Summary of Research

Final Report

On
Visualization of Atmospheric
Water Vapor Data for SAGE

Grant # NAG-1-1944

For period
June 8, 1999 – December 29, 2000

Submitted to

Dr. W.P. Chu
LaRC Technical Officer
Mail Stop 475
NASA Langley Research Center
Hampton, VA 23681-2199

By

Dr. Mou-Liang Kung, Principal Investigator
Department of Computer Science
Norfolk State University
700 Park Ave
Norfolk, VA 23504

Table of Content

I. INTRODUCTION

The goal of this project was to develop visualization tools to study the water vapor dynamics using the Stratospheric Aerosol and Gas Experiment II (SAGE II) water vapor data. During the past years, we completed the development of a visualization tool called EZSAGE, and various Gridded Water Vapor plots, tools deployed on the web to provide users with new insight into the water vapor dynamics. Results and experiences from this project, including papers, tutorials and reviews were published on the main Web page. Additional publishing effort has been initiated to package EZSAGE software for CD production and distribution.

There have been some major personnel changes since Fall, 1998. Dr. Mou-Liang Kung, a Professor of Computer Science assumed the PI position vacated by Dr. Waldo Rodriguez who was on leave. However, former PI, Dr. Rodriguez continued to serve as a research adviser to this project to assure smooth transition and project completion. Typically in each semester, five student research assistants were hired and trained. Weekly group meetings were held to discuss problems, progress, new research direction, and activity planning. Other small group meetings were also held regularly for different objectives of this project. All student research assistants were required to submit reports for conference submission.

OBJECTIVES

Our objectives designed to meet the goal were to develop specific visualization tools and find the correlation of water vapor dynamics between high and low Atlantic and eastern Pacific storm activity years and El Nino Southern Oscillation years. In addition, various gridded water vapor images were plotted to provide us with new insight into the water vapor dynamics. Our results and experiences were disseminated via papers, seminars, tutorials, conference presentations, and the web. We have met our goal through attaining of the following objectives.

To Develop Visualization Tool

Use the IDL Programming Language to develop an interactive visualization tool EzSAGE to allow users to have full control of images created from SAGE II data.

To Study the Water Vapor Dynamics

Use EzSAGE to find the correlation of water vapor dynamics between high and low Atlantic and eastern Pacific storm activity years and El Nino Southern Oscillation years. In addition, exploit SAGE III capabilities should SAGE III water vapor data become available.

To Disseminate Information

Disseminate results and experiences including papers, tutorials and reviews via the Web.

II PROJECT ADMINISTRATION

1. Personnel

There was a major change in personnel. Dr. Mou-Liang Kung, Professor of Computer Science assumed the PI position vacated by Dr. Waldo Rodriguez who was on leave. The transfer was approved by Dr. William Chu. Supported personnel include: Dr. Kung (PI), Dr. Raj Chaudhury (Research Associate), Student Research Assistants: Eyad Youssef, Louay Youssef, Cyntrica Eaton, Bathsheba Farrow, Karim Muhammad, James Riddick, Lerone Banks, Joseph Patrick, Latita Pratt and Tiffany Mapp.

Former PI, Dr. Rodriguez continued to provide research advice and attended our weekly group meetings to discuss problems, progress, new research direction, and activity planning. Other small group meetings were held regularly for different objectives of this project. All student research assistants were required to submit reports in writing. Although research assistants were all undergraduate students, they were encouraged to produce scholarly paper for conference submission.

2. SciViz Laboratory

SciViz Lab is a dedicated visualization research laboratory, a part of the lab group called the BEST (Bringing Education and Science Together) Laboratory. In addition to computer peripherals (e.g. network switches, cables) and software (e.g. IDL, Dreamweaver) acquired in the past years, this grant provided the following additional computing equipment to the SciViz Lab:

- 1 SGI O2 workstations – visualization tool development
- 1 Intel-based Database and Web server – information management and dissemination
- 1 PC Notebook - presentation
- 2 Mac PowerBooks – presentation and demonstration

To complement this computing equipment, other computing resources were secured from NSU. Throughout the three years, new computers, furniture, locks, and two (2) 24 port 10/100 auto port Ethernet switches were secured and installed using the University resources. The SciViz Lab was also moved from a trailer to a new laboratory space in Room 103 of the Woods Science Building. We also

obtained other funding sources to equip the **SciViz Lab** with additional state of the art equipment. They include:

- 1 SGI Onyx2 InfiniteReality
- 1 ImmersaDesk R2
- 2 SGI Octane workstations
- 4 SGI O2 workstations (one was purchased from this grant),
- 2 Windows NT (additional hard disks were purchased from this grant),
- 2 G3 PowerMac,
- 2 printers (1 color, and 1 gray-scale)

The computers are all networked with access to Internet.

III. SUMMARY OF SIGNIFICANT ACCOMPLISHMENTS

During the past years, the following tasks were accomplished:

- Completed and updated the visualization tool, EzSAGE (Source code is in Appendix E),
- Created various Gridded Water Vapor plots to provide us with new insight into the water vapor dynamics,
- Presented our results at 1999 MUSPIN Annual User's Conference: Dr. Mou-Liang Kung (presentation), Dr. S. Raj Chaudhury (presentation) and Mr. Eyad Youssef (poster session);
- Drs. Waldo Rodriguez and S. Raj Chaudhury participated in the Panel Discussion: Earth System Science Education, ESSE/USRA User's meeting, College Park, MD, June 2000
- Packaged EzSAGE software outputs with public-domain visualization software for CD production and distribution,
- Sponsored a regional MUSPIN NRTS Workshop on Data Visualization at Norfolk State University (NSU) with three tutorials (samples are in Appendix B, C, D) given by our faculty members on this project,
- Secured other funding sources to help expand the SciViz Laboratory at Norfolk State University, a dedicated laboratory for scientific visualization research. New equipment includes an SGI Onyx2 InfiniteReality system and an ImmersaDesk R2.
- Additional research awards were received as a direct result of NASA MURED funding: MiLEN^{2}IUM (NASA PAIR): $1,180,000 [4 years 2000-04], Digital Earth: $292,000 [3 years 2000-03], and indirectly, AFOSR $160,000 for visualization instrumentation [1-yr, 2000-2001]
- Many proposals were submitted including a visualization project proposal to NIMA.

RELAVANCE TO NASA STRATEGIC ENTERPRISES

Over the past years, our team developed EzSAGE Visualization Software used for visualizing and analyzing SAGE II data from NASA. It provides insights into the study of the atmospheric water vapor dynamics and its impact on global and local climate changes. Our effort and accomplishment directly support the Mission to Planet Earth in the area of atmospheric science. The project further prepares to utilize SAGE III data when they become available in the future. However, SAGE III data are yet available at this time. The SciViz visualization laboratory houses state of the art computing equipment to conduct earth science related research.

BENEFITS TO SOCIETY

Water vapor plays an important role in the energy and water cycle processes that determine weather and climate. Detailed observations on the water vapor dynamics are crucial to the improvement of the analysis and prediction of convective storms. Our project helps us understand the long and short-term atmospheric water vapor dynamics and its impact on climate changes so that better quality of life can be resulted from harmonizing human inhabitation with the environment.

STUDENT ACHIEVEMENTS

During each academic year, up to six students were divided into three teams to work on three parts of the project. Each team was headed by a faculty leader to work on the application of EzSAGE tool and results dissemination on the web. In the summer of 2000, we collaborated with the REESS program to provide partial support to 12 students in the study of earth science using NASA data.

Student research assistants, Mr. Eyad Youssef and Mr. Louay Youssef (please see Appendix A) worked on the coding of the EzSAGE tool and made a presentation in the poster session of the 1999 MUSPIN Annual User's Conference. Ms. Tiffany Mapp completed her senior physics project titled "Stratospheric Tropospheric Exchange" in which she used the SAGE II water vapor and Ozone data analyzed using the EzSAGE software to look at the occurrence of STE in the eastern Pacific region. The identification of outlier points in the data that do not follow the expected variation of species concentration with height is the signature of STE occurrence.

Most scientific data and documents can be disseminated on the web without much effort. However, interactive science learning tools are rarely ported for web delivery, since typical methods of using embedded Java applets for sophisticated software require a great deal of learning and programming effort. Our students have created a web interface for the EzSAGE tool coded with ION Script (Research System, Inc., http://www.rsinc.com

/ion/index.cfm) , an XML based scripting language. The installation of the ION server is complete and a tutorial has been given (please see Appendix C). The ION server was also used in the CSC 450 Electronic Publishing course. However, the modification of EzSAGE tool (written in IDL programming language) for web delivery is yet completed. Students will continue on this project without additional support from the sponsor.

IV. FUTURE WORK PLANNED

1. Data Transformation and Handling Methods

Since the SAGE II data were sparsely populated over the Earth, short-term synoptic scale observations are difficult. Therefore the use of creative numerical methods to process the data are imperative. In addition to the former method of Gridding, Kriging, and Smoothing, we shall investigate other possibilities such as

- Trajectory Method: Using prediction on the assumption of smooth (continuous and differentiable) data migration
- Genetic Algorithms and Neural Network

These methods may provide approximations of synoptic scale water vapor concentrations, which would allow short-term studies. Faculty release time will be allocated to continue their investigations.

2 Interactive Visualization Techniques

EzSAGE is completed and will be released to the SAGE science team members interested on obtaining a copy. The Principal investigator of the SAGE II and III projects Dr. Patrick McCormick was provided a copy for his studies. A new course on Scientific Visualization (CSC 467) is being offered for Spring, 2001 in the department of Computer Science. EzSAGE to visualize remote sensing data will be incorporated into the course.

Virtual reality files of remote sensing data have also been produced. With the acquisition of an ImmersaDesk R2 and an Onyx2 supercomputer, utilization of virtual reality as a viable tool for remote sensing data analysis will be investigated. Funding from the School of Science and Technology of Norfolk State University will sponsor CaveLib Programming workshop (March 26 and 27 of 2001) to prepare science faculty to utilize the new facility to create immersive visualization tools.

APPENDIX A

EZ-SAGE Report

By Eyad Youssef and Louay Youssef

Over the past two semesters the ViSAGE team has been working on a graphical user interface (gui), which fully exploits the data obtained by Stratospheric Aerosol and Gas Experiment II (SAGE II). The unique aspect of this data is the vertical resolution, which allows one to see vertical changes in the atmosphere. The user can select the various types of data and view it at certain latitudes, longitudes and altitudes. Another aspect of the program is the ability to change the grid size. This is ideal when one is viewing stratospheric changes regionally. The EZ-SAGE, which evolved from the year one program Wigettest, has a variety of applications for analyzing SAGE II data, which make it both a powerful and user-friendly tool.

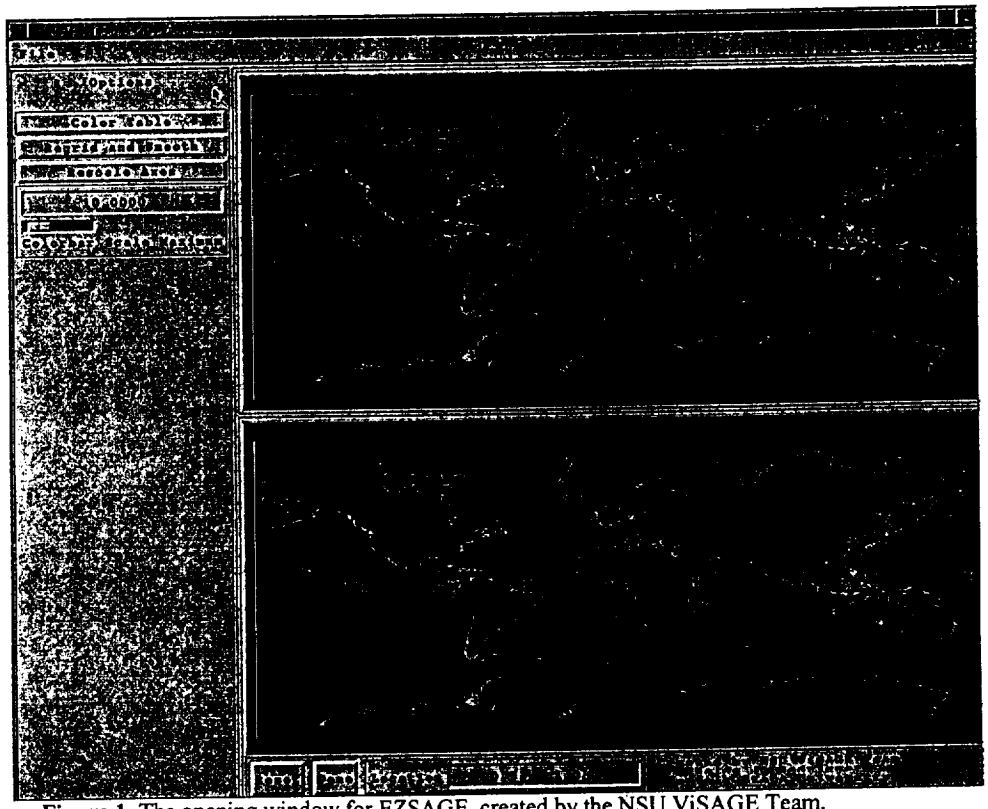


Figure 1. The opening window for EZSAGE, created by the NSU ViSAGE Team.

EZSAGE allows a user to select any of the following data sets: Aerosol Extinction at 1000 nm, 525nm, 450 nm, 385 nm, NO_2 Mixing Ratio and O_3 Mixing Ratio. The user many access the data in a variety of views, for example latitude versus longitude, altitude versus latitude, and altitude versus longitude. Once the user has selected the view, they are then able to select the units to view the altitude. Figure 2 shows all the options allowed in EZSAGE.

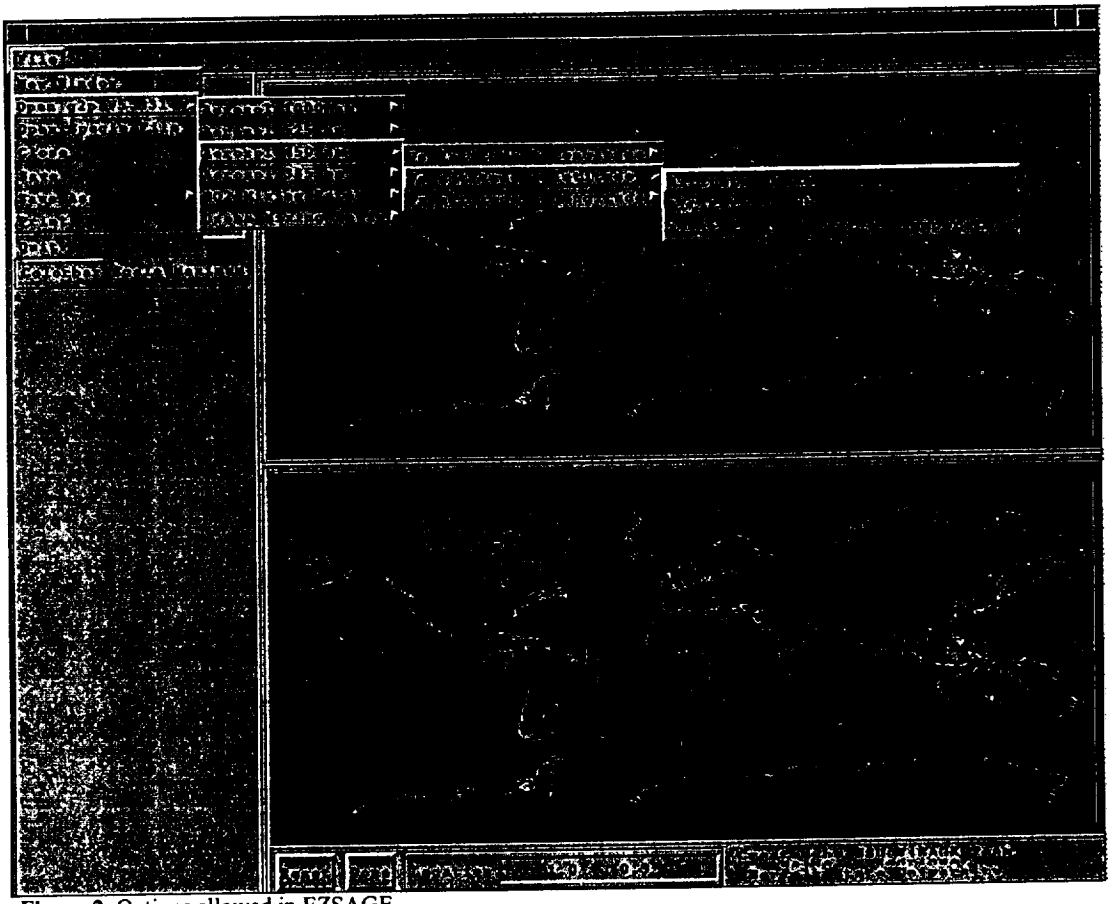


Figure 2. Options allowed in EZSAGE

After the user has selected the data to be view, the user now has an option to resize the grid size, change the color bar, change the scale on the color bar, and resize the axis. These options allow the user to view the data with high accuracy, and detail. EZSAGE fully exploits the data of the SAGE II instrument. The data viewed by this program can be manipulated to view certain atmospheric phenomena.

LinkWinds

By Bathsheba Farrow

The Linked Windows Interactive Data System is one of several visualization tools being explored at Norfolk State University. This system, also known as LinkWinds, was developed by JPL in the ANSI C language. LinkWinds uses a data linking paradigm, one of its most distinguishing features, providing a highly intuitive, easy-to-learn and easy-to-retain user interface. Data linking allows the user to link applications together, giving the applications greater capabilities and power than they would alone. LinkWinds provides interactive 2-dimensional and 3-dimensional graphical displays of data, interactive color manipulation, animation creation and displays. LinkWinds is also able to produce hard copies of graphical displays and text and perform interactive data subsetting either at the input or output. Additions to the list include journal and macro capability, context-sensitive help and network support for collaborative data.

LinkWinds accepts rectangularly gridded data. The datasets have different ranks ranging from 2 to 4. Any rank higher than 4 can not be read by LinkWinds at the present time. Therefore, if a dataset does have a rank higher than 4, only the first 4 dimensions encounter are used. The user should also beware of the fact that only scalar data is currently accepted. The user can add new databases into LinkWinds. However, there are several things may have to be done to successfully add a database. First, the lw.config, which lists the directories which LinkWinds will search, must be changed. This will not need to be edited if data is placed in a directory that is already listed. Next, Databases, found in the top-level LinkWinds "databases" menu, must also undergo alterations unless wildcards have been used correctly. The .db file defining the metadata should be created if LinkWinds can not extract enough information from data in the standard format you are using. The formats accepted by LinkWinds include the Hierarchical Data Format (HDF), HDF-EOS, the Common Data Format (CDF), Network Common Data Format (NetCDF), Silicon Graphics, Inc. native RGB image format, data with Planetary Data System (PDS) headers, the Flexible Image Transport System (FITS), two data formats of UARS, the Upper Atmosphere Research Satellite and SAGE. Raw binary data and ASCII text is also accepted but each must be accompanied by a minimal amount of metadata given in an associated .db file.,

What makes LinkWinds a great visualization tool is the fact that is has so many tools within its environment. Each tool, or application, has its own unique function and combines with other tools to rapidly and interactively access, display and analyze large datasets. *AxisRotator* is one of those tools. It rotates three dimensional applications about any three orthogonal axes. The *LightTool* adds simulated lighting to three-dimensional applications. The *Animator* has the ability to make time-based animations while the *FrameAnimator* makes animations by displaying a user specified number of frames of an application. LinkWinds also has a *ColorTool* that allows the user to alter the palette of a database or to substitute a new pallete for the current one. *Image* displays a single slice of data from any of the three orthogonal axes or a composite RGB slice taken from three slices. *Plane* renders an image in relief with an accompanying height

filed. *VolumeView* displays all the points in an entire data set, rendering both with color and opacity. These

References: http://linkwinds.jpl.nasa.gov

Medium and Long Term Climate Change Predicting

By Cyntrica Eaton

I. INTRODUCTION

Predicting medium and long term climate change has been a topic of significant interest for many years. Unquestionably, the ability to make such predictions would be a powerful tool given the impact of climate change on food supply, energy allocation, and water resources. Satellites have been extremely instrumental in the study of climate change by providing global coverage in short periods of time as well as an extensive data collection. Most scientists currently interpret satellite data using two and three-dimensional graphs which may not fully exploit the capabilities of the available data. Recent increases in computing power allow visualization in more dimensions, which in turn would facilitate the detection of related trends that may otherwise be missed.

Water vapor plays a fundamental role in the energy and water cycle processes that determine weather and climate. Detailed water vapor observations, for example, are essential to the improved analysis and prediction of convective storms. The primary focus of the Visualization of Atmospheric Water Vapor Data for SAGE (ViSAGE) project is to study the long and short term atmospheric water vapor dynamics and its impact on climate changes. To facilitate this study, multi-dimensional scientific visualization procedures for analysis and interpretation of the Stratospheric Aerosol and Gas Experiment II (SAGE II) water vapor data are being developed. We intend to use the resulting images and animations to make conclusions regarding water vapor activity and its related trends.

II. METHOD

The raw data, which was collected by SAGE II between 1986 and 1990, was obtained from NASA Langley Research Center. We were particularly interested in observing monthly data spanning the 6.5 to 30.5 km altitude range. This data was sorted, reformatted, and averaged for ease of visualization by an in-house computer algorithm. The program opens the raw data file and selects only the desired segment to be analyzed. It then arranges the data into a water vapor mixing ratio versus latitude and longitude table, which allows reading by the gridding routine.

Data Conditioning

In order to create three-dimensional images, gridding, kriging, and smoothing of the formatted data had to be performed. Transform™ , software distributed by Fortner Research LLC, assisted in this portion of our work.

Gridding

Due to the nature of the solar occultation technique and the satellite orbit, the latitude and longitude axis of the data do not fall in a uniform grid. Visualization tools generally require the raw data to be arranged into gridded bins. Each bin, which is a longitude-latitude box, is set to intervals appropriate for the data of interest. If more than one point falls within a box, the points are averaged to get one value per box. Gridding generally produces missing fields. The missing data is interpolated with the information from the surrounding bins by the kriging technique.

Kriging

Kriging is a powerful method for interpolating missing fields, which calculates the optimum weighting function and cut off radius at every point. This method estimates the missing data by using a variance curve as a weighting function. By estimating the variance at every location the routine assigns appropriate weight of the point for further extrapolation. During the validation of our visualization technique a linear interpolation with a row and column range of 350 and 150 passes were the optimal kriging parameters. The images produced using these parameters are comparable to those on the SAGE II CD-ROM [4] and to images found in published results [5].

A smooth fill of the kriged data yields smooth images. A smooth fill is a process where the kriged data is smoothed by using the average of its surrounding neighbors. In order to keep the images representative of the data, only a single pass smooth was performed.

Three-Dimensional Generation

Water Vapor Mixing Ratio vs. Latitude and Longitude

After gridding, kriging, and smoothing the water vapor concentration versus latitude and longitude data were overlayed over a cylindrical map projection to produce the images in Figure one. The false color represents the monthly average water vapor mixing ratios from 1986 to 1990. The scale is shown on the color bar below the images. The color pattern is a 16-bit rainbow with a minimum value of 1 part per million water vapor (ppm) and a maximum value of 9.5 ppm.

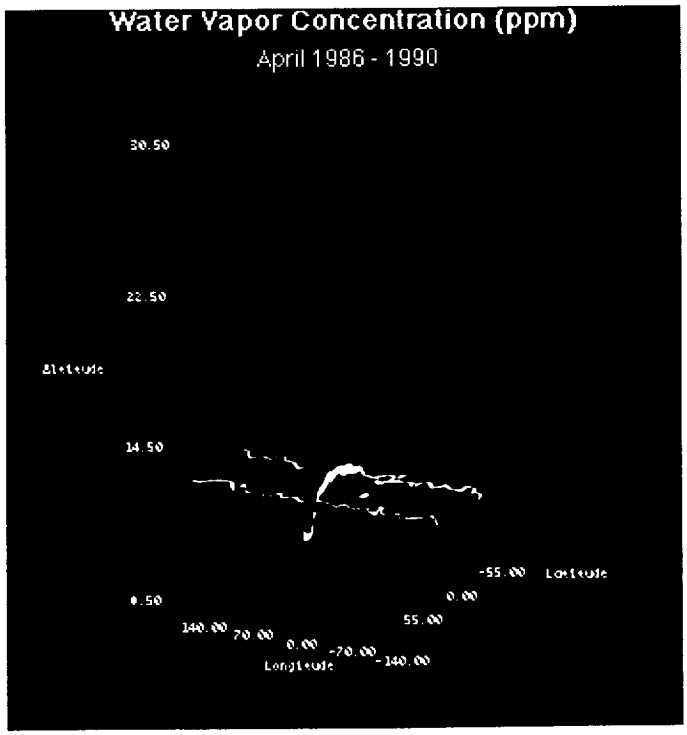
Water Vapor Mixing Ratio vs. Altitude and Month

The plates in Figure 2 represent the water vapor mixing ratio vs. altitude and month for latitudes of 30 to 40 degrees north. This figure shows the movement of the hygropause over the span of a year. To produce these images, the data were sorted and reformatted into a table of water vapor mixing ratio vs. altitude and month for the latitude range of interest using an in-house application. Since there is no missing data in the altitude dimension, kriging and smoothing is not necessary. However, to get a smooth image some interpolation was needed. The interpolation averages the surrounding points to

produce a smooth data set. Image generation was performed using Transform ™ form Fortner LLC. The false color represents the water vapor mixing ratios with the scale shown on the color bar below the images. The color pattern chosen was a 16-bit rainbow with a minimum value of 2 ppm and a maximum value of 12 ppm. This scale clearly shows the hygropause dynamics.

Four-Dimensional Image Generation

Once three-dimensional images were generated for the 25 different altitudes between the 6.5 – 30.5 altitude range for each month we were able to create a four-dimensional image of the atmosphere. Slicer™ software, also distributed by Fortner, took each of the 25 slices and aligned them vertically in the order of increasing altitude. It then allowed us to take horizontal "slices" of the four-dimensional "block" it initially created so that we could observe water vapor patterns at specific attitudes. Slices have to be kept at a minimum to ensure that proper visualization was not inhibited. As a result, five altitudes, 8.5 km, 12.5 km, 17.5 km, 21.5 km, and 25 km, were chosen. A vertical slice was also taken at 0 km longitude. The exceptional aspect of this four dimensional image is the fact that figures can be viewed at the same time and in relation to each other.



Animations

After "slicer" images were produced for each month, we had the components needed to create an animation or movie that could help us observe patterns of water vapor in relation to time. To create these movies we utilized MoviePlayer™ . We generated the January slicer image and saved it as a PICT file. The first image must be stored in this fashion in order to import it into MoviePlayer™. From there however, we only needed to copy and paste the following months into the MoviePlayer™ window.

III. RESULTS

The altitude resolution of the SAGE II instrument facilitates observation of the upper troposphere and lower stratosphere. The animations created further exploit this factor, showing the dynamics of water vapor in these regions over the course of a year. One of the trends observed is the movement of the high concentration >8.5 ppm band throughout the seasons. In particular, it appears to hover around warmer regions. The band, can be found in the Southern Hemisphere during January, a southern summer month, and is found in the Northern Hemisphere by July, a Northern summer month.

The third tier of the Slicer images found in the Results section of this poster seem to turn a lighter shade of blue as the months go on. What it is actually showing is that the hygropause, a dry region of the atmosphere located between the troposphere and stratosphere(?)

IV. CONCLUSION

The advancements in computer technology have facilitated rapid mulit-pronged analyses of satellite data. Atmospheric water vapor data from the SAGE II instrument have been utilized to obtain 3 and 4 dimensional images, animations, and movies which dramatically reveal water vapor patterns in the atmosphere. Sorting, reformatting, kriging, smoothing, and image generation were techniques utilized to get an optimal image. These images were used to make yearly movies showing the dynamics of the hygropause as well as stratospheric and tropospheric water vapor concentration. Animations created from these four-dimensional images show the horizontal and vertical dynamics of atmospheric water vapor, giving scientist and researchers a valuable tool to study global trends which may not otherwise be revealed.

HDF File Viewer

By Karim Muhammud

In order to view a HDF file, it must first be compile. After compiling the HDF file, type the view_hdf into the idl system prompt. When doing is a main menu pops up on the screen. It displays a selection of HDF files. I select one of the files from the pop up menu and open it with attribute, which allow me to view both the SDS and Vdata in the HDF file. After selecting the file from the menu I click the "OK" button. The "filter" button is used to refine the list of file names that is displayed. The filter button allows me to select a file that begins with "CER_BDS". After the file is selected, view_hdf opens the HDF file and search for the names of all the SDS and Vdata structures. Then it lists them in the SDS and Vdata sets areas respectively. Before any data can be accessed, the data must be imported to the Current Subsets Area. For example, select "CERES TOT Filtered Radiance's upwards". A subset Data window pops up with starting, ending, and increment fields for each dimension of the SDS array. A range of data can be selected. The Increment field can be used to selectively sub sample each dimension. For example, the first dimension is for the number of records, and the second is the number of samples in each record. An increment integer n causes every n^{th} record to be selected. After finishing the input, click the "Done" button. The data are imported, and the SDS name will show on the Current Subsets Area. For importing Data, the procedure is slightly different from SDS because the fields in each Data are selected individually. The procedure is to select Data. Click on the desired Data set to import from the Data sets area. For example, select "Converted Temperatures". Select the field from the window by clicking the button next to the field name. Then select "WN Black Body Temperature". A subset Data window pops up with starting, ending, and increment fields. A range of data can then be selected to import. The increment field can be used to selectively sub sample each dimension. To input range. Input the starting and ending record numbers and the increment value for data importing. After finishing the input, click the "Done" button. The data are imported, and the Data field name will show in the Current Subsets area. After the data are imported, click on the name of the data in the Current Subsets. When clicking on the name current subsets automatically pop up the graphics menu. For example, select a "2D Graph" and click the "Done" button. A window with the plot pops up on the screen.

Colors Menu

This menu includes the options for selecting different color tables, editing and saving the color map, and changing the background and foreground colors. Options are used to select colors. When selecting the predefined color tables, the current color table is displayed and a list of available predefined color tables is given. Click on the name of a color table to load the color table. For example, the table "Rainbow + white" is useful for many CERES data products. Click the "Done" button when selection is completed. Edit the current color table. If this option is selected, all the colors in the current color map will displayed. Click on the color index or move the slider to select the color map to

display. Then click on the color index or move the slider to select the color index to edit. An RGB slider window will pop up for adjusting the RGB values. A small sample color square will show on the lower left corner of the select edit colors window. Click the "Done" button on the edit color cell window after adjusting the color. Click the "Done" button on the select edit colors window for the current session without saving to the color table file. Click the "save" button to save the modified color table file for later use. If the "save" button is selected, a window pops up entering the name of the new color table. After entering the name of the new color table, click the "Done" button to save the new color.

Background color

Value zero means the background color is the colors at the index zero in the current color table. Click the "Done" button to use a new value, or the "cancel" button to cancel.

Foreground color

The foreground color is the color table index of the default color to display text or graphics on the screen. The foreground color should not be the same as the background color. Click the "Done" button to use a new value, or the "Cancel" button to cancel.

Right now, I am in the process of viewing over other HDF formats that are inside of idluser. When I come up with something I will surely write it down in my final report.

APPENDIX B

IDL Programming Language Tutorial

Dr. Mou-Liang Kung,
Department of Computer Science,
Norfolk State University, Norfolk, VA 23504
MLKUNG@NSU.EDU

What Is IDL?

IDL (Interactive Data Language) is an integrated software development system for data handling, analysis and visualization from Research System, Inc. (ww.rsi.com).

How Much Does It Cost?

There are two versions: a Student version ($79/copy for PC and Mac) and a Professional version (educational site license, not for production sales is around $200/seat/year). There is a free demo version that can be downloaded directly from RSI (ftp.rsinc.com). As expected, the student version has limited capability (in array sizes, scientific data file formats, etc.) and yet a very good learning tool. The demo version has a time limit, useful only as a demo tool. The on-line Help is a very useful feature. Other important sources include:

Is It All Programming?

IDL software came with an IDL application called **Insight.** Insight is an application (not available for student version) that runs under IDL environment for analyzing, visualizing, importing/exporting, conditioning and generating data. No IDL programming is needed. It does Line plot, scatter plot, Histogram plot, Polar plot, Contour Plot of a 2-dimnl array, Images (2-D array), Surface (2-D array)

Data File Handling

The file formats that IDL can read and/or write include: BMF, GIF, JPEG, TIFF, CDF, netCDF, HDF, ...

IDL Applications

- Line Plot, Surface Plot, Contour Plot in 2D or 3D
- Surface of Revolution
- Display Data or Graphics on 15 different types of Map Projections (e.g. Mercator, Orthographic, ...)
- Image and signal processing (Display and Manipulation: Smoothing, Noise-removal, ...)
- Animation (image or graphics data)
- Spin or fly-through a surface, Shade and illuminate with multiple light sources

The Language

- IDL Variables: character strings NOT begin with digits or $, contain no characters such as "." or " %"
- IDL Data Types:
  - ⇨ **scalar**: e.g. a = 0.0; b = 'Hello'; c = fix(5.8) (truncate to integer)
  - ⇨ **arrays** (up to 8 dimensional): a = intarr(3,4,5) , a 3-dimnl integer array of 3, 4, 5 elements in each dimn; b= fltarr(10, 20), a 2-dimnl array of floating points
    You can generate index automatically: **a = indgen(100)** generates an array a of 100 items with values 0 to 99. **b = findgen(100)** for floating 0.0 to 99.0.
  - ⇨ structures: **student = { name: 'Jim', ssnum: 12345}**
  - ⇨ pointers, objects
- IDL has the look of a procedural language (e.g. Pascal). The control statements should be very familiar to all programmers: **Begin ... end; If ... then ... else; For ... endfor, While and Repeat .. Until** loops; **Case ... of; GOTO.** It even supports **Pointers** for dynamic data structure (e.g. linked lists). IDL also has object-oriented construct that let users encapsulate data and routines into **objects.** Sophisticated object graphics can be created by instantiating atomic graphics objects from the atomic classes (e.g. axis class, image class, ...), then container objects for rendering.
- The IDL also comes with a GUI toolkit to create GUI (with buttons, slider, ...) for the data visualization application. In the Windows and Mac versions, rapid development environment (RDE) a la Visual Basic provides drag and drop capability.
- The Math and Statistics library provides routines for Data Interpolation (e.g. Linear, Cubic Spline), Random Number Generator, Curve (Least Square) and Surface Fitting, Multivariate Analysis (Cluster and Principal Components)
- It provides IDLMiner, an ODBC interface for IDL to access DBMS
- There are also interface to call other programs created in C, FORTRAN, ...

Internet Resources on IDL

Steve's IDL-WWW casa.colorado.edu/~spenton/idl_www/idl-cmd.html
*(You don't need to buy IDL to try and learn IDL, just your Web browser will do. The **ION**, IDL On Internet, is another software package from RSI that uses Java to IDL capabilities to Web Browsers.)*

RSI Site **www.rsinc.com**
(A good place to start)

U of Washngtn IDL http://www.astro.washington.edu/deutsch/idl/index.html
(A good site to search ready to use procedures or functions)

IDL Tutorials
Boston U scv.bu.edu/SCV/Tutorials/IDL/idl_webtut.html
NCSA zonker.ncsa.uiuc.edu/docs/viz/Idl/

The IDL FAQ www.ivsoftware.com/pub/idl_faq.html
(Get it and print it out)

Tips and Tricks: www.dfanning.com
(Good tips)

The newsgroup **comp.lang.idl-pvwave**
(PVWave was a derivative of IDL and became a separate development. Comp.lang.idl became the forum for discussing CORBA's IDL - Interface Definition Language)

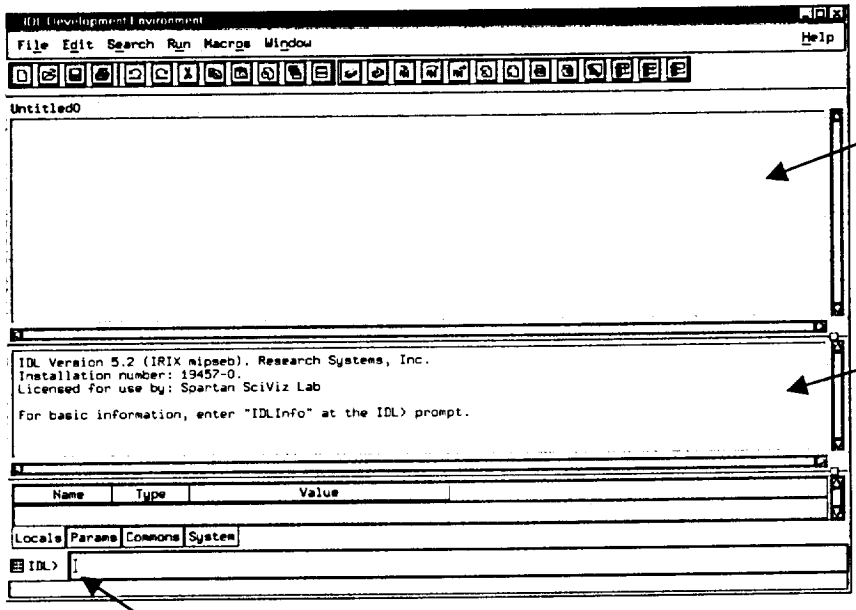
A Quick IDL Tour

(Based on the "Learning IDL, V5", RSI, 1997)
Annotated by Dr. Mou-Liang Kung

1. How to Start IDL

idlde icon or type **idlde** (in a Shell window) to start the IDL Development Environment.

Skip this step, IDL has been started for you



Editing Area

Status Console

2. How to Configure IDL

File -> Preferences

- **Check the box to General -> Change Directory on Open**
  (to change default directory to where the file was opened)
- **Check the box to Graphics -> Bitmap Buffered**
  (to refresh the graphics when covered)
- **Leave Editor and Startup unchanged**
- **Click Path to add any search path.** Click on the box to include the search on all subdirectories as well.

2.1 IDL is a command language (despite what manual says, it is non-case sensitive, except for UNIX file names).

- You can enter interactive commands at the command prompt "**IDL>** ".
- If a command line gets too long, type **$** at the end to mean to continue on the next line.

Note: The next three sections 2.1, 2.2, 2.3 on entering commands interactively are "quick and dirty" methods, not good programming practices

You can enter command lines at the IDL> as follows

Example:

IDL> print, 5+6

(The screen showed **11**)

IDL> print, 'Hello'

(The screen echos **Hello**)

Note: You will not be able to execute these two statements in sequence again, although you can take a peek at what you typed by typing **help, /recall**

Example:

IDL> help, var_name

(to check data type of the variable)

IDL> ?

(to start winhelp)

Example:

IDL> data = randomu(0.123, 50, 100)

; Replace 0.123 by **seed** if system random seed is used.

; Here is A two dimnl array of 50 X 100 uniformly distributed

; random numbers. Use **randomN** for normally distributed random num.

IDL> print, data[10, 3] ; column 10, row 3

IDL> print, data[310] ; 310 = 3 X 100 + 10

IDL> plot, data[2, *] ; plot in memory

IDL> wshow ; Visualize

IDL> subdata = data[10:30, 15:18] ; subdata is a subarray with indicated ranges

IDL> help, subdata ; print out the content of the variable

Example:

IDL> num = [6, 3, 9, 2, 8, 0]

IDL> I = [0, 3, 1]

IDL> print, num[I] ; use I to index num

6 2 3

Example:

IDL> student = { name:'Jim', ssnum=12345}

IDL> print, student.name ; return the field value of "name"

IDL> print, n_tags(student) ; return the number of fields

IDL> print, tag_names(student) ; return field names

2.2 **You can group the interactive commands** as you type so that those commands can be executed a s a group again

Type **.Run** at the beginning, the IDL> prompt will be changed to - (a dash) and you may begin entering IDL commands. When you type **End,** the command input ends and you will get back to IDL> prompt.)

Example:

IDL> **.run**

\- **print, 5+6**

\- **print, 'Hello'**

\- **end**

IDL>

2.3 An IDL script file (no pro, no end) can be created interactively using "journal" command. ***(A technique not too often used)***

Use the "journal" command before and after you enter the same command lines from example 1 at the IDL> prompt

Example 3:

```
journal, 'myscript'              ;Create a new script (choose your own name) from
                                 :the following input:
num = fltarr(40)                 ;num is an array of 40 floating point numbers
for i = 0,39 do num[i] = i * 10  ;initialize these 40 entries
curve= sin(num * !DtoR)          ;change degrees to radians before calling sine function
plot, curve                      ;plot
journal                          ;No more input. The end of the file
```

- Using the journal method to enter the command line, the resulting file will contain all input errors.
- To execute an IDL script, enter the following at the **IDL>** prompt: **@myscript**

Note: Up till now, the interactive command entering method does not allow you to edit out the errors!!! Here is the better way:

2.4 Create an IDL program using the built-in editor (or any other text editor). You must name it with **.PRO** file extension.

Example 4 **(pp.22) On the menu bar near the top, click:** File -> New **and now you can type in the following code in the** editing area **(see the figure in the previous page)**

```
                                 ;This is a comment
num = fltarr(40)                 ;num is an array of 40 floating point numbers
for i = 0,39 do num[i] = i * 10  ;initialize these 40 entries
curve= sin(num * !DtoR)          ;!DtoR changes degrees to radians
plot, curve                      ;plot
end
```

Click: **File -> Save As,** and give it a name, say **your_name.pro.** Now your can click: **Run -> Compile your_name.pro** (same as typing **.COMPILE your_name.pro)**and followed by **Run -> Run your_name.exe** (same as typing **.GO)** to execute.

Example 5: (2-D plot, **pp.24-25) Please type only the commands in** bold. **You don't need to type the commented lines to work. (I have included alternative code that may accomplish the same in the commented commands for you to try out if time permits.)**

```
num = fltarr(40)                 ;Defining array of 40 floating points
for i = 0,39 do num[i] = i*10    ;Initialize num
;num=findgen(40)*10              ;Same as above, Floating Index Generator,
                                 ;magnified 10 times
;for i = 0,39 do print, num[i]   ;Can be used to verify what're stored
curve = sin(num * !DtoR)         ;DtoR to change degrees to radians
```

```
;plot, num, curve, XRange=[0,360]   ;Specifying range of X does not clip the graph
;plot, num, curve, psym=1           ;psym set to 1 (yes) to use plus symbol to mark points
                                    ;Can use  LineStyle = 1  instead to plot dashed line
;Xtitle='Degress', YTitle ='Sine(x)', Title = 'Sine plot'  ;Add labels separately, or
plot, num, curve, XTitle='Degress', $  ;combine the last two with"$" as the continuation mark
            YTitle ='Sine(x)', Title = 'Sine plot', XStyle= 1
polyfill, num[0:18], curve[0:18], color = 120      ;Fill area bounded by x-axis from 0 to
                                                   ;18 and the curve
polyfill, num[18:36], curve[18:36], color=50       ;Use different color for other region
```

Again, click: **File -> Save AS** and name it your_name.pro. You can just click **Run -> run** your_name.**pro.** (The system will automatically try to compile it for you before execute.)

Example 6: (3-D Plot, **pp.26 -28) Click:** File -> Open -> muspin .pro **and you will see the following content loaded:**

```
file = Filepath(Root_dir='/usr/people/idluser/training', 'surface.dat')
                                ;Specify the data file to read
OpenR, lun1, file, /Get_LUN     ;Open the file, assign it logic unit called lun1
peak =fltarr(40,40)             ;Define 2-D array called peak
ReadF, lun1, peak               ;Read in 2-D array from the file
Free_LUN, lun1                  ;Close the file
Print, Max(peak), Min(peak)     ;Max and Min value of the data set
x = findgen(40)*2.5             ;X tic values are 0 thru 100 = 40*2.5
y = findgen(40)+50              ;Y tic values are 50 thru 90
                                ;Draw Wireframe now
surface, peak, x, y, Xtitle='X axis', Ytitle ='Y axis', Title='Surface plot', $
        background=200, color=45, charsize=2.0
                                ;Now we can put a snow cap on the mountain peak
file=Filepath(root_dir='/usr/people/idluser/training', 'surftop.dat')
OpenR, lun2, file, /Get_lun     ;Open the file
snow=fltarr(40,40)              ;Define a 2-dim'l array of 40 by 40 floating points
readf, lun2, snow               ;Read the data file into the 2-dim'l array
free_lun, lun2                  ;Close the file
window, 1                       ;Open another window
surface, snow                   ;Draw  snow cap data in a separate window
window, 2                       ;Open another window
shade_surf, peak, shades=snow   ;Put snow cap data on top of peak
sub1                            ;Call a subroutine called  sub1.  See section 4. next
end                             ;I had placed Example 1 in a procedure called sub1
```

Note: The following code is the **WRONG** way to read in arrays from a file :

```
        x = fltarr(100)
        y = fltarr(100)
        for i, 99 do begin
                readF, lun, x[i], y[i]      ; parameters are pass-by-value
        endfor
```

The **CORRECT** way is:

```
        for i = 0, 99 do begin
```

```
        readF, lun, Tx, Ty        ; you pass the addresses of arrays
        x[i] = Tx;
        y[i] = Ty
endfor
```

4. **An IDL program can call IDL procedures or functions (subroutines).**

- The subroutines must precede the main program code in the same file named with **.PRO** extension.
- If a subroutine is placed in a separate file, then the file name must be identical to the subroutine name.
- An IDL program may consist of multiple modules (procedures or functions). However, only one can be the main program.
- If there are subroutines to be called) by another subroutine, then called subroutines must precede the calling the subroutine that bears same file name.

- **Subroutines can be compiled and executed**

To compile procedure: **IDL> .COMPILE proc_name**
Then you can call it: **IDL> proc_name**
To compile function: **IDL> .COMPILE fcn_name**
To use the compiled function: **IDL> num = fcn_name(value1, value2, .., valuen)**
IDL> print, num

4.1 IDL Procedures

```
Pro    proc_name, param1, param2, ..., paramn
       Idl_cmd
       ...
                                        END
```

Example:

IDL> window
(go to window #0)
IDL> wshow, 3
(to bring the window #3 to the front)
IDL> wshow, 0, 0
(to hide window #0)

Use the editor to create the following procedure that takes one input parameter:

```
pro whide, win
        wshow, win, 0
        return
end
```

Now, save it as whide (identical name). Compile it and later you can use it to hide any window (e.g. IDL> whide, 2 to hide window #2)

Example: There are other window related options

```
IDL> window, 0, xsize=400, ysize=400, xpos=100, ypos=100, title='Test'
IDL> window, /free              ; get the next free window number
IDL> win = !d.window    ; get current window number
IDL> window, /pixmap    ; draw window in memory only. This is a way to
                                ; store multiple images for play back in an animation
                                ; display zoom in/out to a fixed window
```

Example: Here is to create an animation from multiple images:

```
pro imgplay
    file=Filepath(root_dir='/usr/people/idluser/training', 'abnorm.dat')
    OpenR, lun3, file, /Get_lun      ;Open the file
    image = Assoc(lun3, BytArr(64, 64))
    window, 0
    ; for i = 0, 15 do TVScl, image[i]        ; image too small, no wait, too fast
    for i = 0, 15 do TVScl, congrid(image[i], 400, 400) & wait, 0.1
end
```

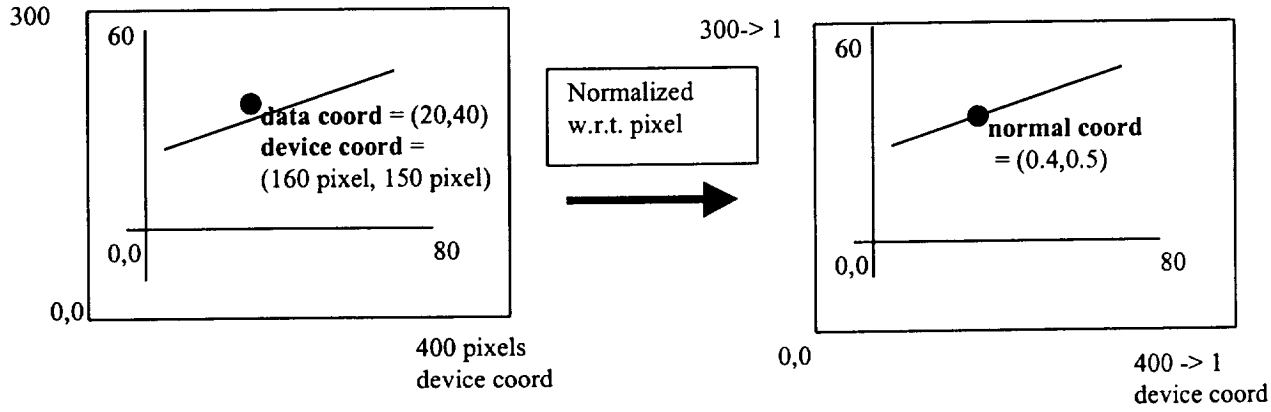
4.2 IDL Functions

```
Function        fcn_name, param1, param2, ..., paramn
        idl_cmd
        ...
        return, value
END
```

4.3 To compile a program and all called subroutines automatically:

- Make sure the directory name where subroutine files are kept is in the **!PATH** system variable
- Names of subroutines are identical to filename used. Lower case used in UNIX.

5. Plotting



Example:

```
X = FINDGEN(11)/10.        ;Make a vector of X values.
D = CONVERT_COORD(X, 2*X, X^2, /T3D, /TO_DEVICE)
        ; Convert the coordinates. D will be an (3,11) element array.
        ; The keywords TO_DATA, TO_DEVICE, and TO_NORMAL
```

Example: Display strings at specific spot of the window

```
xyouts, 20, 30, 'Test,' /data
xyouts, 160, 150, 'Test', /device
xyouts, 0.4, 0.5, 'Test', /normal
```

Example:

```
x = findgen(360) - 180        ; generate index 0 - 180 thru 359 - 180
y = sin (findgen(360)*!DtoR)
plot, y                       ; plot y against 0 -> 359
plot, x, y                    ; plot y against -180 ->179
plot, x, y, xstyle = 1, xrange=[0,10], xtitle='Horizontal', ytitle ='verticle', $
        charsize=20, charthick=2.0, font = 1, background=255, color = 0
        ; white background, black plot
```

Example: **Overlay plotting**

```
plot, y
y = y -0.5
oplot, y
```

6. Direct Graphics and Object Graphics

IDL graphics can be plotted in two different modes. Object Graphics mode is far more flexible than Direct Graphics mode. For example, you can use different fonts to label different axes in OG.

6.1 3D Direct Graphics

Selected Executive Commands (pp.34 – 40)

?	On-line Help
.Compile xxx.pro	compile IDL program only, not script file
.Go	execute the compiled program
retall	reset button
.Run	compile and execute
.Run –l xxx.pro	(or use –t flag) to show a listing of the file
@ scriptfile	to execute an IDL script file, not program
.Continue	when program stopped due to error, can interactively correct the offending statement and resume
&	connect multiple IDL statements on the same line
$cmd	fork out a new process to execute the command **cmd** (on PCs, it is an escape to DOS)
.Skip 10	to skip executing the next 10 lines or one line if not specified
.Step	to step thru
.Stepover	to step over calls to other program
.Trace	display each line and then execute

.Save, Filename='myfile.sav', /ALL
.Save, /Routines, 'sub1', 'sub2'
.Restore 'myfile.sav'
plot, fcnname(param)

(Click Help on the menu bar) **to display help**

Crt-Z	to send it to background. fg to send to foreground
Crt-	to kill IDL instantly

IDL System Variable Examples

!Pi	pi (3.14159 ...)
!Dpi	double precision value of pi
!DtoR	convert degree to radian measure
!d.window	current graphics window

APPENDIX C

Annotation to ION Script User's Guide

By Dr. Mou-Liang Kung

ION Script is an application of XML (eXtensible Markup Language). An ION Script looks similar to an HTML page with additional "tags" defined for IDL. An ION Script is a script file residing on the ION server. When a client (browser) click on the URL for that script file, it invokes the execution of the script on the ION server to produce an HTML page and have it sent back to the client. Before we begin, remember that **ION is case sensitive.**

1. Anatomy of an ION Script

```
<ION_SCRIPT>
        <ION_HEADER>
                <TITLE>My Fist ION Script</TITLE>
        </ION_HEADER>

        <ION_BODY>
                <H1>Hello, World!</H1>
                That's all, folks!
                <ION_IMAGE TYPE="DIRECT">
                        <IDL>
                                shade_surf, dist(30)
                        </IDL>
                </ION_IMAGE>
        </ION_BODY>
</ION_SCRIPT>
```

required

Notice that

- <ION_HEADER> is optional. However, it is required to hold variable and event declarations. <TITLE> is an HTML tag placed inside the <ION_HEADER> section to be exported to the HTML page created by the server.
- <ION_BODY> component contains HTML tags (e.g. H1 tag that you see there) **and/or** ION tags (e.g. <ION_IMAGE>). TYPE is an attribute of ION_IMAGE, a DIRECT graphics image rather than OBJECT graphics.
- In this example, a graphic image is created from the IDL command (surrounded by <IDL> tags)

  In this example, you saw the technique of **placing a graphic image o a web page,** where <ION_IMAGE> tags will be converted to <IMG> HTML tags. You can **place data generated from IDL commands on a web page:**

```
<ION_DATA_OUT>
        <IDL>
                Array=INDGEN(5)
                print, Array
        </IDL>
</ION_DATA_OUT>
```

You can add variables and events to create interactive web pages and flow control tags (e.g. ION_IF) to control the interaction.

2. Variables

There are user-defined variables and system variables. To access variable (e.g. num) value, placed the prefix $ in front of the variable name (e.g. $num).

Declaration of User Defined Variables

XML specification stated that all begin-tags (e.g. <ION_SCRIPT>) must be paired and nested properly with end-tags (</ION_SCRIPT>). However, you can simplify the tagging if there is no content in between the begin-tag and end-tag by using the form of <tag-name attributes .../>. Notice the / is placed before the end bracket, not after the begin-tag. This type of tag is called **empty tag.**

All variables are declared using the empty tags:

<VARIABLE_DCL NAME="num" TYPE="INT" VALUE="10" PERSIST="TRUE"/>

, where the **INT**eger variable num is initialized to 10. Other available data TYPEs are **DOUBLE, BOOL,** and **STR**. This variable is persistent in the sense that it is available to any ION script that this script calls.

Since you can call other ION script from an ION script using <ION_INCLUDE SRC="..."/>, the variables from the calling script are "global" variables and become known to the called scripts. However, if a variable from the calling script is "persistent", then it prevails over the variable with identical name in the called script.

Example:

ex1.ion (calling script)

```
<ION_SCRIPT>
        <ION_HEADER>
                <TITLE>My Fist ION Script</TITLE>
                <VARIABLE_DCL NAME="num1" TYPE="INT" VALUE="10" />
                <VARIABLE_DCL NAME="num2" TYPE="INT" VALUE="10"/>
                <VARIABLE_DCL NAME="num3" TYPE="INT" VALUE="10", PERSIST="TRUE"/>

        </ION_HEADER>

        <ION_BODY>
                <ION_INCLUDE SRC="ion://ex2.ion"/>
        </ION_BODY>
</ION_SCRIPT>
```

ex2.ion (called script)

```
<ION_SCRIPT>
        <ION_HEADER>
                <TITLE>My Fist ION Script</TITLE>
                <!-- num2 and num3 are redefined in this called script-->
                <VARIABLE_DCL NAME="num2" TYPE="INT" VALUE="20" >
```

```
        <VARIABLE_DCL NAME="num3" TYPE="INT" VALUE="20"/>
    </ION_HEADER>

    <ION_BODY>
        The value of num1 is <ION_VARIABLE NAME="$num1"/>
        The value of num2 is <ION_VARIABLE NAME="$num2"/>
        The value of num3 is <ION_VARIABLE NAME="$num3"/>
    </ION_BODY>
</ION_SCRIPT>
```

The resulted web would show

The value of num1 is 10
The value of num2 is 20
The value of num3 is 10

where num1 from ex1.ion is global to ex2.ion; num2 from ex1.ion is superceded by the num2 in ex2.ion, and num3 from ex1.ion is persistent over everything, including the num3 defined in ex2.ion.

System variables

System variables are accessible from all scripts. A system variable is more like a record in Pascal or struct in C or C++. Therefore a system variable may have many fields:

Example: $BROWSER.HTTP_USER_AGENT, where $BROWSER is the variable used to access CGI environment variables, and HTTP_USER_AGENT shows what brand of browser this is.

3. Expressions

Numerical Expressions: +, -, *, /, **MOD, and ^ (power)**

Examples: 1+6
$num1 + $num2

Boolean Expressions: GT, GE, LT, LE, EQ, and NE
Examples: $num GE 5
$num1 NE $num2

String Expressions: EQ, NE, + (concatenation), and CONTAINS
Examples: 'My name is'+ $name
$name CONTAINS 'Kung'

Note: If you assign a STR variable with numerical string constant with decimal point, say "34.5" the constant will end up being "34.500000". Therefore always use " '34.5' " with another layer of single quotes.

Logical Expressions: AND, OR, NOT

4. Flow Control: ION_IF, ION_ELSEIF, ION_ELSE

Example:

```
<ION_IF  EXPR = "$name CONTAINS  'Kung' ">
        <!-- statements when TRUE -->
<ION_ELSEIF  EXPR = "$age GT  50"/>
        <!-- statements when ELSEIF is TRUE -->
<ION_ELSE/>
        <!-- statements for ELSE -->
</ION_IF>
```

5. IDL Commands within <IDL>, </IDL> tags

- <IDL> tags must reside inside of <ION_IMAGE> or <ION_DATA_OUT>
- All commands entered must be on a single line or use $ as continuation mark to extend to the next line. The $ character itself should be prefixed with \ to escape from the continuation interpretation.
- Certain IDL code will not work:
  - ✓ No OBJ_NEW function call to create a new window
  - ✓ Can't use WINDOW procedure
  - ✓ Widgets and compound widgets are not available
  - ✓ Multiline syntax will not work:

```
FOR X=1,20 DO BEGIN
        PRINT, X
ENDFOR
```

6. Creating ION Script Applications

Specifying URL:

HTTP URL: **http://www.junk.com/file**

FILE URL: In Unix: **file:////usr/home/kung/file**

In Windows: **file:///c|dir1/dir2/file**

ION URL: *For files in default location:* ***ion://myscript.ion***

For files elsewhere: In Windows: **ion:///c:\mydir\myscript.001**

In Unix, **ion:////usr/home/kung/myscript.002**

ION Script Forms:

ION forms are more versatile than HTML forms. An ION form can have assign different events to different ION_BUTTON, ION_IMAGE, and ION_LINK. ION form can also have ION_CHECKBOX and ION_RADIO

Example: Create a button on the web page. when the button is clicked, it will load and execute page2.ion

```
<ION_SCRIPT>
        <ION_HEADER>
                <EVENTS>
                        <EVENT_DCL  NAME = "Hello"  ACTION="ion://page2.ion"/>
                 </EVENTS>
        </ION_HEADER>

        <ION_BODY>
                <ION_FORM>
                        <ION_BUTTON  EVENT="PAGE2"  TYPE="BUTTON"/>
                </ION_FORM>
        </ION_BODY>
</ION_SCRIPT>
```

Use HTML Frames with ION Scripts

Example (p.62 -63)

The web page Main.html loads two frames, one with a text box and a button (by invoking Frame1.ion) and one a blank frame (blank.html). Upon the button click, Frame1.ion invokes Frame2.ion, and a graphic image is loaded in the blank frame.

Open a new Window upon Clicking

Select the target window or frame using TARGET attribute of ION_FORM, ION_IMAGE, ION_LINK upon event occurrence. TARGET value is to be the NAME attribute value of the FRAME or IFRAME tag:

```
<FRAME  NAME="form" SRC="http://www.junk.com">
...
<ION_BODY>
        <ION_FORM  TARGET= "form">
        ...
```

Include JavaScript or VBScript in ION Script

- Use HTML <SCRIPT> tag or
- Use ION tags:
  <ION_BODY ONLOAD="script_name">
  <ION_BODY ONUNLOAD="script_name">
  <ION_FORM ONRESET="script_name">
  <ION_FORM ONSUBMIT="script_name">

Appendix D

Unix Workshop Agenda

First Hour (40 minutes Hands-on Lecture)

I. Customize Your Working Environment

1. Keyboard and mouse button shortcuts
2. File manipulation: link, copy, find, and preview
3. Publish Your Own Web Page
4. Configure the Mysterious Dot Files

(10 Minute Independent Hands-on, followed by 10 minute break)

Second Hour (40 minutes)

II. Useful Tricks to Increase Productivity

1. Screen Capture
2. Placing Oftenly Used Tools in One Place
3. Network Utilities: Telnet, FTP, NFS
4. Start X windows from Remote

(10 Minute Independent Hands-on, followed by 10 minute break)

Third Hour (40 minutes)

III. System Configuration and Maintenance

1. Use Someone Else's resources - Hardware manger
2. Find and Install New (free) Software - Software Manager
3. What to do with System Update or patch
4. How to Resolve Software Conflicts
5. User Account Maintenance - User Manager
6. Network Configuration and Simple Debugging - Network manager
7. System Boot and Shutdown

A Unix Workshop

on

Managing Your SGI Workstation (And Making it More Productive)

by

Dr. Mou-Liang Kung
Department of Computer Science
Norfolk State University

May 21, 1999

TABLE OF CONTENT

Part I Customize Your Working Environment

Unit 1 Editors and Basic Shell Commands

1. Changing the font size of Shell command window

(Click **Toolchest -> Desktop -> Open UNIX Shell** to start)

Point anywhere inside a winterm and click the right button. Click **Font** and then choose your font name or size. Note: Different fonts have different sizes, and vice versa.

2. Changing Default Viewer and Editor Utilities

When you double click a text file, you will open the file in a text editor. To choose a different default text editor (or Web browser, image viewer, mail handler, book viewer, or PostScript viewer), click **Toolchest -> Customize -> Utilities** to start Default Viewer and Editor Utilities panel. For example, if you choose **Vi** instead of **NEdit**, any text document icons you double-click will open in **Vi**. To open this panel, choose **Desktop > Customize > Utilities** from the Toolchest.

We have included the use of VI editor in Appendix C. It is nice to get familiar with it when you have to access your workstation from remote using VT100 terminal emulation using telnet or other remote access software (e.g. **procomm**).

3. Using Shell Commands

Click **Toolchest -> Desktop -> Open Unix Shell** to open a Shell window:

to change password:	**passwd**
to find the time and date:	**date**
to find out who is currently login:	**who**
to find out what your login name is:	**who am i**
to find out the full path name of the current directory:	**pwd**
to list all file names in the current directory, but not hidden files:	**ls**
to list all file names in the current directory, including hidden files:	**ls -a**
to list all file names in the current directory in detail:	**ls -l**
to list all file names in the current directory in detail, all:	**ls -al**
to print a message on the console:	**echo You are the Greatest!**
to create a text file called junk:	**echo You are the Greatest! > junk**
to view the content of a text file:	**cat junk**
to concatenate two files into a third:	**cat file1 file2 > file3**
to delay 20 seconds then execute a command:	**sleep 20; echo Tea is Ready!**
to find out what type of terminal emulation	**tset -**

Please see Appendix A for more commands and Appendix B for shell scripts.

Unit 2 File Manipulation

1. Simple File and Directory Commands

to traverse the file system directories: **cd ..**
(.. = parent directory, . = current directory)

to copy file1 to file2: **cp file1 file2**
to create a subdirectory: **mkdir subdir_name**
to copy an entire directory: **cp -r srcdir dstdir**
to rename a file or directory: **mv oldname newname**
to delete a file: **rm filename**
to view text file one screen at a time: **more filename** (or **page filename**)
to find the location of a file: **find / -name filename -print**
to delete a subdirectory and everything in it: **rm -r subdir_name**

2. Change Permissions to Protect Your Files or Directories

Type **chmod 760 filename** when you wish to grant all permissions (**7** = binary 111_2 for read=1, write=1, exceute=1) to yourself, read and write only for your group members (**6**=110_2 for read=1, write=1, execute=0), and no permissions for other users (**0**=000_2 for read=0, write=0, execute=0). If you plan not to share anything with anyone, add the line **umask 077** in your **.cshrc** file.

3. Create a link or a copy to a file icon

Drag and drop an icon will only create a link (filename is in italic), but with Ctrl key pressed down while drag and drop will create a copy.

4. Preview the Contents of Files (text files, images, movies, sound files, VRML, WebJumper, and HTML files)

You can open a file by double clicking its icon or click and highlight a file icon and then choose **Options > Show Content Viewer**.

5. Avoid scrolling the entire file listing in an Icon View of the directory that contains the file you want

Click once in the Icon View background. Then begin typing the name of the file or directory, say "junkmail" you want to select. When you press each letter in sequence, a "tick" sound means there are files begins with "j". If you hear a "ding" sound then that means the unique match is found. So while you are typing "junkmail", it may sound like j (tick), u (tick), n (ding). With only three letters "jun", you have a unique match. Now press **Enter** to execute or open it.

Part I Customize Your Working Environment

Unit 3 Mouse Button Shortcuts to Copy and Paste

1. Insert Previously Typed Text into the Command Line

Crop (highlight) the text and then click the middle button to paste at the command line.

2. "Crop" Any Text and Dump into a New Text File

Crop (highlight) any block of text in an application window, then move the mouse pointer to any place on the desktop and click the middle mouse button. A new file, named "pasted text" will be created containing the selected text.

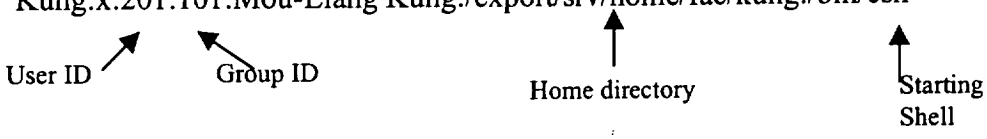
Unit 4 Configure the Dot Files

Dot files (e.g. **.cshrc**) are hidden files (i.e. not visible when you type **ls**. Most dot files are application initialization files. For example, **nedit** has **.nedit, vi** has **.exrc**, ... These dot files are text files that you can edit to customize the behavior of the application.

When you login, UNIX will start a Shell, typically, Bourne Shell or C Shell. However, there are other new and improved Shells such as Bash (Bourne Again Shell, superset of Bourne shell), TCsh (improvement on Csh, in IRIX) and Korn Shell (most recent Bourne shell improvement, in IRIX).

Find out what Shell you login for: As you log in, use **cat /etc/passwd** to check the passwd file to find user's entry (kung in this case):

Kung:x:201:101:Mou-Liang Kung:/export/srv/home/fac/kung:/bin/csh



If you have **su** privilege, then you can edit this text file and change to **/bin/sh** , **/bin/csh, /bin/bash, /bin/tcsh** or **/bin/ksh**.

If you use Bourne Shell (command prompt is **$**), the system will automatically execute the file **.profile**. But if you are using the C Shell (command prompt is **%**), first **.login** and then **.cshrc** will be executed. These files are called hidden files since their names do not appear when you do **ls** . Any file name begins with a **.** (dot) is a hidden file.

.profile, .login, .cshrc are all called Shell Scripts, files contains command lines.

.login can be used to check what type of terminal you are using, check the message of the day (/etc/motd) and whether you have any new mail. You can add more lines to initialize environment variables using **set** and **setenv** commands only once when you login. For example, you want to prevent accidental overwrite of your files, you can use Textedit to add the line:

set noclobber

to the bottom of **.login**.

.cshrc typically contains **set** statements to set up C Shell variables and aliases for favorite command lines. For example

alias dir ls

would mean that **dir** will execute **ls.**

Warning: When you are editing Shell scripts, do not press return key unless you meant that is the end of a command line.

Some common changes users like to make:

- change **umask 002** to (or add the line) **umask 077**
- change the command search path:
  set path=(/bin /usr/bin your-favorite-cmd-path $path)
- change the line of MANPATH to
  setenv MANPATH your-favorite-man-path:/usr/man:
- If you telnet into UNIX system from a VT100 terminal software, add the lines:

```
if ( `tty` != "/dev/console" ) then
        stty erase ^?
        setenv TERM vt100
endif
```

To make changes to **.cshrc** effective immediately, rather than the next time you login, type **source .cshrc** at the **%** command prompt of a shell window.

There is yet another hidden file **.logout** of command lines used by **csh** when you logout to clean up everything when you leave.

For Bourne Shell or Korn Shell users, **.profile** can be edited except that

PATH=.:$PATH:/usr/kung/myfavorite

Part I Customize Your Working Environment

Unit 5 X Windows

X windows X11R6 (generic) comes in different windows managers: Motif, OpenLook (sun), and others (e.g. Tom's Window Manager). X server is the software piece that controls yours console, X clients are X-applications running from remote or local sites. Free X server for PC (X11R5) or Mac (X11R6) is available at

http://tnt.microimages.com/freestuf/mix/download.htm

1. How to start a Remote X-window

Let's say you are on the console of a machine called Poseidon (199.111.120.30) and wish to open an X-window application from a remote machine called nino (199.111.120.40). Type in the following lines in a Shell window (you may be able to use the machine name **nino** instead of 199....).

xhost +199.111.120.40
telnet 199.111.120.40
(login now)
xterm -d 199.111.112.219:0.0 &
(or for a particular X window application such as toolchest)
xterm -d 199.111.112.219:0.0 -e /usr/bin/X11/toolchest -name ToolChest &

If a host is often accessed, you can create a file /etc/X0.hosts and place that host name or IP number there.

Question: Can I use X-window software on my PC, Mac or others to have the exact SGI Desktop on my PC or Mac screen?

Answer: No, you can not. The reason is that the SGI Desktop is the 4Dwm in execution. Your PC or Mac has its own window manager, which is a part of your X-window software.

2. X-Windows Initialization Files (Dot files)

4DWm, the SGI X windows manager is an enhanced Motif Window Manger (**mwm**). A user can alter the default settings of the window manager by

a. **cp /usr/lib/X11/system.4Dwmrc $HOME/.4Dwmrc**

b. Use **nedit** to edit **.4Dwmrc** (**.mwmrc** for **mwm**). Fr example, you can change the menu as you press down left or right button.

If you wish to change the behavior (foreground and background color, fonts, size and location,...) of X-applications, then you can edit the file **.Sgiresources** (or **.Xresources**).

3. Start Generic X-Windows (X11) from the Command line Console

Old Unix systems (or new systems for some reason) start a command line console when you log in. You need to start X windows manually. First use text editor (such as vi) to create a file called **.xinitrc** in the home directory with the following content:

```
xterm -sb 2 >/dev/null &
mwm  2 > /dev/null
```

Then type **xinit** at the command line.
Note: **mwm** is the Motif Window Manager. There are other window managers (e.g. **twm**, Tom's Window Manager) came with the X11 distribution.

Unit 6 In case of Emergency

1. Control-C or Del key

These are the panic buttons to push when you wish to terminate the current foreground process. Nastier programs can be "killed" (see item 4).

2. In case the Mouse Cursor Got Stucked

When the mouse cursor appears to be stuck, try pressing the **Esc** key to release it.

3. Bringing Back the Toolchest

If you accidentally close the Toolchest, you can bring it back by typing `toolchest&` in a shell window

4. In Case the Console Is Frozen

Don't turn the power switch off!!! Go to another networked computer and telnet into this frozen computer. If you can not telnet, the system is definitely crashed and your need to power off and on again. If you can telnet into it, then login, and type the following command:

ps -e (or **ps -ef** for detailed listing)

to list all processes. Take note of the owner (e.g. kung or root) and the number right next to it (PID, the process ID). Try to kill each process:

kill pid or **kill -9 pid** (all 9 lives of it)

from the largest pid **you own** to the smallest one, one by one to see if the console can be unfrozen.

5. (Command): not found?

You know the command exists and was spelled correctly, but the console keeps on saying that **Command Not Found**. Try typing the full path name of the command. If you are root user, you will be forced to type all commands in full path name (or **./cmd** for those in the current working directory) to avoid Trojan Horse commands. If you are not the root user, you can add the directory name to the PATH variable. Type **set** to reveal the search Path for commands. Your command may not be in the search PATH. If it is not, edit PATH in your .cshrc or .profile.

Unit 7 Miscellaneous Commands

1. UNIX even lets you type in a new file without a text editor

This is extremely convenient, if you want to create a simple and short Shell Script without having to get in and then out of an editor:

% cat > newfile
Dear Mr. Kung,
I am so thrilled to be in your class ...
(Ctrl-D when finish and save or Ctrl-C to quit without saving)
%

2. UNIX e-mail

To read, type **mail**
To send someone (say, kung) an e-mail, use the text editor to type up a letter and type
mail kung < newfile

3. Continue executing a **cmd** even one logged out: **nohup cmd &**

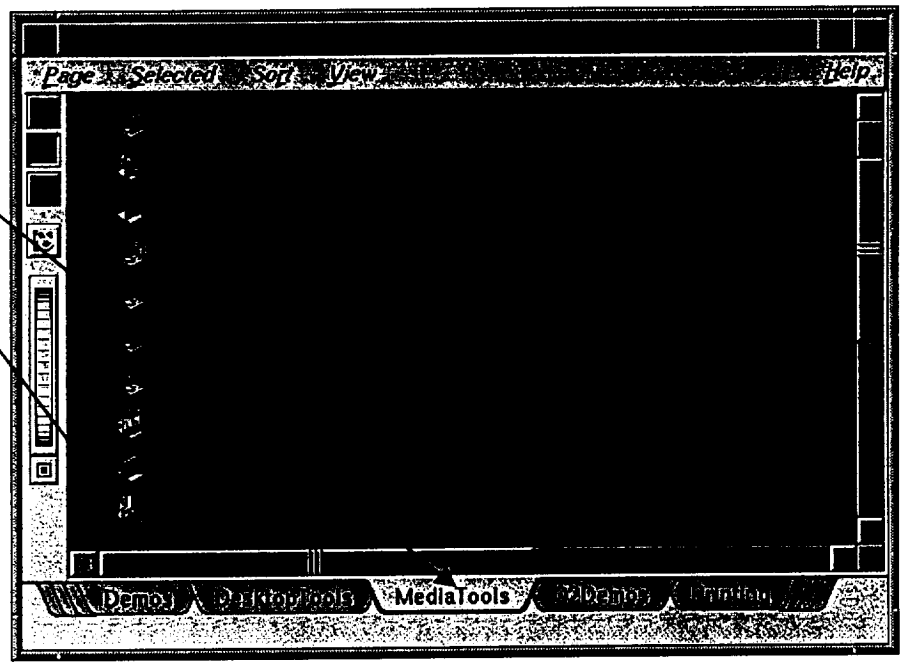
Part II: Screen Capture, Icons, and Web

Unit 1. Screen Capture and Icons

Step 1 In the ToolChest Window, Click Find -> Media Tools

You will then see the next window

Step 2 Click **MediaTools** tab and then **media recorder**

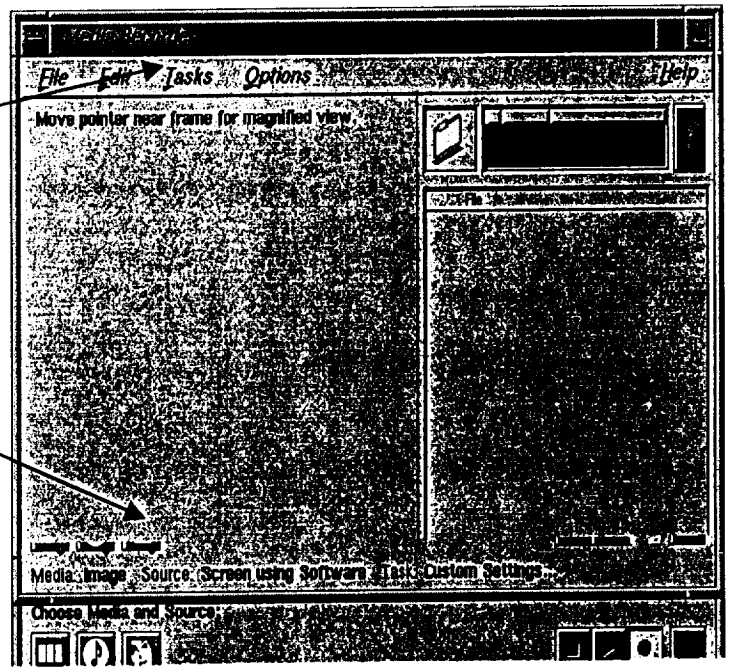


You will see the window on the right. This is the main control for screen capture.

Step3. Select Tasks -> Show Task Setting -> File Format, to select the file format you wish to save as, say GIF.

Step 4 Click down the image icon, you will see Image From Screen▶ and then select one of the four choices as you go to the right of the arrow, and you will see a camera icon.

Select Area and Record - click and crop any area of the screen and let go the button. That's it.



Part II. Screen Capture, Icons, and Web

Select Area - Click and crop any area of the screen and then click the red dot button

Window Area - Click the Title bar (at the top) of a window and then click the red dot button

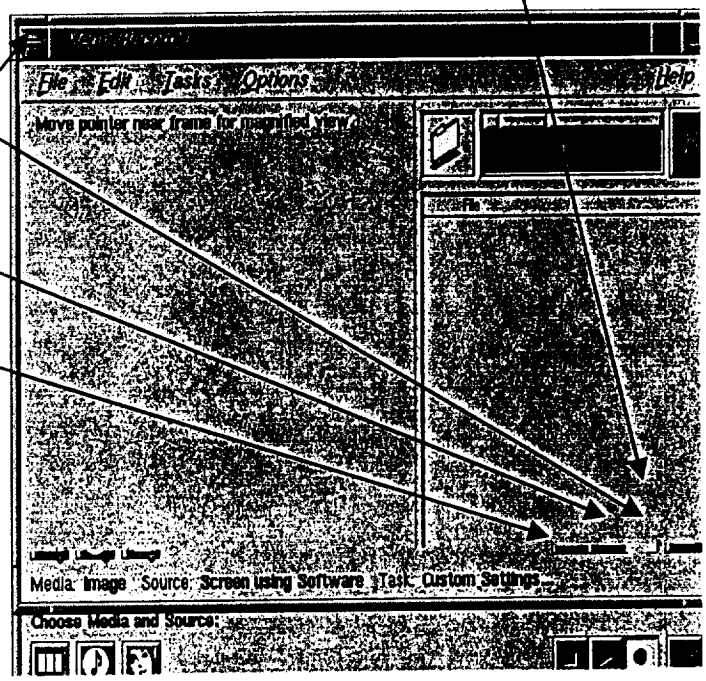
Full Screen - Click the red dot button

Images captured will automatically be named **image?.gif**

Step 5 To View the captured image file, click button. To stop viewing, click button

Now the GIF file can be found in your home directory to be imported into word processor or presentation.

Note: You Can Quickly Close a Window by quickly double-clicking the little bar here.



Unit 2 Create a folder icon on desktop for quick access

Point and select any block of text which is a directory (folder) full path name (e.g. /usr2/kung) in an application window (or type it in at the Shell command prompt), then move the mouse pointer to any place on the desktop and click the middle mouse button. A folder icon with the name "kung" will appear on the desktop.

Unit 3 Web Pages

1. Publish Your Own Web Page (by Putting Them in *public_html)*

Any HTML file, say mywebpage.html that you put in the *public_html* directory (located in your home directory),can be accessed as *http://mysite.nsu.edu/~kung/mywebpage.html*

Note:

- Use chmod to change the permission on your HTML files to 755
- An HTML file named index.html will be the default when a browser requested http://mysite.nsu.edu/~kung

2. Create a desktop icon for a Web page (a WebJumper)

Point and select a Web page URL address in an application window, then move the mouse pointer to any place on the desktop and click the middle mouse button. A WebJumper icon for that Web page will be created.

Unit 4 Use Your Photo as your Login Icon in the Login Window

1. Use the Attached Camera

Start **Media Recorder** by clicking the camera or microphone icon. Click and select **Tasks - > Show Task Settings -> File Format -> GIF** . Click image icon (one with the lady's face) and select **image from camera**. Now click the circular dot button to record.

2. Import a Photo

Step 1 Use a scanner to scan in your photo, trim it into a square shape, and save it as GIF or JPG format, say **kung.jpg**

Step 2 FTP your kung.jpg to your home directory on an SGI station.

3. Make Your Photo into an Icon

Step 1 Execute the command **imgview** at a shell command window, and you will see a window appear

Step 2 Click **File -> Open** and open **kung.jpg**

Step 3 Click **File -> Save As** and change the file extension name in the **Selection** box from **jpg** to **rgb** and also click **infer from file name** bar to select **Classic SGI Image**

Step 4 Click **OK** and exit this window.

Step 5 In a Shell command window, type **/usr/sbin/istat kung.rgb and you'll see**

xsize	ysize	zsize	...
173	182	3	...

Step 6 Use a calculator to compute **100/173 = 0.578** (to the thousandths), and also
Compute **100/182 = 0.549**

Step 7 In a Shell command window, type **/usr/sbin/izoom kung.rgb login.icon 0.578 0.549**

Step 8 In a Shell command window, type **mkdir .icons**

Step 9 In a Shell command window, type **cp login.icon .icons**

You can log out now and find your picture appears as your login icon.

Part III System Configuration and Maintenance

Even though you can log in as **root** to perform system level tasks, it is best that you log in as a common user and switch to **root** using **su** ("switch user"or "superuser") at shell command prompt when needed. Any user can click **Toolchest - > System -> System Manager** to view system settings or perform certain tasks without being a **su.**

Unit 1 Backup and Restore

You need to have a full system backup to a tape system every few months and incremental backup once a week to make sure your important software (especially you configured with software licenses) and data can be restored in case of crash. For example, a SCSI ExaByte 8505 (8 mm) tape drive can be plugged in after you shutdown the system. Reboot the system, the system will automatically recognize it for you to backup or restore.

Unit 2 Resource Sharing (Someone Else's CD, Floppy, or DAT Drive)

If you want to use someone's hardware on your network does, choose **Desktop > Shared Resources > On a Remote Workstation** from the Toolchest. This brings up a window that lets you display another system's shared resources so that you can drag the CD, floppy drive, or DAT drive icon onto your desktop.

Unit 3 Find and Install New Software

Open a Web browser and go to **http://www.sgi.com/fun/** and click on **SGI Free.** You will find all sorts of software already compiled and ready to be installed. Just click the install icon to proceed.

Unit 4 System Update and Resolve System Conflicts

Find all CDs from the past upgrade. When you see **conflicts** red button, click it to see another window asking you which action to take. Select **add** and then eject and insert all past CDs one by one to resolve them.

Unit 5 How to Check Disk Usage (No need to be **su**)

1. In order to check the file or directory size (in 1024 blocks) of everything under the current directory: **du -k dirname**
2. Check all local and remote-mounted (NFS) disk sizes: **df -k**

Part III System Configuration and Maintenance

Unit 6 User Account Maintenance

You can use toolchest or shell commands to add (say **john**) or delete users by directly insert or delete an entry in the **/etc/passwd** file. Doing so also means that later you need to create or delete their home directories (**mkdir /usr/john**) and give the proper ownership (**chown john /home/john**).

Unit 7 Network Configuration and Debugging

You need basic skills to test if the network (software and hardware) is functioning correctly. You do not have to be **su** just to view without change the settings.

1. **inetd** is a network super server that listens for network connection and starts appropriate daemon (server program) to handle the request. It uses a configuration file called **/etc/inetd.conf** (a text file viewable by **cat** or **page**). You should always comment out (insert # sign) the lines starts with **systat, tftp,** and **link** for security reason.

2. If you suspect the network or a host (say, **Vger**) is down, try
   /usr/etc/ping 192.68.217.10 (that's Vger)
   /usr/etc/netstat -nr | grep def (to check if **def**ault router is down)
   /usr/etc/netstat -i (to check network reliability, where Ierrs and Oerrs should be 0)
   /usr/etc/traceroute 192.68.217.10 (to trace which routers the packets hopped over on their way to that destination)

3. **/etc/hosts** (a text file) contains all host aliases. You can add the frequently used host addresses to it

4. **/etc/hosts.equiv** (a text file) contains the host you trust (for such cmds: rsh or nfs) without user authentication. You should delete + (used in NIS) and only add those hosts you can trust.

Unit 8 System Boot and Shutdown Commands

shutdown -g30 -y -in

where **30** means 30 seconds for all to log-off, **y** means all responses to the interaction are yes, and **n** is 0 to halt and drop down to firmware level (when a disk crashed, system will only go into firmware state), n is 6 to reboot, n isl, s or

S to go to single user mode (to mount only the root file system for service). If you wish to power-off as well, add the **-p** flag to that line.

Note: Shutdown can be done using toolchest. When lightening or power failure is imminent, you should protect the equipment by shutting down (state 0)

Unit 9 System or Disk Crashes

Go to another SGI and click on help to search the topic of **Recovering Data After System Corruption** and **Using the Command (PROM) Monitor**, and print them out. To recover root file system crashes, you need an IRIX **System CD** and a **full system backup tape.** When the system is powered on and you see **Staring up the system ...,** press **esc** to bring up the **System Maintenance Menu** (you are in firmware mode). Select option 4 for the **System Recovery Menu** and follow the printed instruction.

Unit 10 System Security

When a workstation owner becomes familiar with his/her station, he/she should invest some time in system security. Get a few books or free guides and implement the recommendations. Nothing can be more valuable than applying the system patches and updates which typically dealt with recently discovered security holes. Here are just a few simple tasks you can perform:

- There should be no login account without a password. A quick glance of /etc/passwd will reveal the logins without passwords. If the text in the second field of the passwd entries is missing, add something in.
- Disable (comment out) any Network services which you do not require from **/etc/inetd.conf**. For example: **tftp, rexd, exec** and **finger**
- Delete the "+" , '!' or '#' from **/etc/hosts.equiv**
- Keep only a small number of "setuid" programs owned by root. Never allow any setuid shell scripts, especially setuid root. To scan the file system for setuid files owned by root, type

  $ find / -user root –perm 4000 -print
- **.profile, .login, .cshrc, .xinitrc, .exrc,** ... are protected with no permission to write or read by anyone except the owner
- Add the following to your **.cshrc** to check the time last time you supposedly logged in:

  echo "Last time logged in was:"
  ls –lc $HOME/.lastlogin | cut –c42-53
  touch $HOME/.lastlogin
  mesg n
- Check the users and login time kept in /var/adm/wtmp by typing:

  who /var/adm/wtmp or **last** or **lactcomm**

Appendix A

C Shell

	Command Execution
cmd &	execute cmd in the background
cmd1 ; cmd2	execute them in sequence
(cmd1 ; cmd2)	two sequential commands as one
cmd1 | cmd2	Two parallel commands, output of cmd1 is PIPED as input to cmd2
cmd1 && cmd2	if cmd1 is successful, then cmd2 will be executed, else cmd2 will not
cmd1 || cmd2	if cmd1 is successful, then cmd2 will NOT be executed, else cmd2 will be
`cmd`	forced execution of cmd in a subshell and the output text will be placed where backward quotes ` ` were.

	I/O Redirection
cmd < file-name	cmd will take input from a file named "file-name"
cmd > file-name	cmd will dump its output to a file named "file-name". if File-name" exists, it will be overwritten!!!, else created
cmd >> file-name	cmd appends output to the file named "file-name"

C Shell Variable Names

A C shell variable name consists of up to 20 letters or digits beginning with a letter. **set** and **unset** are used to create and assign value or destroy the variables

The value of a variable, prefixed with $, is any string bounded by ():

```
% set junk =(Hello, how are you?)
% echo $junk
Hello, how are you?
% echo $#junk
4                    (4 is the number of words in the value of the variable)
% set Junk ="       Hello, world!"
% echo $junk
Hello, world!
```

Quotes

Single quote pairs ' ' and double quote pairs " " both quoting a string as one word. But single quotes' ' suppresses variable-value substitution while double quotes " " always substitute.

```
% set junk = Hello
% echo 'I like to say $junk'
I like to say $junk
% echo "I like to say $junk"
I like to say Hello
```

Appendix A

File Name Substitution

*	matches any, 0 or more, characters
?	matches any single character
[abcde, s -z]	matches any one character a, b, c, d, e or any between s and z
{cat, bat}	matches strings cat or bat
	For example: % ls {cat, bat}* will list all file names preceded by cat or bat, such as "catalog", "battery", ...
~kung	home directory of user named "kung"

Expressions and Operators

+, -, *, /	Arithmetic operations
%	Remainder (Modulo math)
<, >, <=, >=	Logical operators
= =, !=	Logical operators for STRING COMPARISON!!
=~. !~	Checks if matches a filename substitution pattern
-r, -w, -x file-name	Checks user's permission on the file named "file-name"
-e file-name	Check if the file exists
-f file-name	Checks if it is a plain file
-d file-name	Checks if it is a directory

C Shell Script controls:

```
foreach  varname  (word-list separated by spaces)
        ...
end
```

```
if (expression)  single-cmd
```

```
if (expression)  then
        cmds
        ...
else
        ...
endif                   (Any else if pairs allowed, but only a single else)
```

```
switch  (string)
case    pattern1:
        ...
        breaksw
case    pattern2:
        ...
        breaksw
...
default:
        ...
```

```
        breaksw
endsw
while (expression)
        ...
end
```

break	Break out of the loop
continue	Skip the current iteration and go to the next iteration

Environment Variables and Shell Variables

Environment Variables are exported to any new C shell or process it invokes while C shell variables are only valid in the current shell. Use setenv to define environment variables, but use set to define C shell variables. Default environment variables are HOME, SHELL, PATH, MANPATH, USER, TERM, PWD, ... These environment variables are also copied into C shell variables in lower case: home, user, ...

In C Shell,
$0 = cmd argument; $1 = first argument followed cmd, ...,
$argv = all arguments, except the cmd argument
$$ = current process ID

Shell variable numerical conversion and manipulation

@ **is used to set the numerical values instead of string**

```
% set sum = 1
% @ sum = $sum + 1
% echo $sum
2
```

Some Important C Shell Builtin Commands

exec cmd	execute cmd in place of current shell
exit (expr)	exit the shell and assign exit status to be "expr"
jobs	list all current background jobs
kill pid	kill the background job with process ID = pid
kill %2	kill background job number 2
nohup cmd	let the cmd continue executing even logged out
onintr -	ignore all interrupts
onintr	restore shell interrupts
onintr label	go to "label" upon interrupt
shift varname	shift arguments left by one word ($2 becomes $1, ...)
time cmd	time the shell execution of the cmd
wait	wait for the background to finish or an interrupt

Appendix B

Creating C Shell Scripts

1 Edit the Scripts

Double click the Text Editor icon and enter the following text and save as **cmd1**

```
#!/bin/csh
echo Greetings!  This is my first C Shell script
who | wc
date
```

Note: A. If a line gets too long, do not press return to get to the next line, type \ instead.

e.g. echo 'My name is kung and I do like to watch football games \
and fishing as well!'

B. You should always specify what shell is used to run this script and also use #, thecomment statement to document your script:

(No indentation here)

```
#!/bin/csh
#
# cmd1
#
# This command will display non-directory file names only
#
```

2 Make the Script you just created "excutable" in order to run

Go to a cmdtool window and type **chmod 700 cmd1**

Now you can execute **cmd1** by typeing cmd1 in a cmdtool window.

Examples of C Shell Scripts:

Ex1.

```
#!/bin/csh
set cmds = ("date"  "who"  "ls")
$cmds[1]
$cmds[2]
$cmds[3]
echo "$#cmds commands were executed"
```

Ex2.

```
#!/bin/csh
foreach cmd ($argv)
        $cmd
end
echo "$#argv commands were executed"
```

Appendix B

Ex3.

```
#!/bin/csh
@ count = 1
while ($count <= 10)
        echo $count
        @ count = $count + 1
end
```

Ex4.

```
#!/bin/csh
foreach file (`ls –a`)          # for each file in the current directory
        if (! –d $file) then    # test if it is not a directory
                echo $file      # echo that file name if it is true
        endif                   # matches if
end                             # matches foreach
```

Ex5.

```
#!/bin/csh
echo 1.         date
echo 2.         who
echo Please enter your choice (1 or 2)"
set answer = $<
if (answer == 1) then
        echo You wanted to find Date
else    echo  You wanted to find Who
endif
```

Ex6.

```
#!/bin/csh
# This script is called   dirsearch
# Proper usage:   dirsearch filename dirname
#
switch ($#argv)
case 2:
        foreach name (*)
                if (-d name) then
                        dirsearch  $argv[1] $name
                else if ($name = = $argv[1])  then
                        echo Target found at `pwd`
                endif
        end
default:
        echo Usage:  dirsearch filename dirname
        exit(1)
endsw
```

Appendix C

The VI Editor

VI editor has two modes: command mode and edit mode. You should always know what mode it is in.

1. **To invoke vi:**
   $ vi rnyfile
   You are now in command mode, you should see a ~ at the top of the screen.

2. **To exit vi:**
   Press **Esc** key and type **:wq** to save and quit
   Or type **:q!** to quit without saving
   Or type **:w newname** to save in a new file and then
   :q to quit

3. **To get in/out editing mode:**
   Press the letter **a** key, (don't press return key) to start to append
   Or Press the letter **i** to insert after the cursor

4. **To Switch between Command/Edit Modes**
   From command to edit: use **a** or i to edit
   From edit to command: press **Esc** key

5. **Inserting text**
   Move cursor using the arrow keys before pressing a to append or i to insert.
   Note: Back space key will back the cursor one character at a time for you to overstrike, but not necessarily deleted, not until you type something over it.

6. **Deleting text**
   Press **ESC** to change to command mode, move the cursor to the character to be deleted and press x to delete a character or dd to remove a whole line

7. **The deleted line can be pasted below the cursor** by typing **ESC:p**

8. **To search/locate a string**
   Press Esc, and type **:/targetstring** and then press **n** for forward search, **N** for backward search.

9. **To search and replace a string**
   Method A. Press Esc, type **:g/oldstring/s//newstring/c**
   You will be prompted to answer y or n at each occuurrence.
   Method B. Press Esc, type **:g/oldstring/s//newstring/g**
   You will NOT be asked for your approval to replace!

10. **To copy or move a block of text**
Move cursor to the starting character of the block and press Esc and type **:15dd** to remove 15 lines from the cursor or **:15yy** to copy 15 lines. Then move the cursor to the destination and press **:p** to paste below the cursor or **:P** above the cursor.

11. **To insert an external file into the text:**
Move the cursor to the destination and press **ESC : r filename**

12. **To save a block of text (say line 7 thru line 20) into a file**
Press **Esc: 7,20 w newfile** (creating **new** file)
or **:7,20w! oldfile** (**overwrite old** file)
or **:7,20w >> myfile** (append to **myfile**)

13 **To Page-up/down:**
CTRL-F to go Forward , or CTRL-B (back) to go back

14. **To find out which line the cursor is on: Press Esc, and type :nu**

15. **You can execute sh command while** in **vi:** Press Esc, type **:sh** (Now you will see $ prompt.) Type exit to return back to vi.

16. **There are many other useful editor commands.** For example, to change a single character from lower to upper case:
Press Esc, and then type ~ (to toggle)

17. **To insert the output of a command (say, date) directly into the text:**
Press **ESC** and type **:r!date**

18. **Use VI to create a file called .exrc at your home directory to contain the following 4 lines:**
Set nu
Set showmode
Set shiftwidth=3
Set bf

Appendix E

EzSAGE Source Code

NSU Code

cc_sage.pro
colorbar.pro
ez_sage.pro
Getinfo.pro
Get_more_info.pro
get_scale_info.pro
Grid_Smooth_Map.pro
Month_Read.pro
plot_points.pro
ReadSage.pro
reset.pro

Two subroutines from RSI

normal_scat.pro
cw_fslider.pro

```
;*****************************************************************************
**
;Copyright (c) 1999, NSU SciViz Lab.  All rights reserved.
;     Unauthorized reproduction prohibited.
;+
; NAME:
;     Getinfo.pro
;
; RESTRICTIONS:
;     Works with SAGE II data, g2p v5.931
;
;-
;*****************************************************************************
**

pro getinfo_events, event

;Ignores anything that is not a botton

eventname = tag_names(event, /structure_name)


If eventname ne 'WIDGET_BUTTON' then return

;Does button event.

widget_control, event.top, get_uvalue=info, /no_copy
widget_control, event.id, get_value=buttonvalue

case buttonvalue of

      'Cancel':widget_control, event.top, /destroy

      'Accept': begin

            ;Get the values entered in the fields.
            Widget_Control, info.nummonthID, get_value=nummonth
            Widget_Control, info.altitudeID, get_value=altitude
            if (info.surf_type ne 1) then Widget_Control, info.rangeID,
get_value=range
            if (info.file_type eq 3) then begin
                  Widget_control, info.indateID,get_value=in_day
                  Widget_control, info.fidateID,get_value=fi_day
            endif
            Widget_Control, info.headerID, get_value=header

            ;Test for range

            if (info.file_type eq 3) then begin

                  if (info.surf_type ne 1) then begin
                  ;Sends then to the pointer
                        (*info.ptr).nummonth = nummonth
                        (*info.ptr).altitude = altitude
                        (*info.ptr).range = range
                        (*info.ptr).header = header[0]
                        (*info.ptr).in_day = in_day
                        (*info.ptr).fi_day = fi_day
                        (*info.ptr).cancel = 0

                  endif else begin
```

```
                        ;Sends then to the pointer
                        (*info.ptr).nummonth = nummonth
                        (*info.ptr).altitude = altitude
                        (*info.ptr).header = header[0]
                        (*info.ptr).in_day = in_day
                        (*info.ptr).fi_day = fi_day
                        (*info.ptr).cancel = 0

                  endelse

            endif else begin

                  if (info.surf_type ne 1) then begin
                  ;Sends then to the pointer
                        (*info.ptr).nummonth = nummonth
                        (*info.ptr).altitude = altitude
                        (*info.ptr).range = range
                        (*info.ptr).header = header[0]
                        (*info.ptr).cancel = 0

                  endif else begin

                        ;Sends then to the pointer
                        (*info.ptr).nummonth = nummonth
                        (*info.ptr).altitude = altitude
                        (*info.ptr).header = header[0]
                        (*info.ptr).cancel = 0

                  endelse

            endelse

            widget_control, event.top, /destroy

            end
endcase
end
;-------------------------------------------------------------------------
------------------------
pro getinfo, nummonth, altitude, header, alt_unit, surf_type, range,file_type,
in_day, fi_day, cancel=cancel, parent=parent

;Gets information from the user in order to give the proper surface
;nummonth = the number of months to be studied
;altitude = the surface to be studied, lat, lon or altitude



; Error Handler.

On_Error, 2 ;Return to Caller

If n_elements(nummonth) eq  0 then nummonth = 1
If n_elements(in_day) eq  0 then in_day= 1
If n_elements(fi_day) eq  0 then fi_day= 30
If n_elements(file_type) eq  0 then file_type= 1
If n_elements(header) eq  0 then header = 'Do not know.'
;altitude = 700
;alt_unit = 'latitude Degrees'
;surf_type = 2
```

```
; Set dialog in the middle of the screen

device, get_screen_size = screensize
xcenter = fix(screensize[0] / 2.0)
ycenter = fix(screensize[1] / 2.0)
xoff = xcenter - 150
yoff = ycenter - 150

If n_elements(parent) ne  0 then begin
      tlb = widget_base(column=1, xoffset=xoff, yoffset=yoff, title = 'Enter
Output Information',$
      /modal, group_leader=parent, /floating)

endif else begin

      tlb = widget_base(column=1, xoffset=xoff, yoffset=yoff, title = 'Enter
Output Information')

endelse

;Create subbase for text widgets

subbase = widget_base(tlb, column=1, frame=1)
nummonthID= cw_field(subbase,title='Number of month to be studied.',value =
nummonth, /integer)
altitudeID= cw_field(subbase,title='Surface to be studied in '+ alt_unit +'.',
value = altitude, /floating)
If (surf_type ne 1) then rangeID= cw_field(subbase,title='Degree range to be
studied in '+ alt_unit +'.',value = range, /floating)
If (file_type eq 3) then begin

            indateID= cw_field(subbase,title='Initial day number of the date
range.',value = in_day, /integer)
            fidateID= cw_field(subbase,title='Final day number of the date
range.',value = fi_day, /integer)

endif

headerID = cw_field(subbase,title='Enter a 30 character description of the
output file.',value = header, xsize=30, /string)

;Creates subbase for buttons

butbase = widget_base(tlb, row=1)
cancel = widget_Button(butbase, value='Cancel')
accept = widget_Button(butbase, value='Accept')

Widget_Control, tlb, /realize

;Set pointer to store information entered

ptr = ptr_new({nummonth:1, altitude:11.5, range:5.0, in_day:1, fi_day:30,
header:' ', cancel:1})



;Test for range
If file_type eq 3 then begin

      If (surf_type ne 1) then begin
```

```
            info = {nummonthID:nummonthID, altitudeID:altitudeID, rangeID:
rangeID, headerID:headerID,fidateID:fidateID,indateID:indateID, surf_type:
surf_type, file_type:file_type, ptr:ptr}

      endif else begin

            info = {nummonthID:nummonthID, altitudeID:altitudeID, headerID:
headerID,fidateID:fidateID,indateID:indateID, surf_type:surf_type,file_type:
file_type, ptr:ptr}

      endelse

endif else begin

      If (surf_type ne 1) then begin

            info = {nummonthID:nummonthID, altitudeID:altitudeID, rangeID:
rangeID, headerID:headerID, surf_type:surf_type,file_type:file_type, ptr:ptr}

      endif else begin

            info = {nummonthID:nummonthID, altitudeID:altitudeID, headerID:
headerID, surf_type:surf_type,file_type:file_type, ptr:ptr}

      endelse

endelse

;Set information on the widget as a user value
widget_control, tlb, set_uvalue=info, /no_copy

xmanager, 'getinfo', tlb, event_handler='getinfo_events'

;Recover the information from the pointer location

fileinfo = *ptr
ptr_free, ptr

;Error handler on entered info

catch, error

If error ne 0 then begin

      ;ok = dialog_message(!err_string)
      ;cancel = 1

      ;return

endif

;User pressed cancel botton.

cancel = fileinfo.cancel
If cancel eq 1 then return

      nummonth = fileinfo.nummonth
      altitude = fileinfo.altitude
      range = fileinfo.range
      header= fileinfo.header
```

```
      in_day = fileinfo.in_day
      in_day = fileinfo.in_day

;print, nummonth, altitude, range, header

return
end
```

```
PRO colorbar, BOTTOM=bottom, CHARSIZE=charsize, COLOR=color, DIVISIONS=
divisions, $
   FORMAT=format, POSITION=position, MAX=max, MIN=min, NCOLORS=ncolors, $
   PSCOLOR=pscolor, TITLE=title, VERTICAL=vertical, TOP=top, RIGHT=right

; NAME:
;   COLORBAR
;
; PURPOSE:
;       The purpose of this routine is to add a color bar to the current
;       graphics window.
;
; CATEGORY:
;       Graphics, Widgets.
;
; CALLING SEQUENCE:
;       COLORBAR
;
; INPUTS:
;       None.
;
; KEYWORD PARAMETERS:
;
;       BOTTOM: The lowest color index of the colors to be loaded in
;                  the bar.
;
;       CHARSIZE: The character size of the color bar annotations. Default is
1.0.
;
;       COLOR:    The color index of the bar outline and characters. Default
;                  is ncolors - 1 + bottom.
;
;       DIVISIONS: The number of divisions to divide the bar into. There will
;                  be (divisions + 1) annotations. The default is 2.
;
;       FORMAT:   The format of the bar annotations. Default is '(F6.2)'.
;
;       MAX:      The maximum data value for the bar annotation. Default is
;                  NCOLORS-1.
;
;       MIN:      The minimum data value for the bar annotation. Default is 0.
;
;       NCOLORS:  This is the number of colors in the color bar.
;
;       POSITION: A four-element array of normalized coordinates in the same
;                  form as the POSITION keyword on a plot. Default is
;                  [0.88, 0.15, 0.95, 0.95] for a vertical bar and
;                  [0.15, 0.88, 0.95, 0.95] for a horizontal bar.
;
;       PSCOLOR:  This keyword is only applied if the output is being sent to
;                  a PostScript file. It indicates that the PostScript device
;                  is configured for color output. If this keyword is set, then
;                  the annotation is drawn in the color specified by the COLOR
;                  keyword. If the keyword is not set, the annotation is drawn
;                  in the color specified by the !P.COLOR system variable
;                  (usually this will be the color black). In general, this
;                  gives better looking output on non-color or gray-scale
;                  printers. If you are not specifically setting the annotation
;                  color (with the COLOR keyword), it will probably
;                  be better NOT to set this keyword either, even if you
;                  are outputting to a color PostScript printer.
```

```
;
;       RIGHT:    This puts the labels on the right-hand side of a vertical
;                 color bar. It applies only to vertical color bars.
;
;       TITLE:    This is title for the color bar. The default is to have
;                 no title.
;
;       TOP:      This puts the labels on top of the bar rather than under it.
;                 The keyword only applies if a horizontal color bar is
rendered.
;
;       VERTICAL: Setting this keyword give a vertical color bar. The default
;                 is a horizontal color bar.
;
; COMMON BLOCKS:
;       None.
;
; SIDE EFFECTS:
;       Color bar is drawn in the current graphics window.
;
; RESTRICTIONS:
;       The number of colors available on the display device (not the
;       PostScript device) is used unless the NCOLORS keyword is used.
;
; EXAMPLE:
;       To display a horizontal color bar above a contour plot, type:
;
;       LOADCT, 5, NCOLORS=100
;       CONTOUR, DIST(31,41), POSITION=[0.15, 0.15, 0.95, 0.75], $
;          C_COLORS=INDGEN(25)*4, NLEVELS=25
;       COLORBAR, NCOLORS=100
;
; MODIFICATION HISTORY:
;       Written by: David Fanning, 10 JUNE 96.
;       10/27/96: Added the ability to send output to PostScript. DWF
;       11/4/96: Substantially rewritten to go to screen or PostScript
;           file without having to know much about the PostScript device
;           or even what the current graphics device is. DWF
;       1/27/97: Added the RIGHT and TOP keywords. Also modified the
;            way the TITLE keyword works. DWF
;       7/15/97: Fixed a problem some machines have with plots that have
;            no valid data range in them. DWF
;-

   ; Is the PostScript device selected?

postScriptDevice = (!D.NAME EQ 'PS')

  ; Check and define keywords.

IF N_ELEMENTS(ncolors) EQ 0 THEN BEGIN

   ; Most display devices to not use the 256 colors available to
   ; the PostScript device. This presents a problem when writing
   ; general-purpose programs that can be output to the display or
   ; to the PostScript device. This problem is especially bothersome
   ; if you don't specify the number of colors you are using in the
   ; program. One way to work around this problem is to make the
   ; default number of colors the same for the display device and for
   ; the PostScript device. Then, the colors you see in PostScript are
   ; identical to the colors you see on your display. Here is one way to
```

```
   ; do it.

   IF postScriptDevice THEN BEGIN
      oldDevice = !D.NAME

         ; What kind of computer are we using? SET_PLOT to appropriate
         ; display device.

      thisOS = !VERSION.OS_FAMILY
      thisOS = STRMID(thisOS, 0, 3)
      thisOS = STRUPCASE(thisOS)
      CASE thisOS of
         'MAC': SET_PLOT, thisOS
         'WIN': SET_PLOT, thisOS
         ELSE: SET_PLOT, 'X'
      ENDCASE

         ; Open a window (to make sure !D.N_COLORS is accurate).

      WINDOW, /FREE, /PIXMAP, XSIZE=10, YSIZE=10
      WDELETE, !D.WINDOW

         ; Here is how many colors we should use.

      ncolors = !D.N_COLORS
      SET_PLOT, oldDevice
    ENDIF ELSE ncolors = !D.N_COLORS
ENDIF
IF N_ELEMENTS(bottom) EQ 0 THEN bottom = 0B
IF N_ELEMENTS(charsize) EQ 0 THEN charsize = 1.0
IF N_ELEMENTS(format) EQ 0 THEN format = '(F6.2)'
IF N_ELEMENTS(color) EQ 0 THEN color = ncolors - 1 + bottom
IF N_ELEMENTS(min) EQ 0 THEN min = 0.0
IF N_ELEMENTS(max) EQ 0 THEN max = FLOAT(ncolors) - 1
IF N_ELEMENTS(divisions) EQ 0 THEN divisions = 2
IF N_ELEMENTS(title) EQ 0 THEN title = ''
pscolor = KEYWORD_SET(pscolor)

IF KEYWORD_SET(vertical) THEN BEGIN
   bar = REPLICATE(1B,10) # BINDGEN(256)
   IF N_ELEMENTS(position) EQ 0 THEN position = [0.88, 0.15, 0.95, 0.95]
ENDIF ELSE BEGIN
   bar = BINDGEN(256) # REPLICATE(1B, 10)
   IF N_ELEMENTS(position) EQ 0 THEN position = [0.15, 0.88, 0.95, 0.95]
ENDELSE

   ; Scale the color bar.

 bar = BYTSCL(bar, TOP=ncolors-1) + bottom

   ; Get starting locations in DEVICE coordinates.

xstart = position(0) * !D.X_VSIZE
ystart = position(1) * !D.Y_VSIZE

   ; Get the size of the bar in DEVICE coordinates.

xsize = (position(2) - position(0)) * !D.X_VSIZE
ysize = (position(3) - position(1)) * !D.Y_VSIZE

   ; For PostScript output only, draw the annotation in !P.COLOR
```

```
   ; unless "pscolor" is set. This makes better output on grayscale
   ; printers.

IF postScriptDevice AND (pscolor NE 1) THEN BEGIN
   oldcolor = color
   color = !P.COLOR
ENDIF

   ; Display the color bar in the window. Sizing is
   ; different for PostScript and regular display.

IF postScriptDevice THEN BEGIN

   TV, bar, xstart, ystart, XSIZE=xsize, YSIZE=ysize

ENDIF ELSE BEGIN

   bar = CONGRID(bar, CEIL(xsize), CEIL(ysize), /INTERP)
   TV, bar, xstart, ystart

ENDELSE

   ; Annotate the color bar.

IF KEYWORD_SET(vertical) THEN BEGIN

   IF KEYWORD_SET(right) THEN BEGIN

      PLOT, [min,max], [min,max], /NODATA, XTICKS=1, $
         YTICKS=divisions, XSTYLE=1, YSTYLE=9, $
         POSITION=position, COLOR=color, CHARSIZE=charsize, /NOERASE, $
         YTICKFORMAT='(A1)', XTICKFORMAT='(A1)', YTICKLEN=0.1 , $
         YRANGE=[min, max], YTITLE=title

      AXIS, YAXIS=1, YRANGE=[min, max], YTICKFORMAT=format, YTICKS=divisions,
$
         YTICKLEN=0.1, YSTYLE=1, COLOR=color, CHARSIZE=charsize

   ENDIF ELSE BEGIN

      PLOT, [min,max], [min,max], /NODATA, XTICKS=1, $
         YTICKS=divisions, XSTYLE=1, YSTYLE=9, $
         POSITION=position, COLOR=color, CHARSIZE=charsize, /NOERASE, $
         YTICKFORMAT=format, XTICKFORMAT='(A1)', YTICKLEN=0.1 , $
         YRANGE=[min, max]

      AXIS, YAXIS=1, YRANGE=[min, max], YTICKFORMAT='(A1)', YTICKS=divisions,
$
         YTICKLEN=0.1, YTITLE=title, YSTYLE=1, COLOR=color, CHARSIZE=charsize

   ENDELSE

ENDIF ELSE BEGIN

   IF KEYWORD_SET(top) THEN BEGIN

      PLOT, [min,max], [min,max], /NODATA, XTICKS=divisions, $
         YTICKS=1, XSTYLE=9, YSTYLE=1, $
         POSITION=position, COLOR=color, CHARSIZE=charsize, /NOERASE, $
         YTICKFORMAT='(A1)', XTICKFORMAT='(A1)', XTICKLEN=0.1, $
         XRANGE=[min, max], XTITLE=title
```

```
      AXIS, XTICKS=divisions, XSTYLE=1, COLOR=color, CHARSIZE=charsize, $
         XTICKFORMAT=format, XTICKLEN=0.1, XRANGE=[min, max], XAXIS=1

   ENDIF ELSE BEGIN

      PLOT, [min,max], [min,max], /NODATA, XTICKS=divisions, $
         YTICKS=1, XSTYLE=1, YSTYLE=1, $
         POSITION=position, COLOR=color, CHARSIZE=charsize, /NOERASE, $
         YTICKFORMAT='(A1)', XTICKFORMAT=format, XTICKLEN=0.1, $
         XRANGE=[min, max], TITLE=title

    ENDELSE

ENDELSE

   ; Restore color variable if changed for PostScript.

IF postScriptDevice AND (pscolor NE 1) THEN color = oldcolor

END
```

```
;Copyright (c) 1999, NSU SciViz Lab.  All rights reserved.
;      Unauthorized reproduction prohibited.
;+
; NAME:
;      ez_sage.pro
;
; RESTRICTIONS:
;      Works with SAGE II data, g2p v5.931
;
;-
;*******************************************************************************
**


pro ez_sage

call_procedure, 'ReadSage'

end
```

```
;*****************************************************************************
**
;Copyright (c) 1999, NSU SciViz Lab.  All rights reserved.
;      Unauthorized reproduction prohibited.
;+
; NAME:
;      Get_more_info.pro
;
; RESTRICTIONS:
;      Works with SAGE II data, g2p v5.931
;
;-
;*****************************************************************************
**

pro get_more_info_events, event

;Ignores anything that is not a botton

eventname = tag_names(event, /structure_name)

If eventname ne 'WIDGET_BUTTON' then return

;Does button event.

widget_control, event.top, get_uvalue=info_grid, /no_copy
widget_control, event.id, get_value=buttonvalue

case buttonvalue of

      'Cancel':widget_control, event.top, /destroy

      'Accept': begin

            ;Get the values entered in the fields.
            Widget_Control, info_grid.lat_stepID, get_value=lat_step
            Widget_Control, info_grid.lon_stepID, get_value=lon_step
            Widget_Control, info_grid.smooth_passID, get_value=smooth_pass

            ;Sends then to the pointer
            (*info_grid.ptr_grid).lat_step = lat_step
            (*info_grid.ptr_grid).lon_step = lon_step
            (*info_grid.ptr_grid).smooth_pass = smooth_pass
            (*info_grid.ptr_grid).cancel = 0

            ;Resets variables for passing
            widget_control, event.top, set_uvalue=info_grid

            widget_control, event.top, /destroy

            end
endcase
end
;-----------------------------------------------------------------------------
------------------------
pro get_more_info, surf_type, lat_step, lon_step, smooth_pass, cancel=cancel,
parent=parent

; Error Handler.

On_Error, 2 ;Return to Caller
```

```
;Set defaults
If n_elements(lat_step) eq  0 then lat_step = 8.0
If n_elements(lon_step) eq  0 then lon_step = 10.0
If n_elements(smooth_pass) eq  0 then smooth_pass = 9

; Set dialog in the middle of the screen
device, get_screen_size = screensize
xcenter = fix(screensize[0] / 2.0)
ycenter = fix(screensize[1] / 2.0)
xoff = xcenter - 150
yoff = ycenter - 150

If n_elements(parent) ne  0 then begin
      tlb = widget_base(column=1, xoffset=xoff, yoffset=yoff, title = 'Enter
Regridding and Smoothing Information',$
      /modal, group_leader=parent, /floating)

endif else begin

      tlb = widget_base(column=1, xoffset=xoff, yoffset=yoff, title = 'Enter
Gridding and Smoothing Information')

endelse

case surf_type of

      1: begin

            lat_title ='Enter the latitude bin size.'
            lon_title ='Enter the longitude bin size.'

         end

      2:begin

            lat_title ='Enter the altitude bin size.'
            lon_title ='Enter the latitude bin size.'

         end

      3:begin

            lat_title ='Enter the altitude bin size.'
            lon_title ='Enter the longitude bin size.'

         end

endcase

;Create subbase for text widgets

subbase = widget_base(tlb, column=1, frame=1)
lat_stepID= cw_field(subbase,title=lat_title,value = lat_step, /floating)
lon_stepID= cw_field(subbase,title=lon_title,value = lon_step, /floating)
smooth_passID = cw_field(subbase,title='Enter the number of desired smoothing
passes.',value = smooth_pass, /integer)

;Creates subbase for buttons

butbase = widget_base(tlb, row=1)
```

```
cancel = widget_Button(butbase, value='Cancel')
accept = widget_Button(butbase, value='Accept')

Widget_Control, tlb, /realize

;Set pointer to store information entered

ptr_grid = ptr_new({lat_step:3.0, lon_step:10.0, smooth_pass:9.0, cancel:1})

;Set information on the widget as a user value

info_grid = {lat_stepID:lat_stepID, lon_stepID:lon_stepID, smooth_passID:
smooth_passID, ptr_grid:ptr_grid}

widget_control, tlb, set_uvalue=info_grid, /no_copy

xmanager, 'get_more_info', tlb, event_handler='get_more_info_events'

;Recover the information from the pointer location

fileinfo = *ptr_grid
ptr_free, ptr_grid

;Error handler on entered info

catch, error

If error ne 0 then begin

      ok = dialog_message(!err_string)
      cancel = 1

      return

endif

;User pressed cancel botton.

cancel = fileinfo.cancel
If cancel eq 1 then return

;Reset the parameters.

lat_step = fileinfo.lat_step
lon_step = fileinfo.lon_step
smooth_pass= fileinfo.smooth_pass

return
end
```

```
;Copyright (c) 1999, NSU SciViz Lab.  All rights reserved.
;     Unauthorized reproduction prohibited.
;+
; NAME:
;     get_scale_info.pro
;
; RESTRICTIONS:
;     Works with SAGE II data, g2p v5.931
;
;-
;*****************************************************************************
**


pro get_scale_info_events, event

;Ignores anything that is not a botton

eventname = tag_names(event, /structure_name)


If eventname ne 'WIDGET_BUTTON' then return

;Does button event.

widget_control, event.top, get_uvalue=info, /no_copy
widget_control, event.id, get_value=buttonvalue

case buttonvalue of

      'Cancel': begin

            (*info.ptr).cancel = 1
            widget_control, event.top, /destroy

             end

      'Accept': begin

            ;Get the values entered in the fields.
            Widget_Control, info.x_axis_minID, get_value=x_axis_min
            Widget_Control, info.x_axis_maxID, get_value=x_axis_max
            Widget_Control, info.y_axis_minID, get_value=y_axis_min
            Widget_Control, info.y_axis_maxID, get_value=y_axis_max

            (*info.ptr).x_axis_min = x_axis_min
            (*info.ptr).x_axis_max = x_axis_max
            (*info.ptr).y_axis_min = y_axis_min
            (*info.ptr).y_axis_max = y_axis_max
            (*info.ptr).cancel = 0
            widget_control, event.top, /destroy

            end
endcase
end
;-------------------------------------------------------------------------
-----------------------
pro get_scale_info, x_axis_max, x_axis_min, y_axis_max, y_axis_min, cancel=
cancel, parent=parent

;Gets information from the user in order to give the proper surface
```

```
; Error Handler.

On_Error, 2 ;Return to Caller

If n_elements(x_axis_max) eq  0 then x_axis_max = 180
If n_elements(x_axis_min) eq  0 then x_axis_min= -180
If n_elements(y_axis_min) eq  0 then y_axis_min= 10
If n_elements(y_axis_max) eq  0 then y_axis_max= 70


; Set dialog in the middle of the screen

device, get_screen_size = screensize
xcenter = fix(screensize[0] / 2.0)
ycenter = fix(screensize[1] / 2.0)
xoff = xcenter - 150
yoff = ycenter - 150

If n_elements(parent) ne  0 then begin
      tlb = widget_base(column=1, xoffset=xoff, yoffset=yoff, title = 'Enter
Output Information',$
      /modal, group_leader=parent, /floating)

endif else begin

      tlb = widget_base(column=1, xoffset=xoff, yoffset=yoff, title = 'Enter
Output Information')

endelse

;Create subbase for text widgets

subbase = widget_base(tlb, column=1, frame=1)
x_axis_minID= cw_field(subbase,title='Enter desired x axis minimum.',value =
x_axis_min, /integer)
x_axis_maxID= cw_field(subbase,title='Enter desired x axis maximum.',value =
x_axis_max, /integer)
y_axis_minID= cw_field(subbase,title='Enter desired y axis minimum.',value =
y_axis_min, /floating)
y_axis_maxID= cw_field(subbase,title='Enter desired y axis maximum.',value =
y_axis_max, /floating)

;Creates subbase for buttons

butbase = widget_base(tlb, row=1)
cancel = widget_Button(butbase, value='Cancel')
accept = widget_Button(butbase, value='Accept')

Widget_Control, tlb, /realize

;Set pointer to store information entered

ptr = ptr_new({x_axis_min:x_axis_min, x_axis_max:x_axis_max, y_axis_min:
y_axis_min, y_axis_max:y_axis_max, cancel:1})


info = {x_axis_minID:x_axis_minID, x_axis_maxID:x_axis_maxID, y_axis_minID:
y_axis_minID, y_axis_maxID:y_axis_maxID, ptr:ptr}
```

```
;Set information on the widget as a user value
widget_control, tlb, set_uvalue=info, /no_copy

xmanager, 'get_scale_info', tlb, event_handler='get_scale_info_events'

;Recover the information from the pointer location

fileinfo = *ptr
ptr_free, ptr

;Error handler on entered info

catch, error

If error ne 0 then begin

      ok = dialog_message(!err_string)
      cancel = 1
      return

endif

;User pressed cancel botton.

cancel = fileinfo.cancel
If cancel eq 1 then return

x_axis_min = fileinfo.x_axis_min
x_axis_max = fileinfo.x_axis_max
y_axis_min = fileinfo.y_axis_min
y_axis_max = fileinfo.y_axis_max

return
end
```

```
pro cc_sage

call_procedure, 'ReadSage_cc'

return
end
```

```
;Copyright (c) 1999, NSU SciViz Lab.  All rights reserved.
;     Unauthorized reproduction prohibited.
;+
; NAME:
;     Grid_Smooth_Map.pro
;
; RESTRICTIONS:
;     Works with SAGE II data, g2p v5.931
;
;-
;*****************************************************************************
**


pro Grid_Smooth_Map, outfile, y_step, x_step, smooth_pass, cont_levels,
x_axis_max, x_axis_min, y_axis_max, y_axis_min

;Setting graphics
device, pseudo=8, retain=2

@error_catch

;Background color for plot.
!p.background = 0

;Setting variables
norm_factor = 1e1
unit = '    '
y_unit =1
surf_type = 1
max_val = 9.9999e-1
if n_elements(y_step) eq 0 then y_step = 8.0
if n_elements(x_step) eq 0 then x_step = 10.0
if n_elements(smooth_pass) eq 0 then smooth_pass = 9
If n_elements(cont_levels) eq  0 then cont_levels= findgen(11)
If n_elements(x_axis_max) eq  0 then x_axis_max = 0
If n_elements(x_axis_min) eq  0 then x_axis_min = 0
If n_elements(y_axis_min) eq  0 then y_axis_min = 0
If n_elements(y_axis_max) eq  0 then y_axis_max = 0


;!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!
;If you want to run the program alone uncomment the following commands,!!!!!!!
!!!!!!!!!!!!!!!!!!!
;outfile = dialog_pickfile()
;max_val = 9.999999e-1
;window, /free
;!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!

;Opening file

openr, lun, outfile, /get_lun

;Reading header
header = strarr(21)
readf, lun, header, format='(a36)'

;Reading Normalization Factor, Specie Unit, and Surface Type
reads, header(7), unit
```

```
reads, header(10), y_unit
reads, header(13), norm_factor
reads, header(16), surf_type
reads, header(19), max_val

;Reading data

If (surf_type eq 1) then begin

      datacolumn = fltarr(4, 100000)
      temp_peak = fltarr(4)

      i = 0
      while (not eof(lun))  do begin
            readf, lun, temp_peak
            datacolumn[*,i] = temp_peak
            i = i + 1
      endwhile

      datacolumn = datacolumn[*,0:i-1]

      free_lun, lun

      ;Setting arrays

      date = datacolumn[0,*]
      lat = datacolumn[1,*]
      lon = datacolumn[2,*]
      raw_data = datacolumn[3,*]

endif else begin

      datacolumn = fltarr(5, 80000)
      temp_peak = fltarr(5)

      i = 0
      while (not eof(lun))  do begin
            readf, lun, temp_peak
            datacolumn[*,i] = temp_peak
            i = i + 1
      endwhile

      datacolumn = datacolumn[*,0:i-1]

      free_lun, lun

      ;Setting arrays

      date = datacolumn[0,*]
      lat = datacolumn[1,*]
      lon = datacolumn[2,*]
      raw_data = datacolumn[3,*]
      dataerr = datacolumn[4,*]

endelse

sr = SIZE(raw_data)

;Eliminates unusual data.

dr =sr(1)-1
```

```
er =sr(2)-1

for m = 1, dr do begin

      for n = 1, er do begin

            if (raw_data[m,n] LT 0) then raw_data[m,n]=0
            if (raw_data[m,n] GT max_val) then raw_data[m,n]=0

      endfor

endfor

Case surf_type of

      1: begin

            x = lon
            y = lat

            ;Finds the latitude and longitude range of the data for griding
            xMax = MAX(x)
            xMin = Min(x)
            yMax = MAX(y)
            yMin = Min(y)
            max_raw = max(raw_data)*norm_factor

            If x_axis_max eq  0 then x_axis_max = xmax
            If x_axis_min eq  0 then x_axis_min = xmin
            If y_axis_min eq  0 then y_axis_min = ymin
            If y_axis_max eq  0 then y_axis_max = ymax


            ;Set up the map projection.
            MAP_SET, /CONTINENTS, /MERCATOR, limit=[y_axis_min,x_axis_min,
y_axis_max,x_axis_max],position=[0.01, 0.2, 0.99, 0.99], color=255
            colorbar_position = [0.02, 0.06, 0.98, 0.11]



            ;Grids and interpolates spherical data.

            gridData= SPH_SCAT(x, y, raw_data, BOUNDS=[xMin, yMin, xMax, yMax]
, GS=[x_step,y_step], BOUT=bout)

         end

      2: begin

            x = lat
            y = lon

            ;Finds the range of the data for max min defaults

            xMax = MAX(x)
            xMin = Min(x)
            yMax = MAX(y)
            yMin = Min(y)
            max_raw = max(raw_data)*norm_factor
```

```
            If x_axis_max eq  0 then x_axis_max = xmax
            If x_axis_min eq  0 then x_axis_min = xmin
            If y_axis_min eq  0 then y_axis_min = ymin
            If y_axis_max eq  0 then y_axis_max = ymax

            colorbar_position = [0.1, 0.06, 0.97, 0.11]

            case y_unit of

                  2: plot, x, y, xrange=[x_axis_min,x_axis_max], yrange=
[y_axis_min,y_axis_max], position=[0.1, 0.3, 0.97, 0.99], xtitle ='Latitude',
ytitle='Altitude', color=255,/nodata

                  3: plot, x, y, xrange=[x_axis_min,x_axis_max], yrange=
[y_axis_max,y_axis_min], position=[0.1, 0.3, 0.97, 0.99], xtitle ='Latitude',
ytitle='Altitude', color=255, /ylog, /nodata

                  4: plot, x, y, xrange=[x_axis_min,x_axis_max], yrange=
[y_axis_min,y_axis_max], position=[0.1, 0.3, 0.97, 0.99], xtitle ='Latitude',
ytitle='Altitude', color=255,/nodata

            endcase

            ind_test = where(y gt 0.1)

            If ind_test(0) ne -1 then begin

                  date = date(ind_test)
                  x = x(ind_test)
                  y = y(ind_test)
                  raw_data = raw_data(ind_test)
                  dataerr = dataerr(ind_test)

            endif

            ;Finds the range of the data for griding
            xMax = MAX(x)
            xMin = Min(x)
            yMax = MAX(y)
            yMin = Min(y)

            ;Grids and interpolates non spherical data.

            gridData= normal_SCAT(x, y, raw_data, BOUNDS=[xMin, yMin, xMax,
yMax], GS=[x_step,y_step], BOUT=bout)

         end

      3: begin

            x = lat
            y = lon

            ;Finds the latitude and longitude range of the data for griding

            xMax = MAX(x)
            xMin = Min(x)
            yMax = MAX(y)
            yMin = Min(y)
            max_raw = max(raw_data)*norm_factor
```

```
            If x_axis_max eq  0 then x_axis_max = xmax
            If x_axis_min eq  0 then x_axis_min = xmin
            If y_axis_min eq  0 then y_axis_min = ymin
            If y_axis_max eq  0 then y_axis_max = ymax


            colorbar_position = [0.1, 0.06, 0.97, 0.11]

            case y_unit of

                  2: plot, x, y, xrange=[x_axis_min,x_axis_max], yrange=
[y_axis_min,y_axis_max], position=[0.1, 0.3, 0.97, 0.99], xtitle ='Longitude',
ytitle='Altitude',color=255, /nodata

                  3: plot, x, y, xrange=[x_axis_min,x_axis_max], yrange=
[y_axis_max,y_axis_min], position=[0.1, 0.3, 0.97, 0.99], xtitle ='Longitude',
ytitle='Altitude', color=255, /ylog, /nodata

                  4: plot, x, y, xrange=[x_axis_min,x_axis_max], yrange=
[y_axis_min,y_axis_max], position=[0.1, 0.3, 0.97, 0.99], xtitle ='Longitude',
ytitle='Altitude', color=255,/nodata

            endcase

            ind_test = where(y gt 0.1)

            If ind_test(0) ne -1 then begin

                  date = date(ind_test)
                  x = x(ind_test)
                  y = y(ind_test)
                  raw_data = raw_data(ind_test)
                  dataerr = dataerr(ind_test)

            endif

            ;Finds the range of the data for griding
            xMax = MAX(x)
            xMin = Min(x)
            yMax = MAX(y)
            yMin = Min(y)

            ;Grids and interpolates non spherical data.

            gridData= normal_SCAT(x, y, raw_data, BOUNDS=[xMin, yMin, xMax,
yMax], GS=[x_step,y_step], BOUT=bout)

         end
endcase

s = SIZE(gridData)

;Calculate xlon and ylat vectors corresponding to gridded data.

xx = FINDGEN(s(1))*((bout(2) - bout(0))/(s(1)-1)) + bout(0)
yy = FINDGEN(s(2))*((bout(3) - bout(1))/(s(2)-1)) + bout(1)

;Eliminates Data that is too low or too high resulting from the gridding.

d =s(1)-1
e =s(2)-1
```

```
formatData = fltarr(s(1),s(2))

for m = 1, d do begin
      for n = 1, e do begin
            if (gridData[m,n] LT 0) then gridData[m,n]=0
            if (gridData[m,n] GT max_val) then gridData[m,n]=0
            formatData[m,n] = gridData[m,n]*norm_factor
      endfor
endfor

;Smooths data

If (smooth_pass eq 0) then begin

            Final_data = format_data

endif else begin

      Final_data = smooth(formatData, smooth_pass, /Edge_Truncate)

endelse

;Put the contours on the map.

lab_levels = cont_levels[1:9]

CONTOUR, Final_data, xx, yy, LEVELS=lab_levels,$
         C_COLORS=[23,47,70,93,117,140,163,187,210],cell_fill=1, /overplot ; ,
/closed

;Set up the map projection.

If (surf_type eq 1) then MAP_SET,/CONTINENTS,/MERCATOR,/grid,/label,limit=
[y_axis_min,x_axis_min,y_axis_max,x_axis_max],position=[0.01, 0.2, 0.99, 0.99]
,/noerase,color=255

;Plots the tropopause

      index_trop = where(date eq 000000)
      If (surf_type ne 1) then oplot, x(index_trop), y(index_trop), psym = 6,
symsize=.5, color=255

;Color bar routine.  Keep in mind that the max and min  follow the levels set
at the contour.

call_procedure, 'COLORBAR', divisions=9, Position=colorbar_position, Max=
cont_levels[10], Min=cont_levels[0],  $
Format='(f5.2)',Title=header(1) + unit

;Resets Background
!p.background = 255

RETURN
END
```

```
;Copyright (c) 1999, NSU SciViz Lab.  All rights reserved.
;     Unauthorized reproduction prohibited.
;+
; NAME:
;     Month_Read.pro
;
; RESTRICTIONS:
;     Works with SAGE II data, g2p v5.931
;
;-
;*************************************************************************
**


pro Month_Read, specie_type, surf_type, nummonth, chosen_surf, header, range,
outfile, norm_factor, unit, surf_unit, y_unit, max_value

;specie_type = 1
altitude = chosen_surf
large_val = 0.0
;nummonth = 2
;header = ' '

;Opens file for writing

outfile = Dialog_Pickfile(path = '/usr/people/idluser/Desktop', /write)

;Opening output file

      openw, outlun, outfile, /get_lun

;Writing header

      printf, outlun, ' ',                format='(a1)'
      printf, outlun, header,             format='(a30)'
      printf, outlun, ' ',                format='(a1)'
      printf, outlun, 'Surface Units'
      printf, outlun, surf_unit,          format='(a21)'
      printf, outlun, ' ',                format='(a1)'
      printf, outlun, 'Specie Unit'
      printf, outlun, unit,         format='(a13)'
      printf, outlun, ' ',                format='(a1)'
      printf, outlun, 'Y axis unit = 1=degrees, 2=Km, 3=mb, and 4=K.'
      printf, outlun, y_unit
      printf, outlun, ' ',                format='(a1)'
      printf, outlun, 'Normalization Factor'
      printf, outlun, norm_factor
      printf, outlun, ' ',                format='(a1)
      printf, outlun, 'Surface Type = 1=lon-lat, 2=alt-lat, and 3=alt-lon.'
      printf, outlun, surf_type
      printf, outlun, ' ',                format='(a1)
      printf, outlun, 'Maximun Possible value for this specie.  Utilized to
check data.'
      printf, outlun, max_value
      printf, outlun, ' ',                format='(a1)

for h = 0, (nummonth-1) do begin

      ;Enter the file to be read and studied.

      filename = dialog_pickfile(path = '/usr/people/idluser/Desktop')
```

```
	;DEFINE VARIABLES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

	;--------------------------
	; Include the template
	;--------------------------
	;
########################################################################
	;File - G2PSTRUCT.PRO
	;Date - April 28, 1991
	;
	;Purpose-
	;   Contains the Structure for the SAGE I database records which is in
the
	;   original G2P file format.  This file may be included into an IDL
program
	;   with @G2PSTRUCT as needed.
	;
########################################################################
	;
	; Make some temporary arrays
	tmpf4   = fltarr(4)
	tmpf8   = fltarr(8)
	tmpf25  = fltarr(25)
	tmpf60  = fltarr(60)
	tmpf70  = fltarr(70)

	; Define the structure

	g2p  = {G2P_PROF, $
       ;Field        Type              Description
	DATE:        0.,    $; Event Date (yymmdd.0) at 40 km
	TIME:        0.,    $; Event Time (hhmmss.0) at 40 km
	LAT:         0.,    $; Subtangent Latitude (-90 to 90)
	LON:         0.,    $; Subtangent Longitude (-180 to 180)
	TYPE_INST:   0.,    $; Event Type, Instrument (0=sr, 1=ss)
	NMCTEMP:     tmpf25,     $; NMC Temperature (K) profile
	NMCTERR:     tmpf25,     $; Error in NMC Temperature (K)
	NMCALT:            tmpf25,     $; NMC Geopotential Height (meters)
	NMCDEN:            tmpf25,     $; NMC Air Density profile (g/cm^-3)
	NMCDERR:     tmpf25,     $; Error in NMC Air Density (%)
	BETA:        0.,    $; Spacecraft Beta angle (degrees)
	MET_REV_DATE:      0.,    $; LaRC Met Model Rev Date (yymmdd.0)
	DRIVER_REV: 0.,     $; LaRC Driver Revision Level
	TRANS_REV:   0.,    $; LaRC Transmission Revision Level
	INV_REV:     0.,    $; LaRC Inversion Revision Level
	NMC_FLAG:    0.,    $; LaRC Missing Met Data Flag
	TYPE_LOCAL: 0.,     $; Event Type, Local (0=sr,1=ss)
	MET_CODE:    0.,    $; LaRC Met Model Code
	MET_POINTER:       0.,    $; LaRC Met Model Pointer
	TAG:         0.00, $; LaRC Event Tag
	SBTNALT:     tmpf8,      $; Subtangent altitudes (0,10,20,...,70)
	SBTNLAT:     tmpf8,      $; Subtangent Latitudes
	SBTNLON:     tmpf8,      $; Subtangent Longitudes
	TIMEOFYR:    0.,    $; Time of Year (ddd.fraction)
	TIMESPAN:    0.,    $; Duration of event (seconds)
	METCOR:            tmpf4,      $; NMC Temp corrections (5,2,1,0.4 mb)
	GEOALT:            tmpf70,     $; Geometric Altitudes (.5,1.5,2.5,...,70.
5)
	PRESS:             tmpf70,     $; Pressure profile (mb)
	TEMP:        tmpf70,     $; Tempereature profile (K)
```

```
      SPOD600:     tmpf70,      $; Slant Path Optical Depth @600nm
      SPODE600:    tmpf70,      $; Error in Slant Path Optical Depth @ 600nm
      REXT600:     tmpf70,      $; Rayleigh Extintion @ 600 nm (1/km)
      REXTE600:    tmpf70,      $; Error in Rayleigh Extintion @ 600 nm (1/km)
      O3ND:        tmpf70,      $; O3 number density profile (cm^-3)
      O3NDE:             tmpf70,      $; Error in O3 number density profile (cm^
-3)
      O3MR:        tmpf70,      $; O3 mixing ratio profile
      O3MRE:             tmpf70,      $; Error in O3 mixing ratio profile
      QUALF1000:   0.,    $; Quality factor in 1000nm channel
        QUALF940:        0.,      $; Quality factor in 940nm channel
      QUALF600:    0.,    $; Quality factor in 600nm channel
        QUALF525:        0.,      $; Quality factor in 525nm channel
      QUALF453:    0.,    $; Quality factor in 450nm channel
        QUALF448:        0.,      $; Quality factor in 448nm channel
      QUALF385:    0.,    $; Quality factor in 385nm channel
        MEANSBTALT:      0.,      $; Mean subtngt alt evnt cal (v5.6 only)
      MIRROR_CAL:  0.,    $; Scan Mirror Calibration Code(yymmdd.0)
        FILLVAL:         0.,      $; Fill value (vers 5.6 onwards only)
      SPOD1000:    tmpf60,      $; Slant Path Optical Depth @ 1000nm
      SPODE1000:   tmpf60,      $; Error in Slant Path Optical Depth @ 1000nm
      REXT1000:    tmpf60,      $; Rayleigh Extintion @ 1000 nm (1/km)
      REXTE1000:   tmpf60,      $; Error in Rayleigh Extintion @ 1000 nm (1/km)
        SPOD940:         tmpf60, $; Slant Path Optical Depth @ 940nm
        SPODE940:        tmpf60, $; Error in Slant Path Optical Depth @ 940nm
        REXT940:         tmpf60, $; Rayleigh Extintion @ 940nm (1/km)
        REXTE940:        tmpf60, $; Error in Rayleigh Extintion @ 940 nm (1/km)
        SPOD525:         tmpf60, $; Slant Path Optical Depth @ 525nm
        SPODE525:        tmpf60, $; Error in Slant Path Optical Depth @ 525nm
        REXT525:         tmpf60, $; Rayleigh Extintion @ 525nm (1/km)
        REXTE525:        tmpf60, $; Error in Rayleigh Extintion @ 525 nm (1/km)
        SPOD453:         tmpf60, $; Slant Path Optical Depth @ 453nm
        SPODE453:        tmpf60, $; Error in Slant Path Optical Depth @ 453nm
        REXT453:         tmpf60, $; Rayleigh Extintion @ 453 nm (1/km)
        REXTE453:        tmpf60, $; Error in Rayleigh Extintion @ 453 nm (1/km)
      SPOD448:     tmpf60,      $; Slant Path Optical Depth @ 448nm
      SPODE448:    tmpf60,      $; Error in Slant Path Optical Depth @ 448nm
      REXT448:     tmpf60,      $; Rayleigh Extintion @ 448nm (1/km)
      REXTE448:    tmpf60,      $; Error in Rayleigh Extintion @ 448nm (1/km)
      SPOD385:     tmpf60,      $; Slant Path Optical Depth @ 385nm
      SPODE385:    tmpf60,      $; Error in Slant Path Optical Depth @ 385nm
      REXT385:     tmpf60,      $; Rayleigh Extintion @ 385 nm (1/km)
      REXTE385:    tmpf60,      $; Error in Rayleigh Extintion @ 385 nm (1/km)
        EXT1000:         tmpf60, $; 1000nm Extinction (1/km)
        EXTE1000:        tmpf60, $; Error in 1000nm Extinction (1/km)
        EXT525:          tmpf60, $; 525nm Extinction (1/km)
        EXTE525:         tmpf60, $; Error in 525nm Extinction (1/km)
      EXT450:            tmpf60,      $; 450nm Extinction (1/km)
      EXTE450:     tmpf60,      $; Error in 450nm Extinction (1/km)
        EXT385:          tmpf60, $; 385nm  Extinction (1/km)
        EXTE385:         tmpf60, $; Error in 385nm Extinction (1/km)
      EXTRAT:            tmpf60,      $; 1000nm Extinction Ratio
      EXTRATE:     tmpf60,      $; Error in 1000nm Extinction Ratio
      NO2ND:             tmpf60,      $; NO2 number density profile (cm^-3)
      NO2NDE:            tmpf60,      $; Error in NO2 number density profile
(cm^-3)
      NO2MR:             tmpf60,      $; NO2 mixing ratio profile
      NO2MRE:            tmpf60,      $; Error in NO2 mixing ratio profile
      H2OND:       tmpf60,      $; Spare Vector (H2O Number daensity)
        H2ONDE:          tmpf60, $; Spare Vector (H2O Number density error)
        H2OMR:           tmpf60, $; Spare Vector (H2O Mixing ratio)
```

```
        H2OMRE:         tmpf60, $; Spare Vector (H2O Mixing ratio error)
        TRAN940:        tmpf60, $; Transmission ratio (940/1020 nm)
        TRANER940:      tmpf60, $; Transmission ratio error (940/1020 nm)
        TRAN448:        tmpf60, $; Transmission ratio (448/453 nm)
        TRANER448:      tmpf60, $; Transmission ratio error (448/453 nm)
        CRE_DATE:       0.,     $; Profile creation date (yymmdd.)
        CRE_TIME:       0.  }    ; Profile creation time (hhmmss.)

      ; Define the header index structure

      g2pidx  = {G2P_INDEX,   $
     ; Field           Type                    Description
        DATE1:           0L,     $; Event date1 (yymmdd.0) at 40 km
        TIME:            0L,     $; Event Time (hhmmss.0) at 40 km
        LAT:             0L,     $; Subtangent Latitude (-90 to 90)
        LON:             0L,     $; Subtangent Longitude (-180 to 180)
        EVENT_TYPE:      0L,     $; Event Type, Instrument (0=sr, 1=ss)
        LOW_ALT:         0L,     $; Low Alitude (0 to 30km)
        BETA:            0L,     $; Spacecraft Beta angle (degrees)
        TIME_SPAN:       0L,     $; Time span of event (10-180seconds)
        MET_POINTER:     0L,     $; Array pointer to model met (normally 19)
        SPARE:           0L,     $; Spare value
        SPARE1:          0L,     $; Spare Value
        REC_NUM:         0L  }    ; Record Number where event is stored


      padarray = lonarr(105) ; array to pad the index structure to match the
g2p

      index = REPLICATE({G2P_INDEX},300)
      indrec = {IDX, krec: 0L, profile: bytarr(16), num: 0L, numixt: 0L,
idxtag: index,$
               dummy: padarray }

      ;Open input file

      openr,inlun,filename,/get_lun

      ; Read the four header index records
      ;
      head = assoc(inlun,indrec)

      for k = 0,3 do begin
            indx = head(k)
            ;
            ; This tests for the system that the program runs under
            ; If it is DEC OSF, then a byteorder is necessary to read
            ; the ieee 'big-endian' input data
            ;
            if(!version.os eq 'OSF') then begin

                  byteorder,indx,/LSWAP
                  file = indx.profile
                  byteorder,file,/LSWAP  ; the byte array file name has to be
swapped back

            endif else begin

                  file = indx.profile

            endelse
```

```
      endfor

;Depending on the specie chosen by the user, the program does a case.

Case specie_type of

      0: Return

      1:begin; Aerosol Extintion @ 1000nm vs. lat & lon @ Altitude in Km

            ; Read the g2p data record

            data = assoc(inlun,g2p,59392)

            i = 0
            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data
                  recd = data(i)

                  ;Checks for something
                  if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP

                  endif

                  ;Finds the size of the altitude array
                  s = size(recd.ext1000)
                  alt60 =fltarr(s(1))

                  ;Creates altitude array of appropiate size.
                  for j = 0, (s(1)-1) do begin

                        alt60(j) = recd.geoalt(j)

                  endfor

                  ;Interpolation of the specie to the desired altitude
                  A1000_inter = interpol(recd.ext1000,alt60, altitude)

                  ;Writes result to file.
                  ;if((recd.date ge dateini) and (recd.date le datefnl)) then
begin

                  printf,outlun, recd.date, recd.lat,recd.lon,A1000_inter, $
                  format='(I6, 2f7.1, e14.5)'


                  ;endif

                  i = i + 1

             endwhile

          end

      2:begin; Aerosol Extintion @ 1000nm vs. lat & lon @ pressure
```

```
            ; Read the g2p data record

            data = assoc(inlun,g2p,59392)

            i = 0
            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data
                  recd = data(i)

                  ;Checks for something
                  if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP

                  endif

                  ;Finds the size of the altitude array
                  s = size(recd.ext1000)
                  alt60 =fltarr(s(1))

                  ;Creates altitude array of appropiate size.
                  for j = 0, (s(1)-1) do begin

                        alt60(j) = recd.geoalt(j)

                  endfor


                  ;Interpolation of the specie to the desired pressure
                  A1000_inter = interpol(recd.ext1000, alt60, altitude)

                  ;Writes result to file.

                  ;if((recd.date ge dateini) and (recd.date le datefnl)) then
begin

                        printf,outlun, recd.date, recd.lat,recd.lon,
A1000_inter, $
                        format='(I6, 2f7.1, e14.5)'

                  ;endif

                  i = i + 1

            endwhile

          end

      3:begin; Aereosol Extintion @ 1000nm vs. lat & lon @ zeta

            ; Read the g2p data record
            ;
            i = 0
            data = assoc(inlun,g2p,59392)

            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data
                  recd = data(i)
```

```
                ;Checks for something
                if(!version.os eq 'OSF') then begin

                      byteorder,recd,/LSWAP

                endif

                ;Finds the size of the altitude array
                s = size(recd.ext1000)
                alt60 =fltarr(s(1))

                ;Creates altitude array of appropiate size.
                for j = 0, (s(1)-1) do begin

                      alt60(j) = recd.geoalt(j)

                endfor


                ;Finds the size of the altitude array
                s = size(alt60)
                zeta =fltarr(s(1))

                ;Calculates the Potential Temperatures
                for j = 0, (s(1)-1) do begin

                      zeta(j) = recd.temp(j)*((recd.press(j)/1e3)^(-.286))

                endfor

                ;Interpolation of the specie to the desired Potential
Temperature
                A1000_inter = interpol(recd.ext1000,zeta, altitude)

                ;Writes result to file.
                ;if((recd.date ge dateini) and (recd.date le datefnl)) then
begin

                      printf,outlun, recd.date, recd.lat,recd.lon,
A1000_inter, $
                      format='(I6, 2f7.1, e14.5)'

                ;endif

                i = i + 1

          endwhile

        end

    4:begin; Aereosol Extintion @ 1000nm vs. Altitude in Km & lat @ lon

          ; Read the g2p data record
          ;
          i = 0
          data = assoc(inlun,g2p,59392)

          while(not EOF(inlun)) do begin

                ;Sets recd to the association to the file via data
                recd = data(i)
```

```
                ;Checks for something
                if(!version.os eq 'OSF') then begin

                      byteorder,recd,/LSWAP

                endif

                ;Finds the size of the altitude array
                s = size(recd.ext1000)
                alt60 =fltarr(s(1))

                ;Creates altitude array of appropiate size.
                for j = 0, (s(1)-1) do begin

                      alt60(j) = recd.geoalt(j)

                endfor

                If ((((chosen_surf - recd.lon)^2)^(.5)) le range) then
begin


                      ;Finds the size of the altitude array
                      s = size(alt60)

                      for j = 0, (s(1)-1) do begin

                            ;Writes result to file.
                            printf,outlun, recd.date, recd.lat,alt60(j),
recd.ext1000(j), recd.exte1000(j), $
                            format='(I6, 2f7.1, 2e14.5)'

                      endfor

                      ;Saves tropopause Location
                      tropopause = recd.nmcalt(24)/1000
                      if tropopause lt 30.0 and tropopause gt 5 then $
                      printf,outlun, 000000, recd.lat,tropopause,large_val,
large_val, $
                      format='(I6, 2f7.1, 2e14.5)'

                endif

                i = i + 1

          endwhile

       end

    5:begin; Aereosol Extintion @ 1000nm vs. Altitude in mb & lat @ lon

          ; Read the g2p data record
          ;
          i = 0
          data = assoc(inlun,g2p,59392)

          while(not EOF(inlun)) do begin

                ;Sets recd to the association to the file via data
                recd = data(i)
```

```
                ;Checks for something
                if(!version.os eq 'OSF') then begin

                      byteorder,recd,/LSWAP

                endif

                ;Finds the size of the altitude array
                s = size(recd.ext1000)
                alt60 =fltarr(s(1))

                ;Creates altitude array of appropiate size.
                for j = 0, (s(1)-1) do begin

                      alt60(j) = recd.geoalt(j)

                endfor

                If ((((chosen_surf - recd.lon)^2)^(.5)) le range) then
begin


                      ;Finds the size of the altitude array
                      s = size(alt60)

                      for j = 0, (s(1)-1) do begin

                            ;Writes result to file.
                            printf,outlun, recd.date, recd.lat,recd.press(j)
,recd.ext1000(j), recd.exte1000(j), $
                            format='(I6, 2e14.5, 2e14.5)'

                      endfor

                      ;Saves tropopause Location
                      ;Interpolation of the tropopause in terms of mb
                      trop_temp = recd.nmcalt(24)/1000.0
                      tropopause = interpol(recd.press,recd.geoalt,
trop_temp)

                      printf,outlun, 000000, recd.lat,tropopause,large_val,
large_val, $
                      format='(I6, 2e14.5, 2e14.5)'


                endif

                i = i + 1

           endwhile

        end

     6:begin; Aereosol Extintion @ 1000nm vs. Potential Temperature & lat @
lon

           ; Read the g2p data record
           ;
           i = 0
           data = assoc(inlun,g2p,59392)
```

```
        while(not EOF(inlun)) do begin

              ;Sets recd to the association to the file via data
              recd = data(i)

              ;Checks for something
              if(!version.os eq 'OSF') then begin

                    byteorder,recd,/LSWAP

              endif

              ;Finds the size of the altitude array
              g = size(recd.ext1000)
              alt60 =fltarr(g(1))

              ;Creates altitude array of appropiate size.
              for j = 0, (g(1)-1) do begin

                    alt60(j) = recd.geoalt(j)

              endfor

              If ((((chosen_surf - recd.lon)^2)^(.5)) lt range) then
begin


                    ;Finds the size of the altitude array

                    s = size(alt60)
                    zeta =fltarr(s(1))

                    for j = 0, (s(1)-1) do begin

                          ;Calculates the Potential Temperatures
                          zeta(j) = recd.temp(j)*((recd.press(j)/1e3)^(-.
286))

                          ;Writes result to file.
                          printf,outlun, recd.date, recd.lat,zeta(j),
recd.ext1000(j), recd.exte1000(j), $
                          format='(I6, 2f7.1, 2e14.5)'

                    endfor

                    ;Saves tropopause Location
                    ;Interpolation of the tropopause in terms of
Potential Temperature
                    trop_temp = recd.nmcalt(24)/1000.0
                    tropopause = interpol(zeta,alt60,trop_temp)

                    printf,outlun, 000000, recd.lat,tropopause,large_val,
large_val, $
                    format='(I6, 2f7.1, 2e14.5)'

              endif

              i = i + 1

        endwhile
```

```
        end

      7:begin; Ozone Mixing Ratio vs. Altitude in Km & longitude @ latitude

            ; Read the g2p data record
            ;
            i = 0
            data = assoc(inlun,g2p,59392)

            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data
                  recd = data(i)

                  ;Checks for something
                  if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP

                  endif

                  ;Finds the size of the altitude array
                  g = size(recd.ext1000)
                  alt60 =fltarr(g(1))

                  ;Creates altitude array of appropiate size.
                  for j = 0, (g(1)-1) do begin

                        alt60(j) = recd.geoalt(j)

                  endfor

                  If ((((chosen_surf - recd.lat)^2)^(.5)) lt range) then
begin

                        ;Finds the size of the altitude array
                        s = size(alt60)

                        for j = 0, (s(1)-1) do begin

                              ;Writes result to file.
                              printf,outlun, recd.date, recd.lon,recd.
geoalt(j),recd.ext1000(j), recd.exte1000(j), $
                              format='(I6, 2f7.1, 2e14.5)'

                        endfor

                        ;Saves tropopause Location
                        tropopause = recd.nmcalt(24)/1000
                        printf,outlun, 000000, recd.lon,tropopause,large_val,
large_val, $
                        format='(I6, 2f7.1, 2e14.5)'

                  endif

                  i = i + 1

            endwhile
```

```
        end

      8:begin; Ozone Mixing Ratio vs. Altitude in mb & longitude @ latitude

            ; Read the g2p data record
            ;
            i = 0
            data = assoc(inlun,g2p,59392)

            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data
                  recd = data(i)

                  ;Checks for something
                  if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP

                  endif

                  ;Finds the size of the altitude array
                  g = size(recd.ext1000)
                  alt60 =fltarr(g(1))

                  ;Creates altitude array of appropiate size.
                  for j = 0, (g(1)-1) do begin

                        alt60(j) = recd.geoalt(j)

                  endfor

                  If ((((chosen_surf - recd.lat)^2)^(.5)) lt range) then
begin

                        ;Finds the size of the altitude array
                        s = size(alt60)

                        for j = 0, (s(1)-1) do begin

                              ;Writes result to file.
                              printf,outlun, recd.date, recd.lon,recd.press(j)
,recd.ext1000(j), recd.exte1000(j), $
                              format='(I6, 2e14.5, 2e14.5)'

                        endfor

                        ;Saves tropopause Location
                        ;Interpolation of the tropopause in terms of mb
                        trop_temp = recd.nmcalt(24)/1000.0
                        tropopause = interpol(recd.press,recd.geoalt,
trop_temp)

                        printf,outlun, 000000, recd.lon,tropopause,large_val,
large_val, $
                        format='(I6, 2e14.5, 2e14.5)'

                  endif

                  i = i + 1
```

```
            endwhile

         end

      9:begin; Ozone Mixing Ratio vs. Potential Temperature & longitude @ lat

            ; Read the g2p data record
            ;
            i = 0
            data = assoc(inlun,g2p,59392)

            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data
                  recd = data(i)

                  ;Checks for something
                  if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP

                  endif

                  ;Finds the size of the altitude array
                  g = size(recd.ext1000)
                  alt60 =fltarr(g(1))

                  ;Creates altitude array of appropiate size.
                  for j = 0, (g(1)-1) do begin

                        alt60(j) = recd.geoalt(j)

                  endfor

                  If ((((chosen_surf - recd.lat)^2)^(.5)) lt range) then
begin

                        ;Finds the size of the altitude array
                        s = size(alt60)
                        zeta =fltarr(s(1))

                        for j = 0, (s(1)-1) do begin

                              ;Calculates the Potential Temperatures
                              zeta(j) = recd.temp(j)*((recd.press(j)/1e3)^(-.
286))

                              ;Writes result to file.
                              printf,outlun, recd.date, recd.lon,zeta(j),recd.
ext1000(j), recd.exte1000(j), $
                              format='(I6, 2f7.1, 2e14.5)'

                        endfor

                        ;Saves tropopause Location
                        ;Interpolation of the tropopause in terms of
Potential Temperature
                        trop_temp = recd.nmcalt(24)/1000.0
                        tropopause = interpol(zeta,alt60,trop_temp)

                        printf,outlun, 000000, recd.lon,tropopause,large_val,
```

```
large_val, $
                        format='(I6, 2f7.1, 2e14.5)'

                endif

                i = i + 1

            endwhile

        end

    10:begin; Aerosol Extintion @ 525nm vs. lat & lon @ Altitude in Km

        ; Read the g2p data record

        data = assoc(inlun,g2p,59392)

        i = 0
        while(not EOF(inlun)) do begin

            ;Sets recd to the association to the file via data
            recd = data(i)

            ;Checks for something
            if(!version.os eq 'OSF') then begin

                byteorder,recd,/LSWAP

            endif

            ;Finds the size of the altitude array
            s = size(recd.ext525)
            alt60 =fltarr(s(1))

            ;Creates altitude array of appropiate size.
            for j = 0, (s(1)-1) do begin

                alt60(j) = recd.geoalt(j)

            endfor


            ;Interpolation of the specie to the desired altitude
            A525_inter = interpol(recd.ext525,alt60, altitude)

            ;Writes result to file.
            ;if((recd.date ge dateini) and (recd.date le datefnl)) then
begin

                printf,outlun, recd.date, recd.lat,recd.lon,
A525_inter, $
                format='(I6, 2f7.1, e14.5)'

            ;endif

            i = i + 1

        endwhile

    end
```

```
      11:begin; Aerosol Extintion @ 525nm vs. lat & lon @ pressure

            ; Read the g2p data record

            data = assoc(inlun,g2p,59392)

            i = 0
            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data
                  recd = data(i)

                  ;Checks for something
                  if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP

                  endif

                  ;Finds the size of the altitude array
                  s = size(recd.ext525)
                  alt60 =fltarr(s(1))

                  ;Creates altitude array of appropiate size.
                  for j = 0, (s(1)-1) do begin

                        alt60(j) = recd.geoalt(j)

                  endfor


                  ;Interpolation of the specie to the desired pressure
                  A525_inter = interpol(recd.ext525, alt60, altitude)

                  ;Writes result to file.

                  ;if((recd.date ge dateini) and (recd.date le datefnl)) then
begin

                        printf,outlun, recd.date, recd.lat,recd.lon,
A525_inter, $
                        format='(I6, 2f7.1, e14.5)'

                  ;endif

                  i = i + 1

            endwhile

          end

      12:begin; Aereosol Extintion @ 525nm vs. lat & lon @ zeta

            ; Read the g2p data record
            ;
            i = 0
            data = assoc(inlun,g2p,59392)

            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data
```

```
                recd = data(i)

                ;Checks for something
                if(!version.os eq 'OSF') then begin

                      byteorder,recd,/LSWAP

                endif

                ;Finds the size of the altitude array
                s = size(recd.ext525)
                alt60 =fltarr(s(1))

                ;Creates altitude array of appropiate size.
                for j = 0, (s(1)-1) do begin

                      alt60(j) = recd.geoalt(j)

                endfor


                ;Finds the size of the altitude array
                s = size(alt60)
                zeta =fltarr(s(1))

                ;Calculates the Potential Temperatures
                for j = 0, (s(1)-1) do begin

                      zeta(j) = recd.temp(j)*((recd.press(j)/1e3)^(-.286))

                endfor

                ;Interpolation of the specie to the desired Potential
Temperature
                A525_inter = interpol(recd.ext525,zeta, altitude)

                ;Writes result to file.
                ;if((recd.date ge dateini) and (recd.date le datefnl)) then
begin

                      printf,outlun, recd.date, recd.lat,recd.lon,
A525_inter, $
                      format='(I6, 2f7.1, e14.5)'

                ;endif

                i = i + 1

          endwhile

        end

    13:begin; Aereosol Extintion @ 525nm vs. Altitude in Km & lat @ lon

          ; Read the g2p data record
          ;
          i = 0
          data = assoc(inlun,g2p,59392)

          while(not EOF(inlun)) do begin
```

```
                ;Sets recd to the association to the file via data
                recd = data(i)

                ;Checks for something
                if(!version.os eq 'OSF') then begin

                      byteorder,recd,/LSWAP

                endif

                ;Finds the size of the altitude array
                s = size(recd.ext525)
                alt60 =fltarr(s(1))

                ;Creates altitude array of appropiate size.
                for j = 0, (s(1)-1) do begin

                      alt60(j) = recd.geoalt(j)

                endfor

                If ((((chosen_surf - recd.lon)^2)^(.5)) le range) then
begin


                      ;Finds the size of the altitude array
                      s = size(alt60)

                      for j = 0, (s(1)-1) do begin

                            ;Writes result to file.
                            if recd.ext525(j) lt 1.0 and recd.ext525(j) gt
0.0 then $
                            printf,outlun, recd.date, recd.lat,alt60(j),
recd.ext525(j), recd.exte525(j), $
                            format='(I6, 2f7.1, 2e14.5)'

                      endfor

                      ;Saves tropopause Location
                      tropopause = recd.nmcalt(24)/1000
                      if tropopause lt 30.0 and tropopause gt 5 then $
                      printf,outlun, 000000, recd.lat,tropopause,large_val,
large_val, $
                      format='(I6, 2f7.1, 2e14.5)'

                endif

                i = i + 1

          endwhile

       end

    14:begin; Aereosol Extintion @ 525nm vs. Altitude in mb & lat @ lon

          ; Read the g2p data record
          ;
          i = 0
          data = assoc(inlun,g2p,59392)
```

```
            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data
                  recd = data(i)

                  ;Checks for something
                  if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP

                  endif

                  ;Finds the size of the altitude array
                  s = size(recd.ext525)
                  alt60 =fltarr(s(1))

                  ;Creates altitude array of appropiate size.
                  for j = 0, (s(1)-1) do begin

                        alt60(j) = recd.geoalt(j)

                  endfor

                  If ((((chosen_surf - recd.lon)^2)^(.5)) le range) then
begin


                        ;Finds the size of the altitude array
                        s = size(alt60)

                        for j = 0, (s(1)-1) do begin

                              ;Writes result to file.
                              if recd.ext525(j) lt 1.0 and recd.ext525(j) gt
0.0 then $
                              printf,outlun, recd.date, recd.lat,recd.press(j)
,recd.ext525(j), recd.exte525(j), $
                              format='(I6, 2e14.5, 2e14.5)'

                        endfor

                        ;Saves tropopause Location
                        ;Interpolation of the tropopause in terms of mb
                        trop_temp = recd.nmcalt(24)/1000.0
                        tropopause = interpol(recd.press,recd.geoalt,
trop_temp)
                        if tropopause(0) lt 800 and tropopause(0) gt 50 then $
                        printf,outlun, 000000, recd.lat,tropopause,large_val,
large_val, $
                        format='(I6, 2e14.5, 2e14.5)'


                  endif

                  i = i + 1

             endwhile

          end

      15:begin; Aereosol Extintion @ 525nm vs. Potential Temperature & lat @
```

```
lon

            ; Read the g2p data record
            ;
            i = 0
            data = assoc(inlun,g2p,59392)

            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data
                  recd = data(i)

                  ;Checks for something
                  if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP

                  endif

                  ;Finds the size of the altitude array
                  g = size(recd.ext525)
                  alt60 =fltarr(g(1))

                  ;Creates altitude array of appropiate size.
                  for j = 0, (g(1)-1) do begin

                        alt60(j) = recd.geoalt(j)

                  endfor

                  If ((((chosen_surf - recd.lon)^2)^(.5)) lt range) then
begin


                        ;Finds the size of the altitude array

                        s = size(alt60)
                        zeta =fltarr(s(1))

                        for j = 0, (s(1)-1) do begin

                              ;Calculates the Potential Temperatures
                              zeta(j) = recd.temp(j)*((recd.press(j)/1e3)^(-.
286))

                              ;Writes result to file.
                              if recd.ext525(j) lt 1.0 and recd.ext525(j) gt
0.0 then $
                              printf,outlun, recd.date, recd.lat,zeta(j),
recd.ext525(j), recd.exte525(j), $
                              format='(I6, 2f7.1, 2e14.5)'

                        endfor

                        ;Saves tropopause Location
                        ;Interpolation of the tropopause in terms of
Potential Temperature
                        trop_temp = recd.nmcalt(24)/1000.0
                        tropopause = interpol(zeta,alt60,trop_temp)

                        printf,outlun, 000000, recd.lat,tropopause,large_val,
```

```
large_val, $
                        format='(I6, 2f7.1, 2e14.5)'

                endif

                i = i + 1

            endwhile

        end

    16:begin; Ozone Mixing Ratio vs. Altitude in Km & longitude @ latitude

            ; Read the g2p data record
            ;
            i = 0
            data = assoc(inlun,g2p,59392)

            while(not EOF(inlun)) do begin

                ;Sets recd to the association to the file via data
                recd = data(i)

                ;Checks for something
                if(!version.os eq 'OSF') then begin

                    byteorder,recd,/LSWAP

                endif

                ;Finds the size of the altitude array
                g = size(recd.ext525)
                alt60 =fltarr(g(1))

                ;Creates altitude array of appropiate size.
                for j = 0, (g(1)-1) do begin

                    alt60(j) = recd.geoalt(j)

                endfor

                If ((((chosen_surf - recd.lat)^2)^(.5)) lt range) then
begin

                    ;Finds the size of the altitude array
                    s = size(alt60)

                    for j = 0, (s(1)-1) do begin

                        ;Writes result to file.
                        if recd.ext525(j) lt 1.0 and recd.ext525(j) gt
0.0 then $
                        printf,outlun, recd.date, recd.lon,recd.
geoalt(j),recd.ext525(j), recd.exte525(j), $
                        format='(I6, 2f7.1, 2e14.5)'

                    endfor

                    ;Saves tropopause Location
                    tropopause = recd.nmcalt(24)/1000
                    if tropopause lt 30.0 and tropopause gt 5 then $
```

```
                        printf,outlun, 000000, recd.lon,tropopause,large_val,
large_val, $
                        format='(I6, 2f7.1, 2e14.5)'

                  endif

                  i = i + 1

            endwhile

         end

      17:begin; Ozone Mixing Ratio vs. Altitude in mb & longitude @ latitude

            ; Read the g2p data record
            ;
            i = 0
            data = assoc(inlun,g2p,59392)

            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data
                  recd = data(i)

                  ;Checks for something
                  if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP

                  endif

                  ;Finds the size of the altitude array
                  g = size(recd.ext525)
                  alt60 =fltarr(g(1))

                  ;Creates altitude array of appropiate size.
                  for j = 0, (g(1)-1) do begin

                        alt60(j) = recd.geoalt(j)

                  endfor

                  If ((((chosen_surf - recd.lat)^2)^(.5)) lt range) then
begin

                        ;Finds the size of the altitude array
                        s = size(alt60)

                        for j = 0, (s(1)-1) do begin

                              ;Writes result to file.
                              if recd.ext525(j) lt 1.0 and recd.ext525(j) gt
0.0 then $
                              printf,outlun, recd.date, recd.lon,recd.press(j)
,recd.ext525(j), recd.exte525(j), $
                              format='(I6, 2e14.5, 2e14.5)'

                        endfor

                        ;Saves tropopause Location
                        ;Interpolation of the tropopause in terms of mb
```

```
                        trop_temp = recd.nmcalt(24)/1000.0
                        tropopause = interpol(recd.press,recd.geoalt,
trop_temp)

                        printf,outlun, 000000, recd.lon,tropopause,large_val,
large_val, $
                        format='(I6, 2e14.5, 2e14.5)'

                endif

                i = i + 1

            endwhile

        end

    18:begin; Ozone Mixing Ratio vs. Potential Temperature & longitude @ lat

            ; Read the g2p data record
            ;
            i = 0
            data = assoc(inlun,g2p,59392)

            while(not EOF(inlun)) do begin

                ;Sets recd to the association to the file via data
                recd = data(i)

                ;Checks for something
                if(!version.os eq 'OSF') then begin

                    byteorder,recd,/LSWAP

                endif

                ;Finds the size of the altitude array
                g = size(recd.ext525)
                alt60 =fltarr(g(1))

                ;Creates altitude array of appropiate size.
                for j = 0, (g(1)-1) do begin

                    alt60(j) = recd.geoalt(j)

                endfor

                If ((((chosen_surf - recd.lat)^2)^(.5)) lt range) then
begin

                    ;Finds the size of the altitude array
                    s = size(alt60)
                    zeta =fltarr(s(1))

                    for j = 0, (s(1)-1) do begin

                        ;Calculates the Potential Temperatures
                        zeta(j) = recd.temp(j)*((recd.press(j)/1e3)^(-.
286))

                        ;Writes result to file.
                        if recd.ext525(j) lt 1.0 and recd.ext525(j) gt
```

```
0.0 then $
                              printf,outlun, recd.date, recd.lon,zeta(j),recd.
ext525(j), recd.exte525(j), $
                              format='(I6, 2f7.1, 2e14.5)'

                        endfor

                        ;Saves tropopause Location
                        ;Interpolation of the tropopause in terms of
Potential Temperature
                        trop_temp = recd.nmcalt(24)/1000.0
                        tropopause = interpol(zeta,alt60,trop_temp)

                        printf,outlun, 000000, recd.lon,tropopause,large_val,
large_val, $
                        format='(I6, 2f7.1, 2e14.5)'

                endif

                i = i + 1

            endwhile

        end

    19:begin; Aerosol Extintion @ 450nm vs. lat & lon @ Altitude in Km

        ; Read the g2p data record

        data = assoc(inlun,g2p,59392)

        i = 0
        while(not EOF(inlun)) do begin

            ;Sets recd to the association to the file via data
            recd = data(i)

            ;Checks for something
            if(!version.os eq 'OSF') then begin

                byteorder,recd,/LSWAP

            endif

            ;Finds the size of the altitude array
            s = size(recd.ext450)
            alt60 =fltarr(s(1))

            ;Creates altitude array of appropiate size.
            for j = 0, (s(1)-1) do begin

                alt60(j) = recd.geoalt(j)

            endfor


            ;Interpolation of the specie to the desired altitude
            A450_inter = interpol(recd.ext450,alt60, altitude)

            ;Writes result to file.
            ;if((recd.date ge dateini) and (recd.date le datefnl)) then
```

```
begin

                        printf,outlun, recd.date, recd.lat,recd.lon,
A450_inter, $
                        format='(I6, 2f7.1, e14.5)'

                  ;endif

                  i = i + 1

            endwhile

          end

      20:begin; Aerosol Extintion @ 450nm vs. lat & lon @ pressure

            ; Read the g2p data record

            data = assoc(inlun,g2p,59392)

            i = 0
            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data
                  recd = data(i)

                  ;Checks for something
                  if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP

                  endif

                  ;Finds the size of the altitude array
                  s = size(recd.ext450)
                  alt60 =fltarr(s(1))

                  ;Creates altitude array of appropiate size.
                  for j = 0, (s(1)-1) do begin

                        alt60(j) = recd.geoalt(j)

                  endfor


                  ;Interpolation of the specie to the desired pressure
                  A450_inter = interpol(recd.ext450, alt60, altitude)

                  ;Writes result to file.

                  ;if((recd.date ge dateini) and (recd.date le datefnl)) then
begin

                        printf,outlun, recd.date, recd.lat,recd.lon,
A450_inter, $
                        format='(I6, 2f7.1, e14.5)'

                  ;endif

                  i = i + 1
```

```
            endwhile

          end

      21:begin; Aereosol Extintion @ 450nm vs. lat & lon @ zeta

            ; Read the g2p data record
            ;
            i = 0
            data = assoc(inlun,g2p,59392)

            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data
                  recd = data(i)

                  ;Checks for something
                  if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP

                  endif

                  ;Finds the size of the altitude array
                  s = size(recd.ext450)
                  alt60 =fltarr(s(1))

                  ;Creates altitude array of appropiate size.
                  for j = 0, (s(1)-1) do begin

                        alt60(j) = recd.geoalt(j)

                  endfor


                  ;Finds the size of the altitude array
                  s = size(alt60)
                  zeta =fltarr(s(1))

                  ;Calculates the Potential Temperatures
                  for j = 0, (s(1)-1) do begin

                        zeta(j) = recd.temp(j)*((recd.press(j)/1e3)^(-.286))

                  endfor

                  ;Interpolation of the specie to the desired Potential
Temperature
                  A450_inter = interpol(recd.ext450,zeta, altitude)

                  ;Writes result to file.
                  ;if((recd.date ge dateini) and (recd.date le datefnl)) then
begin

                        printf,outlun, recd.date, recd.lat,recd.lon,
A450_inter, $
                        format='(I6, 2f7.1, e14.5)'

                  ;endif

                  i = i + 1
```

```
        endwhile

      end

22:begin; Aereosol Extintion @ 450nm vs. Altitude in Km & lat @ lon

        ; Read the g2p data record
        ;
        i = 0
        data = assoc(inlun,g2p,59392)

        while(not EOF(inlun)) do begin

              ;Sets recd to the association to the file via data
              recd = data(i)

              ;Checks for something
              if(!version.os eq 'OSF') then begin

                    byteorder,recd,/LSWAP

              endif

              ;Finds the size of the altitude array
              s = size(recd.ext450)
              alt60 =fltarr(s(1))

              ;Creates altitude array of appropiate size.
              for j = 0, (s(1)-1) do begin

                    alt60(j) = recd.geoalt(j)

              endfor

              If ((((chosen_surf - recd.lon)^2)^(.5)) le range) then
begin


                    ;Finds the size of the altitude array
                    s = size(alt60)

                    for j = 0, (s(1)-1) do begin

                          ;Writes result to file.
                          if recd.ext450(j) lt 1.0 and recd.ext450(j) gt
0.0 then $
                          printf,outlun, recd.date, recd.lat,alt60(j),
recd.ext450(j), recd.exte450(j), $
                          format='(I6, 2f7.1, 2e14.5)'

                    endfor

                    ;Saves tropopause Location
                    tropopause = recd.nmcalt(24)/1000
                    printf,outlun, 000000, recd.lat,tropopause,large_val,
large_val, $
                    format='(I6, 2f7.1, 2e14.5)'

              endif
```

```
                i = i + 1

            endwhile

         end

      23:begin; Aereosol Extintion @ 450nm vs. Altitude in mb & lat @ lon

            ; Read the g2p data record
            ;
            i = 0
            data = assoc(inlun,g2p,59392)

            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data
                  recd = data(i)

                  ;Checks for something
                  if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP

                  endif

                  ;Finds the size of the altitude array
                  s = size(recd.ext450)
                  alt60 =fltarr(s(1))

                  ;Creates altitude array of appropiate size.
                  for j = 0, (s(1)-1) do begin

                        alt60(j) = recd.geoalt(j)

                  endfor

                  If ((((chosen_surf - recd.lon)^2)^(.5)) le range) then
begin


                        ;Finds the size of the altitude array
                        s = size(alt60)

                        for j = 0, (s(1)-1) do begin

                              ;Writes result to file.
                              if recd.ext450(j) lt 1.0 and recd.ext450(j) gt
0.0 then $
                              printf,outlun, recd.date, recd.lat,recd.press(j)
,recd.ext450(j), recd.exte450(j), $
                              format='(I6, 2e14.5, 2e14.5)'

                        endfor

                        ;Saves tropopause Location
                        ;Interpolation of the tropopause in terms of mb
                        trop_temp = recd.nmcalt(24)/1000.0
                        tropopause = interpol(recd.press,recd.geoalt,
trop_temp)

                        printf,outlun, 000000, recd.lat,tropopause,large_val,
```

```
large_val, $
                        format='(I6, 2e14.5, 2e14.5)'

                endif

                i = i + 1

            endwhile

        end

    24:begin; Aereosol Extintion @ 450nm vs. Potential Temperature & lat @
lon

            ; Read the g2p data record
            ;
            i = 0
            data = assoc(inlun,g2p,59392)

            while(not EOF(inlun)) do begin

                ;Sets recd to the association to the file via data
                recd = data(i)

                ;Checks for something
                if(!version.os eq 'OSF') then begin

                    byteorder,recd,/LSWAP

                endif

                ;Finds the size of the altitude array
                g = size(recd.ext450)
                alt60 =fltarr(g(1))

                ;Creates altitude array of appropiate size.
                for j = 0, (g(1)-1) do begin

                    alt60(j) = recd.geoalt(j)

                endfor

                If ((((chosen_surf - recd.lon)^2)^(.5)) lt range) then
begin


                    ;Finds the size of the altitude array

                    s = size(alt60)
                    zeta =fltarr(s(1))

                    for j = 0, (s(1)-1) do begin

                        ;Calculates the Potential Temperatures
                        zeta(j) = recd.temp(j)*((recd.press(j)/1e3)^(-.
286))


                        ;Writes result to file.
                        if recd.ext450(j) lt 1.0 and recd.ext450(j) gt
0.0 then $
```

```
                              printf,outlun, recd.date, recd.lat,zeta(j),
recd.ext450(j), recd.exte450(j), $
                              format='(I6, 2f7.1, 2e14.5)'

                        endfor

                        ;Saves tropopause Location
                        ;Interpolation of the tropopause in terms of
Potential Temperature
                        trop_temp = recd.nmcalt(24)/1000.0
                        tropopause = interpol(zeta,alt60,trop_temp)

                        printf,outlun, 000000, recd.lat,tropopause,large_val,
large_val, $
                        format='(I6, 2f7.1, 2e14.5)'

                endif

                i = i + 1

            endwhile

        end


    25:begin; Ozone Mixing Ratio vs. Altitude in Km & longitude @ latitude

        ; Read the g2p data record
        ;
        i = 0
        data = assoc(inlun,g2p,59392)

        while(not EOF(inlun)) do begin

            ;Sets recd to the association to the file via data
            recd = data(i)

            ;Checks for something
            if(!version.os eq 'OSF') then begin

                byteorder,recd,/LSWAP

            endif

            ;Finds the size of the altitude array
            g = size(recd.ext450)
            alt60 =fltarr(g(1))

            ;Creates altitude array of appropiate size.
            for j = 0, (g(1)-1) do begin

                alt60(j) = recd.geoalt(j)

            endfor

            If ((((chosen_surf - recd.lat)^2)^(.5)) lt range) then
begin

                ;Finds the size of the altitude array
                s = size(alt60)
```

```
                        for j = 0, (s(1)-1) do begin

                              ;Writes result to file.
                              if recd.ext450(j) lt 1.0 and recd.ext450(j) gt
0.0 then $
                              printf,outlun, recd.date, recd.lon,recd.
geoalt(j),recd.ext450(j), recd.exte450(j), $
                              format='(I6, 2f7.1, 2e14.5)'

                        endfor

                        ;Saves tropopause Location
                        tropopause = recd.nmcalt(24)/1000
                        printf,outlun, 000000, recd.lon,tropopause,large_val,
large_val, $
                        format='(I6, 2f7.1, 2e14.5)'

                  endif

                  i = i + 1

            endwhile

         end

      26:begin; Ozone Mixing Ratio vs. Altitude in mb & longitude @ latitude

            ; Read the g2p data record
            ;
            i = 0
            data = assoc(inlun,g2p,59392)

            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data
                  recd = data(i)

                  ;Checks for something
                  if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP

                  endif

                  ;Finds the size of the altitude array
                  g = size(recd.ext450)
                  alt60 =fltarr(g(1))

                  ;Creates altitude array of appropiate size.
                  for j = 0, (g(1)-1) do begin

                        alt60(j) = recd.geoalt(j)

                  endfor

                  If ((((chosen_surf - recd.lat)^2)^(.5)) lt range) then
begin

                        ;Finds the size of the altitude array
                        s = size(alt60)
```

```
                        for j = 0, (s(1)-1) do begin

                              ;Writes result to file.
                              if recd.ext450(j) lt 1.0 and recd.ext450(j) gt
0.0 then $
                              printf,outlun, recd.date, recd.lon,recd.press(j)
,recd.ext450(j), recd.exte450(j), $
                              format='(I6, 2e14.5, 2e14.5)'

                        endfor

                        ;Saves tropopause Location
                        ;Interpolation of the tropopause in terms of mb
                        trop_temp = recd.nmcalt(24)/1000.0
                        tropopause = interpol(recd.press,recd.geoalt,
trop_temp)

                        printf,outlun, 000000, recd.lon,tropopause,large_val,
large_val, $
                        format='(I6, 2e14.5, 2e14.5)'

                  endif

                  i = i + 1

            endwhile

         end

      27:begin; Ozone Mixing Ratio vs. Potential Temperature & longitude @ lat

            ; Read the g2p data record
            ;
            i = 0
            data = assoc(inlun,g2p,59392)

            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data
                  recd = data(i)

                  ;Checks for something
                  if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP

                  endif

                  ;Finds the size of the altitude array
                  g = size(recd.ext450)
                  alt60 =fltarr(g(1))

                  ;Creates altitude array of appropiate size.
                  for j = 0, (g(1)-1) do begin

                        alt60(j) = recd.geoalt(j)

                  endfor

                  If ((((chosen_surf - recd.lat)^2)^(.5)) lt range) then
begin
```

```
                ;Creates altitude array of appropiate size.
                for j = 0, (s(1)-1) do begin

                      alt60(j) = recd.geoalt(j)

                endfor


                ;Interpolation of the specie to the desired altitude
                A385_inter = interpol(recd.ext385,alt60, altitude)

                ;Writes result to file.
                ;if((recd.date ge dateini) and (recd.date le datefnl)) then
begin

                      printf,outlun, recd.date, recd.lat,recd.lon,
A385_inter, $
                      format='(I6, 2f7.1, e14.5)'

                ;endif

                i = i + 1

          endwhile

        end

     29:begin; Aerosol Extintion @ 385nm vs. lat & lon @ pressure

          ; Read the g2p data record

          data = assoc(inlun,g2p,59392)

          i = 0
          while(not EOF(inlun)) do begin

                ;Sets recd to the association to the file via data
                recd = data(i)

                ;Checks for something
                if(!version.os eq 'OSF') then begin

                      byteorder,recd,/LSWAP

                endif

                ;Finds the size of the altitude array
                s = size(recd.ext385)
                alt60 =fltarr(s(1))

                ;Creates altitude array of appropiate size.
                for j = 0, (s(1)-1) do begin

                      alt60(j) = recd.geoalt(j)

                endfor


                ;Interpolation of the specie to the desired pressure
                A385_inter = interpol(recd.ext385, alt60, altitude)
```

```
                ;Writes result to file.
                ;if((recd.date ge dateini) and (recd.date le datefnl)) then
begin
                      printf,outlun, recd.date, recd.lat,recd.lon,
A385_inter, $
                      format='(I6, 2f7.1, e14.5)'
                ;endif
                i = i + 1
          endwhile
        end
    30:begin; Aereosol Extintion @ 385nm vs. lat & lon @ zeta
          ; Read the g2p data record
          ;
          i = 0
          data = assoc(inlun,g2p,59392)
          while(not EOF(inlun)) do begin
                ;Sets recd to the association to the file via data
                recd = data(i)
                ;Checks for something
                if(!version.os eq 'OSF') then begin
                      byteorder,recd,/LSWAP
                endif
                ;Finds the size of the altitude array
                s = size(recd.ext385)
                alt60 =fltarr(s(1))
                ;Creates altitude array of appropiate size.
                for j = 0, (s(1)-1) do begin
                      alt60(j) = recd.geoalt(j)
                endfor

                ;Finds the size of the altitude array
                s = size(alt60)
                zeta =fltarr(s(1))
                ;Calculates the Potential Temperatures
                for j = 0, (s(1)-1) do begin
                      zeta(j) = recd.temp(j)*((recd.press(j)/1e3)^(-.286))
                endfor
                ;Interpolation of the specie to the desired Potential
 Temperature
```

```
                A385_inter = interpol(recd.ext385,zeta, altitude)

                ;Writes result to file.
                ;if((recd.date ge dateini) and (recd.date le datefnl)) then
begin

                      printf,outlun, recd.date, recd.lat,recd.lon,
A385_inter, $
                      format='(I6, 2f7.1, e14.5)'

                ;endif

                i = i + 1

          endwhile

        end

    31:begin; Aereosol Extintion @ 385nm vs. Altitude in Km & lat @ lon

          ; Read the g2p data record
          ;
          i = 0
          data = assoc(inlun,g2p,59392)

          while(not EOF(inlun)) do begin

                ;Sets recd to the association to the file via data
                recd = data(i)

                ;Checks for something
                if(!version.os eq 'OSF') then begin

                      byteorder,recd,/LSWAP

                endif

                ;Finds the size of the altitude array
                s = size(recd.ext385)
                alt60 =fltarr(s(1))

                ;Creates altitude array of appropiate size.
                for j = 0, (s(1)-1) do begin

                      alt60(j) = recd.geoalt(j)

                endfor

                If ((((chosen_surf - recd.lon)^2)^(.5)) le range) then
begin


                      ;Finds the size of the altitude array
                      s = size(alt60)

                      for j = 0, (s(1)-1) do begin

                            ;Writes result to file.
                            if recd.ext385(j) lt 1.0 and recd.ext385(j) gt
0.0 then $
                            printf,outlun, recd.date, recd.lat,alt60(j),
```

```
recd.ext385(j), recd.exte385(j), $
                                format='(I6, 2f7.1, 2e14.5)'

                        endfor

                        ;Saves tropopause Location
                        tropopause = recd.nmcalt(24)/1000
                        printf,outlun, 000000, recd.lat,tropopause,large_val,
large_val, $
                        format='(I6, 2f7.1, 2e14.5)'

                endif

                i = i + 1

            endwhile

         end

      32:begin; Aereosol Extintion @ 385nm vs. Altitude in mb & lat @ lon

            ; Read the g2p data record
            ;
            i = 0
            data = assoc(inlun,g2p,59392)

            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data
                  recd = data(i)

                  ;Checks for something
                  if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP

                  endif

                  ;Finds the size of the altitude array
                  s = size(recd.ext385)
                  alt60 =fltarr(s(1))

                  ;Creates altitude array of appropiate size.
                  for j = 0, (s(1)-1) do begin

                        alt60(j) = recd.geoalt(j)

                  endfor

                  If ((((chosen_surf - recd.lon)^2)^(.5)) le range) then
begin


                        ;Finds the size of the altitude array
                        s = size(alt60)

                        for j = 0, (s(1)-1) do begin

                              ;Writes result to file.
                              if recd.ext385(j) lt 1.0 and recd.ext385(j) gt
0.0 then $
```

```
                              printf,outlun, recd.date, recd.lat,recd.press(j)
,recd.ext385(j), recd.exte385(j), $
                              format='(I6, 2e14.5, 2e14.5)'

                        endfor

                        ;Saves tropopause Location
                        ;Interpolation of the tropopause in terms of mb
                        trop_temp = recd.nmcalt(24)/1000.0
                        tropopause = interpol(recd.press,recd.geoalt,
trop_temp)

                        printf,outlun, 000000, recd.lat,tropopause,large_val,
large_val, $
                        format='(I6, 2e14.5, 2e14.5)'


                  endif

                  i = i + 1

            endwhile

         end

      33:begin; Aereosol Extintion @ 385nm vs. Potential Temperature & lat @
lon

            ; Read the g2p data record
            ;
            i = 0
            data = assoc(inlun,g2p,59392)

            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data
                  recd = data(i)

                  ;Checks for something
                  if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP

                  endif

                  ;Finds the size of the altitude array
                  g = size(recd.ext385)
                  alt60 =fltarr(g(1))

                  ;Creates altitude array of appropiate size.
                  for j = 0, (g(1)-1) do begin

                        alt60(j) = recd.geoalt(j)

                  endfor

                  If ((((chosen_surf - recd.lon)^2)^(.5)) lt range) then
begin


                        ;Finds the size of the altitude array
```

```
                        s = size(alt60)
                        zeta =fltarr(s(1))

                        for j = 0, (s(1)-1) do begin

                              ;Calculates the Potential Temperatures
                              zeta(j) = recd.temp(j)*((recd.press(j)/1e3)^(-.
286))

                              ;Writes result to file.
                              if recd.ext385(j) lt 1.0 and recd.ext385(j) gt
0.0 then $
                              printf,outlun, recd.date, recd.lat,zeta(j),
recd.ext385(j), recd.exte385(j), $
                              format='(I6, 2f7.1, 2e14.5)'

                        endfor

                        ;Saves tropopause Location
                        ;Interpolation of the tropopause in terms of
Potential Temperature
                        trop_temp = recd.nmcalt(24)/1000.0
                        tropopause = interpol(zeta,alt60,trop_temp)

                        printf,outlun, 000000, recd.lat,tropopause,large_val,
large_val, $
                        format='(I6, 2f7.1, 2e14.5)'

                  endif

                  i = i + 1

            endwhile

      end


34:begin; Ozone Mixing Ratio vs. Altitude in Km & longitude @ latitude

      ; Read the g2p data record
      ;
      i = 0
      data = assoc(inlun,g2p,59392)

      while(not EOF(inlun)) do begin

            ;Sets recd to the association to the file via data
            recd = data(i)

            ;Checks for something
            if(!version.os eq 'OSF') then begin

                  byteorder,recd,/LSWAP

            endif

            ;Finds the size of the altitude array
            g = size(recd.ext385)
            alt60 =fltarr(g(1))
```

```
                ;Creates altitude array of appropiate size.
                for j = 0, (g(1)-1) do begin

                      alt60(j) = recd.geoalt(j)

                endfor

                If ((((chosen_surf - recd.lat)^2)^(.5)) lt range) then
begin

                      ;Finds the size of the altitude array
                      s = size(alt60)

                      for j = 0, (s(1)-1) do begin

                            ;Writes result to file.
                            if recd.ext385(j) lt 1.0 and recd.ext385(j) gt
0.0 then $
                            printf,outlun, recd.date, recd.lon,recd.
geoalt(j),recd.ext385(j), recd.exte385(j), $
                            format='(I6, 2f7.1, 2e14.5)'

                      endfor

                      ;Saves tropopause Location
                      tropopause = recd.nmcalt(24)/1000
                      printf,outlun, 000000, recd.lon,tropopause,large_val,
large_val, $
                      format='(I6, 2f7.1, 2e14.5)'

                endif

                i = i + 1

          endwhile

       end

     35:begin; Ozone Mixing Ratio vs. Altitude in mb & longitude @ latitude

          ; Read the g2p data record
          ;
          i = 0
          data = assoc(inlun,g2p,59392)

          while(not EOF(inlun)) do begin

                ;Sets recd to the association to the file via data
                recd = data(i)

                ;Checks for something
                if(!version.os eq 'OSF') then begin

                      byteorder,recd,/LSWAP

                endif

                ;Finds the size of the altitude array
                g = size(recd.ext385)
                alt60 =fltarr(g(1))
```

```
                ;Creates altitude array of appropiate size.
                for j = 0, (g(1)-1) do begin

                        alt60(j) = recd.geoalt(j)

                endfor

                If ((((chosen_surf - recd.lat)^2)^(.5)) lt range) then
begin

                        ;Finds the size of the altitude array
                        s = size(alt60)

                        for j = 0, (s(1)-1) do begin

                                ;Writes result to file.
                                if recd.ext385(j) lt 1.0 and recd.ext385(j) gt
0.0 then $
                                printf,outlun, recd.date, recd.lon,recd.press(j)
,recd.ext385(j), recd.exte385(j), $
                                format='(I6, 2e14.5, 2e14.5)'

                        endfor

                        ;Saves tropopause Location
                        ;Interpolation of the tropopause in terms of mb
                        trop_temp = recd.nmcalt(24)/1000.0
                        tropopause = interpol(recd.press,recd.geoalt,
trop_temp)

                        printf,outlun, 000000, recd.lon,tropopause,large_val,
large_val, $
                        format='(I6, 2e14.5, 2e14.5)'

                endif

                i = i + 1

            endwhile

        end

      36:begin; Ozone Mixing Ratio vs. Potential Temperature & longitude @ lat

            ; Read the g2p data record
            ;
            i = 0
            data = assoc(inlun,g2p,59392)

            while(not EOF(inlun)) do begin

                ;Sets recd to the association to the file via data
                recd = data(i)

                ;Checks for something
                if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP

                endif
```

```
                ;Finds the size of the altitude array
                g = size(recd.ext385)
                alt60 =fltarr(g(1))

                ;Creates altitude array of appropiate size.
                for j = 0, (g(1)-1) do begin

                      alt60(j) = recd.geoalt(j)

                endfor

                If ((((chosen_surf - recd.lat)^2)^(.5)) lt range) then
begin

                      ;Finds the size of the altitude array
                      s = size(alt60)
                      zeta =fltarr(s(1))

                      for j = 0, (s(1)-1) do begin

                            ;Calculates the Potential Temperatures
                            zeta(j) = recd.temp(j)*((recd.press(j)/1e3)^(-.
286))

                            ;Writes result to file.
                            if recd.ext385(j) lt 1.0 and recd.ext385(j) gt
0.0 then $
                            printf,outlun, recd.date, recd.lon,zeta(j),recd.
ext385(j), recd.exte385(j), $
                            format='(I6, 2f7.1, 2e14.5)'

                      endfor

                      ;Saves tropopause Location
                      ;Interpolation of the tropopause in terms of
Potential Temperature
                      trop_temp = recd.nmcalt(24)/1000.0
                      tropopause = interpol(zeta,alt60,trop_temp)

                      printf,outlun, 000000, recd.lon,tropopause,large_val,
large_val, $
                      format='(I6, 2f7.1, 2e14.5)'

                endif

                i = i + 1

          endwhile

       end

    37:begin; NO2 Mixing Ratio vs. lat & lon @ Altitude in Km

          ; Read the g2p data record

          data = assoc(inlun,g2p,59392)

          i = 0
          while(not EOF(inlun)) do begin

                ;Sets recd to the association to the file via data
```

```
                recd = data(i)

                ;Checks for something
                if(!version.os eq 'OSF') then begin

                      byteorder,recd,/LSWAP

                endif

                ;Finds the size of the altitude array
                s = size(recd.no2mr)
                alt60 =fltarr(s(1))

                ;Creates altitude array of appropiate size.
                for j = 0, (s(1)-1) do begin

                      alt60(j) = recd.geoalt(j)

                endfor


                ;Interpolation of the specie to the desired altitude
                NO2MR_inter = interpol(recd.no2mr,alt60, altitude)

                ;Writes result to file.
                ;if((recd.date ge dateini) and (recd.date le datefnl)) then
begin

                      printf,outlun, recd.date, recd.lat,recd.lon,
NO2MR_inter, $
                      format='(I6, 2f7.1, e14.5)'

                ;endif

                i = i + 1

           endwhile

         end

      38:begin; NO2 Mixing Ratio vs. lat & lon @ pressure

           ; Read the g2p data record

           data = assoc(inlun,g2p,59392)

           i = 0
           while(not EOF(inlun)) do begin

                ;Sets recd to the association to the file via data
                recd = data(i)

                ;Checks for something
                if(!version.os eq 'OSF') then begin

                      byteorder,recd,/LSWAP

                endif

                ;Finds the size of the altitude array
                s = size(recd.no2mr)
```

```
                alt60 =fltarr(s(1))

                ;Creates altitude array of appropiate size.
                for j = 0, (s(1)-1) do begin

                      alt60(j) = recd.geoalt(j)

                endfor


                ;Interpolation of the specie to the desired pressure
                NO2MR_inter = interpol(recd.no2mr, alt60, altitude)

                ;Writes result to file.

                ;if((recd.date ge dateini) and (recd.date le datefnl)) then
begin

                      printf,outlun, recd.date, recd.lat,recd.lon,
NO2MR_inter, $
                      format='(I6, 2f7.1, e14.5)'

                ;endif

                i = i + 1

          endwhile

        end

     39:begin; NO2 Mixing Ratio vs. lat & lon @ zeta

          ; Read the g2p data record
          ;
          i = 0
          data = assoc(inlun,g2p,59392)

          while(not EOF(inlun)) do begin

                ;Sets recd to the association to the file via data
                recd = data(i)

                ;Checks for something
                if(!version.os eq 'OSF') then begin

                      byteorder,recd,/LSWAP

                endif

                ;Finds the size of the altitude array
                s = size(recd.no2mr)
                alt60 =fltarr(s(1))

                ;Creates altitude array of appropiate size.
                for j = 0, (s(1)-1) do begin

                      alt60(j) = recd.geoalt(j)

                endfor
```

```
                ;Finds the size of the altitude array
                s = size(alt60)
                zeta =fltarr(s(1))

                ;Calculates the Potential Temperatures
                for j = 0, (s(1)-1) do begin

                      zeta(j) = recd.temp(j)*((recd.press(j)/1e3)^(-.286))

                endfor

                ;Interpolation of the specie to the desired Potential
Temperature
                NO2MR_inter = interpol(recd.no2mr,zeta, altitude)

                ;Writes result to file.
                ;if((recd.date ge dateini) and (recd.date le datefnl)) then
begin

                      printf,outlun, recd.date, recd.lat,recd.lon,
NO2MR_inter, $
                      format='(I6, 2f7.1, e14.5)'

                ;endif

                i = i + 1

          endwhile

        end

    40:begin; NO2 Mixing Ratio Altitude in Km & lat @ lon

          ; Read the g2p data record
          ;
          i = 0
          data = assoc(inlun,g2p,59392)

          while(not EOF(inlun)) do begin

                ;Sets recd to the association to the file via data
                recd = data(i)

                ;Checks for something
                if(!version.os eq 'OSF') then begin

                      byteorder,recd,/LSWAP

                endif

                ;Finds the size of the altitude array
                s = size(recd.no2mr)
                alt60 =fltarr(s(1))

                ;Creates altitude array of appropiate size.
                for j = 0, (s(1)-1) do begin

                      alt60(j) = recd.geoalt(j)

                endfor
```

```
                If ((((chosen_surf - recd.lon)^2)^(.5)) le range) then
begin

                        ;Finds the size of the altitude array
                        s = size(alt60)

                        for j = 0, (s(1)-1) do begin

                                ;Writes result to file.
                                if recd.no2mr(j) lt 1.0 and recd.no2mr(j) gt 0.
0 then $
                                printf,outlun, recd.date, recd.lat,alt60(j),
recd.no2mr(j), recd.no2mre(j), $
                                format='(I6, 2f7.1, 2e14.5)'

                        endfor

                        ;Saves tropopause Location
                        tropopause = recd.nmcalt(24)/1000
                        printf,outlun, 000000, recd.lat,tropopause,large_val,
large_val, $
                        format='(I6, 2f7.1, 2e14.5)'

                endif

                i = i + 1

            endwhile

         end

      41:begin; NO2 Mixing Ratio Altitude in mb & lat @ lon

            ; Read the g2p data record
            ;
            i = 0
            data = assoc(inlun,g2p,59392)

            while(not EOF(inlun)) do begin

                ;Sets recd to the association to the file via data
                recd = data(i)

                ;Checks for something
                if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP

                endif

                ;Finds the size of the altitude array
                s = size(recd.no2mr)
                alt60 =fltarr(s(1))

                ;Creates altitude array of appropiate size.
                for j = 0, (s(1)-1) do begin

                        alt60(j) = recd.geoalt(j)

                endfor
```

```
                If ((((chosen_surf - recd.lon)^2)^(.5)) le range) then
begin

                        ;Finds the size of the altitude array
                        s = size(alt60)

                        for j = 0, (s(1)-1) do begin

                                ;Writes result to file.
                                if recd.no2mr(j) lt 1.0 and recd.no2mr(j) gt 0.
0 then $
                                printf,outlun, recd.date, recd.lat,recd.press(j)
,recd.no2mr(j), recd.no2mre(j), $
                                format='(I6, 2e14.5, 2e14.5)'

                        endfor

                        ;Saves tropopause Location
                        ;Interpolation of the tropopause in terms of mb
                        trop_temp = recd.nmcalt(24)/1000.0
                        tropopause = interpol(recd.press,recd.geoalt,
trop_temp)

                        printf,outlun, 000000, recd.lat,tropopause,large_val,
large_val, $
                        format='(I6, 2e14.5, 2e14.5)'


                endif

                i = i + 1

            endwhile

         end

      42:begin; NO2 Mixing Ratio Potential Temperature & lat @ lon

            ; Read the g2p data record
            ;
            i = 0
            data = assoc(inlun,g2p,59392)

            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data
                  recd = data(i)

                  ;Checks for something
                  if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP

                  endif

                  ;Finds the size of the altitude array
                  g = size(recd.no2mr)
                  alt60 =fltarr(g(1))
```

```
                ;Creates altitude array of appropiate size.
                for j = 0, (g(1)-1) do begin

                      alt60(j) = recd.geoalt(j)

                endfor

                If ((((chosen_surf - recd.lon)^2)^(.5)) lt range) then
begin


                      ;Finds the size of the altitude array

                      s = size(alt60)
                      zeta =fltarr(s(1))

                      for j = 0, (s(1)-1) do begin

                            ;Calculates the Potential Temperatures
                            zeta(j) = recd.temp(j)*((recd.press(j)/1e3)^(-.
286))

                            ;Writes result to file.
                            if recd.no2mr(j) lt 1.0 and recd.no2mr(j) gt 0.
0 then $
                            printf,outlun, recd.date, recd.lat,zeta(j),
recd.no2mr(j), recd.no2mre(j), $
                            format='(I6, 2f7.1, 2e14.5)'

                      endfor

                      ;Saves tropopause Location
                      ;Interpolation of the tropopause in terms of
Potential Temperature
                      trop_temp = recd.nmcalt(24)/1000.0
                      tropopause = interpol(zeta,alt60,trop_temp)

                      printf,outlun, 000000, recd.lat,tropopause,large_val,
large_val, $
                      format='(I6, 2f7.1, 2e14.5)'

                endif

                i = i + 1

          endwhile

       end


     43:begin; NO2 Mixing Ratio vs. Altitude in Km & longitude @ latitude

          ; Read the g2p data record
          ;
          i = 0
          data = assoc(inlun,g2p,59392)

          while(not EOF(inlun)) do begin

                ;Sets recd to the association to the file via data
                recd = data(i)
```

```
                ;Checks for something
                if(!version.os eq 'OSF') then begin

                      byteorder,recd,/LSWAP

                endif

                ;Finds the size of the altitude array
                g = size(recd.no2mr)
                alt60 =fltarr(g(1))

                ;Creates altitude array of appropiate size.
                for j = 0, (g(1)-1) do begin

                      alt60(j) = recd.geoalt(j)

                endfor

                If ((((chosen_surf - recd.lat)^2)^(.5)) lt range) then
begin

                      ;Finds the size of the altitude array
                      s = size(alt60)

                      for j = 0, (s(1)-1) do begin

                            ;Writes result to file.
                            if recd.no2mr(j) lt 1.0 and recd.no2mr(j) gt 0.
0 then $
                            printf,outlun, recd.date, recd.lon,recd.
geoalt(j),recd.no2mr(j), recd.no2mre(j), $
                            format='(I6, 2f7.1, 2e14.5)'

                      endfor

                      ;Saves tropopause Location
                      tropopause = recd.nmcalt(24)/1000
                      printf,outlun, 000000, recd.lon,tropopause,large_val,
large_val, $
                      format='(I6, 2f7.1, 2e14.5)'

                endif

                i = i + 1

          endwhile

       end

    44:begin; NO2 Mixing Ratio vs. Altitude in mb & longitude @ latitude

          ; Read the g2p data record
          ;
          i = 0
          data = assoc(inlun,g2p,59392)

          while(not EOF(inlun)) do begin

                ;Sets recd to the association to the file via data
                recd = data(i)
```

```
                ;Checks for something
                if(!version.os eq 'OSF') then begin

                      byteorder,recd,/LSWAP

                endif

                ;Finds the size of the altitude array
                g = size(recd.no2mr)
                alt60 =fltarr(g(1))

                ;Creates altitude array of appropiate size.
                for j = 0, (g(1)-1) do begin

                      alt60(j) = recd.geoalt(j)

                endfor

                If ((((chosen_surf - recd.lat)^2)^(.5)) lt range) then
begin

                      ;Finds the size of the altitude array
                      s = size(alt60)

                      for j = 0, (s(1)-1) do begin

                            ;Writes result to file.
                            if recd.no2mr(j) lt 1.0 and recd.no2mr(j) gt 0.
0 then $
                            printf,outlun, recd.date, recd.lon,recd.press(j)
,recd.no2mr(j), recd.no2mre(j), $
                            format='(I6, 2e14.5, 2e14.5)'

                      endfor

                      ;Saves tropopause Location
                      ;Interpolation of the tropopause in terms of mb
                      trop_temp = recd.nmcalt(24)/1000.0
                      tropopause = interpol(recd.press,recd.geoalt,
trop_temp)

                      printf,outlun, 000000, recd.lon,tropopause,large_val,
large_val, $
                      format='(I6, 2e14.5, 2e14.5)'

                endif

                i = i + 1

          endwhile

       end

     45:begin; NO2 Mixing Ratio vs. Potential Temperature & longitude @ lat

          ; Read the g2p data record
          ;
          i = 0
          data = assoc(inlun,g2p,59392)
```

```
            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data
                  recd = data(i)

                  ;Checks for something
                  if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP

                  endif

                  ;Finds the size of the altitude array
                  g = size(recd.no2mr)
                  alt60 =fltarr(g(1))

                  ;Creates altitude array of appropiate size.
                  for j = 0, (g(1)-1) do begin

                        alt60(j) = recd.geoalt(j)

                  endfor

                  If ((((chosen_surf - recd.lat)^2)^(.5)) lt range) then
begin

                        ;Finds the size of the altitude array
                        s = size(alt60)
                        zeta =fltarr(s(1))

                        for j = 0, (s(1)-1) do begin

                              ;Calculates the Potential Temperatures
                              zeta(j) = recd.temp(j)*((recd.press(j)/1e3)^(-.
286))

                              ;Writes result to file.
                              if recd.no2mr(j) lt 1.0 and recd.no2mr(j) gt 0.
0 then $
                              printf,outlun, recd.date, recd.lon,zeta(j),recd.
no2mr(j), recd.no2mre(j), $
                              format='(I6, 2f7.1, 2e14.5)'

                        endfor

                        ;Saves tropopause Location
                        ;Interpolation of the tropopause in terms of
Potential Temperature
                        trop_temp = recd.nmcalt(24)/1000.0
                        tropopause = interpol(zeta,alt60,trop_temp)

                        printf,outlun, 000000, recd.lon,tropopause,large_val,
large_val, $
                        format='(I6, 2f7.1, 2e14.5)'

                  endif

                  i = i + 1

            endwhile
```

```
        end

      64:begin; Ozone Mixing Ratio vs. lat & lon @ Altitude in Km

            ; Read the g2p data record

            data = assoc(inlun,g2p,59392)

            i = 0
            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data
                  recd = data(i)

                  ;Checks for something
                  if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP

                  endif

                  ;Interpolation of the specie to the desired altitude
                  o3mr_inter = interpol(recd.o3mr,recd.geoalt, altitude)

                  ;Writes result to file.
                  ;if((recd.date ge dateini) and (recd.date le datefnl)) then
begin

                        printf,outlun, recd.date, recd.lat,recd.lon,
o3mr_inter, $
                        format='(I6, 2f7.1, e14.5)'

                  ;endif

                  i = i + 1

            endwhile

          end

      65:begin; Ozone Mixing Ratio vs. lat & lon @ pressure

            ; Read the g2p data record

            data = assoc(inlun,g2p,59392)

            i = 0
            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data
                  recd = data(i)

                  ;Checks for something
                  if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP

                  endif

                  ;Interpolation of the specie to the desired pressure
                  o3mr_inter = interpol(recd.o3mr,recd.press, altitude)
```

```
                ;Writes result to file.

                ;if((recd.date ge dateini) and (recd.date le datefnl)) then
begin

                      printf,outlun, recd.date, recd.lat,recd.lon,
o3mr_inter, $
                      format='(I6, 2f7.1, e14.5)'

                ;endif

                i = i + 1

          endwhile

        end

    66:begin; Ozone Mixing Ratio vs. lat & lon @ zeta

          ; Read the g2p data record
          ;
          i = 0
          data = assoc(inlun,g2p,59392)

          while(not EOF(inlun)) do begin

                ;Sets recd to the association to the file via data
                recd = data(i)

                ;Checks for something
                if(!version.os eq 'OSF') then begin

                      byteorder,recd,/LSWAP

                endif

                ;Finds the size of the altitude array
                s = size(recd.geoalt)
                zeta =fltarr(s(1))

                ;Calculates the Potential Temperatures
                for j = 0, (s(1)-1) do begin

                      zeta(j) = recd.temp(j)*((recd.press(j)/1e3)^(-.286))

                endfor

                ;Interpolation of the specie to the desired Potential
Temperature
                o3mr_inter = interpol(recd.o3mr,zeta, altitude)

                ;Writes result to file.
                ;if((recd.date ge dateini) and (recd.date le datefnl)) then
begin

                      printf,outlun, recd.date, recd.lat,recd.lon,
o3mr_inter, $
                      format='(I6, 2f7.1, e14.5)'

                ;endif
```

```
                i = i + 1

            endwhile

        end

    67:begin; Ozone Mixing Ratio vs. Altitude in Km & lat @ lon

            ; Read the g2p data record
            ;
            i = 0
            data = assoc(inlun,g2p,59392)

            while(not EOF(inlun)) do begin

                ;Sets recd to the association to the file via data
                recd = data(i)

                ;Checks for something
                if(!version.os eq 'OSF') then begin

                    byteorder,recd,/LSWAP

                endif

                If ((((chosen_surf - recd.lon)^2)^(.5)) le range) then
begin


                    ;Finds the size of the altitude array
                    s = size(recd.geoalt)

                    for j = 0, (s(1)-1) do begin

                        ;Writes result to file.
                        printf,outlun, recd.date, recd.lat,recd.
geoalt(j),recd.o3mr(j), recd.o3mre(j), $
                        format='(I6, 2f7.1, 2e14.5)'

                    endfor

                    ;Saves tropopause Location
                    tropopause = recd.nmcalt(24)/1000
                    printf,outlun, 000000, recd.lat,tropopause,large_val,
large_val, $
                    format='(I6, 2f7.1, 2e14.5)'

                endif

                i = i + 1

            endwhile

        end

    68:begin; Ozone Mixing Ratio vs. Altitude in mb & lat @ lon

            ; Read the g2p data record
            ;
            i = 0
```

```
        data = assoc(inlun,g2p,59392)

        while(not EOF(inlun)) do begin

              ;Sets recd to the association to the file via data
              recd = data(i)

              ;Checks for something
              if(!version.os eq 'OSF') then begin

                    byteorder,recd,/LSWAP

              endif

              ;Finds the size of the altitude array
              s = size(recd.o3mr)
              alt70 =fltarr(s(1))

              ;Creates altitude array of appropiate size.
              for j = 0, (s(1)-1) do begin

                    alt70(j) = recd.geoalt(j)

              endfor

              If ((((chosen_surf - recd.lon)^2)^(.5)) le range) then
begin


                    ;Finds the size of the altitude array
                    s = size(alt70)

                    for j = 0, (s(1)-1) do begin

                          ;Writes result to file.
                          if recd.o3mr(j) lt 1.0 and recd.o3mr(j) gt 0.0
then $
                          printf,outlun, recd.date, recd.lat,recd.press(j)
,recd.o3mr(j), recd.o3mre(j), $
                          format='(I6, 2e14.5, 2e14.5)'

                    endfor

                    ;Saves tropopause Location
                    ;Interpolation of the tropopause in terms of mb
                    trop_temp = recd.nmcalt(24)/1000.0
                    tropopause = interpol(recd.press,recd.geoalt,
trop_temp)

                    printf,outlun, 000000, recd.lat,tropopause,large_val,
large_val, $
                    format='(I6, 2e14.5, 2e14.5)'


              endif

              i = i + 1

        endwhile

      end
```

```
69:begin; Ozone Mixing Ratio vs. Potential Temperature & lat @ lon

      ; Read the g2p data record
      ;
      i = 0
      data = assoc(inlun,g2p,59392)

      while(not EOF(inlun)) do begin

            ;Sets recd to the association to the file via data
            recd = data(i)

            ;Checks for something
            if(!version.os eq 'OSF') then begin

                  byteorder,recd,/LSWAP

            endif

            If ((((chosen_surf - recd.lon)^2)^(.5)) lt range) then
begin

                  ;Finds the size of the altitude array
                  s = size(recd.geoalt)
                  zeta =fltarr(s(1))

                  for j = 0, (s(1)-1) do begin

                        ;Calculates the Potential Temperatures
                        zeta(j) = recd.temp(j)*((recd.press(j)/1e3)^(-.
286))

                        ;Writes result to file.
                        printf,outlun, recd.date, recd.lat,zeta(j),recd.
o3mr(j), recd.o3mre(j), $
                        format='(I6, 2f7.1, 2e14.5)'

                  endfor

                  ;Saves tropopause Location
                  ;Interpolation of the tropopause in terms of
Potential Temperature
                  trop_temp = recd.nmcalt(24)/1000.0
                  tropopause = interpol(zeta,recd.geoalt,trop_temp)

                  printf,outlun, 000000, recd.lat,tropopause,large_val,
large_val, $
                  format='(I6, 2f7.1, 2e14.5)'

            endif

            i = i + 1

      endwhile

   end

70:begin; Ozone Mixing Ratio vs. Altitude in Km & longitude @ latitude

      ; Read the g2p data record
```

```
            ;
            i = 0
            data = assoc(inlun,g2p,59392)

            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data
                  recd = data(i)

                  ;Checks for something
                  if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP

                  endif

                  If ((((chosen_surf - recd.lat)^2)^(.5)) lt range) then
begin

                        ;Finds the size of the altitude array
                        s = size(recd.geoalt)

                        for j = 0, (s(1)-1) do begin

                              ;Writes result to file.
                              printf,outlun, recd.date, recd.lon,recd.
geoalt(j),recd.o3mr(j), recd.o3mre(j), $
                              format='(I6, 2f7.1, 2e14.5)'

                        endfor

                        ;Saves tropopause Location
                        tropopause = recd.nmcalt(24)/1000
                        printf,outlun, 000000, recd.lon,tropopause,large_val,
large_val, $
                        format='(I6, 2f7.1, 2e14.5)'

                  endif

                  i = i + 1

            endwhile

         end

      71:begin; Ozone Mixing Ratio vs. Altitude in mb & longitude @ latitude

            ; Read the g2p data record
            ;
            i = 0
            data = assoc(inlun,g2p,59392)

            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data
                  recd = data(i)

                  ;Checks for something
                  if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP
```

```
                endif

                If ((((chosen_surf - recd.lat)^2)^(.5)) lt range) then
begin

                      ;Finds the size of the altitude array
                      s = size(recd.geoalt)

                      for j = 0, (s(1)-1) do begin

                            ;Writes result to file.
                            if recd.o3mr(j) lt 1.0 and recd.o3mr(j) gt 0.0
then $
                            printf,outlun, recd.date, recd.lon,recd.press(j)
,recd.o3mr(j), recd.o3mre(j), $
                            format='(I6, 2e14.5, 2e14.5)'

                      endfor

                      ;Saves tropopause Location
                      ;Interpolation of the tropopause in terms of mb
                      trop_temp = recd.nmcalt(24)/1000.0
                      tropopause = interpol(recd.press,recd.geoalt,
trop_temp)

                      printf,outlun, 000000, recd.lon,tropopause,large_val,
large_val, $
                      format='(I6, 2e14.5, 2e14.5)'

                endif

                i = i + 1

          endwhile

       end

    72:begin; Ozone Mixing Ratio vs. Potential Temperature & longitude @ lat

          ; Read the g2p data record
          ;
          i = 0
          data = assoc(inlun,g2p,59392)

          while(not EOF(inlun)) do begin

                ;Sets recd to the association to the file via data
                recd = data(i)

                ;Checks for something
                if(!version.os eq 'OSF') then begin

                      byteorder,recd,/LSWAP

                endif

                If ((((chosen_surf - recd.lat)^2)^(.5)) lt range) then
begin

                      ;Finds the size of the altitude array
```

```
                        s = size(recd.geoalt)
                        zeta =fltarr(s(1))

                        for j = 0, (s(1)-1) do begin

                              ;Calculates the Potential Temperatures
                              zeta(j) = recd.temp(j)*((recd.press(j)/1e3)^(-.
286))

                              ;Writes result to file.
                              printf,outlun, recd.date, recd.lon,zeta(j),recd.
o3mr(j), recd.o3mre(j), $
                              format='(I6, 2f7.1, 2e14.5)'

                        endfor

                        ;Saves tropopause Location
                        ;Interpolation of the tropopause in terms of
Potential Temperature
                        trop_temp = recd.nmcalt(24)/1000.0
                        tropopause = interpol(zeta,recd.geoalt,trop_temp)

                        printf,outlun, 000000, recd.lon,tropopause,large_val,
large_val, $
                        format='(I6, 2f7.1, 2e14.5)'

                  endif

                  i = i + 1

            endwhile

         end

;---------------------------------------------------------------------------
--
;---------------------------------------------------------------------------
--
;---------------------------------------------------------------------------
--

      82:begin; Water Vapor Mixing Ratio vs. lat & lon @ Altitude in Km

            ; Read the g2p data record

            data = assoc(inlun,g2p,59392)

            i = 0
            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data
                  recd = data(i)

                  ;Checks for something
                  if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP

                  endif

                  ;Interpolation of the specie to the desired altitude
```

```
                H2Omr_inter = interpol(recd.H2Omr,recd.geoalt, altitude)

                ;Writes result to file.
                ;if((recd.date ge dateini) and (recd.date le datefnl)) then
begin

                      printf,outlun, recd.date, recd.lat,recd.lon,
H2Omr_inter, $
                      format='(I6, 2f7.1, e14.5)'

                ;endif

                i = i + 1

          endwhile

       end
     83:begin; Water Vapor Ratio vs. lat & lon @ pressure

          ; Read the g2p data record

          data = assoc(inlun,g2p,59392)

          i = 0
          while(not EOF(inlun)) do begin

                ;Sets recd to the association to the file via data
                recd = data(i)

                ;Checks for something
                if(!version.os eq 'OSF') then begin

                      byteorder,recd,/LSWAP

                endif

                ;Interpolation of the specie to the desired pressure
                H2Omr_inter = interpol(recd.H2Omr,recd.press, altitude)

                ;Writes result to file.

                ;if((recd.date ge dateini) and (recd.date le datefnl)) then
begin

                      printf,outlun, recd.date, recd.lat,recd.lon,
H2Omr_inter, $
                      format='(I6, 2f7.1, e14.5)'

                ;endif

                i = i + 1

          endwhile

       end

     84:begin; Water Vapor Mixing Ratio vs. lat & lon @ zeta

          ; Read the g2p data record
          ;
          i = 0
```

```
            data = assoc(inlun,g2p,59392)

            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data
                  recd = data(i)

                  ;Checks for something
                  if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP

                  endif

                  ;Finds the size of the altitude array
                  s = size(recd.H2Omr)
                  zeta =fltarr(s(1))

                  ;Calculates the Potential Temperatures
                  for j = 0, (s(1)-1) do begin

                        zeta(j) = recd.temp(j)*((recd.press(j)/1e3)^(-.286))

                  endfor

                  ;Interpolation of the specie to the desired Potential
Temperature
                  H2Omr_inter = interpol(recd.H2Omr,zeta, altitude)

                  ;Writes result to file.
                  ;if((recd.date ge dateini) and (recd.date le datefnl)) then
begin

                        printf,outlun, recd.date, recd.lat,recd.lon,
H2Omr_inter, $
                        format='(I6, 2f7.1, e14.5)'

                  ;endif

                  i = i + 1

            endwhile

          end

      85:begin; Water Vapor Ratio vs. Altitude in Km & lat @ lon

            ; Read the g2p data record
            ;
            i = 0
            data = assoc(inlun,g2p,59392)

            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data
                  recd = data(i)

                  ;Checks for something
                  if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP
```

```
                endif

                If ((((chosen_surf - recd.lon)^2)^(.5)) le range) then
begin


                      ;Finds the size of the altitude array
                      s = size(recd.H2Omr)

                      for j = 0, (s(1)-1) do begin

                            ;Writes result to file.
                            printf,outlun, recd.date, recd.lat,recd.
geoalt(j),recd.H2Omr(j), recd.H2Omre(j), $
                            format='(I6, 2f7.1, 2e14.5)'

                      endfor

                      ;Saves tropopause Location
                      tropopause = recd.nmcalt(24)/1000
                      printf,outlun, recd.date, recd.lat,tropopause,
large_val,large_val, $
                      format='(I6, 2f7.1, 2e14.5)'

                endif

                i = i + 1

          endwhile

       end

    86:begin; Water Vapor Ratio vs. Altitude in mb & lat @ lon

          ; Read the g2p data record
          ;
          i = 0
          data = assoc(inlun,g2p,59392)

          while(not EOF(inlun)) do begin

                ;Sets recd to the association to the file via data
                recd = data(i)

                ;Checks for something
                if(!version.os eq 'OSF') then begin

                      byteorder,recd,/LSWAP

                endif

                If ((((chosen_surf - recd.lon)^2)^(.5)) le range) then
begin


                      ;Finds the size of the altitude array
                      s = size(recd.H2Omr)

                      for j = 0, (s(1)-1) do begin
```

```
                              ;Writes result to file.
                              printf,outlun, recd.date, recd.lat,recd.press(j)
,recd.H2Omr(j), recd.H2Omre(j), $
                              format='(I6, f7.1, f7.3, 2e14.5)'

                        endfor

                        ;Saves tropopause Location
                        ;Interpolation of the tropopause in terms of mb
                        trop_temp = recd.nmcalt(24)/1000.0
                        tropopause = interpol(recd.press,recd.geoalt,
trop_temp)

                        printf,outlun, recd.date, recd.lat,tropopause,
large_val,large_val, $
                        format='(I6, 2f7.1, 2e14.5)'



                  endif

                  i = i + 1

            endwhile

         end

      87:begin; Water Vapor Ratio vs. Potential Temperature & lat @ lon
      Print, 'OK'
            ; Read the g2p data record
            ;
            i = 0
            data = assoc(inlun,g2p,59392)

            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data.  Is
going to read one record.
                  recd = data(i)

                  ;Checks for something
                  if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP

                  endif

                  If ((((chosen_surf - recd.lon)^2)^(.5)) lt range) then
begin

                        ;Finds the size of the altitude array
                        s = size(recd.H2Omr)
                        zeta =fltarr(s(1))
                        alt_fixed = fltarr(s(1))

                        for j = 0, (s(1)-1) do begin

                              ;Calculates the Potential Temperatures
                              zeta(j) = recd.temp(j)*((recd.press(j)/1e3)^(-.
286))
                              alt_fixed(j) = recd.geoalt(j)
```

```
                              ;Writes result to file.
                              printf,outlun, recd.date, recd.lat,zeta(j),recd.
H2Omr(j), recd.H2Omre(j), $
                              format='(I6, 2f7.1, 2e14.5)'

                        endfor

                        ;Saves tropopause Location
                        ;Interpolation of the tropopause in terms of
Potential Temperature
                        trop_temp = recd.nmcalt(24)/1000.0
                        tropopause = interpol(zeta,alt_fixed,trop_temp)

                        printf,outlun, recd.date, recd.lat,tropopause,
large_val,large_val, $
                        format='(I6, 2f7.1, 2e14.5)'

                  endif

                  i = i + 1 ;RECORD (profile) counter

            endwhile

         end

      88:begin; Water Vapor Mixing Ratio vs. Altitude in Km & longitude @
latitude

            ; Read the g2p data record
            ;
            i = 0
            data = assoc(inlun,g2p,59392)

            while(not EOF(inlun)) do begin

                  ;Sets recd to the association to the file via data
                  recd = data(i)

                  ;Checks for something
                  if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP

                  endif

                  If ((((chosen_surf - recd.lat)^2)^(.5)) lt range) then
begin

                        ;Finds the size of the altitude array
                        s = size(recd.H2Omr)

                        for j = 0, (s(1)-1) do begin

                              ;Writes result to file.
                              printf,outlun, recd.date, recd.lon,recd.
geoalt(j),recd.H2Omr(j), recd.H2Omre(j), $
                              format='(I6, 2f7.1, 2e14.5)'

                        endfor
```

```
                        ;Saves tropopause Location
                        tropopause = recd.nmcalt(24)/1000
                        printf,outlun, recd.date, recd.lon,tropopause,
large_val,large_val, $
                        format='(I6, 2f7.1, 2e14.5)'

                endif

                i = i + 1

        endwhile

    end

  89:begin; Water Vapor Mixing Ratio vs. Altitude in mb & longitude @
latitude

        ; Read the g2p data record
        ;
        i = 0
        data = assoc(inlun,g2p,59392)

        while(not EOF(inlun)) do begin

                ;Sets recd to the association to the file via data
                recd = data(i)

                ;Checks for something
                if(!version.os eq 'OSF') then begin

                        byteorder,recd,/LSWAP

                endif

                If ((((chosen_surf - recd.lat)^2)^(.5)) lt range) then
begin

                        ;Finds the size of the altitude array
                        s = size(recd.H2Omr)

                        for j = 0, (s(1)-1) do begin

                                ;Writes result to file.
                                printf,outlun, recd.date, recd.lon,recd.press(j)
,recd.H2Omr(j), recd.H2Omre(j), $
                                format='(I6, f7.1, f7.3, 2e14.5)'

                        endfor

                        ;Saves tropopause Location
                        ;Interpolation of the tropopause in terms of mb
                        trop_temp = recd.nmcalt(24)/1000.0
                        tropopause = interpol(recd.press,recd.geoalt,
trop_temp)

                        printf,outlun, recd.date, recd.lon,tropopause,
large_val,large_val, $
                        format='(I6, 2f7.1, 2e14.5)'

                endif
```

```
                i = i + 1

            endwhile

        end

    90:begin; Water Vapor Mixing Ratio vs. Potential Temperature &
longitude @ lat

            ; Read the g2p data record
            ;
            i = 0
            data = assoc(inlun,g2p,59392)

            while(not EOF(inlun)) do begin

                ;Sets recd to the association to the file via data
                recd = data(i)

                ;Checks for something
                if(!version.os eq 'OSF') then begin

                    byteorder,recd,/LSWAP

                endif

                If ((((chosen_surf - recd.lat)^2)^(.5)) lt range) then
begin

                    ;Finds the size of the altitude array
                    s = size(recd.H2Omr)
                    zeta =fltarr(s(1))
                    alt_fixed = fltarr(s(1))

                    for j = 0, (s(1)-1) do begin

                        ;Calculates the Potential Temperatures
                        zeta(j) = recd.temp(j)*((recd.press(j)/1e3)^(-.
286))
                        alt_fixed(j) = recd.geoalt(j)

                        ;Writes result to file.
                        printf,outlun, recd.date, recd.lon,zeta(j),recd.
H2Omr(j), recd.H2Omre(j), $
                        format='(I6, 2f7.1, 2e14.5)'

                    endfor

                    ;Saves tropopause Location
                    ;Interpolation of the tropopause in terms of
Potential Temperature
                    trop_temp = recd.nmcalt(24)/1000.0
                    tropopause = interpol(zeta,alt_fixed,trop_temp)

                    printf,outlun, recd.date, recd.lon,tropopause,
large_val,large_val, $
                    format='(I6, 2f7.1, 2e14.5)'

                endif

                i = i + 1
```

```
            endwhile

         end

      endcase

      free_lun,inlun

endfor

free_lun,outlun

return
end
```

```
;Copyright (c) 1999, NSU SciViz Lab.  All rights reserved.
;     Unauthorized reproduction prohibited.
;+
; NAME:
;     plot_points.pro
;
; RESTRICTIONS:
;     Works with SAGE II data, g2p v5.931
;
;-
;*****************************************************************************
**


pro plot_points, outfile, surf_type, levels, x_axis_max, x_axis_min,
y_axis_max, y_axis_min

;Setting graphics
device, pseudo=8, retain=2

;Setting Variables

If n_elements(levels) eq  0 then levels= findgen(11)
If n_elements(x_axis_max) eq  0 then x_axis_max = 0
If n_elements(x_axis_min) eq  0 then x_axis_min = 0
If n_elements(y_axis_min) eq  0 then y_axis_min = 0
If n_elements(y_axis_max) eq  0 then y_axis_max = 0

norm_factor = 1e1
unit = ' '
y_unit = 1
surf_type = 1

;Background color for plot.
!p.background = 0

;Setting Symbol
!psym=0

;!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!
;If you want to run the program alone uncomment the following commands,!!!!!!!
!!!!!!!!!!!!!!!!!!
;;outfile = dialog_pickfile()
;;window, /free
;!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!

;Open file
openr, lun, outfile, /get_lun

;Reading header
header = strarr(21)
readf, lun, header, format='(a36)'

;Reading Normalization Factor, Specie Unit, and Surface Type
reads, header(7), unit
reads, header(10), y_unit
reads, header(13), norm_factor
reads, header(16), surf_type
```

```
;Reading data

If (surf_type eq 1) then begin

      datacolumn = fltarr(4, 100000)
      temp_peak = fltarr(4)

      i = 0
      while (not eof(lun))  do begin
            readf, lun, temp_peak
            datacolumn[*,i] = temp_peak
            i = i + 1
      endwhile

      datacolumn = datacolumn[*,0:i-1]

      free_lun, lun

      ;Setting arrays

      date = datacolumn[0,*]
      lat = datacolumn[1,*]
      lon = datacolumn[2,*]
      data = datacolumn[3,*]*norm_factor

endif else begin

      datacolumn = fltarr(5, 80000)
      temp_peak = fltarr(5)

      i = 0
      while (not eof(lun))  do begin
            readf, lun, temp_peak
            datacolumn[*,i] = temp_peak
            i = i + 1
      endwhile

      datacolumn = datacolumn[*,0:i-1]

      free_lun, lun

      ;Setting arrays

      date = datacolumn[0,*]
      lat = datacolumn[1,*]
      lon = datacolumn[2,*]
      data = datacolumn[3,*]*norm_factor
      dataerr = datacolumn[4,*]*norm_factor

endelse


      LonMax = MAX(lon)
      LonMin = Min(Lon)
      LatMax = MAX(Lat)
      LatMin = Min(Lat)

;Sets the axes and the plot parameters for the appropiate surface chosen
case surf_type of

      1:    begin
```

```
                x = lon
                y = lat
                ;Finds the axes range of the data
                xMax = MAX(x)
                xMin = Min(x)
                yMax = MAX(y)
                yMin = Min(y)
                If x_axis_max eq  0 then x_axis_max = xmax
                If x_axis_min eq  0 then x_axis_min = xmin
                If y_axis_min eq  0 then y_axis_min = ymin
                If y_axis_max eq  0 then y_axis_max = ymax


                MAP_SET, /CONTINENTS, /MERCATOR, limit=[y_axis_min,
x_axis_min,y_axis_max,x_axis_max], position=[0.01, 0.2, 0.99, 0.99], color=255


          end

     2:   begin ;Specie vs. Altitude and Latitude

                x = reform(lat)
                y = reform(lon)
                ;Finds the axes range of the data
                xMax = MAX(x)
                xMin = Min(x)
                yMax = MAX(y)
                yMin = Min(y)
                If x_axis_max eq  0 then x_axis_max = xmax
                If x_axis_min eq  0 then x_axis_min = xmin
                If y_axis_min eq  0 then y_axis_min = ymin
                If y_axis_max eq  0 then y_axis_max = ymax

                case y_unit of

                2: plot, x, y, xrange=[x_axis_min,x_axis_max], yrange=
[y_axis_min,y_axis_max], position=[0.1, 0.3, 0.97, 0.99], xtitle ='Latitude',
ytitle='Altitude', color=255,/nodata

                3: plot, x, y, xrange=[x_axis_min,x_axis_max], yrange=
[y_axis_max,y_axis_min], position=[0.1, 0.3, 0.97, 0.99], xtitle ='Latitude',
ytitle='Altitude', color=255, /ylog, /nodata

                4: plot, x, y, xrange=[x_axis_min,x_axis_max], yrange=
[y_axis_min,y_axis_max], position=[0.1, 0.3, 0.97, 0.99], xtitle ='Latitude',
ytitle='Altitude', color=255,/nodata


                endcase


          end

     3:   begin

                x = lat
                y = lon
                ;Finds the axes range of the data
                xMax = MAX(x)
                xMin = Min(x)
```

```
                yMax = MAX(y)
                yMin = Min(y)
                If x_axis_max eq  0 then x_axis_max = xmax
                If x_axis_min eq  0 then x_axis_min = xmin
                If y_axis_min eq  0 then y_axis_min = ymin
                If y_axis_max eq  0 then y_axis_max = ymax


                case y_unit of


                2: plot, x, y, xrange=[x_axis_min,x_axis_max], yrange=
[y_axis_min,y_axis_max], position=[0.1, 0.3, 0.97, 0.99], xtitle ='Longitude',
ytitle='Altitude',color=255, /nodata

                3: plot, x, y, xrange=[x_axis_min,x_axis_max], yrange=
[y_axis_max,y_axis_min], position=[0.1, 0.3, 0.97, 0.99], xtitle ='Longitude',
ytitle='Altitude', color=255, /ylog, /nodata

                4: plot, x, y, xrange=[x_axis_min,x_axis_max], yrange=
[y_axis_min,y_axis_max], position=[0.1, 0.3, 0.97, 0.99], xtitle ='Longitude',
ytitle='Altitude', color=255,/nodata

                endcase
            end
endcase


      val=levels
      index0=where((data ge levels(0)) and (data lt levels(1)))
      index1=where((data ge levels(1)) and (data lt levels(2)))
      index2=where((data ge levels(2)) and (data lt levels(3)))
      index3=where((data ge levels(3)) and (data lt levels(4)))
      index4=where((data ge levels(4)) and (data lt levels(5)))
      index5=where((data ge levels(5)) and (data lt levels(6)))
      index6=where((data ge levels(6)) and (data lt levels(7)))
      index7=where((data ge levels(7)) and (data lt levels(8)))
      index8=where((data ge levels(8)) and (data lt levels(9)))
      index9=where(data ge levels(9))

;Lets Plot
 !psym=4
 xx = [0.0, 0.5, -0.8, 0.8, -0.5,0.0]
 yy = [1.0, -0.8, 0.3, 0.3, -0.8,1.0]
  usersym, xx, yy, /fill

     !color=210
   if index9(0) ne -1 then oplot,x(index9),y(index9), psym=8, symsize=2.0
    !color=187
   if index8(0) ne -1 then oplot,x(index8),y(index8), psym=8, symsize=2.0
   !color=163
   if index7(0) ne -1 then oplot,x(index7),y(index7), psym=8, symsize=2.0
   !color=140
   if index6(0) ne -1 then oplot,x(index6),y(index6), psym=8, symsize=2.0
   !color=117
   if index5(0) ne -1 then oplot,x(index5),y(index5), psym=8, symsize=2.0
   !color=93
   if index4(0) ne -1 then oplot,x(index4),y(index4), psym=8, symsize=2.0
   !color=70
   if index3(0) ne -1 then oplot,x(index3),y(index3), psym=8, symsize=2.0
   !color=47
```

```
   if index2(0) ne -1 then oplot,x(index2),y(index2), psym=8, symsize=2.0
   !color=23
   if index1(0) ne -1 then oplot,x(index1),y(index1), psym=8, symsize=2.0
   !color=10
   if index0(0) ne -1 then oplot,x(index0),y(index0), psym=8, symsize=2.0
  !psym=0

;Plots the tropopause

      index_trop = where(date eq 000000)
      If (surf_type ne 1) then oplot, x(index_trop), y(index_trop), psym = 6,
symsize=.5, color=255

;Set up the map projection.

;If (surf_type eq 1) then MAP_SET, /CONTINENTS, /MERCATOR, /grid, /label, /
noerase, color=255

colorbar_position = [0.02, 0.06, 0.98, 0.11]
      call_procedure, 'COLORBAR', divisions=9, Position=colorbar_position,
Max=levels[10], Min=levels[0],  $
      Format='(f6.2)',Title=header(1) + unit

return
end
```

```
;*****************************************************************************
**
;Copyright (c) 1999, NSU SciViz Lab.  All rights reserved.
;     Unauthorized reproduction prohibited.
;+
; NAME:
;     ReadSage.pro
;
; RESTRICTIONS:
;     Works with SAGE II data, g2p v5.931
;
;-
;*****************************************************************************
**
;-----------------------------------------------------------------------------
--

pro New, ev
       CALL_Procedure, 'ReadSage'
return
end
;-----------------------------------------------------------------------------
--

pro rescale, ev

widget_control, ev.top, get_uvalue=state
parent = ev.top
x_axis_max = state.x_axis_max
x_axis_min = state.x_axis_min
y_axis_max = state.y_axis_max
y_axis_min = state.y_axis_min

call_procedure, 'get_scale_info', x_axis_max, x_axis_min, y_axis_max,
y_axis_min, cancel=cancel, parent=parent

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid


call_procedure, 'plot_points', state.outfile, state.surf_type, state.levels,
x_axis_max, x_axis_min, y_axis_max, y_axis_min

;Sets window to the appropiate draw widget
widget_control, state.draw, get_value=winid
wset, winid

;Grids the data and draws a cellfilled contour.
call_procedure, 'Grid_Smooth_Map', state.outfile, state.lat_step, state.
lon_step, state.smooth_pass, state.levels, x_axis_max, x_axis_min, y_axis_max,
y_axis_min

state.x_axis_max = x_axis_max
state.x_axis_min = x_axis_min
state.y_axis_max = y_axis_max
state.y_axis_min = y_axis_min

widget_control, ev.top, set_uvalue=state

return
```

```
end

;-------------------------------------------------------------------------
--

pro regrid, ev

;Gets the passing variables throught uvalue structure STATE.
widget_control, ev.top, get_uvalue=state

;Sets the local variables to be passed.
parent = ev.top
;
specie_type = state.specie_type
surf_type = state.surf_type
;
outfile = state.outfile
max_val = state.max_val
norm_factor = state.norm_factor
unit = state.unit
lat_step = state.lat_step
lon_step = state.lon_step
smooth_pass = state.smooth_pass
levels = state.levels

;Gets information from the user about the file to be opened.
call_procedure, 'get_more_info', surf_type, lat_step, lon_step, smooth_pass,
cancel=cancel, parent=parent

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

;Sets window to the appropiate draw widget
widget_control, state.draw, get_value=winid
wset, winid

;Grids the data and draws a cellfilled contour.
call_procedure, 'Grid_Smooth_Map', outfile, lat_step, lon_step, smooth_pass,
levels, state.x_axis_max, state.x_axis_min, state.y_axis_max, state.y_axis_min

;Resets passing variable uvalue
state.lat_step = lat_step
state.lon_step = lon_step
state.smooth_pass = smooth_pass
widget_control, ev.top, set_uvalue=state

return
end
;-------------------------------------------------------------------------
--
pro specie_file, ev

;Gets the passing variables throught uvalue structure STATE.
widget_control, ev.top, get_uvalue=state

;Prompts for file name
outfile = dialog_pickfile()

;Sets the local variables to be passed.
lat_step = 8.0
```

```
lon_step = 10.0
smooth_pass = 9.0

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

;Plots the raw data.
call_procedure,'plot_points', outfile, surf_type

;Sets window to the appropiate draw widget
widget_control, state.draw, get_value=winid
wset, winid

;Grids the data and draws a cellfilled contour.
call_procedure, 'Grid_Smooth_Map', outfile, lat_step, lon_step, smooth_pass

;Resets passing variable uvalue
state.lat_step = lat_step
state.lon_step = lon_step
state.smooth_pass = smooth_pass
state.outfile = outfile
state.surf_type = surf_type
widget_control, ev.top, set_uvalue=state

return
end

;-------------------------------------------------------------------------------
--

pro OpenA1000llkm, ev

;Set default variables and widget values for Aerosol Extintion @ 1000nm.
specie_type = 1
surf_type = 1
max_val = 9.9999999999999e-1
norm_factor=1e3 ;Will give ppt.
unit = '(10^-3)(1/km)'
surf = 14.5
surf_unit = 'km'
y_unit = 1
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
```

```
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end

;-------------------------------------------------------------------------
--

pro OpenA100011mb, ev

;Set default variables and widget values for Aerosol extintion @ 1000nm at
Pressure (mb).
specie_type = 2
surf_type = 1
max_val = 9.9999999999999e-1
norm_factor=1e5 ;Will give ppm.
unit = '(10^-5)(1/km)'
surf = 144
surf_unit = 'mb'
y_unit = 1
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end
;-------------------------------------------------------------------------
--
```

```
pro OpenA1000llK, ev

;Set default variables and widget values for Aerosol Extintion @ 1000nm at
Potential Temperature.
specie_type = 3
surf_type = 1
max_val = 9.9999999999999e-1
norm_factor=1e5 ;Will give ppm.
unit = '(10^-5)(1/km)'
surf = 700.0
surf_unit = 'Potential Temperature'
y_unit = 1
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end


;-------------------------------------------------------------------------
-----

pro OpenA1000alakm, ev

;Set default variables and widget values for Aerosol 1000 nm in km.
specie_type = 4
surf_type = 2
max_val = 9.9999999999999e-1
norm_factor=1e4 ;Will give ppm.
unit = '(10^-4)(1/km)'
surf = 70
range = 10
surf_unit = 'longitude degrees'
y_unit = 2
```

```
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end

;-------------------------------------------------------------------------
--

pro OpenA1000alamb, ev

;Set default variables and widget values for Aerosol 1000 nm at Pressure (mb).
specie_type = 5
surf_type = 2
max_val = 9.9999999999999e-1
norm_factor=1e5 ;Will give ppm.
unit = '(10^-5)(1/km)'
surf = 70
range = 10
surf_unit = 'longitude degrees'
y_unit = 3
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
```

```
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end

;-------------------------------------------------------------------------------
--

pro OpenA1000alaK, ev

;Set default variables and widget values for Aerosol 1000 nm at Potential
Temperature.
specie_type = 6
surf_type =2
max_val = 9.9999999999999e-1
norm_factor=1e4 ;Will give ppm.
unit = '(10^-4)(1/km)'
surf = 70
range = 10
surf_unit = 'longitude degrees'
y_unit = 4
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev
```

```
return
end

;-------------------------------------------------------------------------
-----

pro OpenA1000alokm, ev

;Set default variables and widget values for Aerosol extinction @ km
specie_type = 7
surf_type =3
max_val = 9.9999999999999e-1
norm_factor=1e4 ;Will give ppm.
unit = '(10^-4)(1/km)'
surf = 40
range = 5
surf_unit = 'latitude degrees'
y_unit = 2
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end

;-------------------------------------------------------------------------
--

pro OpenA1000alomb, ev

;Set default variables and widget values for Aerosol extinction @ mb.
specie_type = 8
surf_type =3
```

```
max_val = 9.9999999999999e-1
norm_factor=1e5 ;Will give ppm.
unit = '(10^-5)(1/km)'
surf = 40
range = 5
surf_unit = 'latitude degrees'
y_unit = 3
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end

;-------------------------------------------------------------------------
--

pro OpenA1000aloK, ev

;Set default variables and widget values for Aerosol extinction @ Potential
Temperature.
specie_type = 9
surf_type =3
max_val = 9.9999999999999e-1
norm_factor=1e4 ;Will give ppm.
unit = '(10^-4)(1/km)'
surf = 40
range = 5
surf_unit = 'latitude degrees'
y_unit = 4
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
```

```
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end

;---------------------------------------------------------------------------
--

pro OpenA525llkm, ev

;Set default variables and widget values for Aerosol Extintion @ 525 nm in
Kilometers (km).
specie_type = 10
surf_type = 1
max_val = 9.9999999999999e-1
norm_factor=1e3 ;Will give ppt.
unit = '(10^-3)(1/km)'
surf = 14.5
surf_unit = 'km'
y_unit = 1
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
```

```
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end

;-------------------------------------------------------------------------
--

pro OpenA525llmb, ev

;Set default variables and widget values for Aerosol extintion @ 525nm at
Pressure (mb).
specie_type = 11
surf_type = 1
max_val = 9.9999999999999e-1
norm_factor=1e5 ;Will give ppm.
unit = '(10^-5)(1/km)'
surf = 144
surf_unit = 'mb'
y_unit = 1
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end
;-------------------------------------------------------------------------
--
```

```
pro OpenA525llK, ev

;Set default variables and widget values for Aerosol Extintion @ 525 nm at
Potential Temperature.
specie_type = 12
surf_type = 1
max_val = 9.9999999999999e-1
norm_factor=1e5 ;Will give ppm.
unit = '(10^-5)(1/km)'
surf = 700.0
surf_unit = 'Potential Temperature'
y_unit = 1
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end

;-------------------------------------------------------------------------
-----

pro OpenA525alakm, ev

;Set default variables and widget values for Aerosol 525 nm in km.
specie_type = 13
surf_type = 2
max_val = 9.9999999999999e-1
norm_factor=1e4 ;
unit = '(10^-4)(1/km)'
surf = 70
range = 10
surf_unit = 'longitude degrees'
y_unit = 3
x_unit = 1
outfile = 'temp'
```

```
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev

return
end

;-------------------------------------------------------------------------
--

pro OpenA525alamb, ev

;Set default variables and widget values for Aerosol 525 nm at Pressure (mb).
specie_type = 14
surf_type = 2
max_val = 9.9999999999999e-1
norm_factor=1e5 ;Will give ppm.
unit = '(10^-5)(1/km)'
surf = 70
range = 10
surf_unit = 'longitude degrees'
y_unit = 3
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
```

```
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end

;-------------------------------------------------------------------------------
--

pro OpenA525alaK, ev

;Set default variables and widget values for Aerosol 525 nm at Potential
Temperature.
specie_type = 15
surf_type =2
max_val = 9.9999999999999e-1
norm_factor=1e4 ;Will give ppm.
unit = '(10^-4)(1/km)'
surf = 70
range = 10
surf_unit = 'longitude degrees'
y_unit = 4
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
```

```
end

;-------------------------------------------------------------------------
-----

pro OpenA525alokm, ev

;Set default variables and widget values for Aerosol extinction @ km
specie_type = 16
surf_type =3
max_val = 9.9999999999999e-1
norm_factor=1e4 ;Will give ppm.
unit = '(10^-4)(1/km)'
surf = 40
range = 5
surf_unit = 'latitude degrees'
y_unit = 2
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end

;-------------------------------------------------------------------------
--

pro OpenA525alomb, ev

;Set default variables and widget values for Aerosol extinction @ mb.
specie_type = 17
surf_type =3
max_val = 9.9999999999999e-1
norm_factor=1e5 ;Will give ppm.
unit = '(10^-5)(1/km)'
surf = 40
```

```
range = 5
surf_unit = 'latitude degrees'
y_unit = 3
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev

return
end

;-------------------------------------------------------------------------
--

pro OpenA525aloK, ev

;Set default variables and widget values for Aerosol extinction @ Potential
Temperature.
specie_type = 18
surf_type =3
max_val = 9.9999999999999e-1
norm_factor=1e4 ;Will give ppm.
unit = '(10^-4)(1/km)'
surf = 40
range = 5
surf_unit = 'latitude degrees'
y_unit = 4
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
```

```
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end

;-------------------------------------------------------------------------------
--

pro OpenA450llkm, ev

;Set default variables and widget values for Aerosol Extintion @ 450nm at
Kilometer (km).
specie_type = 19
surf_type = 1
max_val = 9.9999999999999e-1
norm_factor=1e3 ;Will give ppt.
unit = '(10^-3)(1/km)'
surf = 14.5
surf_unit = 'km'
y_unit = 1
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid
```

```
call_procedure,'reset', ev


return
end

;-------------------------------------------------------------------------
--


pro OpenA450llmb, ev

;Set default variables and widget values for Aerosol extintion @ 450nm at
Pressure (mb).
specie_type = 20
surf_type = 1
max_val = 9.9999999999999e-1
norm_factor=1e5 ;Will give ppm.
unit = '(10^-5)(1/km)'
surf = 144
surf_unit = 'mb'
y_unit = 1
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end
;-------------------------------------------------------------------------
--

pro OpenA450llK, ev

;Set default variables and widget values for Aerosol Extintion @ 450nm at
```

```
Potential Temperature.
specie_type = 21
surf_type = 1
max_val = 9.9999999999999e-1
norm_factor=1e5 ;Will give ppm.
unit = '(10^-5)(1/km)'
surf = 700.0
surf_unit = 'Potential Temperature'
y_unit = 1
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end

;-------------------------------------------------------------------------
-----

pro OpenA450alakm, ev

;Set default variables and widget values for Aerosol 450 nm in km.
specie_type = 22
surf_type = 2
max_val = 9.9999999999999e-1
norm_factor=1e4 ;Will give ppm.
unit = '(10^-4)(1/km)'
surf = 70
range = 10
surf_unit = 'longitude degrees'
y_unit = 2
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top
```

```
;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end

;-------------------------------------------------------------------------------
--

pro OpenA450alamb, ev

;Set default variables and widget values for Aerosol 450 nm at Pressure (mb).
specie_type = 23
surf_type = 2
max_val = 9.9999999999999e-1
norm_factor=1e5 ;Will give ppm.
unit = '(10^-5)(1/km)'
surf = 70
range = 10
surf_unit = 'longitude degrees'
y_unit = 3
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
```

```
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev

return
end

;-------------------------------------------------------------------------------
--

pro OpenA450alaK, ev

;Set default variables and widget values for Aerosol 450 nm at Potential
Temperature.
specie_type = 24
surf_type =2
max_val = 9.9999999999999e-1
norm_factor=1e4 ;Will give ppm.
unit = '(10^-4)(1/km)'
surf = 70
range = 10
surf_unit = 'longitude degrees'
y_unit = 4
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end
```

```
;-------------------------------------------------------------------------
-----

pro OpenA450alokm, ev

;Set default variables and widget values for Aerosol extinction @ km
specie_type = 25
surf_type =3
max_val = 9.9999999999999e-1
norm_factor=1e4 ;Will give ppm.
unit = '(10^-4)(1/km)'
surf = 40
range = 5
surf_unit = 'latitude degrees'
y_unit = 2
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev



return
end

;-------------------------------------------------------------------------
--

pro OpenA450alomb, ev

;Set default variables and widget values for Aerosol extinction @ mb.
specie_type = 26
surf_type =3
max_val = 9.9999999999999e-1
norm_factor=1e5 ;Will give ppm.
unit = '(10^-5)(1/km)'
surf = 40
range = 5
```

```
surf_unit = 'latitude degrees'
y_unit = 3
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end

;-------------------------------------------------------------------------
--

pro OpenA450aloK, ev

;Set default variables and widget values for Aerosol extinction @ Potential
Temperature.
specie_type = 27
surf_type =3
max_val = 9.9999999999999e-1
norm_factor=1e4 ;Will give ppm.
unit = '(10^-4)(1/km)'
surf = 40
range = 5
surf_unit = 'latitude degrees'
y_unit = 4
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
```

```
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end



;---------------------------------------------------------------------------
--

pro OpenA38511km, ev

;Set default variables and widget values for Aerosol Extintion @ 1000nm at
Kilometers (km).
specie_type = 28
surf_type = 1
max_val = 9.9999999999999e-1
norm_factor=1e3 ;Will give ppt.
unit = '(10^-3)(1/km)'
surf = 14.5
surf_unit = 'km'
y_unit = 1
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
```

```
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end

;-------------------------------------------------------------------------
--

pro OpenA38511mb, ev

;Set default variables and widget values for Aerosol extintion @ 1000nm at
Pressure (mb).
specie_type = 29
surf_type = 1
max_val = 9.9999999999999e-1
norm_factor=1e5 ;Will give ppm.
unit = '(10^-5)(1/km)'
surf = 144
surf_unit = 'mb'
y_unit = 1
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end
;-------------------------------------------------------------------------
--

pro OpenA38511K, ev

;Set default variables and widget values for Aerosol Extintion @ 1000nm at
```

```
Potential Temperature.
specie_type = 30
surf_type = 1
max_val = 9.9999999999999e-1
norm_factor=1e5 ;Will give ppm.
unit = '(10^-5)(1/km)'
surf = 700.0
surf_unit = 'Potential Temperature'
y_unit = 1
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end


;-------------------------------------------------------------------------
-----

pro OpenA385alakm, ev

;Set default variables and widget values for Aerosol 1000 nm in km.
specie_type = 31
surf_type = 2
max_val = 9.9999999999999e-1
norm_factor=1e4 ;Will give ppm.
unit = '(10^-4)(1/km)'
surf = 70
range = 10
surf_unit = 'longitude degrees'
y_unit = 2
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top
```

```
;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end

;-------------------------------------------------------------------------
--

pro OpenA385alamb, ev

;Set default variables and widget values for Aerosol 1000 nm at Pressure (mb).
specie_type = 32
surf_type = 2
max_val = 9.9999999999999e-1
norm_factor=1e5 ;Will give ppm.
unit = '(10^-5)(1/km)'
surf = 70
range = 10
surf_unit = 'longitude degrees'
y_unit = 3
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
```

```
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev

return
end

;-------------------------------------------------------------------------
--

pro OpenA385alaK, ev

;Set default variables and widget values for Aerosol 1000 nm at Potential
Temperature.
specie_type = 33
surf_type =2
max_val = 9.9999999999999e-1
norm_factor=1e4 ;Will give ppm.
unit = '(10^-4)(1/km)'
surf = 70
range = 10
surf_unit = 'longitude degrees'
y_unit = 4
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end
```

```
;-------------------------------------------------------------------------
-----

pro OpenA385alokm, ev

;Set default variables and widget values for Aerosol extinction @ km
specie_type = 34
surf_type =3
max_val = 9.9999999999999e-1
norm_factor=1e4 ;Will give ppm.
unit = '(10^-4)(1/km)'
surf = 40
range = 5
surf_unit = 'latitude degrees'
y_unit = 2
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end

;-------------------------------------------------------------------------
--

pro OpenA385alomb, ev

;Set default variables and widget values for Aerosol extinction @ mb.
specie_type = 35
surf_type =3
max_val = 9.9999999999999e-1
norm_factor=1e5 ;Will give ppm.
unit = '(10^-5)(1/km)'
surf = 40
range = 5
```

```
surf_unit = 'latitude degrees'
y_unit = 3
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end

;-------------------------------------------------------------------------------
--

pro OpenA385aloK, ev

;Set default variables and widget values for Aerosol extinction @ Potential
Temperature.
specie_type = 36
surf_type =3
max_val = 9.9999999999999e-1
norm_factor=1e4 ;Will give ppm.
unit = '(10^-4)(1/km)'
surf = 40
range = 5
surf_unit = 'latitude degrees'
y_unit = 4
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
```

```
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end

;-------------------------------------------------------------------------------
--

pro OpenNO2MR11km, ev

;Set default variables and widget values for NO2 Mixing Ratio @ km.
specie_type = 37
surf_type = 1
max_val = 9.9999999999999e-1
norm_factor=1e9 ;Will give ppb.
unit = '(ppb)'
surf = 14.5
surf_unit = 'km'
y_unit = 1
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid
```

```
call_procedure,'reset', ev


return
end

;-------------------------------------------------------------------------
--

pro OpenNO2MRllmb, ev

;Set default variables and widget values for NO2 Mixing Ratio at Pressure (mb)
.
specie_type = 38
surf_type = 1
max_val = 9.9999999999999e-1
norm_factor=1e9 ;Will give ppb.
unit = '(ppb)'
surf = 144
surf_unit = 'mb'
y_unit = 1
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end
;-------------------------------------------------------------------------
--

pro OpenNO2MRllK, ev

;Set default variables and widget values for NO2 Mixing Ratio at Potential
Temperature.
specie_type = 39
surf_type = 1
```

```
max_val = 9.9999999999999e-1
norm_factor=1e9 ;Will give ppb.
unit = '(ppb)'
surf = 700.0
surf_unit = 'Potential Temperature'
y_unit = 1
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end


;-------------------------------------------------------------------------
-----

pro OpenNO2MRalakm, ev

;Set default variables and widget values for NO2 Mixing Ratio in km.
specie_type = 40
surf_type = 2
max_val = 9.9999999999999e-1
norm_factor=1e9 ;Will give ppb.
unit = '(ppb)'
surf = 70
range = 10
surf_unit = 'longitude degrees'
y_unit = 2
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
```

```
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end

;-------------------------------------------------------------------------------
--

pro OpenNO2MRalamb, ev

;Set default variables and widget values for NO2 Mixing Ratio at Pressure (mb)
.
specie_type = 41
surf_type = 2
max_val = 9.9999999999999e-1
norm_factor=1e9 ;Will give ppb.
unit = '(ppb)'
surf = 70
range = 10
surf_unit = 'longitude degrees'
y_unit = 3
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
```

```
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end

;-------------------------------------------------------------------------
--

pro OpenNO2MRalaK, ev

;Set default variables and widget values for NO2 Mixing Ratio at Potential
Temperature.
specie_type = 42
surf_type =2
max_val = 9.9999999999999e-1
norm_factor=1e9 ;Will give ppb.
unit = '(ppb)'
surf = 70
range = 10
surf_unit = 'longitude degrees'
y_unit = 4
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end
```

```
;-------------------------------------------------------------------------
-----

pro OpenNO2MRalokm, ev

;Set default variables and widget values for NO2 Mixing Ratio@ km
specie_type = 43
surf_type =3
max_val = 9.9999999999999e-1
norm_factor=1e9 ;Will give ppb.
unit = '(ppb)'
surf = 40
range = 5
surf_unit = 'latitude degrees'
y_unit = 2
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end

;-------------------------------------------------------------------------
--

pro OpenNO2MRalomb, ev

;Set default variables and widget values for NO2 Mixing Ratio @ mb.
specie_type = 44
surf_type =3
max_val = 9.9999999999999e-1
norm_factor=1e9 ;Will give ppb.
unit = '(ppb)'
surf = 40
range = 5
surf_unit = 'latitude degrees'
```

```
y_unit = 3
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end

;-------------------------------------------------------------------------
--

pro OpenNO2MRaloK, ev

;Set default variables and widget values for NO2 Mixing Ratio @ Potential
Temperature.
specie_type = 45
surf_type =3
max_val = 9.9999999999999e-1
norm_factor=1e9 ;Will give ppb.
unit = '(ppb)'
surf = 40
range = 5
surf_unit = 'latitude degrees'
y_unit = 4
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val
```

```
;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev

return
end

;-------------------------------------------------------------------------------
--

pro OpenO3MRllkm, ev

;Set default variables and widget values for O3 Mixing Ratios at Kilometers
(km).
specie_type = 64
surf_type = 1
max_val = 9.9999999999999e-1
norm_factor=1e6 ;Will give ppm for the mixing ratio.
unit = '(ppm)'
surf = 14.5
surf_unit = 'km'
y_unit = 1
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev
```

```
return
end

;-------------------------------------------------------------------------
--

pro OpenO3MR1lmb, ev

;Set default variables and widget values for O3 Mixing Ratios at Pressure (mb)
.
specie_type = 65
surf_type = 1
max_val = 9.9999999999999e-1
norm_factor=1e6 ;Will give ppm for the mixing ratio.
unit = '(ppm)'
surf = 144
surf_unit = 'mb'
y_unit = 1
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end
;-------------------------------------------------------------------------
--

pro OpenO3MR1lK, ev

;Set default variables and widget values for O3 Mixing Ratios at Potential
Temperature.
specie_type = 66
surf_type = 1
max_val = 9.9999999999999e-1
```

```
norm_factor=1e6 ;Will give ppm for the mixing ratio.
unit = '(ppm)'
surf = 700.0
surf_unit = 'Potential Temperature'
y_unit = 1
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end
;-------------------------------------------------------------------------
--

pro OpenO3MRalakm, ev

;Set default variables and widget values for O3 Mixing Ratios at kilometers
(km).
specie_type = 67
surf_type = 2
max_val = 9.9999999999999e-1
norm_factor=1e6 ;Will give ppm for the mixing ratio.
unit = '(ppm)'
surf = 70
range = 10
surf_unit = 'longitude degrees'
y_unit = 2
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent
```

```
;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end

;-------------------------------------------------------------------------------
--

pro OpenO3MRalamb, ev

;Set default variables and widget values for O3 Mixing Ratios at Pressure (mb)
.
specie_type = 68
surf_type = 2
max_val = 9.9999999999999e-1
norm_factor=1e6 ;Will give ppm for the mixing ratio.
unit = '(ppm)'
surf = 70
range = 10
surf_unit = 'longitude degrees'
y_unit = 3
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state
```

```
;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end

;---------------------------------------------------------------------------
--

pro OpenO3MRalaK, ev

;Set default variables and widget values for O3 Mixing Ratios at Potential
Temperature.
specie_type = 69
surf_type =2
max_val = 9.9999999999999e-1
norm_factor=1e6 ;Will give ppm for the mixing ratio.
unit = '(ppm)'
surf = 70
range = 10
surf_unit = 'longitude degrees'
y_unit = 4
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end

;---------------------------------------------------------------------------
--
```

```
pro OpenO3MRalokm, ev

;Set default variables and widget values for O3 Mixing Ratios at Kilometers
(km).
specie_type = 70
surf_type =3
max_val = 9.9999999999999e-1
norm_factor=1e6 ;Will give ppm for the mixing ratio.
unit = '(ppm)'
surf = 40
range = 5
surf_unit = 'latitude degrees'
y_unit = 2
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev


return
end
;-------------------------------------------------------------------------
--

pro OpenO3MRalomb, ev

;Set default variables and widget values for O3 Mixing Ratios at Pressure (mb)
.
specie_type = 71
surf_type =3
max_val = 9.9999999999999e-1
norm_factor=1e6 ;Will give ppm for the mixing ratio.
unit = '(ppm)'
surf = 40
range = 5
surf_unit = 'latitude degrees'
y_unit = 3
x_unit = 1
```

```
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev

return
end

;-------------------------------------------------------------------------
--


pro Open03MRaloK, ev

;Set default variables and widget values for 03 Mixing Ratios at Potential
Temperature.
specie_type = 72
surf_type =3
max_val = 9.9999999999999e-1
norm_factor=1e6 ;Will give ppm for the mixing ratio.
unit = '(ppm)'
surf = 40
range = 5
surf_unit = 'latitude degrees'
y_unit = 4
x_unit = 1
outfile = 'temp'
widget_control, ev.top, get_uvalue=state
parent = ev.top

;Gets information from the user about the file to be opened.
call_procedure, 'getinfo', nummonth, surf, header, surf_unit, surf_type,
range, cancel=cancel, parent=parent

;Reads the data and creates and output file.
call_procedure,'month_read', specie_type, surf_type, nummonth, surf, header,
range, outfile, norm_factor, unit, surf_unit, y_unit, max_val

;Sets the file name for passing.
```

```
state.specie_type = specie_type
state.surf_type = surf_type
state.max_val = max_val
state.norm_factor = norm_factor
state.unit = unit
state.outfile = outfile
widget_control, ev.top, set_uvalue=state

;Sets window to the appropiate draw widget
widget_control, state.draw2, get_value=winid
wset, winid

call_procedure,'reset', ev

return
end

;-------------------------------------------------------------------------
--
pro reset, ev

widget_control, ev.top, get_uvalue=state
parent = ev.top

      ;Sets window to the appropiate draw widget
      widget_control, state.draw2, get_value=winid
      wset, winid

      ;Plots the raw data.
      call_procedure, 'plot_points', state.outfile, state.surf_type, state.
levels,x_axis_max, x_axis_min, y_axis_max, y_axis_min

      ;Sets window to the appropiate draw widget
      widget_control, state.draw, get_value=winid
      wset, winid

      ;Grids the data and draws a cellfilled contour.
      call_procedure, 'Grid_Smooth_Map', state.outfile, state.lat_step, state.
lon_step, state.smooth_pass, state.levels, x_axis_max, x_axis_min, y_axis_max,
y_axis_min

state.x_axis_max = x_axis_max
state.x_axis_min = x_axis_min
state.y_axis_max = y_axis_max
state.y_axis_min = y_axis_min

widget_control, ev.top, set_uvalue=state




;     ;Sets window to the appropiate draw widget
;     widget_control, state.draw2, get_value=winid
;     wset, winid

      ;Plots the raw data.
;     call_procedure,'plot_points', state.outfile

      ;Sets window to the appropiate draw widget
;     widget_control, state.draw, get_value=winid
```

```
;     wset, winid

      ;Grids the data and draws a cellfilled contour.
;     call_procedure, 'Grid_Smooth_Map', state.outfile


return
end
;-------------------------------------------------------------------------------
--
;-------------------------------------------------------------------------------
--
;-------------------------------------------------------------------------------
--
pro Color_Table, ev
      CALL_Procedure, 'xloadct'

return
end


;-------------------------------------------------------------------------------
--

pro WidgetDone, ev

Widget_Control, ev.top, /destroy

return
end

;-------------------------------------------------------------------------------
--

pro Process_Gif, ev

; This event is responsible for creating a GIF file

thisfile = dialog_pickfile(/write, file='DATA.gif')
if thisfile eq '' then return

Widget_Control, ev.top, Get_UValue=state, /No_Copy
WSet, state.winid
thisImage = TVRD()
TVLCT, r, g, b, /Get
Write_GIF, thisFile, thisImage, r, g, b
Widget_Control, ev.top, Set_UValue=state, /No_Copy
END

;-------------------------------------------------------------------------------
--

pro Process_JPEG, ev

   ; This event handler is responsible for creating a JPEG file.

thisFile = Dialog_Pickfile(/Write, File='DATA.jpg')
IF thisFile EQ '' THEN RETURN

Widget_Control, ev.top, Get_UValue=state, /No_Copy
WSet, state.winid
```

```
thisImage = TVRD()
TVLCT, r, g, b, /Get

   ; Create 24-bit color image.

image24 = BytArr(3, state.xsize, state.ysize)
image24(0,*,*) = r(thisImage)
image24(1,*,*) = g(thisImage)
image24(2,*,*) = b(thisImage)

   ; Write file. Normal image quality in this lossy format.

Write_JPEG, thisFile, image24, True=1, Quality=75
Widget_Control, ev.top, Set_UValue=state, /No_Copy
END
;-------------------------------------------------------------------------
--

PRO Postscript_Save, ev

filename = Dialog_Pickfile(/Write, File='DATA.ps')
IF filename EQ '' THEN RETURN

      ; Get the screen dump of the current graphics window.

   screenDump = TVRD()

      ; Open a PostScript file and dump it.

   thisDevice = !D.NAME
   SET_PLOT, 'PS'

      ; Make a PostScript window with the same aspect
      ; ratio as the current display window. Use color
      ; PostScript if the COLOR keyword is set.

 ;;;; aspect = PSWINDOW()
   DEVICE, FILENAME=filename, XSIZE=aspect.xsize, YSIZE=aspect.ysize, $
      XOFFSET=aspect.xoffset, YOFFSET=aspect.yoffset, COLOR=KEYWORD_SET(color)

      ; Display the screen dump. Fill up the window.

   TV, screenDump, XSIZE=!D.X_SIZE, YSIZE=!D.Y_SIZE
   DEVICE, /CLOSE_FILE
   SET_PLOT, thisDevice
   END

;-------------------------------------------------------------------------
--

pro ReadSage_EVENT ,ev

widget_control,ev.top,get_uvalue=state

; What type of event was it?
wtype = tag_names(ev,/struct)


;Slider Widget

if(wtype eq 'WIDGET_DRAW')then begin
```

```
      dx = ev.x
      dy = ev.y

      xyd = convert_coord(dx,dy, /device, /to_normal)

      xd = xyd(0,0)
      yd = xyd(1,0)

      xyid = widget_info(ev.top, find_by_uname = 'xy')

      loc = string(FORMAT='(f7.1,f7.1)',xd,yd)

      widget_control, xyid, set_value=loc

endif

;Button Widget

if(wtype eq 'WIDGET_BUTTON')then begin

      widget_control,ev.id,get_value=typ

        case typ of

            'Done': widget_control,/destroy,ev.top



        endcase

 endif

;Slider Widget

if(wtype eq 'WIDGET_FSLIDER')then begin

      levels = state.levels

            step = ev.value/10.0

            for j = 0, 10 do begin

                  levels(j) = j*step

            endfor

      ;Sets window to the appropiate draw widget
      widget_control, state.draw2, get_value=winid
      wset, winid

      ;Plots the raw data.
      call_procedure, 'plot_points', state.outfile, state.surf_type, levels,
state.x_axis_max, state.x_axis_min, state.y_axis_max, state.y_axis_min

      ;Sets window to the appropiate draw widget
      widget_control, state.draw, get_value=winid
      wset, winid

      ;Grids the data and draws a cellfilled contour.
      call_procedure, 'Grid_Smooth_Map', state.outfile, state.lat_step, state.
```

```
lon_step, state.smooth_pass, levels, state.x_axis_max, state.x_axis_min,
state.y_axis_max, state.y_axis_min

      state.levels = levels

      widget_control,ev.top,set_uvalue=state
endif

return
end

;-------------------------------------------------------------------------
--

pro ReadSage

device, pseudo=8, retain=2
!p.background = 0

; Set dialog in the middle of the screen

device, get_screen_size = screensize
xcenter = fix(screensize[0] / 2.0)
ycenter = fix(screensize[1] / 2.0)



base = widget_base(mbar=bar, title = 'SAGE Data Surface Plate Visualization',
ysize=screensize[1], xsize=screensize[0], /row, /tlb_size_events)

; Creates FILE menu button

menu1 = widget_button(bar, value='File', /menu)
  button1 = widget_button(menu1, value='New Window'       , event_pro='New')
  button2 = widget_button(menu1, value='Open g2p V5.931'             , /menu)

;Aerosol Menu
      but1 = widget_button(button2, value='Aerosol 1000 nm'        , /menu)
            but11 = widget_button(but1, value='Vs latitude & longitude', /
menu)
                  but111 = widget_button(but11, value='At altitute in km',
event_pro='OpenA1000llkm')
                  but112 = widget_button(but11, value='At altitute in mb',
event_pro='OpenA1000llmb')
                  but113 = widget_button(but11, value='At altitute in
Potential Temperature', event_pro='OpenA1000llK')
            but12 = widget_button(but1, value='Vs altitude & latitude', /menu)
                  but121 = widget_button(but12, value='Altitute in km',
event_pro='OpenA1000alakm')
                  but122 = widget_button(but12, value='Altitute in mb',
event_pro='OpenA1000alamb')
                  but123 = widget_button(but12, value='Altitute in Potential
Temperature',     event_pro='OpenA1000alaK')
            but13 = widget_button(but1, value='Vs altitude & longitude', /
menu)
                  but131 = widget_button(but13, value='Altitute in km',
event_pro='OpenA1000alokm')
                  but132 = widget_button(but13, value='Altitute in mb',
event_pro='OpenA1000alomb')
                  but133 = widget_button(but13, value='Altitute in Potential
Temperature',     event_pro='OpenA1000aloK')
```

```
      but2 = widget_button(button2, value='Aerosol 525 nm'       , /menu)
            but21 = widget_button(but2, value='Vs latitude & longitude', /
menu)
                  but211 = widget_button(but21, value='At altitute in km',
event_pro='OpenA52511km')
                  but212 = widget_button(but21, value='At altitute in mb',
event_pro='OpenA52511mb')
                  but213 = widget_button(but21, value='At altitute in
Potential Temperature', event_pro='OpenA52511K')
            but22 = widget_button(but2, value='Vs altitude & latitude', /menu)
                  but221 = widget_button(but22, value='Altitute in km',
event_pro='OpenA525alakm')
                  but222 = widget_button(but22, value='Altitute in mb',
event_pro='OpenA525alamb')
                  but223 = widget_button(but22, value='Altitute in Potential
Temperature',      event_pro='OpenA525alaK')
            but23 = widget_button(but2, value='Vs altitude & longitude', /
menu)
                  but231 = widget_button(but23, value='Altitute in km',
event_pro='OpenA525alokm')
                  but232 = widget_button(but23, value='Altitute in mb',
event_pro='OpenA525alomb')
                  but233 = widget_button(but23, value='Altitute in Potential
Temperature',      event_pro='OpenA525aloK')

      but3 = widget_button(button2, value='Aerosol 450 nm'       , /menu)
            but31 = widget_button(but3, value='Vs latitude & longitude', /
menu)
                  but311 = widget_button(but31, value='At altitute in km',
event_pro='OpenA45011km')
                  but312 = widget_button(but31, value='At altitute in mb',
event_pro='OpenA45011mb')
                  but313 = widget_button(but31, value='At altitute in
Potential Temperature', event_pro='OpenA45011K')
            but32 = widget_button(but3, value='Vs altitude & latitude', /menu)
                  but321 = widget_button(but32, value='Altitute in km',
event_pro='OpenA450alakm')
                  but322 = widget_button(but32, value='Altitute in mb',
event_pro='OpenA450alamb')
                  but323 = widget_button(but32, value='Altitute in Potential
Temperature',      event_pro='OpenA450alaK')
            but33 = widget_button(but3, value='Vs altitude & longitude', /
menu)
                  but331 = widget_button(but33, value='Altitute in km',
event_pro='OpenA450alokm')
                  but332 = widget_button(but33, value='Altitute in mb',
event_pro='OpenA450alomb')
                  but333 = widget_button(but33, value='Altitute in Potential
Temperature',      event_pro='OpenA450aloK')

      but4 = widget_button(button2, value='Aerosol 385 nm'       , /menu)
            but41 = widget_button(but4, value='Vs latitude & longitude', /
menu)
                  but411 = widget_button(but41, value='At altitute in km',
event_pro='OpenA38511km')
                  but412 = widget_button(but41, value='At altitute in mb',
event_pro='OpenA38511mb')
                  but413 = widget_button(but41, value='At altitute in
Potential Temperature', event_pro='OpenA38511K')
```

```
            but42 = widget_button(but4, value='Vs altitude & latitude', /menu)
                  but421 = widget_button(but42, value='Altitute in km',
event_pro='OpenA385alakm')
                  but422 = widget_button(but42, value='Altitute in mb',
event_pro='OpenA385alamb')
                  but423 = widget_button(but42, value='Altitute in Potential
Temperature',     event_pro='OpenA385alaK')
            but43 = widget_button(but4, value='Vs altitude & longitude', /
menu)
                  but431 = widget_button(but43, value='Altitute in km',
event_pro='OpenA385alokm')
                  but432 = widget_button(but43, value='Altitute in mb',
event_pro='OpenA385alomb')
                  but433 = widget_button(but43, value='Altitute in Potential
Temperature',     event_pro='OpenA385aloK')


; NO2 Menu
      but2 = widget_button(button2, value= 'NO2 Mixing Ratio'        , /menu)
            but21 = widget_button(but2, value='Vs latitude & longitude', /
menu)
                  but211 = widget_button(but21, value='At altitute in km',
event_pro='OpenNO2MRllkm')
                  but212 = widget_button(but21, value='At altitute in mb',
event_pro='OpenNO2MRllmb')
                  but213 = widget_button(but21, value='At altitute in
Potential Temperature', event_pro='OpenNO2MRllK')
            but22 = widget_button(but2, value='Vs altitude & latitude', /menu)
                  but221 = widget_button(but22, value='Altitute in km',
event_pro='OpenNO2MRalakm')
                  but222 = widget_button(but22, value='Altitute in mb',
event_pro='OpenNO2MRalamb')
                  but223 = widget_button(but22, value='Altitute in Potential
Temperature',     event_pro='OpenNO2MRalaK')
            but23 = widget_button(but2, value='Vs altitude & longitude', /
menu)
                  but231 = widget_button(but23, value='Altitute in km',
event_pro='OpenNO2MRalokm')
                  but232 = widget_button(but23, value='Altitute in mb',
event_pro='OpenNO2MRalomb')
                  but233 = widget_button(but23, value='Altitute in Potential
Temperature',     event_pro='OpenNO2MRaloK')

; Ozone Menu
     but3 = widget_button(button2, value= 'Ozone Mixing Ratio'        , /menu)

            but31 = widget_button(but3, value='Vs latitude & longitude', /
menu)
                  but311 = widget_button(but31, value='At altitute in km',
event_pro='OpenO3MRllkm')
                  but312 = widget_button(but31, value='At altitute in mb',
event_pro='OpenO3MRllmb')
                  but313 = widget_button(but31, value='At altitute in
Potential Temperature', event_pro='OpenO3MRllK')
            but32 = widget_button(but3, value='Vs altitude & latitude', /menu)
                  but321 = widget_button(but32, value='Altitute in km',
event_pro='OpenO3MRalakm')
                  but322 = widget_button(but32, value='Altitute in mb',
event_pro='OpenO3MRalamb')
                  but323 = widget_button(but32, value='Altitute in Potential
Temperature',     event_pro='OpenO3MRalaK')
```

```
            but33 = widget_button(but3, value='Vs altitude & longitude', /
menu)
                  but331 = widget_button(but33, value='Altitute in km',
event_pro='OpenO3MRalokm')
                  but332 = widget_button(but33, value='Altitute in mb',
event_pro='OpenO3MRalomb')
                  but333 = widget_button(but33, value='Altitute in Potential
Temperature',     event_pro='OpenO3MRaloK')

  ;Water Vapor menu
  ;   but4 = widget_button(button2, value='Water Vapor'    , /menu)
  ;                but41 = widget_button(but4, value='Mixing Ratio', /menu)
  ;                but411 = widget_button(but41, value='Vs latitude &
longitude', /menu)
  ;                       but4111 = widget_button(but411, value='Altitute in
km',                                  event_pro='OpenH2OMRllkm')
  ;                       but4112 = widget_button(but411, value='Altitute in
mb',                                  event_pro='OpenH2OMRllmb')
  ;                       but4113 = widget_button(but411, value='Altitute in K',
event_pro='OpenH2OMRllK')
  ;                but412 = widget_button(but41, value='Vs altitude &
latitude', /menu)
  ;                       but4121 = widget_button(but412, value='Altitute in
km',                                  event_pro='OpenH2OMRalakm')
  ;                       but4122 = widget_button(but412, value='Altitute in
mb',                                  event_pro='OpenH2OMRalamb')
  ;                       but4123 = widget_button(but412, value='Altitute in K',
event_pro='OpenH2OMRalaK')
  ;                but413 = widget_button(but41, value='Vs altitude &
latitude', /menu)
  ;                       but4131 = widget_button(but413, value='Altitute in
km',                                  event_pro='OpenH2OMRalokm')
  ;                       but4132 = widget_button(but413, value='Altitute in
mb',                                  event_pro='OpenH2OMRalomb')
  ;                       but4133 = widget_button(but413, value='Altitute in K',
event_pro='OpenH2OMRaloK')


  button3 = widget_button(menu1, value='Open Specie File' , event_pro=
'Specie_file')
  button4 = widget_button(menu1, value='Close'             , event_pro='three')
  button6 = widget_button(menu1, value='Save Plates As'          , /menu)
    but1 = widget_button(button6, value='GIF File'        , event_pro=
'Process_gif')
    but1 = widget_button(button6, value='JPEG File'       , event_pro=
'Process_jpeg')
    but1 = widget_button(button6, value='PostScript File' , event_pro=
'Postscript_save')
  ;button22 = widget_button(menu1, value='Print'           , event_pro=
'Print_it')
  button6 = widget_button(menu1, value='Quit', /separator , event_pro=
'WidgetDone')

; Manipulation Options

Man_baseID = widget_base(base,/column,/frame)
label = widget_label(Man_baseID, value= 'Options')
label = widget_label(Man_baseID, value='  ')
lb20 = widget_Button(Man_baseID, value='Color Table', event_pro='Color_Table')
Grid_butID = widget_Button(Man_baseID, value= 'Regrid and Smooth', event_pro=
'regrid')
```

```
scale_butID = widget_Button(Man_baseID, value= 'Rescale Axes', event_pro=
'rescale')
sliderID = cw_fslider(Man_baseID, Title = 'Colorbar Scale Maximum', value=10,
maximum=100.0, /frame, /edit)


;Sections for drawing
Draw_BaseID = widget_base(base, /column, /frame)

windowx=screensize[0]*.75
windowy=screensize[1]*.42

; Draw Window for raw data
rbot3 = widget_base(Draw_baseID, /column, /frame)
 draw2 = widget_draw(rbot3, xsize=windowx, ysize=windowy, /frame, /
motion_events)

; Draw Window for gridded data

rbot2 = widget_base(Draw_baseID, /column, /frame)
 draw = widget_draw(rbot2, xsize=windowx, ysize=windowy, /frame, /
motion_events)

 MapStuff_baseID = widget_base(rbot2,space=10,/row,/frame)
    but2 = widget_button(MapStuff_baseID, value='Reset', event_pro='reset')
    but3 = widget_button(MapStuff_baseID, value='Done')

  rbt0 = widget_base(MapStuff_baseID,/row,/frame)
    lbl0 = widget_label(rbt0,value='Location')
    txt0 = widget_text( rbt0,value=' ', uname = 'xy')
    InfoID = widget_base(MapStuff_baseID, /column)
      meID = widget_label(InfoID,value = 'Created by NSU ViSAGE Team', ysize =
10)
      NAPFID = widget_label(InfoID,value = 'Waldo J. Rodriguez, PI', ysize =
7)
      NSUID = widget_label(InfoID,value = 'Norfolk State University', ysize =
10)

; Realizes the Widget
widget_control, base, /realize

; Draws First map
widget_control, draw2, get_value=winid
WSET, winid
;MAP_SET, /CONTINENTS, /MERCATOR
MAP_SET, /MERCATOR
MAP_CONTINENTS, color=255
MAP_Grid, /Label, color=255

; Draws Second map
widget_control, draw, get_value=winid
WSET, winid
;MAP_SET, /CONTINENTS, /MERCATOR
MAP_SET, /MERCATOR
MAP_CONTINENTS, color=255
MAP_Grid, /Label, color=255


; State is utilize
state ={draw:draw, $
        draw2:draw2, $
```

```
        winid:winid, $
        xsize:windowx, $
        ysize:windowy, $
        specie_type:1,$
        surf_type:1,$
        max_val:9.99999e-1,$
        norm_factor:1e6,$ ; for ppm
        unit:'ppm',$
        outfile:' ',$
        lat_step:8.0,$
      lon_step:10.0,$
      smooth_pass:9, $
      levels:findgen(11), $
      x_axis_max:0, $
      x_axis_min:0, $
      y_axis_max:0, $
      y_axis_min:0}

widget_control, base, set_uvalue = state
xmanager, 'ReadSage', base

return
end
```

```
pro reset, outfile, surf_type, levels, x_axis_max, x_axis_min, y_axis_max,
y_axis_min, y_step, x_step, smooth_pass, cont_levels

widget_control, ev.top, get_uvalue=state
parent = ev.top

      ;Sets window to the appropiate draw widget
      widget_control, state.draw2, get_value=winid
      wset, winid

      ;Plots the raw data.
      call_procedure, 'plot_points', state.outfile, state.surf_type, state.
levels,x_axis_max, x_axis_min, y_axis_max, y_axis_min

      ;Sets window to the appropiate draw widget
      widget_control, state.draw, get_value=winid
      wset, winid

      ;Grids the data and draws a cellfilled contour.
      call_procedure, 'Grid_Smooth_Map', state.outfile, state.lat_step, state.
lon_step, state.smooth_pass, state.levels, x_axis_max, x_axis_min, y_axis_max,
y_axis_min

state.x_axis_max = x_axis_max
state.x_axis_min = x_axis_min
state.y_axis_max = y_axis_max
state.y_axis_min = y_axis_min

widget_control, ev.top, set_uvalue=state




;     ;Sets window to the appropiate draw widget
;     widget_control, state.draw2, get_value=winid
;     wset, winid

      ;Plots the raw data.
;     call_procedure,'plot_points', state.outfile

      ;Sets window to the appropiate draw widget
;     widget_control, state.draw, get_value=winid
;     wset, winid

      ;Grids the data and draws a cellfilled contour.
;     call_procedure, 'Grid_Smooth_Map', state.outfile


return
end
```

```
; $Id: cw_fslider.pro,v 1.11 1998/03/10 20:33:20 alan Exp $
;
; Copyright (c) 1992-1998, Research Systems, Inc.  All rights reserved.
;     Unauthorized reproduction prohibited.
;+
; NAME:
;     CW_FSLIDER
;
; PURPOSE:
;     The standard slider provided by the WIDGET_SLIDER() function is
;     integer only. This compound widget provides a floating point
;     slider.
;
; CATEGORY:
;     Compound widgets.
;
; CALLING SEQUENCE:
;     widget = CW_FSLIDER(Parent)
;
; INPUTS:
;       Parent:          The ID of the parent widget.
;
; KEYWORD PARAMETERS:
;     DRAG:        Set this keyword to zero if events should only
;                  be generated when the mouse is released. If it is
;                  non-zero, events will be generated continuously
;                  when the slider is adjusted. Note: On slow systems,
;                  /DRAG performance can be inadequate. The default
;                  is DRAG=0.
;       EDIT:            Set this keyword to make the slider label be
;                  editable. The default is EDIT=0.
;     FORMAT:            Provides the format in which the slider value is
;                  displayed. This should be a format as accepted by
;                  the STRING procedure. The default is FORMAT='(G13.6)'
;     FRAME:             Set this keyword to have a frame drawn around the
;                  widget. The default is FRAME=0.
;     MAXIMUM:     The maximum value of the slider. The default is
;                  MAXIMUM=100.
;     MINIMUM:     The minimum value of the slider. The default is
;                  MINIMUM=0.
;     SCROLL             Sets the SCROLL keyword to the WIDGET_SLIDER
underlying
;                  this compound widget. Unlike WIDGET_SLIDER, the
;                  value given to SCROLL is taken in the floating units
;                  established by MAXIMUM and MINIMUM, and not in pixels.
;     SUPPRESS_VALUE:    If true, the current slider value is not displayed.
;                  The default is SUPPRESS_VALUE=0.
;     TITLE:             The title of slider. (The default is no title.)
;     UVALUE:            The user value for the widget.
;     VALUE:             The initial value of the slider
;     VERTICAL:    If set, the slider will be oriented vertically.
;                  The default is horizontal.
;     XSIZE:             For horizontal sliders, sets the length.
;     YSIZE:             For vertical sliders, sets the height.
;
; OUTPUTS:
;       The ID of the created widget is returned.
;
; SIDE EFFECTS:
;     This widget generates event structures containing a field
;     named value when its selection thumb is moved. This is a
```

```
;      floating point value.
;
; PROCEDURE:
;      WIDGET_CONTROL, id, SET_VALUE=value can be used to change the
;            current value displayed by the widget.
;
;      WIDGET_CONTROL, id, GET_VALUE=var can be used to obtain the current
;            value displayed by the widget.
;
; MODIFICATION HISTORY:
;      April 2, 1992, SMR and AB
;            Based on the RGB code from XPALETTE.PRO, but extended to
;            support color systems other than RGB.
;      5 January 1993, Mark Rivers, Brookhaven National Labs
;            Added EDIT keyword.
;        7 April 1993, AB, Removed state caching.
;      28 July 1993, ACY, set_value: check labelid before setting text.
;      3 October 1995, AB, Added SCROLL keyword.
;-


PRO fslider_set_value, id, value

  ; Set the value of both the slider and the label
  ON_ERROR, 2                                   ;return to caller

  stash = WIDGET_INFO(id, /CHILD)
  WIDGET_CONTROL, stash, GET_UVALUE=state, /NO_COPY

  WIDGET_CONTROL, state.slideid, $
      SET_VALUE = 1000000. * $
            (float(value) - state.bot) / (float(state.top) - state.bot)
  IF (state.labelid NE 0) THEN $
      WIDGET_CONTROL, state.labelid, $
            SET_VALUE = STRING(FLOAT(value), format=state.format)

  WIDGET_CONTROL, stash, SET_UVALUE=state, /NO_COPY
END



FUNCTION fslider_get_value, id

  ; Return the value of the slider
  ON_ERROR, 2                                   ;return to caller

  stash = WIDGET_INFO(id, /CHILD)
  WIDGET_CONTROL, stash, GET_UVALUE=state, /NO_COPY

  WIDGET_CONTROL, state.slideid, GET_VALUE = tmp
  ret = ((tmp / 1000000.) * (float(state.top) - state.bot)) + state.bot

  WIDGET_CONTROL, stash, SET_UVALUE=state, /NO_COPY
  return, ret
END


;-----------------------------------------------------------------------

FUNCTION fslide_event, ev
```

```
   ; Retrieve the structure from the child that contains the sub ids
   parent=ev.handler
   stash = WIDGET_INFO(parent, /CHILD)
   WIDGET_CONTROL, stash, GET_UVALUE=state, /NO_COPY


   ; See which widget was adjusted, the slider or the label

   if (ev.id eq state.slideid) then begin
     ; Get the non-adjusted value
     WIDGET_CONTROL, state.slideid, GET_VALUE = nonadj
     ; Compute the floating point value
     value = ((nonadj / 1000000.) * (float(state.top) - state.bot)) + state.bot
     drag = ev.drag
     ; Update label
     IF (state.labelid NE 0) THEN $
       WIDGET_CONTROL, state.labelid, $
            SET_VALUE=STRING(value, format=state.format)

   endif else if (ev.id eq state.labelid) then begin

     WIDGET_CONTROL, state.labelid, GET_VALUE = tmp

     value = float(tmp[0])
     value = value > state.bot
     value = value < state.top
     ;Update the slider, set new value
     WIDGET_CONTROL, state.slideid, $
       SET_VALUE = 1000000. * $
             (value - state.bot) / (float(state.top) - state.bot)

     drag = 0
     ; Update the label so it has desired format
     WIDGET_CONTROL, state.labelid, $
           SET_VALUE=STRING(value, format=state.format)
   endif

   WIDGET_CONTROL, stash, SET_UVALUE=state, /NO_COPY
   RETURN, {WIDGET_FSLIDER, ID:parent, TOP:ev.top, HANDLER:0L, VALUE:value,
DRAG:drag }
END

;-------------------------------------------------------------------------

FUNCTION cw_f1slider, parent, $
             DRAG = drag, $
                 EDIT = edit, $
             FRAME = frame, $
             MAXIMUM = max, $
             MINIMUM = min, $
             SCROLL = scroll, $
             SUPPRESS_VALUE = sup, $
             TITLE = title, $
             UVALUE = uval, $
             VALUE = val, $
             VERTICAL = vert, $
             XSIZE = xsize, $
             YSIZE = ysize, $
             FORMAT=format

  IF (N_PARAMS() EQ 0) THEN MESSAGE, 'Incorrect number of arguments'
```

```
 ON_ERROR, 2                                          ;return to caller

 ; Defaults for keywords
 IF NOT (KEYWORD_SET(drag))  THEN drag = 0
 IF NOT (KEYWORD_SET(edit))  THEN edit = 0
 IF NOT (KEYWORD_SET(frame)) THEN frame = 0
 IF N_ELEMENTS(max) EQ 0     THEN max = 100.0
 IF N_ELEMENTS(min) EQ 0     THEN min = 0.0
 IF NOT (KEYWORD_SET(scroll)) THEN scroll = 10000 ELSE $
 scroll = ABS(LONG((float(scroll) / (max - min)) * 1000000))
 IF NOT (KEYWORD_SET(sup))   THEN sup = 0
 IF NOT (KEYWORD_SET(title)) THEN title = ""
 IF NOT (KEYWORD_SET(uval))  THEN uval = 0
 IF N_ELEMENTS(val) EQ 0     THEN val = min
 IF NOT KEYWORD_SET(format)  THEN format='(G13.6)'

 state = {slideid:0L, labelid:0L, top:max, bot:min, format:format }

 ; Motif 1.1 and newer sliders react differently to XSIZE and YSIZE
 ; keywords than Motif 1.0 or OpenLook. These defs are for horizontal sliders
 version = WIDGET_INFO(/version)
 newer_motif = (version.style eq 'Motif') and (version.release ne '1.0')

 ; The sizes of the parts depend on keywords and whether or not the
 ; float slider is vertical or horizontal
 ;these are display specific and known to be inherently evil
 sld_thk = 16
 chr_wid = 7
 IF (KEYWORD_SET(vert)) THEN BEGIN
   if (newer_motif) then begin
     if (not KEYWORD_SET(xsize)) then xsize = 0
   endif else begin
     title_len = STRLEN(title) * chr_wid
     xsize = (sld_thk * 1.4) + title_len ; Take label into account
   endelse
   IF NOT (KEYWORD_SET(ysize)) THEN ysize = 100
   l_yoff = ysize / 2
 ENDIF ELSE BEGIN                                     ;horizontal slider
   vert = 0
   tmp = not keyword_set(xsize)
   if (newer_motif) then begin
     if (tmp) then xsize = 0
     IF NOT (KEYWORD_SET(ysize)) THEN ysize = 0
   endif else begin
     if (tmp) then xsize = 100
     IF (TITLE NE '') THEN sld_thk = sld_thk + 21
     ysize = sld_thk          ; Make the slider not waste label space
   endelse
   l_yoff = 0
 ENDELSE

 if (vert) then begin
   mainbase = WIDGET_BASE(parent, FRAME = frame, /ROW)
   labelbase = WIDGET_BASE(mainbase)
 endif else begin
   mainbase = WIDGET_BASE(parent, FRAME = frame, /COLUMN)
   labelbase = mainbase
 endelse
 WIDGET_CONTROL, mainbase, SET_UVALUE = uval, EVENT_FUNC = 'fslide_event', $
     PRO_SET_VALUE='FSLIDER_SET_VALUE', $
```

```
      FUNC_GET_VALUE='FSLIDER_GET_VALUE'


  IF (sup EQ 0) THEN $
    ; Only build the label if suppress_value is FALSE
    state.labelid = WIDGET_TEXT(labelbase, YOFFSET = l_yoff, $
            VALUE = STRING(FLOAT(val), format=state.format), $
                edit=edit) $
  ELSE state.labelid = 0

    state.slideid = WIDGET_SLIDER(mainbase, $
            TITLE = TITLE, $
            XSIZE = xsize, $
            YSIZE = ysize, $
            /SUPPRESS_VALUE, $
            MINIMUM = 0, $
            MAXIMUM = 1000000, $
            VALUE = 1000000. * $
                  (float(val) - state.bot) / $
                  (float(state.top) - state.bot), $
            VERTICAL = vert, $
            DRAG=drag, $
            SCROLL=scroll)


  WIDGET_CONTROL, WIDGET_INFO(mainbase, /CHILD), SET_UVALUE=state, /NO_COPY
  RETURN, mainbase

END
```

```
; $Id: sph_scat.pro,v 1.4 1998/01/15 18:44:11 scottm Exp $
;
; Copyright (c) 1994-1998, Research Systems, Inc.  All rights reserved.
;     Unauthorized reproduction prohibited.

function normal_scat, lon, lat, f, GS = gs, BOUNDS = bounds, NLON=nlon, $
      NLAT = nlat, GOUT = gsout, BOUT = boundsout
;+
; NAME:
;     SPH_SCAT
;
; PURPOSE:
;     Interpolate to a regular grid given scattered samples on the
;     surface of a sphere.
; CATEGORY:
;     Interpolation.
; CALLING SEQUENCE:
;     Result = SPH_SCAT(lon, lat, f)
; INPUTS:
;     lon = sample longitudes, a vector, in degrees.  lon, lat, and
;           f must have the same number of points.
;     lat = sample latitudes, a vector, in degreees.
;     f = data values measured at lon and lat.  f(i) = sample value
;         at lon(i), lat(i).
; KEYWORD PARAMETERS:
;     GS:      If present, GS must be a two-element vector [XS, YS],
;              where XS is the spacing between grid points in longitude,
;              and YS is the spacing in latitude. The default is based on
;              the extents of lon and lat. If the grid starts at longitude
;              Lonmin and ends at Lonmax, then the default horizontal
;              spacing is (Lonmax - Lonmin)/(NX-1). YS is computed in the
;              same way. The default grid size, if neither NX or NY
;              are specified, is 26 by 26.
;     BOUNDS:   If present, BOUNDS must be a four element array containing
;              the grid limits in longitude and latitude of the output grid:
;              [Lonmin, Latmin, Lonmax, Latmax]. If not specified, the grid
;              limits are set to the extent of lon and lat.   Warning:
;              to cover all longitudes, you must directly specify BOUNDS.
;     NX:       The output grid size in the longitude direction. NX need not
;              be specified if the size can be inferred from GS and
;              BOUNDS. The default value is 26.
;     NY:       The output grid size in the latitude direction. See NX.
;     BOUT:    the actual extent of the regular grid, arranged as in
;              bounds.  An optional output parameter.
;     GOUT:     The actual grid spacing, a two element optional output array.
;
; OUTPUTS:
;     Result = regularly interpolated result.
; COMMON BLOCKS:
;     None.
; SIDE EFFECTS:
;     None.
; RESTRICTIONS:
;     Timing. on a Sun SPARCstation LX producing a 36 x 36 output
;     grid (1296 points), t is ~ .578 + .00368 * N + 2.39e-06 * N^2.
;     For example:
;     N     16    64    256   1024  4096
;     Time  .7    .8    1.6   6.6   56
;     Output points are produced at a rate of approximately 2000
;     points per second.
;
```

```
; PROCEDURE:
;     This routine is a convenience interface to the Spherical gridding
;     and interpolation provided by TRIANGULATE and TRIGRID.  The
;     methods are based on the work of Robert Renka, Interpolation of Data
;     on the Surface of a Sphere, Oak Ridge Natl Lab Technical Paper
;     CSD-108.  The procedure consists of generating a triangulation of the
;     scattered data points, estimating the gradients with a local method,
;     and then constructing a triangle based interpolant of the data and
;     gradient estimates.  The interpolant is C(1) continuous.
; EXAMPLE:
;     Create 50 random longitudes and latitudes, make a function value,
;     and then interpolate, obtaining a 360 x 360 array of
;     10 degree by 5 degree resolution that covers the sphere:
;
;     lon = randomu(seed, 50) * 360. -180.  ;Make random scattered points
;     lat = randomu(seed, 50) * 180. -90.
;     z = sin(lat*!DTOR)            ;Make a function to fit
;     c = cos(lat*!DTOR)
;     x = cos(lon*!DTOR) * c
;     y = sin(lon*!DTOR) * c
;     f =  sin(x+y) * sin(x*z)      ;The dependent variable
;  ** Now, given lon, lat, and f, interpolate the data:
;     result = sph_scat(lon, lat, f, bounds=[0, -90, 350, 85], gs=[10,5])
;
; MODIFICATION HISTORY:
;     DMS, November, 1994.  Written.
;-

n = n_elements(lon)
if n ne n_elements(lat) or n ne n_elements(f) then $
      message, 'lon, lat, and f must have the same number of elements'
if n le 3 then $
      message, 'Must have at least 3 points'
;           Construct bounds if necessary
if n_elements(bounds) ne 4 then $
    boundsout = [ min(lon, max=lonmax), min(lat, max=latmax), lonmax, latmax]
$
else boundsout = bounds
;           Get gs, nx, and ny.
if n_elements(gs) ne 2 then begin
    if n_elements(nx) le 0 then nx = 10
    if n_elements(ny) le 0 then ny = 10
    gsout = [boundsout[2]-boundsout[0], boundsout[3]-boundsout[1]] / $
            float([nx-1, ny-1])
endif else gsout = gs
fcopy = f         ;will be rearranged.
TRIANGULATE, 1.0*lon, 1.0*lat, tr;, FVALUE=fcopy
return, TRIGRID(1.0*lon, 1.0*lat, fcopy, tr, gsout, boundsout)
end
```