,D,XO,YO,ZO,LFLAG)
      SUBROUTINE INTEXT (MAXN,N,XYZ,P,A,B,C,EPS,LFLAG)
      SUBROUTINE EQPLAN (NVPF,IFACE,MAXN,XYZ,A,B,C,D)
      SUBROUTINE DUPNOD (XO,MAXN,NNODE,XYZ,EPSN,NINDX,IERR)
      SUBROUTINE ORDCON (NEDGE,MAXC,NUMC,IFACE,ITAG,MAXN,XYZ)
      SUBROUTINE XCHANG (MAXC,IFACE,ITAG,INDX,JNDX)
      SUBROUTINE CRNRPT (LFLAG,MAXC,NUMC,IFACE,ITAG)
      SUBROUTINE SUBFAC (MAXC,NUMC,IFACE,ITAG,NSF,JFACE,MAXS,ITYPE,
      SUBROUTINE REDUNE (MAXC,JFACE,IFACE,ITAG,IROOT,NVPF,
      SUBROUTINE COLINR (IV1,IV2,IV3,MAXP,XYZ,LCOLIN)
      SUBROUTINE CNVXFC (MAXC,JFACE,IFACE,ITAG,IROOT,
      SUBROUTINE NNCNVX (MAXC,KFACE,ITAG,IROOT,NFL,NSF,
      SUBROUTINE ADDCON (MAX,ICONN,INDX,NFL,IERR)
      SUBROUTINE DISTSQ (MAX,XYZ,INDX,JNDX,DIST)
      SUBROUTINE REVRSE (IROOT,MAXL,LLIST)
      SUBROUTINE NUEDGE (IEV1,IEV2,MAXD,IFACE,IROOT,LFLAG)
      SUBROUTINE CONCAV (IIN,ITV1,ITV2,ITV3,MAXC,IFACE,MAXP,XYZ,LFLAG)
      SUBROUTINE CLEANN

******************************
SLDSMOOTH.FOR

       SUBROUTINE SLDSMOOTH(NN, N, IF     )
```

```
*****************************
SLDSUBS.FOR

        SUBROUTINE SODRWA(XW, YW, IDASH     )
        SUBROUTINE SODRWS(IXS, IYS, IDASH     )
        SUBROUTINE SOMOVA(XW, YW     )
        SUBROUTINE SOMOVS(IXS, IYS     )
        SUBROUTINE SEND_2_SOLID(N, X, NORMS, IDASH     )
        SUBROUTINE SLDSMOOTH(NN, N, IF     )
        SUBROUTINE SLDZWIND(ZMINSLD, ZMAXSLD     )

*****************************
SMPMISC.FOR

      SUBROUTINE SINITT
      SUBROUTINE GINITT
      SUBROUTINE SAVEGM
      SUBROUTINE SAVMODL
      SUBROUTINE SAVGEO
      SUBROUTINE TOMOV
      SUBROUTINE COLMOV(IASSEM1)
      SUBROUTINE TOPAT
      SUBROUTINE FILMAT(I,ONE,TWO,TRE,FOR,T)
      SUBROUTINE FLGCOMP(LSELECT)
      SUBROUTINE READGM
      SUBROUTINE READER(IREAD)
      SUBROUTINE RDRNEW(IREAD)
      SUBROUTINE RDGEO(IREAD)
      SUBROUTINE GETNUM(IREAD,ISTART,ICODE)
      SUBROUTINE RDMOV (IREAD)
      SUBROUTINE HELP (I)
      SUBROUTINE SMPDOC (INDEX,ICODE)
      SUBROUTINE OPENR(DESC,ICODE)
      SUBROUTINE OPENW(DESC,ICODE)
      SUBROUTINE FICLOSE(ICODE)
      SUBROUTINE RIDBLK (FILNAM,LEN,ISTART,IEND)
      SUBROUTINE ERRLOG (NUMERR,NAMROU)
      SUBROUTINE BREAKIT (IPNTR)
      SUBROUTINE STRCOMP (NUM,NCOMPNU,PCOMPNU,ICODE)
      SUBROUTINE PICWNDO(IVW,IX,IY)
      FUNCTION AMAX(A,NPTS)
      FUNCTION AMIN(A,NPTS)

*****************************
STRMDL.FOR

      SUBROUTINE STRMDL (NMAX,NN,X,N,IF,T,IPC,INERR)

*****************************
U.FOR

      SUBROUTINE UCLR (A)
      SUBROUTINE UCOPY (A, B)
```

```
      SUBROUTINE UTRNSP (A, AT)
      SUBROUTINE UMULT (A, B, C)
      SUBROUTINE UIDENT (A)
      SUBROUTINE UAPPLY (X,Y,Z, A, U,V,W)
      SUBROUTINE UINVRT (A, B)
      SUBROUTINE UPRINT (LUN, LABEL, A)
      SUBROUTINE USETAU (NAME, VAL)
      SUBROUTINE USCLC (SX, SY, SZ, A)
      SUBROUTINE USCLB (SX, SY, SZ, A)
      SUBROUTINE USCLA (SX, SY, SZ, A)
      SUBROUTINE UTRNC (DX, DY, DZ, A)
      SUBROUTINE UTRNB (DX, DY, DZ, A)
      SUBROUTINE UTRNA (DX, DY, DZ, A)
      SUBROUTINE UROTOC (IAXIS, ANGLE, A)
      SUBROUTINE UROTOB (IAXIS, ANGLE, A)
      SUBROUTINE UROTOA (IAXIS, ANGLE, A)
      SUBROUTINE UROT1C (PX, PY, PZ, ANGLE, A)
      SUBROUTINE UROT1B (PX, PY, PZ, ANGLE, A)
      SUBROUTINE UROT1A (PX, PY, PZ, ANGLE, A)
      SUBROUTINE UROT2C (OX,OY,OZ, PX,PY,PZ, ANGLE, A)
      SUBROUTINE UROT2B (OX,OY,OZ, PX,PY,PZ, ANGLE, A)
      SUBROUTINE UROT2A (OX,OY,OZ, PX,PY,PZ, ANGLE, A)
      SUBROUTINE UFIXC (I, VAL, A)
      SUBROUTINE UFIXB (I, VAL, A)
      SUBROUTINE UFIXA (I, VAL, A)
      SUBROUTINE UMAPC (A1,A2, X1,X2,   B1,B2, Y1,Y2,
      SUBROUTINE UMAPB (A1,A2, X1,X2,   B1,B2, Y1,Y2,
      SUBROUTINE UMAPA (A1,A2, X1,X2,   B1,B2, Y1,Y2,
      SUBROUTINE USCTRC (SX,DX, SY,DY, SZ,DZ, A)
      SUBROUTINE USCTRB (SX,DX, SY,DY, SZ,DZ, A)
      SUBROUTINE USCTRA (SX,DX, SY,DY, SZ,DZ, A)
      SUBROUTINE UPRMC (I, J, K, A)
      SUBROUTINE UPRMB (I, J, K, A)
      SUBROUTINE UPRMA (I, J, K, A)

*****************************
XBS.FOR

        SUBROUTINE XBS(N, STRTAD, IODATA, IMODE,     IERR)
```

APPENDIX E

VAX/VMS COMPILATION COMMAND FILE

```
$ FOR ADSUBS.FOR
$ FOR ALPHA.FOR
$ FOR CLIP.FOR
$ FOR /NOI4 /NOF77 CMAPLD.FOR
$ FOR CTABLE.FOR
$ FOR /NOI4 /NOF77 CURSLD.FOR
$ FOR DICTAB.FOR
$ FOR DIDDGS.FOR
$ FOR DISPLAY.FOR
$ FOR /NOI4 DSPLSTRTI
$ FOR /NOI4 /NOF77 FBC.FOR
$ FOR GETCOLORM.FOR
$ FOR GRAPHICS.FOR
$ FOR HSDRVR.FOR
$ FOR LINKDUMMY.FOR
$ FOR MASPROP.FOR
$ FOR PF2GF.FOR
$ FOR PRIMTV.FOR
$ FOR PRIORTY.FOR
$ FOR SCANLINE.FOR
$ FOR SETOP.FOR
$ FOR SLDSMOOTH.FOR
$ FOR SLDSUBS.FOR
$ FOR SMPMISC.FOR
$ FOR STRMDL.FOR
$ FOR U.FOR
$ FOR /NOI4 /NOF77 XBS.FOR
```

APPENDIX F

INCLUDE FILES: VMS LOGICALS REQUIRED FOR SMP SYSTEM

1) Include Files.-

```
INCLUDE 'ADAGE$SOLID:SPARMS.INC'
INCLUDE 'ADAGE$SOLID:DSPLSTDEF.INC'
INCLUDE 'ADAGE$SOLID:SMNAME.INC'
INCLUDE 'ADAGE.CMN'
INCLUDE 'AEDDUM.FOR'
INCLUDE 'AEDSTAT.FOR'
INCLUDE 'DEVICE.CMN'
INCLUDE 'HS.CMN'
INCLUDE 'LL.CMN'
INCLUDE 'PARTS.PRM'
INCLUDE 'UNITS.CMN'
```

2) VMS Logicals Required for SMP Source Compilation.-

```
$ DEFINE        ADAGE$SOLID     US2$:[ADAGE.SMP.SMP.SOLID]
```

3) VMS Logicals Required for SMP Link.-

```
$ DEFINE        ADAGE$SOLID     US2$:[ADAGE.SMP.SMP.SOLID]
$ DEFINE        ADAGE$AIDS      US2$:[ADAGE.V4]
$ DEFINE        TEK$LIB         US2$:[LIBS]
```

4) VMS Logicals Required for SMP Execution.-

```
$ DEFINE        SMP$SCRATCH     US2$:[ADAGE.SMP.SMP.SCRATCH]
$ DEFINE        ADAGE$SOLID     US2$:[ADAGE.SMP.SMP.SOLID]
```

APPENDIX G

SMP LINK COMMAND FILE: OBJECT MODULES AND LIBRARIES

```
$ SET VERIFY
$ !
$ ! COMPILE    FBC, CURSLD, XBS, & CMAPLD     USING FOR/NOI4/NOF77 !!!!
$ !
$ ! (NOTE:  Its OK to have %LINK-W-MULDEF for DICTAB, STRMDL, SLD_SMOOTH,
$ !                              HSDRVR)
$ !
$ DEFINE        ADAGE$SOLID     US2$:[ADAGE.SMP.SMP.SOLID]
$ DEFINE        ADAGE$AIDS      US2$:[ADAGE.V4]
$ DEFINE        TEK$LIB         US2$:[LIBS]
$ SET NOVERIFY
$ !
$LINK/EXE=SMP -
$ ! SUBROUTINES PUT IMMEDIATELY FOLLOWING THIS LINE ARE FOR DEBUG ONLY.
CTABLE,-
HSDRVR,-        ! GOES BACK INTO DISPLAY
GETCOLORM,-     ! THIS SHOULD GO INTO SLDSUBS.FOR
SLDSMOOTH,-     ! GOES BACK INTO SLDSUBS.FOR
STRMDL,-        ! GOES BACK INTO PRIORTY.FOR
$ ! END OF DEBUG SUBROUTINES
ADSUBS,-
ALPHA,-
CLIP,-
CMAPLD,-
CURSLD,-
DICTAB,-
DIDDGS,-
DISPLAY,-
FBC,-
GRAPHICS,-
LINKDUMMY,-     ! <--------------------THIS SHOULD BE AN AED LIBRARY
MASPROP,-
PF2GF,-
PRIMTV,-
PRIORTY,-
SCANLINE,-
```

```
SETOP,-
SLDSUBS,-
SMPMISC,-
U,-
XBS,-
!.................LIBRARY DEFS:.........................................
ADAGE$AIDS:DMAVMS,ALUVMS,-
ADAGE$SOLID:SOLID/LIB,UTLLIB/LIB,-
TEK$LIB:TEKLIB/LIB
```

APPENDIX H

SUBROUTINE STRMDL

This appendix includes a description of modifications to and listing of subroutine STRMDL. This subroutine is singled out for description because it is the SMP subroutine that modify the most in the process of integrating SMP with the ADAGE 3000. Other required changes to SMP often resulted in new routines being written so that the actual change to SMP source code usually was only an added call to the new routine.

Subroutine STRMDL required a significant amount of modification because it is the routine which interfaces the model data structures to the rendering method. Thus, when the SOLID 3000 system was added as an optional rendering method, SMP's model data structures had to be interfaced to the data structures required by SOLID 3000. Briefly, each of the model's polygon vertices and normals are stored in arrays indexed by coordinate axis (1, 2, and 3 signify x-axis, y-axis, and z-axis, respectively) and vertex number. Normals are also indexed by coordinate axis and vertex number. These arrays of vertices and normals can then be passed to SOLID 3000. To see details of this process, such as code to avoid divide by zero) see the listing which follows.

```
      SUBROUTINE STRMDL (NMAX,NN,X,N,IF,T,IPC,INERR)
C
C*          ROUTINE TO STORE ALL MODEL NODES AND FACES IN ONE PART
CSLD:
C           N.B. DEVECE DEPENDENT CODE
C
C     NMAX        = MAXNPP (MAX NODES PER PART PARAMETER IN PARTS.PRM
C                   ACTUALLY SHOULD NOT BE PASSED)
C     NN          = NPOINT (# OF VERTICES (NODES) IN THIS PART)
C     X(3,4)      = XKNOT  (VERTEX (NODE) LIST)
C     N           = NFACE  (# OF FACES IN THIS PART)
C     IF(5,MAXFPP)= IFACE  (LIST OF FACES (3 OR 4 VERTEX POLYGONS) )
C     T(4,4)      = TRANSFORMATION ARRAY TO RE-ORIENT PART FOR VIEWING)
C     IPC         = THIS PARTS COLOR. (SET BY SUBR. STRCOLR)
C     INERR       = ARRAY OVERFLOW INDICATOR = 0,1, OR 2. (RETURNED)
C
C     NPF         = # OF NODES PER FACE IN THIS FACE
```

```
C
        INCLUDE 'PARTS.PRM'
        INCLUDE 'PARTS.CMN'
        INCLUDE 'DEVICE.CMN'
CSLDEND
        INCLUDE 'HS.CMN'
C
      COMMON /TMNMX/ TXMIN,TXMAX,TYMIN,TYMAX,TZMIN,TZMAX
      COMMON /HSOPT/ SORL,BFCULL,ISMOTH,DIF,EN,DORD
      CHARACTER*1 SORL,BFCULL
      COMMON /OBSRVR/ OX,OY,OZ
CSLD: DIMENSION X(NMAX,3),IF(5,1),T(4,4)
CSLD:
      DIMENSION X(NMAX,3),IF(5,MAXFPP),T(4,4)
      INTEGER*2 COLORA, COLORB, COLORC, ON, NORMS(4,5), NRMLST, DUMMY
      INTEGER*2 ICOMP, I2NPF, OFF, LIGHT1, SCOLOR
      INTEGER*2 XLITVEC, YLITVEC, ZLITVEC, VECTOR, IBFINV
      INTEGER*2 AMBLIT, DIFUSE, RFLCTD, EXPONT, WEIGHT, WHITE
      REAL*4    XT(4,5), NORM
      CHARACTER*1 ANS
      DATA NRMLST/1/, ON/1/, OFF/0/
      DATA XLITVEC/100/, YLITVEC/100/, ZLITVEC/-500/, VECTOR/0/
C
C-------------------------------------------------------------------------
CSLDEND
C
C     CHECK FOR NODE NUMBER AND FACE NUMBER LIMITS
      INERR = 0
      IF ((NG+NN) .GT. MAXNOD)  THEN
         INERR = 1
         GO TO 999
      END IF
      IF ((NF+N ) .GT. MAXFAC)  THEN
         INERR = 2
         GO TO 999
      END IF
C
C     TRANSFORM AND STORE NODES
CSLD:
      IF (IDEV .EQ. 6)   THEN
         DO 10  I=1,NN
           NODENO = I + NG
           CALL UAPPLY (X(I,1),X(I,2),X(I,3), T,
           XG(NODENO),YG(NODENO),ZG(NODENO))
           TXMAX = MAX (TXMAX , XG(NODENO))
           TXMIN = MIN (TXMIN , XG(NODENO))
           TYMAX = MAX (TYMAX , YG(NODENO))
           TYMIN = MIN (TYMIN , YG(NODENO))
           TZMAX = MAX (TZMAX , ZG(NODENO))
           TZMIN = MIN (TZMIN , ZG(NODENO))
 10      CONTINUE
      ELSE
CSLDEND
        DO 11 I=1, NN
```

```
            II = I+NG
            CALL UAPPLY (X(I,1),X(I,2),X(I,3), T, XG(II),YG(II),ZG(II))
            TXMAX = MAX (TXMAX , XG(II))
            TXMIN = MIN (TXMIN , XG(II))
            TYMAX = MAX (TYMAX , YG(II))
            TYMIN = MIN (TYMIN , YG(II))
            TZMAX = MAX (TZMAX , ZG(II))
            TZMIN = MIN (TZMIN , ZG(II))
 11       CONTINUE
CSLD:
         END IF
CSLDEND
CSLD 10 CONTINUE
C
C        DEFINE OBSERVER POSITION
CRG      OZ = TZMAX + 0.1*ABS(TZMAX-TZMIN)
         OZ = TZMAX + 2.0* ABS(TZMAX-TZMIN)
CSLD:
         IF (IDEV .EQ. 6)   THEN
C
C           GIVE COLOR OF PART TO SOLID-3000:
            CALL GET_SLD_COLR(IPC,    COLORA, COLORB, COLORC)
            ICOMP = SCOLOR(COLORA, COLORB, COLORC     )
C
C           IF REQUESTED, DO SMOOTH SHADING:
            IF (ISMOTH .EQ. 1) THEN
               CALL SLDSMOOTH(NN, N, IF     )
               RETURN
            END IF
         END IF
CSLDEND
C
C       STORE FACES AND COLOR
         II = NF
         DO 30  I=1,N
CSLD:
            NPF = IF(1,I)
C
C           FOR EACH FACE, GIVE FACE DATA, COLOR, & NORMALS TO SOLID:
            IF (IDEV .EQ. 6)   THEN
C
C              FORM THIS FACE'S NORMAL (REQUIRED BY SOLID-3000):
C
C              1ST, COPY THE FACE POINTERS INTO FACE(J,II) FOR PLANEQ
C              IGNORING POINT OR LINE ELEMENTS:
               IF (NPF .GE. 3)  THEN
                  II = II+1
                  FACE(1,II) = NPF
C
                  DO 14  J=2,NPF+1
                     FACE(J,II) = IF(J,I) + NG
   14             CONTINUE
C
C                 LET PLANEQ COMPUTE THE EQUATION OF THE PLANE:
```

```
               CALL PLANEQ (II,A,B,C,D)
C
C              NORMALIZE EQN OF THE PLANE OF THIS FACE TO GET ITS NORMAL
C              AND CONVERT IT TO A 16-BIT SIGNED FRACTION FOR SOLID-3000:
               NORM = SQRT(A*A + B*B + C*C     ) / 32767.0
C
C              INSURE THAT THERE WILL BE NO FLOATING DIVIDE BY ZERO:
               IF (NORM .EQ. 0.0) THEN
                  A = 32767.0
                  B = 32767.0
                  C = 32767.0
               ELSE
                  A = A / NORM
                  B = B / NORM
                  C = C / NORM
               END IF
C
C              SOLID-3000 ONLY REQUIRES 1 NORM FOR EACH FACE:
C              THE NORMALS ARE INVERTED BECAUSE SMP HAS A
C              RIGHT-HANDED COORDINATE SYSTEM AND
C              SOLID-3000 HAS A LEFT-HANDED COORDINATE SYSTEM:
C
C              INSURE NO OVERFLOW WHEN CONVERTING FROM REAL*4 TO I*2:
               IF (A .GT. 32767.0) THEN
                  NORMS(1,1) = -32767
               ELSE
                  NORMS(1,1) = -NINT(A)
               END IF
C
               IF (B .GT. 32767.0) THEN
                  NORMS(2,1) = -32767
               ELSE
                  NORMS(2,1) = -NINT(B)
               END IF
C
               IF (C .GT. 32767.0) THEN
                  NORMS(3,1) = -32767
               ELSE
                  NORMS(3,1) = -NINT(C)
               END IF
C
C              COPY FACE'S POINTS TO XT(4,5) SO SEND_2_SOLID CAN GET THEM:
               DO 12 J = 2, NPF + 1
                  JM1 = J - 1
                  INDEX = IF(J, I) + NG
                  XT(1,JM1) = XG(INDEX)
                  XT(2,JM1) = YG(INDEX)
                  XT(3,JM1) = ZG(INDEX)
 12            CONTINUE
C
C              GIVE FACE'S POINTS & NORMALS TO SOLID-3000:
               I2NPF = NPF
               CALL SEND_2_SOLID(I2NPF, XT, NORMS, IDASH     )
            END IF
```

```
         ELSE
CSLDEND
            NPF = IF(1,I)
C
C           IGNORE POINT OR LINE ELEMENTS
            IF (NPF .LT. 3)  GO TO 30
            II = II+1
            FACE(1,II) = NPF
            IFC(II) = IPC
            DO 20  J=2,NPF+1
               FACE(J,II) = IF(J,I) + NG
   20       CONTINUE
C
C           "CULL" THE BACK-FACES BY TAKING DOT PRODUCT OF PLANE
C           NORMAL WITH VECTOR IN DIRECTION OF OBSERVER
            IF (BFCULL .EQ. 'Y')  THEN
C
C              COMPUTE THE PLANE EQUATION OF THE GIVEN FACE
               CALL PLANEQ (II,A,B,C,D)
               DOTPR = A*OX + B*OY + C*OZ
               IF (DOTPR .LE. 0.0)  THEN
                  PFLAG(II) = .FALSE.
                  II = II-1
               END IF
            END IF
CSLD:
         END IF
CSLDEND
   30 CONTINUE
C
C     UPDATE NODE AND FACE COUNTERS
      NG = NG+NN
      NF = II
C
  999 CONTINUE
C
      RETURN
      END
```

APPENDIX I

REFERENCES

1. Randall, D. P., K. H. Jones, et. al., "SMP - A Solid Modeling Program." NASA CR 172473, November 1984.

2. Anon., RDS 3000 User's Guide. ADAGE, Inc., No. 10-301-095-10A, Billerica, Massachusetts. March 1982.

3. Anon., BPS 32 Programming Guide. ADAGE, Inc., Billerica, Massachusetts, July 1982.

4. Anon., SOLID 3000 Users' Guide. ADAGE, Inc., No. 104174, Rev. C. Billerica, Massachusetts, August 1985.

5. Anon., IDL2 Reference Manual. ADAGE, Inc. Billerica, Massachusetts, March 1983.

6. Foley, J. D. and A. VanDam: Fundamentals of Interactive Computer Graphics. Addison-Wesley Publishing Company, Reading, Massachusetts, 1983.

General

7. Schumacker, R. A. et. al., "Study for Applying Computer-generated Images to Visual Simulation." AFHRL-TR-69-14, U.S. Air Force Human Resources Laboratory, September 1969.

REFERENCES

(Concluded)

8. Newman, W. M. and R. F. Sproull: Principles of Interactive Computer Graphics. McGraw-Hill Book Company, New York, 1973.

9. Anon., VAX-11 FORTRAN Language Reference Manual. Digital Equipment Corporation Order No. AA-D034C-TE. April 1982.

10. Anon., VAX-11 FORTRAN Users Guide. Digital Equipment Corporation Order No. AA-D035C-TE. April 1982.

11. Anon., VAX-11 Linker Reference Manual. Digital Equipment Corporation Order No. AA-D019C-TE. May 1982.

Standard Bibliographic Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-178065		

4. Title and Subtitle	5. Report Date
Display System Software for the Integration of an ADAGE 3000 Programmable Display Generator into the Solid Modeling Package C.A.D. Software	March 1986
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
R. J. Montoya and H. H. Lane, Jr.	RTI/3052/00-01F
9. Performing Organization Name and Address	10. Work Unit No.
Research Triangle Institute P.O. Box 12194 Research Triangle Park, NC 27709	11. Contract or Grant No. NAS1-17890
12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered Contractor Report 9/26/84 3/31/86
National Aeronautics and Space Administration Washington, D.C. 20546	14. Sponsoring Agency Code

15. Supplementary Notes

Langley Technical Monitor: Dariene D. DeRyder

16. Abstract

A software system that integrates an ADAGE 3000 Programmable Display Generator into a C.A.D. software package known as the Solid Modeling Program is described. The software system was designed, implemented, and tested at the Research Triangle Institute and later installed and demonstrated at the Systems and Analysis Branch's display system laboratory.

The Solid Modeling Program (SMP) is an interactive program that is used to model complex solid object through the composition of primitive geometric entities. In addition, SMP provides extensive facilities for model editing and display. SMP was developed at LaRC by Computer Sciences Corporation.

The ADAGE 3000 Programmable Display Generator (PDG) is a color, raster scan, programmable display generator with a 32-bit bit-slice, bipolar microprocessor (BPS). The modularity of the system architecture and the width and speed of the system bus allow for additional co-processors in the system. These co-processors combine to provide efficient operations on and rendering of graphics entities.

The resulting software system takes advantage of the graphics capabilities of the PDG in the operation of SMP by distributing its processing modules between the host and the PDG. Initially, the target host computer was a PRIME 850, which was later substituted with a VAX-11/785. Two versions of the software system were developed, a phase I and a phase II. In phase I, the ADAGE 3000 is used as a frame buffer. In phase II, SMP was functionally partitioned and some of its functions were implemented in the ADAGE 3000 by means of ADAGE's SOLID 3000 software package.

17. Key Words (Suggested by Authors(s))	18. Distribution Statement
Interactive graphics, computer aided design, solid modeling, distributed software system, programmable display generator, shading models	Unclassified - Unlimited

19. Security Classif.(of this report)	20. Security Classif.(of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	159	

For sale by the National Technical Information Service, Springfield, Virginia 22161

NASA Langley Form 63 (June 1985)