ARMY RESEARCH LABORATORY

BLIRB Visualization and BUFR Encoder/Decoder Software User's Guide

by Elton P. Avara
Battlefield Environment Directorate

ARL-TR-273-8

July 1995

DTIC
ELECTE
AUG 25 1995
S B D

19950824 077

DTIC QUALITY INSPECTED 5

Approved for public release; distribution is unlimited.

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

The citation of trade names and names of manufacturers in this report is not to be construed as official Government indorsement or approval of commercial products or services referenced herein.

Destruction Notice

When this document is no longer needed, destroy it by any method that will prevent disclosure of its contents or reconstruction of the document.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
July 1995

3. REPORT TYPE AND DATES COVERED
Final Jul 94 through Apr 95

4. TITLE AND SUBTITLE
BLIRB Visualization and BUFR Encoder/Decoder Software User's Guide

5. FUNDING NUMBERS
Contract
DAAL03-91-C-0034

6. AUTHOR(S)
Elton P. Avara

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U. S. Army Research Laboratory
Battlefield Environment Directorate
Attn: AMSRL-BE-M
White Sands Missile Range, NM 88002-5501

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-273-8

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U. S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
TCN 94291

11. SUPPLEMENTARY NOTES
Task was performed under a Scientific Services Agreement issued by Battelle, Research Triangle Park Office, 200 Park Drive, P.O. Box 12297, Research Triangle Park, NC 27709

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT *(Maximum 200 words)*

The eight-stream extension of the Boundary Layer Illumination Radiation Balance Model (BLIRB8) provides direct and diffuse radiative fluxes at each grid point in a physical BLIRB8 space. This physical space may contain several regions of aerosol concentration (clouds, smoke, fog, etc.), ground level areas of varying albedos, flares, and a searchlight. BLIRB8 inputs are in a file comprised of several records containing the necessary input parameters. These records are complex and require each parameter field to contain a valid input. VISUAL was developed to provide a user-friendly graphical user interface for building the input file and graphically displaying the BLIRB8 output flux fields in an interactive mode. All BLIRB8 inputs (except a savefile name) can be selected using the mouse. The keyboard may be used for hotkeys. The BLIRB8 radiant flux output file may be encoded using the World Meteorological Organization code form FM 94-IX Ext. BUFR (Binary Universal Form for the Representation of meteorological data). BUFR is a binary encoding scheme designed to represent any meteorological data. BLIRB_EN encodes BLIRB8 output to a BUFR binary file. BLIRB_DE recovers the BLIRB8 output ASCII file from the BUFR file.

14. SUBJECT TERMS
BLIRB, BLIRB8, visualization, BUFR, encoding, decoding, GUI

15. NUMBER OF PAGES
560

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

Acknowledgments

This work was supported by the Battlefield Environment Directorate (BED) (Dr. Richard Shirkey) under the auspices of the U.S. Army Research Office Scientific Services Program administered by Battelle (Delivery Order 1355, Contract Number DAAL03-91-C-0034). I am especially grateful for the detailed software requirements clarification, software testing, and software design suggestions offered by Dr. Richard Shirkey, Dr. Alan Wetmore, Mr. David Tofsted, Dr. Donald Hoock, and Mr. John Linder of the U.S. Army Research Laboratory, BED, White Sands Missile Range, NM.

Accession For
NTIS GRA&I ☑
DTIC TAB ☐
Unannounced ☐
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

Contents

Acknowledgments 1

1. Introduction 7
1.1 BLIRB8 Visualization 7
1.2 BLIRB8 Output Encoding/Decoding 8

2. Starting VISUAL 11

3. VISUAL Menubar 13
3.1 File 13
3.1.1 Create New File 13
3.1.2 Open File 14
3.1.3 Save File 14
3.1.4 Save File As 14
3.1.5 Exit Program 15
3.2 View 15
3.2.1 Viewing Axis Options 15
3.2.2 Sun Options 16
3.2.3 Zoom Options 17
3.2.4 Minor Grid Lines On/Off 17
3.2.5 Transparent Colors On/Off 18
3.2.6 Region Definitions On/Off 18
3.3 Flux 18
3.3.1 Flux Options 19
3.3.2 Cross-Section Plane Orientation 21
3.3.3 Cross-Section Plane Value Selection 21
3.3.4 Wave Number Selection 22
3.4 Modify 23
3.4.1 Model Changes 23
3.4.2 Region Selection 37
3.4.3 Albedo Area Selection 48
3.4.4 Grid Mesh Selection 55
3.4.5 Cloud Changes 59

3.4.6 Sun Changes 61
3.4.7 Flare Selection 63
3.4.8 SearchLight Selection 65
3.4.9 Spectral Range Changes 68
3.4.10 Computation Changes 72
3.4.11 Output File Changes 74
3.5 Reset 74
3.6 Help 75
3.6.1 Viewing Options 75
3.6.2 Flux Display Options 76
3.6.3 Input Modifications Options 77

4. BLIRB8 Space Rotation 79

5. BLIRB8 Space Translation 81

6. BLIRB8 Default Input Values 83

7. VISUAL Global Software Variables 85

8. BLIRB_EN 87
8.1 Section 1 87
8.2 Section 3 88
8.3 Section 4 88

9. BLIRB_DE 91

10. BLIRB_CM 93

References 95

Acronyms and Abbreviations 97

Appendices
Appendix A. Listing of VISUALO.H, the Include File for VISUAL.C 99
Appendix B. Listing of BUFR.H 105

Appendix C. Listing of BLIRB_EN.C 109
Appendix D. Listing of BLIRB_DE.C 141
Appendix E. Listing of BLIRB_CM.C 167
Appendix F. Listing of Table B 183
Appendix G. Listing of Table D 193
Appendix H. Listing of VISUAL.C 213

Distribution 547

1. Introduction

The eight-stream extension of Boundary Layer Illumination Radiation Balance Model (BLIRB8) was developed by Andrew Zardecki under contract to the U.S. Army Research Office, P.O. Box 12211, Research Triangle Park, NC 27709. [1,2] BLIRB8 provides direct and diffuse radiative fluxes at each grid point in the physical BLIRB8 space. The physical BLIRB8 space may contain several regions of aerosol concentration (clouds, smoke, fog, etc.), ground level areas of varying albedos, flares, and a searchlight. An ASCII file, comprised of several records (cards) containing the necessary input parameter values, contains inputs to BLIRB8. The ASCII file GRID.ASC and the binary file GRID.BIN contain the BLIRB8 outputs. Both files contain the same information.

1.1 BLIRB8 Visualization

Because of the complexity of the various input records and the requirement for each parameter field of each record to contain a valid input value, VISUAL was developed to provide a user-friendly graphical user interface (GUI) for building a BLIRB8 input file. In addition, VISUAL graphically displays the BLIRB8 output radiative flux fields in an interactive mode.

VISUAL graphically displays the entire physical BLIRB8 space including the following:

- surface albedo areas (shades of green)
- regions of aerosol concentrations (various transparent colors)
- flare positions (red)
- searchlight position (white)
- relative position of the Sun (yellow)
- BLIRB8 output radiative flux fields (red and white)

Optionally, text widgets may be displayed depicting the following:

- the current BLIRB8 input/output filename
- the surface albedo areas information
- the aerosols regions information
- the output radiative flux field information

You can select all BLIRB8 input parameters (except a filename for saving the inputs) using the mouse. You can use the keyboard for hot keys, and you must use it to specify a savefile name.

VISUAL uses the Silicon Graphics Inc. (SGI) IRIS Graphics Library for the display graphics and X-Windows/Motif for the menus and message boxes of the GUI. This software requires an SGI workstation with at least 128 Mb of RAM for program execution. Appendix H contains a listing of the VISUAL source code, VISUAL.C. VISUAL.C is also contained under the directory VISUAL on an accompanying MS-DOS-compatible 3.5-in. floppy disk along with the SGI makefile and the include file, VISUAL0.H. Appendix A contains a listing of VISUAL0.H.

1.2 BLIRB8 Output Encoding/Decoding

The BLIRB8 radiant flux output will be required in the near future to be sent over the Internet to the participants in the Distributed Interactive Simulation groups. This will require that the output ASCII file be encoded using the World Meteorological Organization code form FM 94-IX Ext. BUFR (Binary Universal Form for the Representation of meteorological data). [3] BUFR is a binary encoding scheme, that employs a continuous binary stream, designed to represent any meteorological data. The same scheme can be used to encode any numerical or qualitative data.

BLIRB_EN was created for the purpose of encoding the BLIRB8 radiant flux output ASCII file to a BUFR file. BLIRB_DE was created for the purpose of recovering the BLIRB8 radiant flux output ASCII file from the BUFR file. In addition, BLIRB_CM was created to compare the contents of the original output file and the reconstituted output file item by item.

BLIRB_EN, BLIRB_DE, and BLIRB_CM were tested on SGI and Hewlett Packard workstations and MS-DOS-compatible personal computers (compiled using the Compact Memory Model in Turbo C++ versions 1.0 and 2.0). Appendix B contains a listing of the encoder/decoder include file, BUFR.H. Appendices C, D, and E contain listings of BLIRB_EN.C, BLIRB_DE.C, and BLIRB_CM.C, respectively. Appendices F and G contain listings of two BUFR tables (B and D) required for software execution. The BUFR directory on the accompanying MS-DOS-compatible 3.5-in. floppy disk contains all source and MS-DOS-compatible executable files along with the two required tables.

2. Starting VISUAL

Start VISUAL in any of three ways. Enter **visual** to start VISUAL with default values for each of the input parameters. The default BLIRB8 space is 5- by 4- by 5-km XYZ dimensioned with no aerosol. No aerosol subregions are within the BLIRB8 space. One surface albedo area is 5- by 4-km XY dimensioned with the value of the background albedo (0.2). The Sun is positioned directly overhead with azimuth and zenith angles set to zero. The default case contains no flares or searchlights. The observer is on the negative-Y axis looking toward the center of the BLIRB8 space. The grid points are every 0.5 km in the X, Y, and Z directions.

Enter **visual <inputfile>** in which <inputfile> is a BLIRB8 ASCII input file to start VISUAL with the input parameters set to those in the inputfile.

Enter **visual <outputfile>** in which <outputfile> is a BLIRB8 ASCII radiative flux output file to start VISUAL with the input parameters set to those in the outputfile.

The BLIRB8 radiative flux fields are available for display.

3. VISUAL Menubar

After an input file is entered into VISUAL, the menubar at the top of the window contains five options:

- **File**
- **View**
- **Modify**
- **Reset**
- **Help**

After an output file is entered into VISUAL, the menubar contains another option, **Flux**, that appears between **View** and **Modify**.

3.1 File

Select **File** on the menubar to display a menu with five options:

- **Create New File**
- **Open File**
- **Save File**
- **Save File As**
- **Exit Program**

(Or, enter **Alt-f** or **F10-f** to select **File**.)

3.1.1 Create New File

Select **Create New File** or press **n** in the **File** menu (or, enter **Alt-fn**, **F10-fn**, or **Shift-F1** to select **Create New File**) to display a warning that continuing this option will delete all previous modifications if modifications are made to the initial input parameter values and the changes are not saved.

Choose to **Continue** or **Cancel** the selection. The default BLIRB8 input values are loaded if you choose to continue.

The default BLIRB8 input values are loaded when you select **Create New File** and you have no unsaved modifications pending.

3.1.2 Open File

Select **Open File** or press **o** in the **File** menu (or, enter **Alt-fo**, **F10-fo**, or **Shift-F2** to select **Open File**) to display a warning that continuing this option will delete all previous modifications if modifications are made to the initial input parameter values and the changes are not saved.

Choose to **Continue** or **Cancel** the selection.

Select **Open File** to display a file selection box with the file names in the current directory filtered according to the filename filter. Scroll the list of file names. Select a file to process.

Click on **grid*.*** and type in a new filter to change the filename filter.

Click on **Cancel** to close the file selection box.

The BLIRB8 input parameters from the selected file are loaded into VISUAL.

3.1.3 Save File

Select **Save File** or press **s** in the **File** menu (or, enter **Alt-fs**, **F10-fs**, or **Shift-F3**) to save current inputs to the selected file name, <filename>, after saving the original input file to a backup file <filename.bak>.

The default file name used when you select **Create New File** is **grid_newfile.i**.

3.1.4 Save File As

Select **Save File As** or press **a** in the **File** menu (or, enter **Alt-fa**, **F10-fa**, or **Shift-F4**) to input a file name for saving the current BLIRB8 input parameters.

3.1.5 Exit Program

Select **Exit Program** or press **x** in the **File** menu (or, enter **Alt-fx**, **F10-fx**, or **F12**) to terminate the VISUAL program.

A warning is displayed to allow you to **Cancel** the selection or **Continue** and terminate the program, resulting in the loss of all modifications if the BLIRB8 input parameters are modified and not saved when you select **Exit Program**.

You can abort the VISUAL program by pressing **Esc** or by closing the window. All modifications are lost without warning if you abort.

3.2 View

Select **View** on the menubar to display a menu with six options:

- **Viewing Axis Options**
- **Sun Options**
- **Zoom Options**
- **Minor Grid Lines On/Off**
- **Transparent Colors On/Off**
- **Region Definitions On/Off**

(Or, enter **Alt-v** or **F10-v**.)

3.2.1 Viewing Axis Options

Select **Viewing Axis Options** or press **a** in the **View** menu (or, enter **Alt-va** or **F10-va**) to display a menu with three options:

- **Positive X-axis**
- **Negative Y-axis**
- **Positive Z-axis**

3.2.1.1 *Positive X-Axis.*—Select **Positive X-axis** or press **x** in the **Viewing Axis Options** menu (or, enter **Alt-vax**, **F10-vax**, or **F1**) to cause the observer position to be in a YZ plane on the positive X axis. The viewed point is the center of the BLIRB8 space. Change the observer position on the observer plane by moving the mouse while holding down the left mouse button, which creates a rotation effect on the BLIRB8 space.

3.2.1.2 *Negative Y-Axis.*—Select **Negative Y-axis** or press **y** in the **Viewing Axis Options** menu (or, enter **Alt-vay**, **F10-vay**, or **F2**) to cause the observer position to be in an XZ plane on the negative Y axis. The viewed point is the center of the BLIRB8 space. Change the observer position on the observer plane by moving the mouse while holding down the left mouse button, which creates a rotation effect on the BLIRB8 space. **Negative Y-axis** is the initial viewing axis choice when starting VISUAL.

3.2.1.3 *Positive Z-Axis.*—Select **Positive Z-axis** or press **z** in the **Viewing Axis Options** menu (or, enter **Alt-vaz**, **F10-vaz**, or **F3**) to cause the observer position to be in an XY plane on the positive Z axis. The viewed point is the center of the BLIRB8 space. Change the observer position on the observer plane by moving the mouse while holding down the left mouse button, which creates a rotation effect on the BLIRB8 space.

3.2.2 Sun Options

Select **Sun Options** or press **s** in the **View** menu (or, enter **Alt-vs** or **F10-vs**), to display a menu with two options:

- **Sun Plot On/Off**
- **Select New Sun Position**

3.2.2.1 *Sun Plot On/Off.*—Select **Sun Plot On/Off** or press **p** in the **Sun Options** menu (or, enter **Alt-vsp**, **F10-vsp**, or **F4**) to access a toggle switch that displays the relative Sun position if it was not displayed before making the selection and hides the relative Sun position if it was displayed before making the selection.

Sun Plot On is the default.

3.2.2.2 *Select New Sun Position.*—Select **Select New Sun Position** or press **n** in the **Sun Options** menu (or, enter **Alt-vsn**, **F10-vsn**, or **F5**) to cause the viewing axis to temporarily become the positive Z axis. Change the Sun position by moving the mouse while holding down the left mouse button. Move the mouse until the Sun is positioned at the desired location, then release the mouse button. When you release the mouse button, the display returns to the original viewing position. Only the position of the Sun is changed.

3.2.3 Zoom Options

Select **Zoom Options** or press **z** in the **View** menu (or, enter **Alt-vz** or **F10-vz**) to display the **Zoom Options** menu with two options:

- **Zoom In**
- **Zoom Out**

3.2.3.1 *Zoom In.*—Select **Zoom In** or press **i** in the **Zoom Options** menu (or, enter **Alt-vzi**, **F10-vzi**, or **F6**) to display the BLIRB8 space 5 percent larger than previously displayed.

3.2.3.2 *Zoom Out.*—Select **Zoom Out** or press **o** in the **Zoom Options** menu (or, enter **Alt-vzo**, **F10-vzo**, or **F7**) to display the BLIRB8 space 5 percent smaller than previously displayed.

3.2.4 Minor Grid Lines On/Off

Select **Minor Grid Lines On/Off** or press **m** in the **View** menu (or, enter **Alt-vm**, **F10-vm**, or **F8**) to access a toggle switch that displays the minor grid lines if they were not displayed and hides the minor grid lines if they were displayed.

Minor Grid Lines Off is the default.

3.2.5 Transparent Colors On/Off

Select **Transparent Colors On/Off** or press **t** in the **View** menu (or, enter **Alt-vt**, **F10-vt**, or **F9**) to access a toggle switch that displays the aerosol regions in transparent colors if they were displayed as outlines and displays the aerosol regions as outlines if they were displayed in transparent colors.

Transparent Colors On is the default.

3.2.6 Region Definitions On/Off

Select **Region Definitions On/Off** or press **d** in the **View** menu (or, enter **Alt-vd**, **F10-vd**, or **F11**) to access a toggle switch that displays the text boxes at the bottom of the BLIRB8 window containing the current filename, aerosol regions information, albedo areas information, and radiative flux information if they were not displayed and hides the text boxes if they were displayed.

Region Definitions On is the default.

3.3 Flux

Select **Flux** on the menubar to display a menu with four options:

- **Flux Options**
- **Cross-Section Plane Orientation**
- **Cross-Section Plane Value Selection**
- **Wave Number Selection**

(Or, enter **Alt-l** or **F10-l**.)

3.3.1 *Flux Options*

Select **Flux Options** or press **f** in the **Flux** menu (or, enter **Alt-lf** or **F10-lf**) to display a menu with 13 options:

- **Solar Direct Flux**
- **Solar Reflected Flux**
- **Diffuse Flux - 1**
- **Diffuse Flux - 2**
- **Diffuse Flux - 3**
- **Diffuse Flux - 4**
- **Diffuse Flux - 5**
- **Diffuse Flux - 6**
- **Diffuse Flux - 7**
- **Diffuse Flux - 8**
- **No Flux**
- **Dec by 1 Button**
- **Inc by 1 Button**

3.3.1.1 *Flux Field Choices.*—You can investigate a flux field by displaying the flux magnitude using flux field cross-sections of various orientations and positions. Select any of the first 10 choices or press the corresponding keys (as shown below) to display the corresponding flux within the BLIRB8 space.

Flux Field	Mnemonic Key	Hotkey Combination
Solar Direct Flux	**d**	**Alt-lfd or F10-lfd**
Solar Reflected Flux	**r**	**Alt-lfr or F10-lfr**
Solar Diffuse Flux - 1	**1**	**Alt-lf1 or F10-lf1**
Solar Diffuse Flux - 2	**2**	**Alt-lf2 or F10-lf2**
Solar Diffuse Flux - 3	**3**	**Alt-lf3 or F10-lf3**
Solar Diffuse Flux - 4	**4**	**Alt-lf4 or F10-lf4**
Solar Diffuse Flux - 5	**5**	**Alt-lf5 or F10-lf5**
Solar Diffuse Flux - 6	**6**	**Alt-lf6 or F10-lf6**
Solar Diffuse Flux - 7	**7**	**Alt-lf7 or F10-lf7**
Solar Diffuse Flux - 8	**8**	**Alt-lf8 or F10-lf8**

The flux display is a red three-dimensional (3-D) grid wiremesh plot depicting the magnitude of the appropriate flux corresponding to the choices of flux cross-section plane, distance of the cross-section plane from the BLIRB8 space origin, and radiation wavenumber. The current flux cross-section plane is depicted by a white grid mesh. The distance between the red and white grid meshes is related to the magnitude of the selected flux. The flux amplitude scale can be linear or logarithmic.

3.3.1.2 *No Flux.*—Select **No Flux** or press **n** in the **Flux Options** menu (or, enter **Alt-lfn** or **F10-lfn**) to turn off the flux display and text information box.

No Flux is the default when VISUAL is started without a BLIRB8 output radiation flux file as input.

3.3.1.3 *Dec by 1 Button.*—Select **Dec by 1 Button** in the **Flux Options** menu (or, enter **Ctrl-F1**) to decrease the choice of flux fields one position in the ordered sequence: **Solar Direct Flux**, **Solar Reflected Flux**, **Diffuse Flux - 1**, **Diffuse Flux - 2**, **Diffuse Flux - 3**, **Diffuse Flux - 4**, **Diffuse Flux - 5**, **Diffuse Flux - 6**, **Diffuse Flux - 7**, **Diffuse Flux - 8**.

Dec by 1 Button has a wraparound effect such that making this selection while viewing **Solar Direct Flux** causes the new flux field choice to be **Diffuse Flux - 8**.

3.3.1.4 *Inc by 1 Button.*—Select **Inc by 1 Button** in the **Flux Options** menu (or, enter **Ctrl-F2**) to increase the choice of flux fields by one position in the ordered sequence: **Solar Direct Flux**, **Solar Reflected Flux**, **Diffuse Flux - 1**, **Diffuse Flux - 2**, **Diffuse Flux - 3**, **Diffuse Flux - 4**, **Diffuse Flux - 5**, **Diffuse Flux - 6**, **Diffuse Flux - 7**, **Diffuse Flux - 8**.

Inc by 1 Button has a wraparound effect such that making this selection while viewing **Diffuse Flux - 8** causes the new flux field choice to be **Solar Direct Flux**.

3.3.2 *Cross-Section Plane Orientation*

Select **Cross-Section Plane Orientation** or press **o** in the **Flux** menu (or, enter **Alt-lo** or **F10-lo**) to display a menu with three options:

- **X-Plane Cross-Section**
- **Y-Plane Cross-Section**
- **Z-Plane Cross-Section**

3.3.2.1 *X-Plane Cross-Section.*—Select **X-Plane Cross-Section** or press **x** in the **Cross-Section Plane Orientation** menu (or, enter **Alt-lox**, **F10-lox**, or **Ctrl-F3**) to cause the radiative flux field cross-section plane to be a YZ plane on the positive X axis. The position of the YZ plane on the X axis is dependent upon the choice of **Cross-Section Plane Value**.

3.3.2.2 *Y-Plane Cross-Section.*—Select **Y-Plane Cross-Section** or press **y** in the **Cross-Section Plane Orientation** menu (or, enter **Alt-loy**, **F10-loy**, or **Ctrl-F4**) to cause the radiative flux field cross-section plane to be an XZ plane on the positive Y axis. The position of the XZ plane on the Y axis is dependent upon the choice of **Cross-Section Plane Value**.

3.3.2.3 *Z-Plane Cross-Section.*—Select **Z-Plane Cross-Section** or press **z** in the **Cross-Section Plane Orientation** menu (or, enter **Alt-loz**, **F10-loz**, or **Ctrl-F5**) to cause the radiative flux field cross-section plane to be an XY plane on the positive Z axis. The position of the XY plane on the Z axis is dependent upon the choice of **Cross-Section Plane Value**.

Z-Plane Cross-Section is the default.

3.3.3 *Cross-Section Plane Value Selection*

Select **Cross-Section Plane Value Selection** or press **v** in the **Flux** menu (or, enter **Alt-lv** or **F10-lv**) to display a menu with an arbitrary number of options. All options, except the last two, are xx.xxx km in which the xx.xxx is the value of one of the minor grid line values corresponding to the flux cross-section plane orientation choice. The last two options are **Dec by 1 Button** and **Inc by**

1 Button. For example, minor grid lines every 0.5 km from 0 to 5 km would give 13 choices for this option: 0.000, 0.500, 1.000, 1.500, 2.000, 2.500, 3.000, 3.500, 4.000, 4.500, and 5.000 km, Dec by 1 Button, and Inc by 1 Button.

The **Cross-Section Plane Value Selection** 0.0 km is the default for all three cross-section plane orientations.

Select **Dec by 1 Button** in the **Cross-Section Plane Value Selection** menu (or, enter **Ctrl-F6**) to decrease the flux cross-section value to the next lower minor grid line value. There is a wraparound effect such that, if the cross-section value is at its minimum value before making this selection, it will be at its maximum value after making this selection.

Select **Inc by 1 Button** in the **Cross-Section Plane Value Selection** menu (or, enter **Ctrl-F7**) to increase the flux cross-section value to the next higher minor grid line value. There is a wraparound effect such that, if the cross-section value is at its maximum value before making this selection, it will be at its minimum value after making this selection.

3.3.4 Wave Number Selection

Select **Wave Number Selection** or press **w** in the **Flux** menu (or, enter **Alt-lw** or **F10-lw**) to display a menu with an arbitrary number of options. All options, except the last two, are xxxx.xxxx per cm in which xxxx.xxxx is the value of one of the input wavenumbers. The last two options are **Dec by 1 Button** and **Inc by 1 Button**. For example, BLIRB8 spectral inputs of four subintervals between 450 and 1250 per cm would give seven choices: 450.0000, 650.0000, 850.0000, 1050.0000, and 1250.0000 per cm, Dec by 1 Button, and Inc by 1 Button.

The **Wave Number Selection** minimum value available is the default.

Select **Dec by 1 Button** in the **Wave Number Selection** menu (or, enter **Ctrl-F8**) to decrease the wavenumber value to the next lower available value. There is a wraparound effect such that, if the wavenumber is at its minimum

value before making this selection, it will be at its maximum value after making this selection.

Select **Inc by 1 Button** in the **Wave Number Selection** menu (or, enter **Ctrl-F9**) to increase the wavenumber value to the next higher available value. There is a wraparound effect such that, if the wavenumber is at its maximum value before making this selection, it will be at its minimum value after making this selection.

3.4 Modify

Select **Modify** on the menubar display a menu with 11 options:

- **Model Changes**
- **Region Selection**
- **Albedo Area Selection**
- **Grid Mesh Selection**
- **Cloud Changes**
- **Sun Changes**
- **Flare Selection**
- **SearchLight Selection**
- **Spectral Range Changes**
- **Computation Changes**
- **Output File Changes**

(Or, enter **Alt-m** or **F10-mto**.)

3.4.1 Model Changes

Select **Model Changes** or press **m** in the **Modify** menu (or enter **Alt-mm**, **F10-mm**, or **Shift-F5**) to display a menu with two pushbutton options, five toggle button menus, and an adjustable scale.

The pushbuttons are

- **Finished with Selections**
- **Aerosol Selections**

The toggle button menus are

- **Temperature Profile Model**
- **Meteorological Range**
- **Tropospheric Profile**
- **Albedo**
- **Aerosol Profile Printout**

The adjustable scale is **Surface Temperature (K)**.

Select **Finished with Selections** to terminate the **Model Changes** menu.

The depressed toggle button in each menu and the value depicted by the position of the scale cursor reflect the BLIRB8 model input values. Click on a different toggle button or drag the scale cursor to a new position to change the input values.

3.4.1.1 *Aerosol Selections.*—Select **Aerosol Selections** in the **Model Changes** menu to display a menu with one pushbutton and a toggle button menu with 12 toggle buttons.

The pushbutton is **Finished with Selections**.

The toggle button menu is **AFGL Aerosol Models**.

The toggle buttons are

- **Default aerosol**
- **No aerosols**
- **Rural aerosol**
- **Urban aerosol**

- **Maritime aerosol**
- **Tropospheric aerosol**
- **Fog**
- **Soot-like aerosols**
- **Oceanic component of maritime**
- **Background stratospheric**
- **Volcanic**
- **Meteoric dust**

The depressed toggle button indicates the **Aerosol Selection** choice. Click on a different toggle button to change the input value.

Default aerosol, **No aerosols**, **Soot-like aerosols**, **Oceanic component of maritime**, **Background stratospheric**, and **Meteoric dust** result in no further subchoices.

Select **Rural aerosol**, **Urban aerosol**, **Maritime aerosol**, **Tropospheric aerosol**, **Fog**, and **Volcanic** to display menus with suboptions. Select **Finished with Selections** to terminate the **Aerosol Selections** menu.

No Aerosols is the default.

1. AFGL Rural/Urban/Tropospheric Aerosol Models. Select **AFGL Rural/Urban/Tropospheric Aerosol Models** in the **AFGL Aerosol Models** menu to display a menu with one pushbutton and a toggle button menu with three options (dependent upon the previous selection) and nine toggle buttons.

The pushbutton is **Finished with Selections**.

The menu options are

- **AFGL Rural Aerosol Models**
- **AFGL Urban Aerosol Models**
- **AFGL Tropospheric Aerosol Models**

The toggle buttons are

- **General**
- **0 percent RH**
- **50 percent RH**
- **70 percent RH**
- **80 percent RH**
- **90 percent RH**
- **95 percent RH**
- **98 percent RH**
- **99 percent RH**

Click on a toggle button to change the input value.

Select **Finished with Selections** to terminate the **AFGL Rural Aerosol Models**, **AFGL Urban Aerosol Models**, or **AFGL Tropospheric Aerosol Models** menus.

2. AFGL Maritime Aerosol Models. Select **AFGL Maritime Aerosol Models** in the **AFGL Aerosol Models** menu to display a menu with one pushbutton and a toggle button menu with nine toggle buttons.

The pushbutton is **Finished with Selections**.

The toggle button menu is **AFGL Maritime Aerosol Models**.

The nine toggle buttons are

- **75 percent oceanic**
- **0 percent RH**
- **50 percent RH**
- **70 percent RH**
- **80 percent RH**
- **90 percent RH**
- **95 percent RH**

- **98 percent RH**
- **99 percent RH**

Click on a toggle button to change the input value.

Select **Finished with Selections** to terminate the **AFGL Maritime Aerosol Models** menu.

3. AFGL Fog Models. Select **AFGL Fog Models** in the **AFGL Aerosol Models** menu to display a menu with one pushbutton and a toggle button menu with four toggle buttons.

The pushbutton is **Finished with Selections**.

The toggle button menu is **AFGL Fog Models**.

The four toggle buttons are

- **RRA Fog 1: Advection fog**
- **RRA Fog 2: Advection fog**
- **RRA Fog 3: Radiation fog**
- **RRA Fog 4: Radiation fog**

Click on a toggle button to change the input value.

Select **Finished with Selections** to terminate the **AFGL Fog Models** menu.

4. AFGL Volcanic Dust Models. Select **AFGL Volcanic Dust Models** in the **AFGL Aerosol Models** menu to display a menu with one pushbutton and a toggle button menu with two toggle buttons.

The pushbutton is **Finished with Selections**.

The toggle button menu is **AFGL Volcanic Dust Models**.

The two toggle buttons are

- **Aged**
- **Fresh**

Click on a toggle button to change the input value.

Select **Finished with Selections** to terminate the **AFGL Volcanic Dust Models**.

3.4.1.2 *Temperature Profile Model.*—Select **Temperature Profile Model** in the **Model Changes** menu to display a toggle button menu. The toggle button menu contains seven choices:

- **Tropical Atmosphere**
- **Midlatitude Summer**
- **Midlatitude Winter**
- **Subarctic Summer**
- **Subarctic Winter**
- **1976 U.S. Standard**
- **User Defined Temperature Profile**

The depressed toggle button indicates the **Temperature Profile Model** choice. Click on a different toggle button to change the input value. Select one of the first six choices to use the corresponding temperature profile from the Standard Atmosphere as the BLIRB8 temperature profile input.

Select the last option **User Defined Temperature Profile** to display a menu containing one pushbutton and six adjustable scales.

The pushbutton is **Finished with Selections**.

The adjustable scales are

- **Temperature (K) at 5 km**
- **Temperature at 4 km**

- **Temperature at 3 km**
- **Temperature at 2 km**
- **Temperature at 1 km**
- **Temperature at 0 km**

Drag the scale cursor to a new position to select a new temperature value for each of the six altitudes. After inputting all temperatures, select **Finished with Selections** to terminate the menu.

1976 U.S. Standard is the **Temperature Profile Model** default.

3.4.1.3 *Meteorological Range.*—Select **Meteorological Range** in the **Model Changes** menu to display a toggle button menu. The toggle button menu contains two choices:

- **Met Range < 5 km**
- **Met Range < 50 km**

The depressed toggle button indicates the **Meteorological Range** choice. Click on the other toggle button to change the input value. If the meteorological range is less than 5 km, select the first option to get the best range resolution. If the meteorological range is greater than 5 km, select the second option.

Either option displays a menu containing one pushbutton and an adjustable scale.

The pushbutton is **Finished with Selections**.

The adjustable scale is **Meteorological Range (km)**.

The **Met Range < 5 km** meteorological range scale ranges from 0 to 5 km in increments of 10 m. The **Met Range < 50 km** meteorological range scale ranges from 0 to 50 km in increments of 100 m. Drag the scale cursor to a new position to select a new meteorological range. After the meteorological range is input, select **Finished with Selections** to terminate the menu.

The Meteorological Range 40 km is the default.

3.4.1.4 *Tropospheric Profile.*—Select **Tropospheric Profile** in the **Model Changes** menu to display a toggle button menu. The toggle button menu contains three options:

- **Set by Meteorological Range**
- **Spring-Summer**
- **Fall-Winter**

The depressed toggle button indicates the **Tropospheric Profile** choice. Click on a different toggle button to change the input value. This option defines the aerosol profile above 2 km.

Set by Meteorological Range is the **Tropospheric Profile** default.

3.4.1.5 *Albedo.*—Select **Albedo** in the **Model Changes** menu to display a toggle button menu. The toggle button menu contains three options:

- **Wave Independent, User-defined**
- **Wave Independent, Tabulated**
- **Spectral**

The depressed toggle button indicates the **Albedo** choice. Click on a different toggle button to change the input value. Changing the **Albedo** option displays a menu with one pushbutton and a toggle button menu with several toggle buttons.

The pushbutton is **Finished with Selections**.

The toggle button menu is **Select Albedos for each Area**.

There are as many toggle buttons as there are surface albedo areas defined. Each toggle button is labeled **Area - #** in which # is the ordinal number of the area. You should systematically select each toggle button to input the albedo values for the surface albedo areas.

After you input the albedo for each area, select **Finished with Selections** to terminate the menu.

1. Wave Independent, User-Defined. Select **Wave Independent, User-defined** in the **Albedo** menu when another option value is selected and select an area toggle button to display a menu with one pushbutton and an adjustable scale.

The pushbutton is **Finished with Selections**.

The adjustable scale is **Albedo Value for Area - #** in which # is the ordinal number of the area.

The menu appears each time you select a surface albedo area.

Drag the scale cursor to a new position to select a new albedo value between 0 and 1 for area number #.

After you input the albedo, select **Finished with Selections** to terminate the menu.

Wave Independent, User-defined is the **Albedo** default option and background albedo (0.2) is the default albedo value for each surface albedo area.

2. Wave Independent, Tabulated. Select **Wave Independent, Tabulated** in the **Albedo** menu when another option value is selected and select an area toggle button to display a menu with one pushbutton and a menu with 31 toggle buttons.

The pushbutton is **Finished with Selections**.

The menu is **Broad-Band Albedo Surfaces for Area - #**, in which # is the ordinal number of the area. The menu appears each time you select a surface albedo area.

The 31 toggle buttons are

- **Default**
- **Dark soil**
- **Light soil**
- **Dark-ploughed**
- **Light-ploughed**
- **Clay**
- **Sandy soil**
- **Sand**
- **White sand**
- **Asphalt**
- **Lava**
- **Tundra**
- **Steppe**
- **Concrete**
- **Stone**
- **Desert**
- **Rock**
- **Dirt road**
- **Clay road**
- **Grass**
- **Mowed grass**
- **Deciduous**
- **Coniferous**
- **Rice**
- **Beet, wheat**
- **Potato**
- **Rye**
- **Cotton**
- **Lettuce**
- **Snow**
- **Ice**

Click on a toggle button to select the input value if the albedo choice is **Default** (background albedo = 0.2); **White sand**; **Asphalt**; **Lava**; **Tundra**; **Steppe**; **Concrete**; **Stone**; **Desert**; **Rice**; **Beet, wheat**; **Potato**; **Rye**; **Cotton**; or **Lettuce**.

After choosing the albedo, select **Finished with Selections** to terminate the menu.

A. Soils and Roads. Choose the albedo surface **Dark soil**, **Light soil**, **Dark-ploughed**, **Light-ploughed**, **Clay**, **Sandy soil**, **Sand**, **Rock**, **Dirt road**, or **Clay road** in the **Wave Independent, Tabulated** menu to display a menu with one pushbutton and a toggle button menu.

The pushbutton is **Finished with Selections**.

The toggle button menu contains the albedo choice and two toggle buttons:

- **dry**
- **wet**

Click on the corresponding toggle button to choose **dry** or **wet**.

After choosing the albedo value, select **Finished with Selections** to terminate the dry/wet menu.

B. Grass and Trees. Choose the albedo surface **Grass**, **Mowed grass**, **Deciduous**, or **Coniferous** in the **Wave Independent, Tabulated** menu to display a menu with one pushbutton and a toggle button menu.

The pushbutton is **Finished with Selections**.

The toggle button menu contains the albedo choice and three toggle buttons:

- **growing**
- **dormant**
- **unspecified**

Click on the corresponding toggle button to choose **growing**, **dormant**, or **unspecified**.

After choosing the albedo value, select **Finished with Selections** to terminate the growing/dormant/unspecified menu.

C. Snow. Choose the albedo surface **Snow** in the **Wave Independent, Tabulated** menu to display a menu with one pushbutton and a toggle button menu.

The pushbutton is **Finished with Selections**.

The toggle button menu contains the albedo choice and five toggle buttons:

- **fresh**
- **dense**
- **moist**
- **old**
- **melting**

Click on the corresponding toggle button to choose the snow condition.

After choosing the albedo value, select **Finished with Selections** to terminate the snow condition menu.

D. Ice. Choose the albedo surface **Ice** in the **Wave Independent, Tabulated** menu to display a menu with one pushbutton and a toggle button menu with the albedo choice and four toggle buttons.

The pushbutton is **Finished with Selections**.

The toggle button menu contains the albedo choice and four toggle buttons:

- **white**
- **grey**

- **snow and ice**
- **dark glass**

Click on the corresponding toggle button to choose the ice condition.

After choosing the albedo value, select **Finished with Selections** to terminate the ice condition menu.

3. Spectral. Select **Spectral** in the **Albedo** menu when another option is selected, and select an area toggle button to display a menu with one pushbutton and a menu.

The pushbutton is **Finished with Selections**.

The menu is **Spectral Surface Albedo Models for Area - #**, in which # is the ordinal number of the area. The menu appears each time you select a surface albedo area.

The seven toggle buttons are

- **Spectral Model - 0**
- **Spectral Model - 1**
- **Spectral Model - 2**
- **Spectral Model - 3**
- **Spectral Model - 4**
- **Spectral Model - 5**
- **Spectral Model - 6**

The albedo for each spectral model depends on the wavelength of the radiation in the 0.55- to 12.0-μm range. **Spectral Model - 0** corresponds to the background albedo value (0.2) at all wavelengths.

Click on the corresponding toggle button to select the spectral albedo model input value.

After choosing the spectral albedo model, select **Finished with Selections** to terminate the spectral menu.

3.4.1.6 *Aerosol Profile Printout.*—Select **Aerosol Profile Printout** in the **Model Changes** menu to display a toggle button menu. The toggle button menu contains five choices:

- **None**
- **Ext Coefs & Scale Factors**
- **Adds Cross-sections to Printout**
- **Adds Scar and Absorption Coefs**
- **Full Details**

The depressed toggle button indicates the **Aerosol Profile Printout** choice. Click on a different toggle button to change the input value.

Aerosol Profile Printout controls the amount of detail contained in the aerosol printout, which is different from the binary and ASCII text BLIRB8 output files.

None means the aerosol profile printout is suppressed.

Ext Coefs & Scale Factors means the aerosol profile printout contains the tables of attenuation coefficients as a function of wavelength for each aerosol model and the scale factors used by BLIRB8.

Adds Cross-sections to Printout means the aerosol profile printout contains all previously mentioned aerosol profile outputs and the cross-sections (cm^2/particle) for each aerosol model.

Adds Scar & Absorption Coefs means the aerosol profile printout contains all previously mentioned aerosol profile outputs and the scattering and absorption coefficient tables as a function of wavelength.

Full Details means the aerosol profile printout contains all aerosol model properties.

None is the **Aerosol Profile Printout** default.

3.4.1.7 *Surface Temperature (K).*—Select **Surface Temperature** in the **Model Changes** menu to display an adjustable scale. Drag the scale cursor to a new position on the adjustable scale **Surface Temperature (K)** to input the surface temperature.

The **Surface Temperature (K)** 288.2 K is the default.

3.4.2 Region Selection

Select **Region Selection** or press **r** in the **Modify** menu (or, enter **Alt-mr** or **F10-mr**) to display a secondary menu with one pushbutton for each aerosol region currently defined and, if the maximum number of aerosol regions has not been reached, one pushbutton corresponds to a new region. All pushbuttons, except possibly the last, are in the form **BLIRB Region - #** in which # is the ordinal number of the aerosol region. If it is possible to create a new aerosol region, the last pushbutton is **Add New BLIRB Region**.

Click on the corresponding pushbutton to select the aerosol region of interest. Click outside the menu to terminate the menu.

Select **BLIRB Region - 1** to display a menu with four pushbuttons:

- **Dimensions**
- **Material - 1**
- **Material - 2**
- **Material - 3**

Select **BLIRB Region - #** in which # is > 1 to display a menu with six pushbuttons:

- **Dimensions**
- **Material - 1**
- **Material - 2**
- **Material - 3**

- **Location**
- **Delete Region**

Select **Add New BLIRB Region** to display a menu with the pushbutton **Dimensions**.

Click on the corresponding pushbutton to select the desired operation. Click outside the menu to terminate the menu.

One region (primary BLIRB8 region) with dimensions 5 by 4 by 5 km is the default.

All three materials are **No cloud** with infinite visibility.

3.4.2.1 *Dimensions.*—Select **Dimensions** or press **d** in the **Region Selection** menu to display a menu with one pushbutton and three adjustable scales.

The pushbutton is **Finished with Selections**.

The adjustable scales are

- **Length in X Direction (km)**
- **Length in Y Direction (km)**
- **Length in Z Direction (km)**

Drag the scale cursor to a new position on the adjustable scales **Length in X Direction (km)**, **Length in Y Direction (km)**, and **Length in Z Direction (km)** to input the region dimensions in the X, Y, and Z directions, respectively.

The values 5, 4, and 5 are the X, Y, and Z region dimension default values, respectively, of the primary BLIRB8 region. 0, 0, and 0 are the default dimensions of any other region.

The grid mesh changes when the corresponding dimensions of the primary BLIRB8 region (Region - 1) change. This noticeably changes the positions and dimensions of the other aerosol regions and surface albedo areas. Failure to

monitor these effects can invalidate some inputs. To be safe, you should resize the primary BLIRB8 region before adding other regions or albedo areas.

After inputting the three region dimensions, select **Finished with Selections** to terminate the **Dimensions** menu.

If you selected **Add New BLIRB Region** before **Dimensions**, **Materials** and **Location** are automatically invoked after terminating the **Dimensions** menu.

3.4.2.2 *Materials.*—Select **Material - 1**, **Material - 2**, or **Material - 3** or press 1, 2, or 3, respectively, in the **Region Selection** menu to display a menu with one pushbutton and a toggle button menu with 16 toggle buttons.

The pushbutton is **Finished with Selections**.

The toggle button menu is **Material # for Region $**, in which # is 1, 2, or 3 depending on the material pushbutton selected and $ is the ordinal number of the region.

The 16 toggle buttons are

- **LOWTRAN Default Cloud**
- **No Cloud**
- **Rural Aerosol**
- **Maritime Aerosol**
- **Urban Aerosol**
- **Tropospheric Aerosol**
- **Stratospheric Aerosol**
- **Volcanic**
- **Meteoric Dust**
- **Fog**
- **Clouds**
- **Rain**
- **Snow**
- **Desert Aerosol**

- **Dust and Dirt**
- **Combat Dust and Smoke**

Click on the corresponding toggle button to select **LOWTRAN Default Cloud**, **No Cloud**, **Stratospheric Aerosol**, **Meteoric Dust**, or **Snow**.

Select one of the other material options to display a menu with a list of refinement options. Aerosol regions are displayed as 3-D boxes with transparent color. The color and shade of color correspond to the first material specified for a region.

After choosing a material for a region, select **Finished with Selections** to terminate the **Material** menu and display a new menu with one pushbutton and an adjustable scale.

The pushbutton is **Finished with Selections**.

The adjustable scale is **Aerosol Component Vis (km) - Material # Region $** in which # is the number of the material (1, 2, or 3) and $ is the ordinal number for the region.

Drag the scale cursor to a new position to select the horizontal visibility within the aerosol. Select **Finished with Selections** to terminate the **Aerosol Component Vis** menu when the task is accomplished.

1. Rural or Urban Aerosols. Select **Rural Aerosol** or **Urban Aerosol** in the **Materials** menu to display a menu with one pushbutton and a toggle button menu with 12 toggle buttons.

The pushbutton is **Finished with Selections**.

The toggle button menu is **Rural Aerosol Options** or **Urban Aerosol Options**, depending upon whether **Rural Aerosol** or **Urban Aerosol** was selected.

The 12 toggle buttons are

- **LOWTRAN 0 percent RH**
- **LOWTRAN 70 percent RH**
- **LOWTRAN 80 percent RH**
- **LOWTRAN 99 percent RH**
- **EOSAEL 0 percent RH**
- **EOSAEL 50 percent RH**
- **EOSAEL 70 percent RH**
- **EOSAEL 80 percent RH**
- **EOSAEL 90 percent RH**
- **EOSAEL 95 percent RH**
- **EOSAEL 98 percent RH**
- **EOSAEL 99 percent RH**

LOWTRAN and EOSAEL are the two available rural and urban aerosol characteristic models.

Click on the corresponding toggle button to select the desired rural or urban aerosol option.

After making the selection, click on **Finished with Selections** to terminate the menu.

2. Maritime Aerosol. Select **Maritime Aerosol** in the **Materials** menu to display a menu with one pushbutton and a toggle button menu with 12 toggle buttons.

The pushbutton is **Finished with Selections**.

The toggle button menu is **Maritime Aerosol**.

The toggle buttons are

- **LOWTRAN 0 percent RH**
- **LOWTRAN 70 percent RH**

- **LOWTRAN 90 percent RH**
- **LOWTRAN 99 percent RH**
- **EOSAEL 0 percent RH**
- **EOSAEL 50 percent RH**
- **EOSAEL 70 percent RH**
- **EOSAEL 80 percent RH**
- **EOSAEL 90 percent RH**
- **EOSAEL 95 percent RH**
- **EOSAEL 98 percent RH**
- **EOSAEL 99 percent RH**

LOWTRAN and EOSAEL are the two available maritime aerosol characteristic models.

Click on the corresponding toggle button to select the desired maritime aerosol option.

After making the selection, click on **Finished with Selections** to terminate the menu.

3. Tropospheric Aerosol. Select **Tropospheric Aerosol** in the **Materials** menu to display a menu with one pushbutton and a toggle button menu with four toggle buttons.

The pushbutton is **Finished with Selections**.

The toggle button menu is **Tropospheric Aerosol Options**.

The toggle buttons are

- **LOWTRAN 0 percent**
- **LOWTRAN 70 percent RH**
- **LOWTRAN 90 percent RH**
- **LOWTRAN 99 percent RH**

LOWTRAN is the only available tropospheric aerosol characteristic model.

Click on the corresponding toggle button to select the desired tropospheric aerosol option.

After making the selection, click on **Finished with Selections** to terminate the menu.

4. Volcanic Aerosol. Select **Volcanic** in the **Materials** menu to display a menu with one pushbutton and a toggle button menu with two toggle buttons.

The pushbutton is **Finished with Selections**.

The toggle button menu is **Volcanic Aerosol Options**.

The toggle buttons are

- **LOWTRAN Aged**
- **LOWTRAN Fresh**

LOWTRAN is the only available volcanic aerosol characteristic model.

Click on the corresponding toggle button to select the desired volcanic aerosol option.

After making the selection, click on **Finished with Selections** to terminate the menu.

5. Fog Aerosol. Select **Fog** in the **Materials** menu to display a menu with one pushbutton and a toggle button menu with four toggle buttons.

The pushbutton is **Finished with Selections**.

The toggle button menu is **Fog Aerosol Options**.

The toggle buttons are

- **LOWTRAN Radiative**
- **LOWTRAN Advective**
- **EOSAEL Heavy Advection**
- **EOSAEL Moderate Radiation**

LOWTRAN and EOSAEL are the two available fog aerosol characteristic models.

Click on the corresponding toggle button to select the desired fog aerosol option.

After making the selection, click on **Finished with Selections** to terminate the menu.

6. Clouds. Select **Clouds** in the **Materials** menu to display a menu with one pushbutton and a toggle button menu with 10 toggle buttons.

The pushbutton is **Finished with Selections**.

The toggle button menu is **Cloud Options**.

The toggle buttons are

- **Deirmendjian Model C**
- **LOWTRAN Cumulus**
- **LOWTRAN Altostratus**
- **LOWTRAN Stratus**
- **LOWTRAN Stratus/Strato**
- **LOWTRAN Nimbostratus**
- **LOWTRAN Standard Cirrus**
- **LOWTRAN Subvisual Cirrus**
- **EOSAEL Fairweather Cumulus**
- **EOSAEL Cumulus Congestus**

LOWTRAN and EOSAEL are the two available cloud characteristic models.

Click on the corresponding toggle button to select the desired cloud option.

After making the selection, click on **Finished with Selections** to terminate the menu.

7. Rain. Select **Rain** in the **Materials** menu to display a menu with one pushbutton and a toggle button menu with three toggle buttons.

The pushbutton is **Finished with Selections**.

The toggle button menu is **Rain Options**.

The toggle buttons are

- **EOSAEL Drizzle**
- **EOSAEL Widespread**
- **EOSAEL Thunderstorm**

EOSAEL is the only available rain characteristic model.

Click on the corresponding toggle button to select the desired rain option.

After making the selection, click on **Finished with Selections** to terminate the menu.

8. Desert Aerosol. Select **Desert Aerosol** in the **Materials** menu to display a menu with one pushbutton and a toggle button menu with four toggle buttons.

The pushbutton is **Finished with Selections**.

The toggle button menu is **Desert Aerosol Options**.

The toggle buttons are

- **LOWTRAN Wind 0 mps**
- **LOWTRAN Wind 10 mps**
- **LOWTRAN Wind 20 mps**
- **LOWTRAN Wind 30 mps**

LOWTRAN is the only available desert aerosol characteristic model.

Click on the corresponding toggle button to select the desired desert aerosol option.

After making the selection, click on **Finished with Selections** to terminate the menu.

9. Dust and Dirt. Select **Dust and Dirt** in the **Materials** menu to display a menu with one pushbutton and a toggle button menu with three toggle buttons.

The pushbutton is **Finished with Selections**.

The toggle button menu is **Dust and Dirt Options**.

The toggle buttons are

- **Dirt**
- **EOSAEL Dust Light Loading**
- **EOSAEL Dust Heavy Loading**

EOSAEL is the only available dust characteristic model.

Click on the corresponding toggle button to select the desired dust and dirt option.

After making the selection, click on **Finished with Selections** to terminate the menu.

10. Combat Dust and Smoke. Select **Combat Dust and Smoke** in the **Materials** menu to display a menu with one pushbutton and a toggle button menu with six toggle buttons.

The pushbutton is **Finished with Selections**.

The toggle button menu is **Combat Dust and Smoke Options**.

The toggle buttons are

- **EOSAEL High Explosive Dust**
- **EOSAEL WP Smoke 17 percent RH**
- **EOSAEL WP Smoke 50 percent RH**
- **EOSAEL WP Smoke 90 percent RH**
- **EOSAEL Fog Oil**
- **EOSAEL HC Smoke 85 percent RH**

EOSAEL is the only available combat dust and smoke characteristic model.

Click on the corresponding toggle button to select the desired combat dust and smoke option.

After making the selection, click on **Finished with Selections** to terminate the menu.

3.4.2.3 *Location.*—Select **Location** or press **l** in the **Region Selection** menu to cause the display to snap to a view downward from the positive Z axis. Transparent red with a red lettered prompt below the projection stating `Move Region` depicts the projection of the region of interest onto the XY plane. Drag the cursor to change the position of the region of interest in the XY plane. When you release the mouse button the display snaps to a view toward the BLIRB8 space from the negative Y axis. Transparent red with a red lettered prompt below the projection stating `Move Region` depicts the projection of the region of interest onto the XZ plane. Drag the cursor to change the position of the region of interest in the XZ plane. After you release the mouse button, the

display snaps back to the original view, but the region of interest will be in a new position.

The origin of the BLIRB8 space is the default position of a newly created region. It must be repositioned.

3.4.2.4 *Delete Region.*—Select **Delete Region** or press **r** in the **Region Selection** menu to remove the region of interest and its associated materials from all inputs and the display.

3.4.3 Albedo Area Selection

Select **Albedo Area Selection** or press **a** in the **Modify** menu (or, enter **Alt-ma** or **F10-ma**) to display a menu with one pushbutton for each defined surface albedo area and one pushbutton corresponding to a new area if the maximum number of albedo areas has not been reached. All pushbuttons, except possibly the last, are **Albedo Area - #** in which # is the ordinal number of the albedo area. If it is possible to create a new albedo area, the last pushbutton is **Add New Albedo Area**.

Click on the corresponding pushbutton to select the albedo area of interest. Click outside the menu to terminate the menu.

Select **Albedo Area - 1** to display a menu with one pushbutton **Albedo**.

1. Select **Albedo Area - #**, in which # is greater than 1, to display a menu with four pushbuttons.

The pushbuttons are

- **Dimensions**
- **Albedo**
- **Location**
- **Delete Area**

2. Select **Add New Albedo Area** to display a menu with one pushbutton.

The pushbutton is **Dimensions**.

Click on the corresponding pushbutton to select the desired operation for the albedo area of interest. Click outside the menu to terminate the menu.

One albedo area (Z = 0 plane of the primary BLIRB8 region) with dimensions 5 by 4 km is the default. The background albedo (0.2) is the default albedo.

3.4.3.1 *Dimensions.*—Select **Dimensions** or press **d** in the **Albedo Area Selection** menu to display a menu with one pushbutton and two adjustable scales.

The pushbutton is **Finished with Selections**.

The adjustable scales are

- **Length in X Direction (km)**
- **Length in Y Direction (km)**

Drag the **Length in X Direction (km)** scale cursor to a new position to input the albedo area dimension in the X direction. Drag the **Length in Y Direction (km)** scale cursor to a new position to input the albedo area dimension in the Y direction. The values (5, 4) of the (X, Y) dimensions of the primary BLIRB8 albedo area are the default values. The values (0, 0) of the (X, Y) dimensions of any other albedo area are the default dimensions.

The dimensions of the primary BLIRB8 albedo area change to cover the entire XY projection of Region - 1 onto the Z = 0 plane when you change the dimensions of the primary BLIRB8 region (Region - 1).

After inputting both albedo area dimensions, select **Finished with Selections** to terminate the **Dimensions** menu.

If you selected **Add New Albedo Area** before **Dimensions**, **Albedo** and **Location** are automatically invoked after you terminate the **Dimensions** menu.

3.4.3.2 *Albedo.*—Select **Albedo** or press **b** in the **Albedo Area Selection** menu to get one of three results, depending upon the type of albedo selected in **Model Changes**:

- **Wave Independent, User-defined**
- **Wave Independent, Tabulated**
- **Spectral**

The albedo areas are displayed as XY-plane rectangles in shades of green. The darker the green, the higher the albedo.

1. Wave Independent, User-Defined. If the albedo type is **Wave Independent, User-defined**, a menu with one pushbutton and an adjustable scale is displayed.

The pushbutton is **Finished with Selections**.

The adjustable scale is **Albedo Value for Area - #**, in which # is the ordinal number of the area of interest.

Select the adjustable scale. Drag the scale cursor to a new position to select a new albedo value between 0 and 1 for area number #.

After inputting the albedo, select **Finished with Selections** to terminate the menu.

Wave Independent, User-defined is the **Albedo** default, and the background albedo (0.2) for each surface albedo area is the albedo value default.

2. Wave Independent, Tabulated. If the albedo type is **Wave Independent, Tabulated**, a menu with one pushbutton and a menu with 31 toggle buttons is displayed.

The pushbutton is **Finished with Selections**.

The menu is **Broad-Band Albedo Surfaces for Area - #**, in which # is the ordinal number of the area.

The toggle buttons are

- **Default**
- **Dark soil**
- **Light soil**
- **Dark-ploughed**
- **Light-ploughed**
- **Clay**
- **Sandy soil**
- **Sand**
- **White sand**
- **Asphalt**
- **Lava**
- **Tundra**
- **Steppe**
- **Concrete**
- **Stone**
- **Desert**
- **Rock**
- **Dirt road**
- **Clay road**
- **Grass**
- **Mowed grass**
- **Deciduous**
- **Coniferous**
- **Rice**
- **Beet, wheat**
- **Potato**
- **Rye**
- **Cotton**
- **Lettuce**
- **Snow**
- **Ice**

Click the corresponding toggle button to choose **Default** (background albedo = 0.2); **White sand**; **Asphalt**; **Lava**; **Tundra**; **Steppe**; **Concrete**; **Stone**; **Desert**; **Rice**; **Beet, wheat**; **Potato**; **Rye**; **Cotton**; or **Lettuce**.

After choosing the albedo value, select **Finished with Selections** to terminate the menu.

A. Soils and Roads. Choose **Dark soil**, **Light soil**, **Dark-ploughed**, **Light-ploughed**, **Clay**, **Sandy soil**, **Sand**, **Rock**, **Dirt road**, or **Clay road** in the **Broad-Band Albedo Surfaces for Area - #** menu to display a menu with one pushbutton, a toggle button menu, and two toggle buttons.

The pushbutton is **Finished with Selections**.

The toggle button menu contains the albedo choice.

The toggle buttons are

- **dry**
- **wet**

Click on the corresponding toggle button to choose **dry** or **wet**.

After choosing the albedo value, select **Finished with Selections** to terminate the dry/wet menu.

B. Grass and Trees. Choose **Grass**, **Mowed grass**, **Deciduous**, or **Coniferous** in the **Broad-Band Albedo Surfaces for Area - #** menu to display a menu with one pushbutton, a toggle button menu, and three toggle buttons.

The pushbutton is **Finished with Selections**.

The toggle button menu contains the albedo choice.

The toggle buttons are

- **growing**
- **dormant**
- **unspecified**

Click on the corresponding toggle button to choose **growing**, **dormant**, or **unspecified**.

After choosing the albedo value, select **Finished with Selections** to terminate the growing/dormant/unspecified menu.

C. Snow. Choose **Snow** in the **Broad-Band Albedo Surfaces for Area - #** menu to display a menu with one pushbutton and a toggle button menu with the albedo choice and five toggle buttons.

The pushbutton is **Finished with Selections**

The toggle button menu contains the albedo choice.

The toggle buttons are

- **fresh**
- **dense**
- **moist**
- **old**
- **melting**

Click on the corresponding toggle button to choose the snow condition.

After choosing the albedo value, select **Finished with Selections** to terminate the snow condition menu.

D. Ice. Choose **Ice** in the **Broad-Band Albedo Surfaces for Area - #** menu to display a menu with one pushbutton and a toggle button menu with the albedo and four toggle buttons.

The pushbutton is **Finished with Selections**.

The toggle button menu contains the albedo choice.

The toggle buttons are

- **white**
- **grey**
- **snow and ice**
- **dark glass**

Click on the corresponding toggle button to choose the ice condition.

After choosing the albedo value, select **Finished with Selections** to terminate the ice condition menu.

3. Spectral. If the current albedo type is **Spectral**, a menu with one pushbutton and a menu with seven toggle buttons is displayed.

The pushbutton is **Finished with Selections**.

The menu is **Spectral Surface Albedo Models for Area - #**, in which # is the ordinal number of the area of interest.

The toggle buttons are

- **Spectral Model - 0**
- **Spectral Model - 1**
- **Spectral Model - 2**
- **Spectral Model - 3**
- **Spectral Model - 4**
- **Spectral Model - 5**
- **Spectral Model - 6**

Each spectral model albedo depends on the wavelength of the radiation in the 0.55- to 12.0-μm range. **Spectral Model - 0** corresponds to the background albedo value (0.2) at all wavelengths.

Click on the corresponding toggle button to select the spectral albedo model input value.

After choosing the spectral albedo model, select **Finished with Selections** to terminate the spectral menu.

3.4.3.3 *Location.*—Select **Location** or press **l** in the **Albedo Area Selection** menu to snap the display to a view downward from the positive Z axis. Red with a red lettered prompt below the area stating `Move Area` depicts the albedo area of interest. Drag the cursor to change the position of the albedo area of interest in the XY plane. After you release the left mouse button, the display snaps back to the original view, but the albedo area of interest is in a new position.

The origin of the BLIRB8 space is the default position of a newly created albedo area. It must be repositioned.

3.4.3.4 *Delete Area.*—Select **Delete Area** or press **a** in the **Albedo Area Selection** menu to remove the albedo area of interest and its associated albedo from all inputs and the display.

3.4.4 Grid Mesh Selection

Select **Grid Mesh Selection** or press **g** in the **Modify** menu (or, enter **Alt-mg** or **F10-mg**) to display a menu with three pushbuttons:

- **X Grid Mesh**
- **Y Grid Mesh**
- **Z Grid Mesh**

Click on the corresponding pushbutton or enter **Shift-F6**, **Shift-F7**, or **Shift-F8** to select the axis of interest. Click outside the menu to terminate the menu.

3.4.4.1 *X Grid Mesh.*—Select **X Grid Mesh** to display a menu with one pushbutton and two columns of adjustable scales.

The pushbutton is **Finished with Selections**.

The two columns contain two or more adjustable scales each.

All the scales in the left column, except possibly the last, are **End of X Mesh # Interval (km)** in which # is the ordinal number of a minor grid interval. If the number of X mesh intervals is less than the maximum allowed, the last scale is **End of New X Mesh Interval (km)**. The scales in the right column are **Num Subintervals**. The right column contains one scale for each scale in the left column. The maximum allowable total number of subintervals for all intervals combined is depicted above the right column of scales as `Total <= 40` in which 40 is the number.

The origin of the BLIRB8 space (0.0 km) is the start point of the first X mesh interval. The start point of each X mesh interval, after the first, is the end point of the previous X mesh interval. Adjust the interval end points to modify X mesh intervals of interest. Adjust the number of subintervals within the adjusted intervals to prevent the inadvertent modification of the BLIRB8 regions and albedo areas within these intervals.

Adding a new X mesh interval is the same as subdividing a current X mesh interval and making the adjustments necessary to fit the users model requirements. Adjust the interval end points so the last scale in the left column depicts the greatest end point value to add the new X mesh interval. You must also change the corresponding **Num Subintervals** scales in the right column. Alternately, consider the last scale in the left column as out-of-sequence relative to the other interval end point scales, and add the desired interval end point and the corresponding number of subintervals. The software inserts the scale in the proper sequence after you exit the menu. Examine all previous intervals to determine the start point of the new interval.

After entering the interval data, select **Finished with Selections** to terminate the menu.

One end point of 5.0 km and 10 subintervals are the default initial values for the X grid mesh intervals making the X grid mesh spacing 0.5 km.

3.4.4.2 *Y Grid Mesh.*—Select **Y Grid Mesh** to display a menu with one pushbutton and two columns of adjustable scales.

The pushbutton is **Finished with Selections.**

The two columns contain two or more adjustable scales each.

All the scales in the left column, except possibly the last, are **End of Y Mesh # Interval (km)** in which # is the ordinal number of a minor grid interval. If the number of Y mesh intervals is less than the maximum allowed, the last scale is **End of New Y Mesh Interval (km)**. The scales in the right column are **Num Subintervals**. The right column contains one scale for each scale in the left column. The maximum allowable total number of subintervals for all intervals combined is depicted above the right column of scales as `Total <= 40` where 40 would be that number.

The origin of the BLIRB8 space (0.0 km) is the start point of the first Y mesh interval. The start point of each Y mesh interval, after the first, is the end point of the previous Y mesh interval. Adjust the interval end points to modify Y mesh intervals of interest. Adjust the number of subintervals within the adjusted intervals to prevent the inadvertent modification of BLIRB8 regions and albedo areas within these intervals.

Adding a new Y mesh interval is the same as subdividing a current Y mesh interval and making the adjustments necessary to fit the users model requirements. Adjust the interval end points so the last scale in the left column depicts the greatest end point value to add the new Y mesh interval. You must also change the corresponding **Num Subintervals** scales in the right column. Alternately, consider the last scale in the left column as out-of-sequence relative to the other interval end point scales, and add the desired interval end point and the corresponding number of subintervals. The software inserts the scale in the proper sequence after you exit the menu. Examine all previous intervals to determine the start point of the new interval.

After entering the interval data, select **Finished with Selections** to terminate the menu.

One end point of 4.0 km and 8 subintervals are the default initial values for the Y grid mesh intervals making the Y grid mesh spacing 0.5 km.

3.4.4.3 *Z Grid Mesh.*—Select **Z Grid Mesh** to display a menu with one pushbutton and two columns of adjustable scales.

The pushbutton is **Finished with Selections**.

The two columns contain two or more adjustable scales each.

All the scales in the left column, except possibly the last, are **End of Z Mesh # Interval (km)** in which # is the ordinal number of a minor grid interval. If the number of Z mesh intervals is less than the maximum allowed, the last scale is **End of New Z Mesh Interval (km)**. The scales in the right column are **Num Subintervals**. The right column contains one scale for each scale in the left column. The maximum allowable total number of subintervals for all intervals combined is depicted above the right column of scales as `Total <= 40` in which 40 is the number.

The origin of the BLIRB8 space (0.0 km) is the start point of the first Z mesh interval. The start point of each Z mesh interval, after the first, is the end point of the previous Z mesh interval. Adjust the interval end points to modify Z mesh intervals of interest. Adjust the number of subintervals within the adjusted intervals to prevent the inadvertent modification of the BLIRB8 regions and albedo areas within these intervals.

Adding a new Z mesh interval is the same as subdividing a current Z mesh interval and making the adjustments necessary to fit the users model requirements. Adjust the interval end points so the last scale in the left column depicts the greatest end point value to add the new Z mesh interval. You must also change the corresponding **Num Subintervals** scales in the right column. Alternately, consider the last scale in the left column as out-of-sequence relative to the other interval end point scales, and add the desired interval end point and

the corresponding number of subintervals. The software inserts the scale in the proper sequence after you exit the menu. Examine all previous intervals to determine the start point of the new interval.

After entering the interval data, select **Finished with Selections** to terminate the menu.

One end point of 5.0 km and 10 subintervals are the default initial values for the Z grid mesh intervals making the Z grid mesh spacing 0.5 km.

3.4.5 Cloud Changes

Select **Cloud Changes** or press **c** in the **Modify** menu (or, enter **Alt-mc**, **F10-mc**, or **Shift-F9**) to display a menu with one pushbutton, two toggle button menus, and one adjustable scale **Wind Speed (mps)**.

The pushbutton is **Finished with Selections**.

The toggle button menus are

- **Cloud Structure**
- **Aerosol Outside Physical Region**

The adjustable scale is **Wind Speed (mps)**.

After entering the **Cloud Changes** input data, select **Finished with Selections** to terminate the menu.

3.4.5.1 *Cloud Structure.*—Select **Cloud Structure** in the **Cloud Changes** menu to display a toggle button menu. The toggle button menu contains three choices:

- **No Cloud**
- **Rectangular Structure**
- **CSS Model**

The depressed toggle button indicates the selected **Cloud Structure**. Click on a different toggle button to change the input value.

No Cloud means the BLIRB8 region contains no cloud.

Rectangular Structure means the cloud aerosols (up to three different aerosols or densities) are uniformly distributed over each of the BLIRB8 aerosol regions.

CSS Model means the Cloud Scene Simulation prototype model developed by TASC is used for the entire BLIRB8 space.

No Cloud is the default **Cloud Structure** option.

3.4.5.2 *Aerosol Outside Physical Region.*—Select **Aerosol Outside Physical Region** in the **Cloud Changes** menu to display a toggle button menu. The toggle button menu contains two choices:

- **Background Aerosol**
- **Periodic Boundary Conditions**

The depressed toggle button indicates the selected **Aerosol Outside Physical Region**. Click on a different toggle button to change the input value.

Background Aerosol means BLIRB8 uses a uniform aerosol distribution outside the primary BLIRB8 region that is equal to the background aerosol. **Periodic Boundary Conditions** means BLIRB8 replicates the BLIRB8 space in all directions around and outside the BLIRB8 space.

Periodic Boundary Regions is the default **Aerosol Outside Physical Region** option.

3.4.5.3 *Wind Speed (mps).*—Select **Wind Speed (mps)** in the **Cloud Changes** menu to display an adjustable scale. Drag the scale cursor to a new position to input the surface wind speed using the adjustable scale **Wind Speed (mps)**. The default wind speed is 0.0 mps.

3.4.6 Sun Changes

Select **Sun Changes** or press s in the **Modify** menu (or, enter **Alt-ms**, **F10-ms**, or **Shift-F10**) to display a menu with one pushbutton, three toggle button menus, and two adjustable scales.

The pushbutton is **Finished with Selections**.

The toggle button menus are

- **Solar Flux and Sky Radiance at 5 km**
- **Sky Radiance Input**
- **Spectral Molecular Transmission**

The adjustable scales are

- **Solar Zenith Angle (deg)**
- **Solar Azimuth Angle (deg)**

After entering the **Sun Changes** input data, select **Finished with Selections** to terminate the menu.

3.4.6.1 *Solar Flux and Sky Radiance at 5 km.*—Select **Solar Flux and Sky Radiance at 5 km** in the **Sun Changes** menu to display a toggle button menu. The toggle button menu contains two choices:

- **Parameterized**
- **LOWTRAN**

The depressed toggle button indicates the selected **Solar Flux and Sky Radiance 5 km**. Click on a different toggle button to change the input value.

LOWTRAN is the default option.

3.4.6.2 *Sky Radiance Input.*—Select **Sky Radiance Input** in the **Sun Changes** menu to display a toggle button menu. The toggle button menu contains two toggle buttons:

- **No**
- **Yes**

The depressed toggle button indicates the selected **Sky Radiance Input**. Click on the corresponding toggle button to change the input value.

No is the default option.

3.4.6.3 *Spectral Molecular Transmission.*—Select **Spectral Molecular Transmission** in the **Sun Changes** menu to display a toggle button menu. The toggle button menu contains two choices:

- **No**
- **Yes**

The depressed toggle button indicates the selected **Spectral Molecular Transmission**. Click on a different toggle button to change the input value.

No is the default option.

3.4.6.4 *Solar Zenith Angle (deg).*—Select **Solar Zenith Angle (deg)** in the **Sun Changes** menu to display an adjustable scale. Drag the scale cursor to a new position on the **Solar Zenith Angle (deg)** adjustable scale to input the solar zenith angle.

The **Solar Zenith Angle (deg)** 0.0° is the default.

3.4.6.5 *Solar Azimuth Angle (deg).*—Select **Solar Azimuth Angle (deg)** in the **Sun Changes** menu to display an adjustable scale. Drag the scale cursor to a new position on the **Solar Azimuth Angle (deg)** adjustable scale to input the solar azimuth angle. The solar azimuth angle 0.0° is the default.

3.4.7 Flare Selection

Select **Flare Selection** or press **f** in the **Modify** menu (or, enter **Alt-mf** or **F10-mf**) to display a menu with one pushbutton for each defined flare and, if the maximum number of flares is not reached, one pushbutton corresponding to a new flare.

All pushbuttons, except possibly the last, are **Flare - #** in which # is the ordinal number of the flare. The last pushbutton is **Add New Flare** if it is possible to create a new flare.

Click on the corresponding toggle button to select the flare of interest. Click outside the menu to terminate the menu. The default input contains no flare.

Select **Flare - #** in which # is a number to display a menu with three pushbuttons **Parameter Options**, **Location**, and **Delete Flare**.

Select **Add New Flare** to display a menu to with one pushbutton **Parameter Options**.

Click on the corresponding pushbutton to select the desired operation for the flare of interest. Click outside the menu to terminate the menu.

3.4.7.1 Parameter Options.—Select **Parameter Options** or press **p** in the **Flare Selection** menu to display a menu to with one pushbutton, one toggle button menu, and two adjustable scales.

The pushbutton is **Finished with Selections**.

The toggle button menu is **Flare Type**

The adjustable scales are

- **Flare Intensity (watts)**
- **Flare Temperature (K)**

After entering the **Parameter Options** input data, select **Finished with Selections** to terminate the menu. If **Add New Flare option** is selected, **Location** is invoked after you select **Finished with Selections**.

1. Flare Type. Select **Flare Type** in the **Parameter Options** menu to display a toggle button menu. The toggle button menu contains three choices:

- **Isotropic**
- **10 percent Up & 90 percent Down**
- **User Defined**

The depressed toggle button indicates the input value. Click on a different toggle button to change the input value.

Selecting **Isotropic** means the flare energy radiates uniformly in all directions. Selecting **10 percent Up & 90 percent Down** means that 10 percent of the flare energy radiates uniformly in all directions above the flare and 90 percent of the energy radiates uniformly in all directions below the flare.

Select **User Defined** in the **Flare Type** menu to display another menu with one pushbutton and eight adjustable scales.

The pushbutton is **Finished with Selections**.

Four of the adjustable scales are **Fraction of Energy in Up Direction #**.

Four of the adjustable scales are **Fraction of Energy in Down Direction #**.

In both cases, # ranges from 1 to 4 indicating the octant of interest.

Drag the scale cursors to a new position along the adjustable scales to input the fraction of flare energy for each octant. If the sum of the fractions of flare energy do not total unity, the software normalizes the fractions so the new total is unity.

After entering the energy fractions, select **Finished with Selections** to terminate the **User Defined** menu.

2. Flare Intensity (watts). Select **Flare Intensity (watts)** in the **Parameter Options** menu. Drag the scale cursor to a new position along the **Flare Intensity (watts)** adjustable scale to input the flare intensity in watts.

3. Flare Temperature (K). Select **Flare Temperature (K)** in the **Parameter Options** menu. Drag the scale cursor to a new position on along the **Flare Temperature (K)** adjustable scale to input the flare temperature in Kelvin.

3.4.7.2 *Location.*—Select **Location** or press **l** in the **Flare Selection** menu to snap to a view downward from the positive Z axis. Red with a red lettered prompt below the projection stating `Move Flare` depicts the projection of the flare location onto the XY plane. Hold the left mouse button and move the mouse to change the position of the flare in the XY plane. Release the left mouse button and the display snaps to a view toward the BLIRB8 space from the negative Y axis. Red with a red lettered prompt below the projection stating `Move Flare` depicts the projection of the flare location onto the XZ plane. Hold the left mouse button down and move the mouse to change the position of the flare in the XZ plane. Release the left mouse button and the display snaps back to the original view, but the flare of interest is in a new position. Red stars depict flare positions.

The origin of the BLIRB8 space is the default position of a newly created flare. It must be repositioned.

3.4.7.3 *Delete Flare.*—Select **Delete Flare** or press **d** in the **Flare Selection** menu to remove the flare of interest from all inputs and the display.

3.4.8 SearchLight Selection

Select **SearchLight Selection** or press **l** in the **Modify** menu (or, enter **Alt-ml** or **F10-ml**) to display a menu with one or three pushbuttons.

The pushbutton is **Add/Modify SearchLight** if no searchlight exists.

The menu contains three pushbuttons if a searchlight exists:

- **Add/Modify SearchLight**
- **Location**
- **Delete SearchLight** (if a searchlight exists)

Only one searchlight is allowed.

The default input contains no searchlight.

Click on the corresponding pushbutton to select the desired operation for the searchlight. Click outside the menu to terminate the menu.

3.4.8.1 *Add/Modify SearchLight.*—Select **Add/Modify SearchLight** or press **a** in the **SearchLight Selection** menu (or, enter **Alt-mla** or **F10-mla**) to display a menu with one pushbutton and five adjustable scales.

The pushbutton is **Finished with Selections**.

The adjustable scales are

- **SearchLight Beam Zenith (deg)**
- **SearchLight Beam Azimuth (deg)**
- **SearchLight Intensity (watts)**
- **SearchLight Temperature (K)**
- **SearchLight Diameter (m)**

After entering the **Add/Modify SearchLight** input data, select **Finished with Selections** to terminate the menu. **Location** is automatically invoked after selecting **Finished with Selections** if the searchlight did not exist.

1. SearchLight Beam Zenith (deg). Select **SearchLight Beam Zenith (deg)** in the **Add/Modify SearchLight** menu. Drag the scale cursor to a new position on the **SearchLight Beam Zenith (deg)** adjustable scale to input the searchlight beam zenith angle in degrees.

2. SearchLight Beam Azimuth (deg). Select **SearchLight Beam Azimuth (deg)** in the **Add/Modify SearchLight** menu. Drag the scale cursor to a new position on the **SearchLight Beam Azimuth (deg)** adjustable scale to input the searchlight beam azimuth angle in degrees.

3. SearchLight Intensity (watts). Select **SearchLight Intensity (watts)** in the **Add/Modify SearchLight** menu. Drag the scale cursor to a new position on the **SearchLight Intensity (watts)** adjustable scale to input the searchlight intensity in watts.

4. SearchLight Temperature (K). Select **SearchLight Temperature (K)** in the **Add/Modify SearchLight** menu. Drag the scale cursor to a new position on the **SearchLight Temperature (K)** adjustable scale to input the searchlight temperature in Kelvin.

5. SearchLight Diameter (m). Select **SearchLight Diameter (m)** in the **Add/Modify SearchLight** menu. Drag the scale cursor to a new position on the **SearchLight Diameter (m)** adjustable scale to input the searchlight diameter in meters.

3.4.8.2 *Location.*—Select **Location** or press **l** in the **SearchLight Selection** menu (or enter **Alt-mll** or **F10-mll**) to snap to a view downward from the positive Z axis. White with a red lettered prompt below the projection stating `Move Slite` depicts the projection of the searchlight location onto the XY plane. Hold the left mouse button down and move the mouse to change the position of the searchlight in the XY plane. Release the left mouse button to snap to a view toward the BLIRB8 space from the negative Y axis. White with a red lettered prompt below the projection stating `Move Slite` depicts the projection of the searchlight location onto the XZ plane. Hold the left mouse button down and move the mouse to change the position of the searchlight in the XZ plane. Release the left mouse button to snap back to the original view with the searchlight a new position. A white star depicts the searchlight position.

The origin of the BLIRB8 space is the default position of a newly created searchlight. It must be repositioned.

3.4.8.3 *Delete SearchLight.*—Select **Delete SearchLight** or press **d** in the **SearchLight Selection** menu (or, enter **Alt-mld** or **F10-mld**) to remove the searchlight from all inputs and the display.

3.4.9 Spectral Range Changes

Select **Spectral Range Changes** or press **t** in the **Modify** menu (or, enter **Alt-mt** or **F10-mt**) to display a menu with two pushbuttons. The pushbuttons allow you to choose the units for the spectral range selection options. This spectral range corresponds to the spectral range of the radiation energy wave bands for BLIRB8.

The pushbuttons are

- **Wavenumber**
- **Wavelength**

The pushbuttons allow you to choose the units for the spectral range selection options. The spectral range corresponds to the spectral range of the radiation energy wave bands for BLIRB8.

Click on the corresponding pushbutton to select the spectral range units. Click outside the menu to terminate the menu.

3.4.9.1 *Wavenumber.*—Select **Wavenumber** or press **n** in the **Spectral Range Changes** menu (or, enter **Alt-mtn**, **F10-mtn**, or **Ctrl-F11**) to display a menu with one pushbutton and a toggle button menu with five toggle buttons.

The pushbutton is **Finished with Selections**.

The toggle button menu is **Wavenumber Interval (per cm)**.

The toggle buttons are

- **Visible: 8000 - 28000**
- **Near IR: 3000 - 13000**

- **Mid IR: 1200 - 5200**
- **Far IR: 500 - 1500**
- **2 Color IR: 600 - 3600**

Select the wavenumber range of interest from the menu.

Select one of the five toggle buttons to display a menu with one pushbutton and three adjustable scales.

The pushbutton is **Finished with Selections**.

The adjustable scales are

- **Lowest Wavenumber (per cm)**
- **Highest Wavenumber (per cm)**
- **Number of Wavenumber Intervals**

The range of values and increments available for the first two scales are the only differences between the scales in one toggle button selection and the scales in another toggle button selection. The scale value ranges correspond to the numbers in the toggle button label.

After entering the scale input data, select **Finished with Selections** to terminate the three-scale menu.

After entering the **Wavenumber** input data, select **Finished with Selections** to terminate the **Wavenumber** menu.

1. Lowest Wavenumber (per cm). Select **Lowest Wavenumber (per cm)** in the **Wavenumber Interval (per cm)** menu. Drag the scale cursor to a new position on the **Lowest Wavenumber (per cm)** adjustable scale to input the lowest wavenumber.

The **Lowest Wavenumber (per cm)** 10,000 per cm is the default value.

2. Highest Wavenumber (per cm). Select **Highest Wavenumber (per cm)** in the **Wavenumber Interval (per cm)** menu. Drag the scale cursor to a new position on the **Highest Wavenumber (per cm)** adjustable scale to input the highest wavenumber.

The **Highest Wavenumber (per cm)** 25,000 per cm is the default value.

3. Number of Wavenumber Intervals. Select **Number of Wavenumber Intervals** in the **Wavenumber Interval (per cm)** menu. Drag the scale cursor to a new position on the **Number of Wavenumber Intervals** adjustable scale to input the number of wavenumber intervals used for integration over the spectral range within BLIRB8.

The **Number of Wavenumber Intervals** 15 is the default value.

3.4.9.2 *Wavelength.*—Select **Wavelength** or press **l** in the **Spectral Range Changes** menu (or, enter **Alt-mtl**, **F10-mtl**, or **Ctrl-F12**) to display a menu with one pushbutton and a toggle button menu with five toggle buttons.

The pushbutton is **Finished with Selections**.

The toggle button menu is **Wavelength Interval (micrometers)**.

The toggle buttons are

- **Visible: 0.3 - 1.3**
- **Near IR: 0.7 - 3.2**
- **Mid IR: 2.0 - 7.0**
- **Far IR: 6.0 - 16.0**
- **2 Color IR: 3.0 - 13.0**

Select the wavelength range of interest from the menu.

Select one of the five toggle buttons to display a menu with one pushbutton and three adjustable scales.

The pushbutton is **Finished with Selections.**

The adjustable scales are

- **Lowest Wavelength (micrometers)**
- **Highest Wavelength (micrometers)**
- **Number of Wavelength Intervals**

The range of values and increments available for the first two scales are the only differences between the scales in one toggle button selection and the scales in another toggle button selection. The scale value ranges correspond to the numbers in the toggle button label.

After entering the scale input data, select **Finished with Selections** to terminate the three-scale menu.

After entering the **Wavelength** input data, select **Finished with Selections** to terminate the **Wavelength** menu.

1. Lowest Wavelength (micrometers). Select **Lowest Wavelength (micrometers)** to display an adjustable scale. Drag the scale cursor to a new position on the **Lowest Wavelength (micrometers)** adjustable scale to input the lowest wavelength.

2. Highest Wavelength (micrometers). Drag the scale cursor to a new position on the **Highest Wavelength (micrometers)** adjustable scale to input the highest wavelength.

3. Number of Wavelength Intervals. Drag the scale cursor to a new position on the **Number of Wavelength Intervals** adjustable scale to input the number of wavelength intervals used for integration over the spectral range within BLIRB8.

3.4.10 Computation Changes

Select **Computation Changes** or press **p** in the **Modify** menu (or, enter **Alt-mp**, **F10-mp**, or **Shift-F11**) to display a menu with one pushbutton, two toggle button menus, and three adjustable scales.

The pushbutton is **Finished with Selections**.

The toggle button menus are

- **Delta Function Adjustment**
- **Order of Spherical Harmonics**

The adjustable scales are

- **Maximum Number of Iterations**
- **Convergence Criterion**
- **Number of Convergence Fail Points**

After entering the **Computation Changes** input data, select **Finished with Selections** to terminate the menu.

3.4.10.1 *Delta Function Adjustment.*—Select **Delta Function Adjustment** in the **Computation Changes** menu to display a toggle button menu.

The toggle button menu contains two choices:

- **No**
- **Yes**

The depressed toggle button indicates the **Delta Function Adjustment** choice. Click on the corresponding toggle button to change the input value.

Yes is the default choice.

3.4.10.2 *Order of Spherical Harmonics.*—Select **Order of Spherical Harmonics** in the **Computation Changes** menu to display a toggle button. The toggle button menu contains seven choices for the order of the spherical harmonics used in the BLIRB8 computations.

The toggle buttons are **Order #** in which # is an integer between 0 and 6.

The depressed toggle button indicates the **Order of Spherical Harmonics** choice. Click on the corresponding toggle button to change the input value.

Order 2 is the default choice.

3.4.10.3 *Maximum Number of Iterations.*—Select **Maximum Number of Iterations** in the **Computation Changes** menu to display an adjustable scale. Drag the scale cursor to a new position on the **Maximum Number of Iterations** adjustable scale to input the maximum number of iterations to be used in the computation.

The **Maximum Number of Iterations** 10 is the default value.

3.4.10.4 *Convergence Criterion.*—Select **Convergence Criterion** in the **Computation Changes** menu to display an adjustable scale. Drag the scale cursor to a new position on the **Convergence Criterion** adjustable scale to input the convergence criterion to be used in the computation.

The **Convergence Criterion** 0.002 is the default value.

3.4.10.5 *Number of Convergence Fail Points.*—Select **Number of Convergence Fail Points** in the **Computation Changes** menu to display an adjustable scale. Drag the scale cursor to a new position on the **Number of Convergence Fail Points** adjustable scale to input the number of convergence fail points used in the computation.

The **Number of Convergence Fail Points** 5 is the default.

3.4.11 Output File Changes

Select **Output File Changes** or press **o** in the **Modify** menu (or, enter **Alt-mo, F10-mo**, or **Shift-F12**) to display a menu with one pushbutton and a toggle button menu with four toggle buttons.

The pushbutton is **Finished with Selections**.

The toggle button menu is **Radiant Flux Output Options**.

The toggle buttons are

- **No Output Flux**
- **Formatted Output File (GRID.ASC)**
- **Unformatted (binary) Output File (GRID.BIN)**
- **Both Formatted and Unformatted (GRID.ASC and GRID.BIN)**

The options govern the type of BLIRB8 radiant flux output file(s) created.

Both Formatted and Unformatted is the default option.

After entering the **Output File Changes** input data, select **Finished with Selections** to terminate the menu.

3.5 Reset

Select **Reset** on the menubar to remove rotation and/or translation of the BLIRB8 space display. (Or, enter **Alt-r** or **F10-r**.) The display resets to a view along an axis looking at the center of the BLIRB8 space.

3.6 Help

Select **Help** on the menubar to display a menu with eight options:

- **General Information**
- **File Options**
- **Viewing Options**
- **Flux Display Options**
- **Input Modifications Options**
- **Reset**
- **BLIRB8 Space Rotations**
- **BLIRB Space Translations**

(Or, enter **Alt-h** or **F10-h**.)

Select an option or press **g**, **f**, **v**, **x**, **m**, **s**, **r**, or **t**. (Or, enter **Alt-hg** or **F10-hg**; **Alt-hf** or **F10-hf**; **Alt-hv** or **F10-hv**; **Alt-hx** or **F10-hx**; **Alt-hm** or **F10-hm**; **Alt-hs** or **F10-hs**; **Alt-hr** or **F10-hr**; or **Alt-ht** or **F10-ht**.)

Click outside the menu to terminate the menu.

Select any options except **Viewing Options**, **Flux Display Options**, and **Input Modifications Options** to display a corresponding help message on the screen.

Click on **ok** to terminate the help message display.

3.6.1 Viewing Options

Select **Viewing Options** in the **Help** menu to display a menu with one pushbutton and a toggle button menu with four options.

The pushbutton is **Finished with Selections**.

The toggle button menu is **View Options**.

The options are

- **Viewing Axis Options**
- **Sun Options**
- **Zoom Options**
- **Toggle Switch Options**

Select one of the four options to display a corresponding help message on the screen.

Click on **ok** to terminate the help message display.

After reading the help messages, select **Finished with Selections** to terminate the menu.

3.6.2 Flux Display Options

Select **Flux Display Options** in the **Help** menu to display a menu with one pushbutton **Finished with Selections** and a toggle button menu **Flux Display Options** with four options.

The pushbutton is **Finished with Selections**.

The toggle button menu is **Flux Display Options**.

The options are

- **Flux Options**
- **Cross-Section Plane Orientation**
- **Cross-Section Plane Value Selection**
- **Wave Number Selection**

Select one of the four options to display a corresponding help message on the screen.

Click on **ok** to terminate the help message display.

After reading the help messages, select **Finished with Selections** to terminate the menu.

3.6.3 Input Modifications Options

Select **Input Modifications Options** in the **Help** menu to display a menu with one pushbutton and a toggle button menu with 11 options.

The pushbutton is **Finished with Selections**.

The toggle button menu is **Modify Options**.

The options are

- **Model Changes**
- **Region Selection**
- **Albedo Area Selection**
- **Grid Mesh Selection**
- **Cloud Changes**
- **Sun Changes**
- **Flare Selection**
- **Searchlight Selection**
- **Spectral Range Changes**
- **Computation Changes**
- **Output File Changes**

Select one of the 11 options to display a corresponding help message on the screen.

Click on **ok** to terminate the help message display.

After reading the help messages, select **Finished with Selections** to terminate the menu.

4. BLIRB8 Space Rotation

Hold the left mouse button down and move the mouse to perform rotations of the BLIRB8 space. The display is an ortho projection; therefore, everything in the BLIRB8 space is projected, using reasonably parallel projection lines, onto an observer position plane. Ortho projection helps conserve length and angular relationships between the objects in the display (near objects do not appear much larger than distant objects). Unfortunately, ortho projection does not allow the observer to select arbitrary positions around the BLIRB8 space from which to view the objects. The projections are onto a plane not a point.

There are three observer position planes to choose from, a YZ plane on the positive-X axis, an XZ plane on the negative-Y axis, and an XY plane on the positive-Z axis. Hold down the left mouse button and move the mouse to move the observer position on one of these planes. The center of the BLIRB8 space is the point looked at from all planes using all observer positions. The BLIRB8 space can disappear from the screen if a severe rotation is performed on any observer plane. If the BLIRB8 space disappears, undo some of the rotation and select another observer plane from which to view the BLIRB8 space. Severe rotations are not necessary.

5. BLIRB8 Space Translation

Hold down the right mouse button and move the mouse to perform translations of the BLIRB8 space. Translations cause the point looked at in the BLIRB8 space to change from the center of the BLIRB8 space. Perform translations with caution as they can cause unexpected viewing problems when followed by rotations.

6. BLIRB8 Default Input Values

The BLIRB8 default input values are loaded for the BLIRB8 inputs when you select the default **Create New File** or start VISUAL without a filename on the command line. Change the BLIRB8 default input values under the various options available under **Modify** on the menubar. The default values follow:

BLIRB8 Input	Default Value
Aerosol Model	No aerosols
Temperature Profile Model	1976 U.S. Standard
Meteorological Range	40 km
Tropospheric Profile	Set by Meteorological Range
Albedo Type	Wave Independent, User-defined
Aerosol Profile Printout	None
Surface Temperature	288.2 K
Number of Regions	1 (primary BLIRB8 region)
Dimensions of Region	5 by 4 by 5 km beginning at (X,Y,Z) = (0,0,0)
Materials for Region	No cloud
Material Visibilities for Region	Infinite
Number of Albedo Areas	1
Dimensions of Area	5 by 4 km beginning at (X,Y,Z) = (0,0,0)
Albedo for Area	Background Albedo = 0.2
Number of X Grid Meshes	1
Start and End Points of X Grid Mesh	0 and 5 km
Number of Subintervals	10
Number of Y Grid Meshes	1
Start and End Points of Y Grid Mesh	0 and 4 km
Number of Subintervals	8

Number of Z Grid Meshes	1
Start and End Points of Z Grid Mesh	0 and 5 km
Number of Subintervals	10
Cloud Structure	No Cloud
Aerosol Outside Physical Region	Periodic Boundary Conditions
Wind Speed	0 mps
Solar Flux and Sky Radiance at 5 km	LOWTRAN
Sky Radiance Input	No
Spectral Molecular Transmission	No
Solar Zenith Angle	0°
Solar Azimuth Angle	0°
Number of Flares	0
Number of Searchlights	0
Lowest Wavenumber	10,000 per cm
Highest Wavenumber	25,000 per cm
Number of Wavenumber Intervals	15
Delta Function Adjustment	Yes
Order of Spherical Harmonics	2
Maximum Number of Iterations	10
Convergence Criterion	0.002
Number of Convergence Fail Points	5
Radiant Flux Output Option	Both Formatted and Unformatted

7. VISUAL Global Software Variables

The file VISUAL0.H, under the heading Definitions used in the BLIRB8 Visualization Program, contains the global variables for the VISUAL.C code. Appendix A contains a listing of VISUAL0.H. You can assign new values to the global variables and recompile the VISUAL.C code to produce a new executable VISUAL. However, remember that these global variables are set to be consistent with the FORTRAN parameters in the BLIRB8 code. If you change the variables, change the corresponding parameters in the BLIRB8 code.

8. BLIRB_EN

The BUFR binary stream contains the BLIRB8 radiant flux field output encoded to save space. A BUFR binary file is composed of as many as six sections in series. Section 0, the indicator section, is required and includes the total length of the BUFR file or message. Section 1, the identification section, is required and includes the length of the section and identifying information about the originating site and the BLIRB8 data. Section 2, the optional section, is not required and not used for encoding the BLIRB8 data. Section 3, the data description section, is required and contains the length of the section and a collection of descriptors which define the form and content of the individual data elements comprising the BLIRB8 data. Section 4, the data section, is required and contains the length of the section and the encoded BLIRB8 data elements as defined by descriptors in Section 3. Section 5, the end section, is required and contains ASCII 7777 to signal the end of the BUFR file or message.

The size of a BUFR message can be quite large depending upon the size of the BLIRB8 radiant flux field output file. The approach used to develop the BLIRB_EN source code, BLIRB_EN.C, so it could create the complete BUFR message in one pass through a BLIRB8 radiant flux field output file was to concurrently construct Sections 0, 1, 3, and 4 by analyzing the BLIRB8 data as it is read and writing the required encoding information for each section to a corresponding temporary file. After all the BLIRB8 data are processed, the four temporary files are reread in sequence and the binary coded data from each along with the Section 5, 7777, are written to the final BUFR binary file.

Enter **blirb_en -i <infile> -o <outfile>** in which <infile> is the BLIRB8 radiant flux field ASCII output filename and <outfile> is the BLIRB8 BUFR binary encoded output filename to execute BLIRB_EN.

8.1 Section 1

Section 1 contains the year, month, day, hour, minute, and second taken from the computer system clock at runtime to specify when the BUFR message was

generated. If this information is to apply to the date and time for the BLIRB8 data used in a prescribed scenario, the corresponding C code in procedure Section 1 must be modified.

As of the end of this contract period, several parameters used in this section are not officially defined. As a result, dummy values are being used until the official values are assigned. The dummy values are assigned in the include file, BUFR.H appendix B. The Local Originating Center Code (LOCC) (byte 18) is given a temporary value of 111. The Model Identifier (MODEL) (byte 19) is given a temporary value of 1. The Database Sequence Number (DBSN) (bytes 21 and 22) is given a temporary value of 0. The assigned values must be changed after official values are available.

8.2 Section 3

Section 3 is composed of data element descriptors, replication operators, data description operators, and data descriptor sequences. These operators and sequences can be nested as deeply as desired. A data element descriptor simply defines the type of data element, unit of measurement, reference value, scale value, and data field width in bits. A replication operator replicates all descriptors and operators following its execution until it is signaled to stop. A data descriptor operator is primarily used to temporarily change reference value, scale value, or data field width for a data element descriptor. A descriptor sequence is a sequence of data element descriptors, replication operators, and data descriptor operators. Nesting of sequences, operators, and data descriptors is quite common. For each data element value encoded in Section 4, a unique description of that data element exists somewhere encoded in Section 3. The better the nesting and encoding, the shorter the section length.

To construct this section for the BLIRB8 radiant flux field data, a series of new data descriptors had to be added to the original BUFR Table B, the list of all data descriptors. Appendix F (beginning on page 183) contains the additional data descriptors. The F (column 1) of Table B is required to be 0 for all data descriptors (records). The value of X (column 2) is assigned a temporary value of 55 until an official value is designated. This number will delineate the BLIRB8 data descriptors from all other data descriptors.

The BLIRB8 radiant flux field output file contains a copy of the BLIRB8 input records from the ASCII input file in addition to the flux field output. The values of Y (column 3) of Table B contain the integers 1 through 83 and 120 through 147. The values 1 through 83 denote BLIRB8 input data elements. The values 120 through 147 denote BLIRB8 output values or arrays.

In addition to modifying BUFR Table B, BUFR Table D, a list of data descriptor sequences, was modified. The sequences contained in Table D are like a shorthand method of assigning the data descriptors from Table B to sets of data requiring several data descriptors. An additional 10 sequences added to Table D pertain to the BLIRB8 input/output data descriptors. Appendix G (beginning on page 193) contains these sequences. A sequence is a series of records in which the first record is `3 X Y` with `X` and `Y` being positive integers. The first record defines the sequence; and, the record `-1 -1 -1` denotes the end of the sequence. All records between the first and end records are the data descriptor members of the sequence. For the BLIRB8 data, a temporary value of 25 was assigned to `X`. The value of `Y` is the sequence number, 1 through 10.

The values are temporarily assigned and can be changed later when the official Tables B and D are defined. The pertinent definitions are contained in BUFR.H. The value assigned to BB (currently 55) is the temporary value of X denoting BLIRB8 data descriptors. The value assigned to YYI (currently 1) is the temporary value of Y corresponding to the beginning of the BLIRB8 inputs. The value of YYO (currently 120) is the temporary value of Y corresponding to the beginning of the BLIRB8 outputs. The value of YYM (currently 147) is the temporary value of Y corresponding to the last data descriptor of the BLIRB8 data. The value of DD (currently 25) is the temporary value of X denoting BLIRB8 descriptor sequences.

8.3 Section 4

Section 4 contains the encoded BLIRB8 data element values. Positive and negative values, real, integer, and ASCII data elements are included. If it is necessary to keep the Data Field Width (number of bits per data element) as small as possible yet accommodate the full range of data values, the software

breaks up the original data arrays into smaller arrays in which the ratio of the largest data element to the smallest data element is less than 1000. The smaller data arrays can be rescaled and re-referenced differently from the original data array. Selecting 1000 for the ratio represents a reasonable guess at its best value to achieve a minimum total BUFR message length over many different BLIRB8 output files. The larger the ratio, the longer the Section 4 component of the message. The smaller the ratio, the longer the Section 3 component of the message. Replication, rescaling, and re-referencing are used extensively to shorten the Section 4 length.

9. BLIRB_DE

The BUFR binary stream contains pertinent information about the BLIRB8 radiant flux field output file, except the ASCII output file formatting specification. Except for this shortcoming, a universal BUFR decoder could be written and applied to all data. BLIRB_DE was developed to specifically decode a BUFR encoded BLIRB8 radiant flux field output binary stream and create an ASCII file containing the data formatted as the original data. The approach used to develop the BLIRB_DE source code, BLIRB_DE.C, was to first decode Sections 0 and 1 of the BUFR stream, then fully expand Section 3 into its simplest descriptors, and last decode the Section 4 data values using the expanded version of the Section 3 data descriptors. The decoded data values are written to an ASCII file with the proper format.

The expansion of the BUFR Section 3 data is performed using a multipass process. Each pass entails expanding another layer of nesting and writing the intermediate results to a temporary file. The process terminates when there are no more nests, replications, or sequences to expand.

All writing to the formatted ASCII output file is performed in the procedure WRITE_FILE. The integer arrays CH_DESC, CH_LAST, IN_DESC, and OUT_DESC in the procedure contain the values of Y (Table B) minus the value of YYI or YYO (BUFR.H). These values represent the data descriptor offsets for particular data elements. CH_DESC contains the offsets for all sixteen ASCII data elements. Each of these ASCII data elements represent the beginning of a different type of BLIRB8 input record. CH_LAST contains the offsets of the last data element for each type of BLIRB8 input record. IN_DESC contains the offsets of the seven integer input data elements. OUT_DESC contains the offsets of the five integer output data elements.

Enter **blirb_de -i <infile> -o <outfile>** in which <infile> is the BLIRB8 BUFR encoded binary output filename and <outfile> is the BLIRB8 ASCII output filename to execute BLIRB_DE.

10. BLIRB_CM

BLIRB_CM was developed to compare the original BLIRB8 radiant flux field ASCII output file with the reconstituted version of the file after BUFR encoding with BLIRB_EN and decoding with BLIRB_DE. Because of a possible difference of one bit in the least significant position between the original real data element value before encoding and the real data element value after decoding, a UNIX diff or cmp comparison is generally not feasible. The comparison criterion for the data element values is based on the ratio of the absolute difference between the two data values to the absolute average of the two data values. If the data elements are BLIRB8 input data, they are considered the same value if the ratio is less than 0.001. If the data elements are BLIRB8 radiant flux field output data, they are considered the same value if the ratio is less than 0.0001.

Enter **blirb_cm <file1> <file2>** in which <file1> is the original BLIRB8 radiant flux field ASCII output filename or the filename of the reconstituted version of the same file and <file2> is the other filename to execute BLIRB_CM.

References

1. Zardecki, A., and R. Davis, STC Technical Report 6211 *Boundary Layer Illumination Radiation Balance Model: BLIRB*, April 1991.

2. Zardecki, A., *Three-Dimensional Extension of Boundary Layer Illumination Radiation Balance Model for Imaging Applications*, Contract TCN 92-480, Delivery Order 541 Final Report, U.S Army Research Office, January 1994.

3. Thorpe, W., *A Guide to the WMO Code Form FM 94-IX Ext. BUFR*, Fleet Numerical Oceanography Center, Monterey, CA, date unknown.

Acronyms and Abbreviations

AFGL	Air Force Geophysical Laboratory
BLIRB	Boundary Layer Illumination Radiation Balance Model
BUFR	Binary Universal Form for the Representation of meteorological data
EOSAEL	Electro-Optical Systems Atmospheric Effects Library
GUI	graphical user interface
LOCC	Local Originating Center Code
MODEL	model identifier
RH	relative humidity
SGI	Silicon Graphics Inc.
3-D	three-dimensional

Appendix A

Listing of VISUAL0.H, the Include File for VISUAL.C

```
/*=====================================================================
 * --- Definitions used in the BLIRB Visualization Program
 *=====================================================================
 */
#define RELEASED    0
#define PRESSED     1
#define WINDOW_HEIGHT 10.0                  /* Relative Window Hgt (in) */
#define RLEN        90                      /* Input/Output Record Len  */

                        /* Max Number of Input Cards of a given type. */
#define IAM     6                           /* Max number Areas         */
#define IRM     12                          /* Max number Regions       */
#define NEST    10                          /* Max number of Flares     */
#define ISM     8                           /* Max number Grid Mesh     */
#define IDM     6                           /* Max num Surface Albedos  */
#define ITM     10                          /* Max num Mtrl Composites  */
#define MXMTR   3                           /* Max Material per Region  */
#define ISCT    6                           /* Max number Spherical Harm*/

                        /* Max Output Array Dimensions                 */
#define NV      47                          /* Max number Wave Number   */
#define NA      5                           /* Num of Atmospheric Layrs */
#define ITN     (ITM+1)*NA                  /* Num misc. coefficients  */
#define ITN1    ITN+1                       /* ITN + 1                  */
#define MAXMX   40                          /* Max number X grid points */
#define MAXMY   40                          /* Max number Y grid points */
#define MAXMZ   40                          /* Max number Z grid points */
#define MAX_LEN_X 10                        /* Max X BLIRB Box (km)     */
#define MAX_LEN_Y 10                        /* Max Y BLIRB Box (km)     */
#define MAX_LEN_Z 12                        /* Max Z BLIRB Box (km)     */

#define MAX(X,Y) ((X) > (Y) ? (X) : (Y))
#define MAXXYZ MAX(MAXMX,MAX(MAXMY,MAXMZ)) /* Max (MAXMX,MAXMY,MAXMZ) */

/*=====================================================================
 * --- Structure and Function Pointer Prototypes
 *=====================================================================
 */
typedef struct _mov {
        float magfactor;                    /* Plot Magnification Factor */
      int ndx;                            /* Left Mouse Posn in X dir */
      int ndy;                            /* Left Mouse Posn in Y dir */
      int tdx;                            /* Rgt Mouse Posn in X dir  */
      int tdy;                            /* Rgt Mouse Posn in Y dir  */
} MOVEPLOT, *PLOTPTR;

typedef void (*FctnPointer);

/*=====================================================================
 * --- Procedure (function) Prototypes
 *=====================================================================
 */
void main(int argc, char **argv);

/*---------------------------------------------------------------------
 * --- Widget creation functions
 *---------------------------------------------------------------------
 */
Widget create_menubar(Widget w);
```

```
void create_filemenu(Widget w);
void create_viewmenu(Widget w);
void create_outputsmenu(Widget w);
void create_modifymenu(Widget w);
void create_helpmenu(Widget w, Widget helpcas);
void create_axismenu(Widget w);
void create_sunmenu(Widget w);
void create_zoommenu(Widget w);
void create_fluxmenu(Widget w);
void create_csectmenu(Widget w);
void create_cvaluemenu(Widget w);
void create_wavemenu(Widget w);
void create_areamenu(Widget w);
void create_area1menu(Widget w, int *n);
void create_regionmenu(Widget w);
void create_region1menu(Widget w, int *n);
void create_flarmenu(Widget w);
void create_flar1menu(Widget w, int *n);
void create_srchmenu(Widget w);
void create_meshmenu(Widget w);
void create_spectmenu(Widget w);
void create_message(Widget w, char *string[], unsigned char dialogtype);
void create_messagef(Widget w, char *string[],unsigned char dialogtype);
Widget create_rowcol(Widget w, char *label, FctnPointer cbfctn);
Widget create_bboard(Widget w, char *label);
Widget create_togglebutton(Widget w, char *tog_label, int *index,
                           FctnPointer cbfctn);
Widget create_radiobox(Widget w, int *nc, char *cat_label);
Widget create_scale(Widget w, char *scale_label, int *wid, int *min,
                    int *max, int *inc, int *dec, int *val, int *swid,
                    int *index, FctnPointer cbfctn);
void create_separator(Widget w, int *hv, int *sd);
void create_cascadebutton(Widget w, Widget pane, char label[], int nm);
void create_pushbuttons(Widget w, int nct, int index[], char *label[],
                        int nm[], FctnPointer cbfctn);
void create_pushbuttonss(Widget w, int *index, char *label, int nm,
                         FctnPointer cbfctn);
void create_pushbuttonfn(Widget w, char *fctn_key, char *fc_key,
                         char *button_label, int nm_key,
                         FctnPointer cbfctn);
void create_buttonsf(Widget w, int nct, int index[], char *fctn_key[],
                     char *fc_key[], char *button_label[],
                     int nm_key[], FctnPointer cbfctn);

/*-----------------------------------------------------------------------
 * --- Data input, output, check, and miscellaneous functions
 *-----------------------------------------------------------------------
 */
void newfile (void);
void fileopen (void);
void blirb_inout(void);
void getdata(void);
void readcards(void);
void checkinputs(void);
void readoutput(FILE *fp);
void set_albedo(void);
void set_aerosol(void);
void set_mtrl_color(void);
void order_mtrl(void);
void order_albd(void);
void set_axis_pts(void);
void writecards(void);
```

```
void reset(void);
void model2(void);
void flar2(int ind);
void albedo_chg(void);
void obsc(void);
void plot_out_def1(void);
void area_fix(void);
void regn_fix(void);
void regn_mtl1(void);
void aero0(int *n);
void aero1(void);
void aero2(void);
void aero3(void);
void albe0(int *n);
void albe1(int *n);
void albe2(void);
void albe3(void);
void mtrl0(int *n);
void mtrl1(int *n);
void mtrl2(int *n);
void mtrl3(int *n);
void mtrl4(int *n);
void mtrl5(int *n);
void mtrl6(int *n);
void mtrl7(int *n);
void mtrl8(int *n);
void mtrl9(int *n);

XmString xstr2xmstr(char *chararray[], int n);

/*-----------------------------------------------------------------------
 * --- GL window plot related functions
 *-----------------------------------------------------------------------
 */
void drawscene(void);

void dist_sun(void);
void plot_sun(void);

void plot_areas(void);

void plot_axes(void);
void plot_xaxis(void);
void plot_yaxis(void);
void plot_zaxis(void);

void plot_regions(void);
void regn_sides(float regn_vertex[8][3]);
void bottom_face(float regn_vertex[8][3]);
void left_face(float regn_vertex[8][3]);
void right_face(float regn_vertex[8][3]);
void front_face(float regn_vertex[8][3]);
void back_face(float regn_vertex[8][3]);
void top_face(float regn_vertex[8][3]);

void plot_flars(void);
void plot_slite(void);

void plot_flux(void);
void plot_flux_axis(int out_indx[3], int cur_imx);
void plot_flux_tran(int out_indx[3], int cur_imx, int which_flux);
void plot_flux_base(int out_indx[3], int cur_imx, int which_flux);
```

```
/*-----------------------------------------------------------------------
 * --- Callback functions
 *-----------------------------------------------------------------------
 */
void okCB (Widget w, XtPointer c, XtPointer call_data);
void okfCB (Widget w, XtPointer c, XtPointer call_data);
void cancelCB (Widget w, XtPointer c, XtPointer call_data);
void cancelobCB (Widget w, XtPointer c, XtPointer call_data);
void canceloCB (Widget w, XtPointer c, XtPointer call_data);
void cancelsCB (Widget w, XtPointer c, XtPointer call_data);
void cancelaCB (Widget w, XtPointer c, XtPointer call_data);
void cancelbCB (Widget w, XtPointer c, XtPointer call_data);
void cancelbbCB (Widget w, XtPointer c, XtPointer call_data);
void cancelrCB (Widget w, XtPointer c, XtPointer call_data);
void cancelmCB (Widget w, XtPointer c, XtPointer call_data);
void cancelmeCB (Widget w, XtPointer c, XtPointer call_data);
void cancelfCB (Widget w, XtPointer c, XtPointer call_data);
void cancelslCB (Widget w, XtPointer c, XtPointer call_data);
void helpCB (Widget w, XtPointer c, XtPointer call_data);
void exitCB (Widget w, XtPointer c, XtPointer call_data);
void initCB (Widget w, XtPointer c, XtPointer call_data);
void exposeCB (Widget w, XtPointer c, XtPointer call_data);
void resizeCB(Widget w, XtPointer c, XtPointer call_data);
void inputCB (Widget w, XtPointer c, XtPointer call_data);
void resetCB (Widget w, XtPointer c, XtPointer call_data);
void fileCB (Widget w, XtPointer c, XtPointer call_data);
void checkfiletypeCB (Widget w, XtPointer c, XtPointer call_data);
void newfCB (Widget w, XtPointer c, XtPointer call_data);
void savefileCB (Widget w, XtPointer c, XtPointer call_data);
void savefileasCB (Widget w, XtPointer c, XtPointer call_data);
void getfilenameCB (Widget w, XtPointer c, XtPointer call_data);
void zoomCB (Widget w, XtPointer c, XtPointer call_data);
void minor_gridCB (Widget w, XtPointer c, XtPointer call_data);
void axisCB (Widget w, XtPointer c, XtPointer call_data);
void fluxCB (Widget w, XtPointer c, XtPointer call_data);
void cross_sectionCB (Widget w, XtPointer c, XtPointer call_data);
void transCB (Widget w, XtPointer c, XtPointer call_data);
void sun_posCB (Widget w, XtPointer c, XtPointer call_data);
void sunCB (Widget w, XtPointer c, XtPointer call_data);
void obscCB (Widget w, XtPointer c, XtPointer call_data);
void planeCB (Widget w, XtPointer c, XtPointer call_data);
void waveCB (Widget w, XtPointer c, XtPointer call_data);
void model_optCB(Widget w, XtPointer c, XtPointer call_data);
void modelCB(Widget w, XtPointer c, XtPointer call_data);
void model3CB(Widget w, XtPointer c, XtPointer call_data);
void cloud_optCB(Widget w, XtPointer c, XtPointer call_data);
void cloudCB(Widget w, XtPointer c, XtPointer call_data);
void aerosol_optCB(Widget w, XtPointer c, XtPointer call_data);
void aerosolCB(Widget w, XtPointer c, XtPointer call_data);
void aeros0CB(Widget w, XtPointer c, XtPointer call_data);
void metrng_optCB(Widget w, XtPointer c, XtPointer call_data);
void metrngCB(Widget w, XtPointer c, XtPointer call_data);
void area_optCB(Widget w, XtPointer c, XtPointer call_data);
void areaCB(Widget w, XtPointer c, XtPointer call_data);
void area_locCB(Widget w, XtPointer c, XtPointer call_data);
void area_delCB(Widget w, XtPointer c, XtPointer call_data);
void area_albCB(Widget w, XtPointer c, XtPointer call_data);
void albedo1CB(Widget w, XtPointer c, XtPointer call_data);
void albed1CB(Widget w, XtPointer c, XtPointer call_data);
void albedo2CB(Widget w, XtPointer c, XtPointer call_data);
void albed2CB(Widget w, XtPointer c, XtPointer call_data);
void albe0CB(Widget w, XtPointer c, XtPointer call_data);
```

```
void albedo3CB(Widget w, XtPointer c, XtPointer call_data);
void albed3CB(Widget w, XtPointer c, XtPointer call_data);
void regn_optCB(Widget w, XtPointer c, XtPointer call_data);
void regnCB(Widget w, XtPointer c, XtPointer call_data);
void regn_locCB(Widget w, XtPointer c, XtPointer call_data);
void regn_delCB(Widget w, XtPointer c, XtPointer call_data);
void regn_mtlCB(Widget w, XtPointer c, XtPointer call_data);
void mtrlCB(Widget w, XtPointer c, XtPointer call_data);
void mtrl0CB(Widget w, XtPointer c, XtPointer call_data);
void mtrl1CB(Widget w, XtPointer c, XtPointer call_data);
void denmtlCB(Widget w, XtPointer c, XtPointer call_data);
void meshxmenuCB(Widget w, XtPointer c, XtPointer call_data);
void meshymenuCB(Widget w, XtPointer c, XtPointer call_data);
void meshzmenuCB(Widget w, XtPointer c, XtPointer call_data);
void meshCB(Widget w, XtPointer c, XtPointer call_data);
void flar_optCB(Widget w, XtPointer c, XtPointer call_data);
void flarinCB(Widget w, XtPointer c, XtPointer call_data);
void flar3CB(Widget w, XtPointer c, XtPointer call_data);
void flar_locCB(Widget w, XtPointer c, XtPointer call_data);
void flar_delCB(Widget w, XtPointer c, XtPointer call_data);
void srch_optCB(Widget w, XtPointer c, XtPointer call_data);
void srchinCB(Widget w, XtPointer c, XtPointer call_data);
void srch_locCB(Widget w, XtPointer c, XtPointer call_data);
void srch_delCB(Widget w, XtPointer c, XtPointer call_data);
void sun_optCB(Widget w, XtPointer c, XtPointer call_data);
void suninCB(Widget w, XtPointer c, XtPointer call_data);
void spect_optCB(Widget w, XtPointer c, XtPointer call_data);
void spectCB(Widget w, XtPointer c, XtPointer call_data);
void spect1menuCB(Widget w, XtPointer c, XtPointer call_data);
void spect1CB(Widget w, XtPointer c, XtPointer call_data);
void comp_optCB(Widget w, XtPointer c, XtPointer call_data);
void compCB(Widget w, XtPointer c, XtPointer call_data);
void output_optCB(Widget w, XtPointer c, XtPointer call_data);
void outputCB(Widget w, XtPointer c, XtPointer call_data);
void help_generalCB(Widget w, XtPointer c, XtPointer call_data);
void help_fileCB(Widget w, XtPointer c, XtPointer call_data);
void help_viewCB(Widget w, XtPointer c, XtPointer call_data);
void help_fluxCB(Widget w, XtPointer c, XtPointer call_data);
void help_modifyCB(Widget w, XtPointer c, XtPointer call_data);
void help_resetCB(Widget w, XtPointer c, XtPointer call_data);
void help_rotationCB(Widget w, XtPointer c, XtPointer call_data);
void help_translationCB(Widget w, XtPointer c, XtPointer call_data);
void help_axisCB(Widget w, XtPointer c, XtPointer call_data);
void help_sunoCB(Widget w, XtPointer c, XtPointer call_data);
void help_zoomCB(Widget w, XtPointer c, XtPointer call_data);
void help_togCB(Widget w, XtPointer c, XtPointer call_data);
void help_optCB(Widget w, XtPointer c, XtPointer call_data);
void help_orCB(Widget w, XtPointer c, XtPointer call_data);
void help_valCB(Widget w, XtPointer c, XtPointer call_data);
void help_waveCB(Widget w, XtPointer c, XtPointer call_data);
void help_modCB(Widget w, XtPointer c, XtPointer call_data);
void help_regCB(Widget w, XtPointer c, XtPointer call_data);
void help_areaCB(Widget w, XtPointer c, XtPointer call_data);
void help_meshCB(Widget w, XtPointer c, XtPointer call_data);
void help_cldCB(Widget w, XtPointer c, XtPointer call_data);
void help_sunCB(Widget w, XtPointer c, XtPointer call_data);
void help_flareCB(Widget w, XtPointer c, XtPointer call_data);
void help_sliteCB(Widget w, XtPointer c, XtPointer call_data);
void help_spectCB(Widget w, XtPointer c, XtPointer call_data);
void help_compCB(Widget w, XtPointer c, XtPointer call_data);
void help_outCB(Widget w, XtPointer c, XtPointer call_data);
```

Appendix B

Listing of BUFR.H

```
/*=====================================================================
 * --- Definitions
 *=====================================================================
 */
#define MISSING 999999                    /* missing value indicator  */
#define RLEN    190                       /* Input Record Length bytes*/
#define SEQ_SIZE 1200                     /* Size of sequence array   */

                         /* ARL-WSMR Specific Values for debugging     */
#define LOCC    111                       /* Local Orig Center Code   */
#define MODEL   1                         /* Model Identifier         */
#define DBSN    0                         /* Database Sequence Number */

#define BB      55                        /* BUFR "x0" from Table B   */
#define YYI     1                         /* init y0 for BLIRB inputs */
#define YYO     120                       /* init y0 for BLIRB outputs*/
#define YYM     147                       /* maximum y0 for BLIRB data*/

#define DD      25                        /* BUFR "x0" from Table D   */

/*=====================================================================
 * --- Structure Prototypes
 *=====================================================================
 */
/*---------------------------------------------------------------------
 * --- Identification section
 *---------------------------------------------------------------------
 */
struct identif
   {
   long length;        /* length of section                          */
   int master_tbl;     /* master table (zero if standard WMO FM 94-IX) */
   unsigned orig_cntr; /* originating centre Code table 0 01 031     */
   int updt_seq;       /* update sequence number (0 for original BUFR) */
   int opt_sec;        /* optional section 0 missing, 1 exists       */
   int msg_type;       /* BUFR message type TABLE A                  */
   int msg_subtype;    /* BUFR message type defined by local ADP     */
   int mstr_vrsn;      /* master table version (currently 2)         */
   int locl_vrsn;      /* local table version                        */
   int year;           /* year of century                            */
   int month;          /* month                                      */
   int day;            /* day                                        */
   int hour;           /* hour                                       */
   int minute;         /* minute                                     */
   int locc;           /* local originating center code              */
   int model;          /* model identifier                           */
   int second;         /* second                                     */
   unsigned dbsn;      /* database sequence number                   */
   };

/*---------------------------------------------------------------------
 * --- Structure table_b holds the information about element
 *     descriptors, their scale value, reference value, # of bits and
 *     units.
 *---------------------------------------------------------------------
 */
struct table_b
   {
   long descriptor;    /* element descriptor                         */
```

```
   int scale;          /* scaling factor                                  */
   long ref_val;       /* reference value                                 */
   int data_width;     /* data width                                      */
   char units[32];     /* units                                           */
   };

/*-----------------------------------------------------------------------
 * --- Structure table_d holds information about sequences, number of
 *     descriptors that are associated to a sequence, and the index of
 *     the first descriptor in the seq_desc array.
 *-----------------------------------------------------------------------
 */
struct table_d
   {
   long sequence;      /* sequence descriptor                             */
   long seq_index;     /* pointer to where the first descriptor
                          associated to this sequence is stored in
                          seq_desc array                                  */
   long seq_expand;    /* number of descriptors associated to this
                          sequence                                        */
   };

/*-----------------------------------------------------------------------
 * --- Structure to hold the operands modifications
 *-----------------------------------------------------------------------
 */
struct operation
   {
   int dt_width_op;    /* value for change data width                     */
   int scale_op;       /* value for change scale                          */
   long ref_val_op;    /* value for change reference value                */
   long assoc_fld;     /* the significance of associated field
                          (0 31 021)                                      */
   long assoc_width;   /* the size of the associated field                */
   };

/*=======================================================================
 * --- Procedure (function) Prototypes
 *=======================================================================
 */
void main(int argc, char **argv);
void ParseInput(int argc, char **argv);
void read_table_b(void);
void read_table_d(void);
void section_0(void);
void section_1(void);
void write_data(long out, char *tempx, int size_byte);
void readcards(void);
void out_data(char c1, char c2, char c3, char c4, int nct, float x[10],
              unsigned long *n4, int desc);
void putbits(unsigned long *n, long input, int *bit, int inbits,
             FILE *sx, long *out_len);
void readoutput(unsigned long *n4, unsigned long *n3);
void get_trans(int y0, int *scale, int *width, float *ref);
void group_bytes(unsigned long *n4, unsigned long *n3, float *data,
                 long nct, int scale, int nochg, int width, float ref,
                 int param);
void write_bytes(unsigned long *n4, unsigned long *n3, float *data,
                 long nct, int scale, int nochg, int width, float min1,
                 float max1, long zero, float ref, int param);
void factor(long nct, long *nct1, long *nct2, long *nct3, long *nct4);
void gbyte(unsigned long *iout, int nbyte);
```

```
void bufridenti(struct identif *id);
void bufrgetind(int file, int *fin, unsigned long *status);
void get_data(void);
void bufruncomp(struct operation *oper);
void write_file(char *string, long index);
```

Appendix C

Listing of BLIRB_EN.C

```
#include <string.h>
#include <stdio.h>
#include <math.h>
#include <fcntl.h>
#include <ctype.h>
#include <time.h>
#include <stdlib.h>
#include "bufr.h"                        /* Prototypes & Definitions      */

/*-----------------------------------------------------------------------
 * --- Array declarations and assignments
 *-----------------------------------------------------------------------
 */
                                         /* Main Structures               */
struct table_b *tbl_b;

                                         /* Filename and I/O Variables    */
char *file_name;                             /* BLIRB Data Filename       */
FILE *fp;                                    /* Pointer to Input File     */
FILE *fpo;                                   /* Pointer to Output File    */
FILE *fpo0;                                  /* Pointer to Sect 0 tempfil*/
FILE *fpo1;                                  /* Pointer to Sect 1 tempfil*/
FILE *fpo3;                                  /* Pointer to Sect 3 tempfil*/
FILE *fpo4;                                  /* Pointer to Sect 4 tempfil*/
static char *temp0 = "temp0.tmp";            /* Tempfile 0 name           */
static char *temp1 = "temp1.tmp";            /* Tempfile 1 name           */
static char *temp3 = "temp3.tmp";            /* Tempfile 2 name           */
static char *temp4 = "temp4.tmp";            /* Tempfile 3 name           */
long out0, out1, out3, out4;                 /* Num bytes in Sect 0 - 4   */
int bit0, bit1, bit3, bit4;                  /* Bits counts in Sect 0 - 4*/
long out_length;                             /* Num bytes in output file */
int counter;                                 /* Num records in Table B    */

                                         /* BLIRB Input/Output Variables  */
int area, regn, mesx, mesy, mesz, albd, mtrl, recl; /* Input card cnt */
float mdl1_model;                            /* The BLIRB Temp model      */
float domd_isc;                              /* Order Spherical Harmonic */
float wavn_v1, wavn_v2, wavn_dv;             /* Wavenumber info           */
int out_imx[3];                              /* Num of X, Y, & Z grid pt */
int out_nwave;                               /* Number of Waves           */

/**********************************************************************
 *                          VOID MAIN
 **********************************************************************
 *<Begin>
 *<Identification>          Name:  main
 *                          Type:  C Main Program
 *                      Filename:  blirb_en.c
 *                        Parent:  None
 *=====================================================================
 *<Description>
 *    This program encodes a BLIRB ASCII output file into a binary
 *    output file using the BUFR encoding system.
 *=====================================================================
 *<Called routines>
 *    ParseInput            - parses the command line inputs
 *    read_table_b          - reads the structure records from Table_B
 *    section_0             - generates section 0 BUFR temporary file
 *    section_1             - generates section 1 BUFR temporary file
```

```
 *    readcards              - reads the BLIRB input records and writes
 *                             them to a binary output file.
 *    write_data             - reads the BUFR data from a temporary file
 *                             and writes it to final composite file
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void main(int argc, char **argv)
 *    Input:
 *       argc                - (int) the number of command lines inputs
 *       argv                - (char pointer) the array of command lines
 *                             inputs
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    02/01/95  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void main(int argc, char **argv)
{
  int loop;

  tbl_b = NULL;
  read_table_b();
  if(tbl_b == NULL)
  { printf("Could not get the records from Table_B.\n");
    printf("Program Aborting!\n");
    exit(0);
  }

  ParseInput(argc, argv);

  out_length = (long)0;

  section_0();
  section_1();
  readcards();

  out_length += out3;
  out_length += out4;
  out_length += (long)4;

  free(tbl_b);

  write_data(out_length, temp0, 4);
  write_data(out1, temp1, 0);
  write_data(out3, temp3, 0);
  write_data(out4, temp4, 0);

  for(loop = 0; loop < 4; loop++)
    fputc((char)7, fpo);
  fclose(fpo);

  printf("\nSection 0 had %8ld bytes\n", out0);
  printf("Section 1 had %8ld bytes\n", out1);
  printf("Section 3 had %8ld bytes\n", out3);
  printf("Section 4 had %8ld bytes\n", out4);
  printf("Section 5 had        4 bytes\n");
```

```
  printf("There were     %8ld bytes total\n", out_length);
}       /* end main() */

/*********************************************************************
 *                            VOID PARSEINPUT
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  ParseInput
 *                          Type:  C void
 *                      Filename:  blirb_en.c
 *                        Parent:  main
 *====================================================================
 *<Description>
 *    Parses the command line inputs.
 *====================================================================
 *<Called routines>
 *    None.
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void ParseInput( int argc, char **argv)
 *    Input:
 *       argc                - (int) the number of command lines inputs
 *       argv                - (char pointer) the array of command lines
 *                             inputs
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    02/02/94  CSC Monterey, CA   Mugur Georgescu
 *              Probably developed the original source code
 *    05/31/94  PL Hanscom AFB, MA   Rodger Biasca
 *              Probably modified the original source code
 *    02/01/95  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Modified the source code again
 *====================================================================
 *<End>
 *********************************************************************
 */
void ParseInput(int argc, char **argv)
{
  int i, status;

  status = 0;
  for (i=1; i<argc; i++)
  { if(argv[i][0] == '-')
    { switch(argv[i][1])
      { case 'o':
          fpo = fopen(argv[i+1], "wb");
          fpo0 = fopen(temp0, "wb");
          fpo1 = fopen(temp1, "wb");
          fpo3 = fopen(temp3, "wb");
          fpo4 = fopen(temp4, "wb");
          break;
        case 'i':
          file_name = argv[i+1];
          fp = fopen(file_name, "r");
        break;
        default:
          status = 1;
        break;
      }
```

```
    }
  }

  if(status == 1 || fpo == NULL || fp == NULL)
  { printf("Usage: bufr_enc -o outfile -i infile\n");
    exit(0);
  }
}       /* end ParseInput() */

/*********************************************************************
 *                          VOID SECTION_0
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  section_0
 *                          Type:  C void
 *                      Filename:  blirb_en.c
 *                        Parent:  main
 *====================================================================
 *<Description>
 *    Generates the BUFR section 0 temporary output file.
 *====================================================================
 *<Called routines>
 *    None.
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void section_0(void)
 *    Input:
 *       None
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    02/01/95  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void section_0(void)
{
  unsigned long n0;
  long nn;

  n0 = (unsigned long)0;
  out0 = (long)0;
  bit0 = 0;

  nn = ('B' << 8) + 'U';
  putbits(&n0, nn, &bit0, 16, fpo0, &out0);
  nn = ('F' << 8) + 'R';
  putbits(&n0, nn, &bit0, 16, fpo0, &out0);
  putbits(&n0, 0, &bit0, 24, fpo0, &out0);
  putbits(&n0, 2, &bit0, 8, fpo0, &out0);

  out_length += out0;
  fclose(fpo0);
}       /* end section_0() */

/*********************************************************************
 *                          VOID SECTION_1
 *********************************************************************
```

```
 *<Begin>
 *<Identification>          Name:  section_1
 *                          Type:  C void
 *                      Filename:  blirb_en.c
 *                        Parent:  main
 *=====================================================================
 *<Description>
 *    Generates the BUFR section 1 temporary output file.
 *=====================================================================
 *<Called routines>
 *    None.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void section_1(void)
 *    Input:
 *       None
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    02/01/95  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void section_1(void)
{
  time_t ltime;
  struct tm *times;
  unsigned long n1;
  char c1;
  int loop;

  n1 = (unsigned long)0;
  bit1 = 0;
  out1 = (long)0;
  putbits(&n1, 0, &bit1, 24, fpo1, &out1);
  putbits(&n1, 0, &bit1, 8, fpo1, &out1);
  putbits(&n1, 128, &bit1, 16, fpo1, &out1);
  putbits(&n1, 0, &bit1, 8, fpo1, &out1);
  putbits(&n1, 0, &bit1, 8, fpo1, &out1);
  putbits(&n1, 255, &bit1, 8, fpo1, &out1);
  putbits(&n1, 0, &bit1, 8, fpo1, &out1);
  putbits(&n1, 3, &bit1, 8, fpo1, &out1);
  putbits(&n1, 0, &bit1, 8, fpo1, &out1);

  ltime = time(NULL);
  times = localtime(&ltime);

  putbits(&n1, (long)(*times).tm_year, &bit1, 8, fpo1, &out1);
  putbits(&n1, (long)((*times).tm_mon+1), &bit1, 8, fpo1, &out1);
  putbits(&n1, (long)(*times).tm_mday, &bit1, 8, fpo1, &out1);
  putbits(&n1, (long)(*times).tm_hour, &bit1, 8, fpo1, &out1);
  putbits(&n1, (long)(*times).tm_min, &bit1, 8, fpo1, &out1);
  putbits(&n1, (long)LOCC, &bit1, 8, fpo1, &out1);
  putbits(&n1, (long)MODEL, &bit1, 8, fpo1, &out1);
  putbits(&n1, (long)(*times).tm_sec, &bit1, 8, fpo1, &out1);
  putbits(&n1, (long)DBSN, &bit1, 16, fpo1, &out1);

  if(bit1 > 0)
```

```
    { loop = 0;
      n1 <<= (32 - bit1);
      while (bit1 > 0)
      { c1 = (char)((n1 << (8 * loop)) >> 24);
        fputc(c1, fpo1);
        out1++;
        bit1 -= 8;
        loop++;
      }
    }
    if(out1 % (long)2 == (long)1)
    { fputc((char)0, fpo1);
      out1++;
    }

    out_length += out1;
    fclose(fpo1);
  }      /* end section_1() */

/*********************************************************************
 *                          VOID READ_TABLE_B
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  read_table_b
 *                          Type:  C void
 *                      Filename:  blirb_en.c
 *                        Parent:  main
 *====================================================================
 *<Description>
 *    Reads the BUFR descriptors from Table_B.
 *====================================================================
 *<Called routines>
 *    None.
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void read_table_b( void)
 *    Input:
 *       None.
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    02/02/94  CSC Monterey, CA   Mugur Georgescu
 *              Probably developed the original source code
 *    05/31/94  PL Hanscom AFB, MA   Rodger Biasca
 *              Probably modified the original source code
 *    02/01/95  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Modified the source code again
 *====================================================================
 *<End>
 *********************************************************************
 */
void read_table_b(void)
{
  FILE *ftb;
  long f0, x0, y0, scale, ref, width;
  int size, num;
  char card[RLEN], name[200], unit[16];

  size = 50;
  num = 0;
```

```
  if((ftb = fopen("table_b", "r")) == 0)
  { printf("Table_B file not found.  Program Aborting!\n");
    exit(0);
  }
  else
  { tbl_b = (struct table_b *) malloc(sizeof(struct table_b) * size);

    while(!feof(ftb))
    { fgets(card, RLEN, ftb);
      sscanf(card, "%ld%ld%ld%ld%ld%ld%s%s", &f0, &x0, &y0,
             &scale, &ref, &width, unit, name);

      tbl_b[num].descriptor = f0*(long)100000 + x0*(long)1000 + y0;
      tbl_b[num].scale = (int)scale;
      tbl_b[num].ref_val = ref;
      tbl_b[num].data_width = (int)width;
      strcpy(tbl_b[num].units, unit);

      num++;
      if(num >= size)
      { size += 50;
        tbl_b = (struct table_b *) realloc(tbl_b,
                 sizeof(struct table_b) * size);
      }
    }

    counter = num - 1;
    fclose(ftb);
  }
}      /* end read_table_b() */

/*********************************************************************
 *                          VOID WRITE_DATA
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  write_data
 *                          Type:  C void
 *                      Filename:  blirb_en.c
 *                        Parent:  main
 *====================================================================
 *<Description>
 *    Reads the BUFR data from a temporary file and writes it to the
 *    final composite BUFR output file.
 *====================================================================
 *<Called routines>
 *    None.
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void write_data(long out, char *tempx, int size_byte)
 *    Input:
 *       out                - the number of bytes in this section
 *       *tempx             - Name of the temporary output file
 *       size_byte          - first byte to hold the section size
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    02/01/95  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
```

```
 ***********************************************************************
 */
void write_data(long out, char *tempx, int size_byte)
{
  static int iter = 0;
  long nn, cnt;
  char c1;
  FILE *fpox;

  iter++;
  if(iter == 1)
    cnt = (long)8;
  else
    cnt = out;

  fpox = fopen(tempx, "rb");

  for (nn = (long)0; nn < cnt; nn++)
  { c1 = fgetc(fpox);

    if(nn == size_byte)
    { c1 = (char) ((out << 8) >> 24);
      fputc(c1, fpo);
    }
    else if(nn == size_byte + 1)
    { c1 = (char) ((out << 16) >> 24);
      fputc(c1, fpo);
    }
    else if(nn == size_byte + 2)
    { c1 = (char) ((out << 24) >> 24);
      fputc(c1, fpo);
    }
    else
      fputc(c1, fpo);
  }

  fclose(fpox);
  remove(tempx);
}      /* end write_data() */

/***********************************************************************
 *                          VOID READCARDS
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  readcards
 *                          Type:  C void
 *                      Filename:  blirb_en.c
 *                        Parent:  main
 *=====================================================================
 *<Description>
 *    Reads the BLIRB input records from the BLIRB output file and
 *    encodes the records into the BUFR binary output file.
 *=====================================================================
 *<Called routines>
 *    readoutput            - reads BLIRB output from the output file
 *                            and encodes the data in the BUFR format
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void readcards( void)
 *    Input:
 *       None
```

```
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    02/01/95  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void readcards(void)
{
  char cardlabel[5], card[RLEN], c[5];
  unsigned long n3, n4;
  float x[10], ref;
  long nn, y[10];
  int j, w[10], sca;
  int loop;

/*---------------------------------------------------------------------
 * --- Initialize the Input Card Type Counters
 *---------------------------------------------------------------------
 */
  n4 = (unsigned long)0;
  out4 = (long)0;
  bit4 = 0;
  putbits(&n4, 0, &bit4, 32, fpo4, &out4); /* Number bytes in Sect 4  */

  n3 = (unsigned long)0;
  out3 = (long)0;
  bit3 = 0;
  putbits(&n3, 0, &bit3, 32, fpo3, &out3); /* Number bytes in Sect 3  */
  putbits(&n3, 1, &bit3, 16, fpo3, &out3); /* data subset = 1         */
  putbits(&n3, 0, &bit3, 8, fpo3, &out3);  /* "other" data & no comp. */

  area = regn = mesx = mesy = mesz = albd = mtrl = recl = -1;

  do                                       /* From VIEW.RJOB Subroutine*/
/*---------------------------------------------------------------------
 * --- Read a BLIRB input card from the file.
 *---------------------------------------------------------------------
 */
  { fgets(card, RLEN, fp);

/*---------------------------------------------------------------------
 * --- Get the first 5 characters (card identifier) from the card.
 *---------------------------------------------------------------------
 */
    sscanf(card, "%s", cardlabel);

/*---------------------------------------------------------------------
 * --- Process the rest of the card depending upon the identifier.
 *---------------------------------------------------------------------
 */
    if(strncmp(cardlabel,"MDL1",4) == 0)
    { sscanf(card, "%s%10e%10e%10e%10e%10e", c, &x[0], &x[1], &x[2],
             &x[3], &x[4]);
      mdl1_model = x[1];
      out_data('M', 'D', 'L', '1', 5, x, &n4, YYI + 1);
    }

    else if(strncmp(cardlabel,"MDL2",4) == 0)
```

```
{ sscanf(card, "%s%10e%10e%10e%10e", c, &x[0], &x[1], &x[2], &x[3]);
  out_data('M', 'D', 'L', '2', 4, x, &n4, YYI + 7);

  if((int)mdl1_model != 7)
  { fgets(card, RLEN, fp);
    sscanf(card, "%ld%ld%ld%ld%ld%ld%ld", &y[0], &y[1], &y[2],
           &y[3], &y[4], &y[5], &y[6]);

    for (j = 0; j < 7; j++)
    { get_trans(j + YYI + 18, &sca, &w[j], &ref);
      if(sca != 0)
        y[j] *= (long) pow((double)10, (double)sca);
      y[j] -= (long )ref;
      putbits(&n4, y[j], &bit4, w[j], fpo4, &out4);
    }
  }
}

else if(strncmp(cardlabel,"MDL3",4) == 0)
{ sscanf(card, "%s%10e%10e%10e%10e%10e%10e", c, &x[0], &x[1], &x[2],
         &x[3], &x[4], &x[5]);
  out_data('M', 'D', 'L', '3', 6, x, &n4, YYI + 12);

  fgets(card, RLEN, fp);
  sscanf(card, "%ld%ld%ld%ld%ld%ld%ld", &y[0], &y[1], &y[2], &y[3],
         &y[4], &y[5], &y[6]);

  for (j = 0; j < 7; j++)
  { get_trans(j + YYI + 18, &sca, &w[j], &ref);
    if(sca != 0)
      y[j] *= (long) pow((double)10, (double)sca);
    y[j] -= (long) ref;
    putbits(&n4, y[j], &bit4, w[j], fpo4, &out4);
  }
}

else if(strncmp(cardlabel,"AREA",4) == 0)
{ area++;                                /* AREA card                    */
  sscanf(card, "%s%10e%10e%10e%10e%10e", c, &x[0], &x[1], &x[2],
         &x[3], &x[4]);
  out_data('A', 'R', 'E', 'A', 5, x, &n4, YYI + 26);
}

else if(strncmp(cardlabel,"REGN",4) == 0)
{ regn++;                                /* REGN card                    */
  sscanf(card, "%s%10e%10e%10e%10e%10e%10e%10e", c, &x[0], &x[1],
         &x[2], &x[3], &x[4], &x[5], &x[6]);
  out_data('R', 'E', 'G', 'N', 7, x, &n4, YYI + 32);
}

else if(strncmp(cardlabel,"MESX",4) == 0)
{ mesx++;                                /* MESX card                    */
  sscanf(card, "%s%10e%10e", c, &x[0], &x[1]);
  out_data('M', 'E', 'S', 'X', 2, x, &n4, YYI + 40);
}

else if(strncmp(cardlabel,"MESY",4) == 0)
{ mesy++;                                /* MESY card                    */
  sscanf(card, "%s%10e%10e", c, &x[0], &x[1]);
  out_data('M', 'E', 'S', 'Y', 2, x, &n4, YYI + 43);
}
```

```
    else if(strncmp(cardlabel,"MESZ",4) == 0)
    { mesz++;                                /* MESZ card                  */
      sscanf(card, "%s%10e%10e", c, &x[0], &x[1]);
      out_data('M', 'E', 'S', 'Z', 2, x, &n4, YYI + 46);
    }

    else if(strncmp(cardlabel,"ALBD",4) == 0)
    { albd++;                                /* ALBD card                  */
      sscanf(card, "%s%10e%10e%10e", c, &x[0], &x[1], &x[2]);
      out_data('A', 'L', 'B', 'D', 3, x, &n4, YYI + 49);
    }

    else if(strncmp(cardlabel,"MTRL",4) == 0)
    { mtrl++;                                /* MTRL card                  */
      sscanf(card,"%s%10e%10e%10e%10e%10e", c, &x[0], &x[1],
             &x[2], &x[3], &x[4], &x[5]);
      out_data('M', 'T', 'R', 'L', 6, x, &n4, YYI + 53);
    }

    else if(strncmp(cardlabel,"CLDS",4) == 0)
    { sscanf(card, "%s%10e%10e%10e", c, &x[0], &x[1], &x[2]);
      out_data('C', 'L', 'D', 'S', 3, x, &n4, YYI + 60);
    }

    else if(strncmp(cardlabel,"DOMD",4) == 0)
    { sscanf(card, "%s%10e%10e%10e%10e%10e", c, &x[0], &x[1], &x[2],
             &x[3], &x[4]);
      domd_isc = x[0];
      out_data('D', 'O', 'M', 'D', 5, x, &n4, YYI + 64);
    }

    else if(strncmp(cardlabel,"SUN",3) == 0)
    { sscanf(card, "%s%10e%10e%10e%10e%10e", c, &x[0], &x[1], &x[2],
             &x[3], &x[4]);
      out_data('S', 'U', 'N', ' ', 5, x, &n4, YYI + 70);
    }

    else if(strncmp(cardlabel,"WAVN",4) == 0)
    { sscanf(card, "%s%10e%10e%10e", c, &x[0], &x[1], &x[2]);
      x[2] = (x[1] - x[0]) / x[2];
      wavn_v1 = x[0];
      wavn_v2 = x[1];
      wavn_dv = x[2];
      out_data('W', 'A', 'V', 'N', 3, x, &n4, YYI + 76);
    }

    else if(strncmp(cardlabel,"ASCI",4) == 0)
    { sscanf(card, "%s%10e", c, &x[0]);
      out_data('A', 'S', 'C', 'I', 1, x, &n4, YYI + 80);
    }

    else if(strncmp(cardlabel,"RECL",4) == 0)
    { recl = 0;                              /* RECL card                  */
      sscanf(card, "%s%10e", c, &x[0]);
      out_data('R', 'E', 'C', 'L', 1, x, &n4, YYI + 82);
    }

    else
      printf(" Record ID %s not identified.\n", cardlabel); /* Unknown*/

  } while (recl != 0);
```

```
putbits(&n3, 3, &bit3, 2, fpo3, &out3);
putbits(&n3, (long)DD, &bit3, 6, fpo3, &out3);
if((int)mdl1_model != 7)
  putbits(&n3, 2, &bit3, 8, fpo3, &out3);
else
  putbits(&n3, 1, &bit3, 8, fpo3, &out3);

if(area > 0)
{ putbits(&n3, 1, &bit3, 2, fpo3, &out3);
  putbits(&n3, 1, &bit3, 6, fpo3, &out3);
  putbits(&n3, (long)(area + 1.0), &bit3, 8, fpo3, &out3);
}
putbits(&n3, 3, &bit3, 2, fpo3, &out3);
putbits(&n3, (long)DD, &bit3, 6, fpo3, &out3);
putbits(&n3, 3, &bit3, 8, fpo3, &out3);

if(regn > 0)
{ putbits(&n3, 1, &bit3, 2, fpo3, &out3);
  putbits(&n3, 1, &bit3, 6, fpo3, &out3);
  putbits(&n3, (long)(regn + 1.0), &bit3, 8, fpo3, &out3);
}
putbits(&n3, 3, &bit3, 2, fpo3, &out3);
putbits(&n3, (long)DD, &bit3, 6, fpo3, &out3);
putbits(&n3, 4, &bit3, 8, fpo3, &out3);

if(mesx > 0)
{ putbits(&n3, 1, &bit3, 2, fpo3, &out3);
  putbits(&n3, 1, &bit3, 6, fpo3, &out3);
  putbits(&n3, (long)(mesx + 1.0), &bit3, 8, fpo3, &out3);
}
putbits(&n3, 3, &bit3, 2, fpo3, &out3);
putbits(&n3, (long)DD, &bit3, 6, fpo3, &out3);
putbits(&n3, 5, &bit3, 8, fpo3, &out3);

if(mesy > 0)
{ putbits(&n3, 1, &bit3, 2, fpo3, &out3);
  putbits(&n3, 1, &bit3, 6, fpo3, &out3);
  putbits(&n3, (long)(mesy + 1.0), &bit3, 8, fpo3, &out3);
}
putbits(&n3, 3, &bit3, 2, fpo3, &out3);
putbits(&n3, (long)DD, &bit3, 6, fpo3, &out3);
putbits(&n3, 6, &bit3, 8, fpo3, &out3);

if(mesz > 0)
{ putbits(&n3, 1, &bit3, 2, fpo3, &out3);
  putbits(&n3, 1, &bit3, 6, fpo3, &out3);
  putbits(&n3, (long)(mesz + 1.0), &bit3, 8, fpo3, &out3);
}
putbits(&n3, 3, &bit3, 2, fpo3, &out3);
putbits(&n3, (long)DD, &bit3, 6, fpo3, &out3);
putbits(&n3, 7, &bit3, 8, fpo3, &out3);

if(albd > -1)
{ if(albd > 0)
  { putbits(&n3, 1, &bit3, 2, fpo3, &out3);
    putbits(&n3, 1, &bit3, 6, fpo3, &out3);
    putbits(&n3, (long)(albd + 1.0), &bit3, 8, fpo3, &out3);
  }
  putbits(&n3, 3, &bit3, 2, fpo3, &out3);
  putbits(&n3, (long)DD, &bit3, 6, fpo3, &out3);
  putbits(&n3, 8, &bit3, 8, fpo3, &out3);
}
```

```
  if(mtrl > 0)
  { putbits(&n3, 1, &bit3, 2, fpo3, &out3);
    putbits(&n3, 1, &bit3, 6, fpo3, &out3);
    putbits(&n3, (long)(mtrl + 1.0), &bit3, 8, fpo3, &out3);
  }
  putbits(&n3, 3, &bit3, 2, fpo3, &out3);
  putbits(&n3, (long)DD, &bit3, 6, fpo3, &out3);
  putbits(&n3, 9, &bit3, 8, fpo3, &out3);

  putbits(&n3, 3, &bit3, 2, fpo3, &out3);
  putbits(&n3, (long)DD, &bit3, 6, fpo3, &out3);
  putbits(&n3, 10, &bit3, 8, fpo3, &out3);

  readoutput(&n4, &n3);

/*-----------------------------------------------------------------------
 * --- When finished reading all the data, close the file.
 *-----------------------------------------------------------------------
 */
  fclose(fp);
}      /* end readcards() */

/***********************************************************************
 *                          VOID OUT_DATA
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  out_data
 *                          Type:  C void
 *                      Filename:  blirb_en.c
 *                        Parent:  readcards
 *=====================================================================
 *<Description>
 *    Transforms the raw data into BUFR form and sends it to be written
 *    to the section 4 temporary data stream
 *=====================================================================
 *<Called routines>
 *    get_trans             - gets the Scale, Data Width, and Reference
 *                            Value for a variable.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void out_data(char c1, char c2, char c3, char c4, int nct,
 *                     float x[10], unsigned long *n4, int desc)
 *    Input:
 *       c1                 - the first character of the card ID
 *       c2                 - the second character of the card ID
 *       c3                 - the third character of the card ID
 *       c4                 - the fourth character of the card ID
 *       nct                - the number of data values to output
 *       x[10]              - the array of raw data values
 *       *n4                - pointer to section 4 output stream
 *       desc               - the Table B descriptor (0 BB desc) for the
 *                            first data value in the array
 *    Output
 *       None
 *=====================================================================
 *<History>
 *    02/01/95  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
```

```
 */
void out_data(char c1, char c2, char c3, char c4, int nct, float x[10],
              unsigned long *n4, int desc)
{
  float ref;
  long nn;
  int j, w[10], sca;

  nn = ((int)c1 << 8) + (int)c2;
  putbits(n4, nn, &bit4, 16, fpo4, &out4);
  nn = ((int)c3 << 8) + (int)c4;
  putbits(n4, nn, &bit4, 16, fpo4, &out4);

  for (j = 0; j < nct; j++)
  { get_trans(j + desc, &sca, &w[j], &ref);
    if(sca != 0)
      x[j] *= (float) pow((double)10, (double)sca);
    x[j] -= ref;
    putbits(n4, (long)x[j], &bit4, w[j], fpo4, &out4);
  }
}      /* end out_data() */

/*********************************************************************
 *                          VOID PUTBITS
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  putbits
 *                          Type:  C void
 *                      Filename:  blirb_en.c
 *                        Parent:  main, readcards, readoutput,
 *                                 write_bytes
 *====================================================================
 *<Description>
 *    This routine puts some BLIRB output bits into the BUFR output
 *    data stream.
 *====================================================================
 *<Called routines>
 *    None.
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void putbits(unsigned long *n, long input, int *bitx,
 *                    int inbits, FILE *sx, long *out_len)
 *    Input:
 *       *n                - pointer to part of the output data stream
 *       input             - the input value
 *       *bitx             - pointer to the number of bits in "n"
 *       inbits            - the number of bits in "input"
 *       *sx               - the output stream pointer
 *       *out_len          - pointer to section output length in bytes
 *    Output
 *       None
 *====================================================================
 *<History>
 *    02/01/95  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void putbits(unsigned long *n, long input, int *bitx, int inbits,
             FILE *sx, long *out_len)
```

```
{
  int sum1, sum2;
  char c1;

  if(input < 0)
    input = ((long)1 << (inbits - 1)) | labs(input);

  if(*bitx + inbits <= 32)
  { *n <<= inbits;
    *n += input;
    *bitx += inbits;

    if(*bitx == 32)
    { c1 = (char) (*n >> 24);
      fputc(c1, sx);
      c1 = (char) ((*n << 8) >> 24);
      fputc(c1, sx);
      c1 = (char) ((*n << 16) >> 24);
      fputc(c1, sx);
      c1 = (char) ((*n << 24) >> 24);
      fputc(c1, sx);
      *out_len += (long)4;

      *n = (long)0;
      *bitx = 0;
    }
  }
  else
  { sum1 = *bitx + inbits;
    sum2 = sum1 - 32;
    sum1 = inbits - sum2;
    *n <<= sum1;
    *n += input >> sum2;

    c1 = (char) (*n >> 24);
    fputc(c1, sx);
    c1 = (char) ((*n << 8) >> 24);
    fputc(c1, sx);
    c1 = (char) ((*n << 16) >> 24);
    fputc(c1, sx);
    c1 = (char) ((*n << 24) >> 24);
    fputc(c1, sx);
    *out_len += (long)4;

    *n = (input << (32 - sum2)) >> (32 - sum2);
    *bitx = sum2;
  }
}      /* end putbits() */

/*********************************************************************
 *                          VOID READOUTPUT
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  readoutput
 *                          Type:  C void
 *                      Filename:  blirb_en.c
 *                        Parent:  readcards
 *====================================================================
 *<Description>
 *    Reads the BLIRB output data from the BLIRB output file and sends
 *    it to be written to sections 3 & 4 streams
 *====================================================================
```

```
 *<Called routines>
 *    group_bytes            - Collects the sequential BLIRB data bytes
 *                             together into commensurate groups
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void readoutput( unsigned long *n4, unsigned long *n3)
 *    Input:
 *       *n4                 - part of the section 4 output data stream
 *       *n3                 - part of the section 3 output data stream
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    02/01/95  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 *********************************************************************
 */
void readoutput(unsigned long *n4, unsigned long *n3)
{
  float dum, *data, ref;
  int i, k, m, l, w[3], wid, sca, loop, na, itn, itn1;
  long j, incr, y[3], dum1;
  char c1;

/*-------------------------------------------------------------------
 * --- Determine the number of wavenumbers.
 *-------------------------------------------------------------------
 */
  out_nwave = wavn_dv;

/*-------------------------------------------------------------------
 * --- Read the values of NA and ITN.
 *-------------------------------------------------------------------
 */
  fscanf(fp, "%d%d", &na, &itn);

  y[0] = (long)na;
  get_trans(YYO, &sca, &w[0], &ref);
  if(sca != 0)
    y[0] *= (long) pow((double)10, (double)sca);
  y[0] -= (long) ref;
  putbits(n4, y[0], &bit4, w[0], fpo4, &out4);

  y[0] = (long)itn;
  get_trans(YYO + 1, &sca, &w[0], &ref);
  if(sca != 0)
    y[0] *= (long) pow((double)10, (double)sca);
  y[0] -= (long) ref;
  putbits(n4, y[0], &bit4, w[0], fpo4, &out4);

  itn1 = itn + 1;

/*-------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine RJOB, get the X, Y, and
 *     Z BLIRB main region grid points and calculate the flux grid
 *     points from them.
 *-------------------------------------------------------------------
 */
  fscanf(fp, "%d%d%d", &out_imx[0], &out_imx[1], &out_imx[2]);
```

```
  for (i = 0; i < 3; i++)
  { get_trans(YYO + 2 + i, &sca, &w[i], &ref);
    y[i] = (long)out_imx[i];
    if(sca != 0)
      y[i] *= (long) pow((double)10, (double)sca);
    y[i] -= (long) ref;
    putbits(n4, y[i], &bit4, w[i], fpo4, &out4);
  }

  dum1 = (float)(out_imx[0] * out_imx[1] * out_imx[2] + 1);
  if(dum1 < (na + 1))
    dum1 = (float)(na + 1);
  if(dum1 < 4 * itn1)
    dum1 = (float)(4 * itn1);
  if(dum1 < (domd_isc + (float)1.0) * (float)itn1)
    dum1 = (domd_isc + (float)1.0) * (float)itn1;

  data = (float *) malloc((long)sizeof(float) * dum1);

  for (i=0; i<3; i++)
  { if(out_imx[i] > 0)
    { get_trans(YYO + 5 + i, &sca, &wid, &ref);

      for (j = 0; j<=out_imx[i]; j++)
        fscanf(fp, "%12e", (data + j));   /* Read X,Y,Z grid points   */

      group_bytes(n4, n3, data, (long)(out_imx[i] + 1), sca, 1, wid,
                  ref, YYO + 5 + i);
    }
  }

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine RJOB, get the surface
 *     albedo indices at each (X,Y) grid point (ISURF).
 *-----------------------------------------------------------------------
 */
  if(out_imx[0] > 0 && out_imx[1] > 0)
  { get_trans(YYO + 8, &sca, &wid, &ref);

    for (incr = 0, i = 0; i<out_imx[1]; i++)
      for (j = 0; j<out_imx[0]; incr++, j++)
        fscanf(fp, "%12e", (data + incr));

    group_bytes(n4, n3, data, (long)(out_imx[0] * out_imx[1]), sca, 1,
                wid, ref, YYO + 8);
  }

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine RJOB, get the region
 *     material indices at each (X,Y,Z) grid point (IVOLM).
 *-----------------------------------------------------------------------
 */
  if(out_imx[0] > 0 && out_imx[1] > 0 && out_imx[2] > 0)
  { get_trans(YYO + 9, &sca, &wid, &ref);

    for (incr = 0, k = 0; k<out_imx[2]; k++)
      for (i = 0; i<out_imx[1]; i++)
        for (j = 0; j<out_imx[0]; incr++, j++)
          fscanf(fp, "%12e", (data + incr));

    group_bytes(n4, n3, data, (long)(out_imx[0] * out_imx[1] *
                out_imx[2]), sca, 1, wid, ref, YYO + 9);
```

```
  }

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine RFLX, get the LOWTRAN
 *     molecular transmission (TRLW).
 *-----------------------------------------------------------------------
 */
  for (m = 0; m<out_nwave; m++)
  { get_trans(YYO + 10, &sca, &wid, &ref);

    for (j = 0; j<=na; j++)
      fscanf(fp, "%12e", (data + j));

    group_bytes(n4, n3, data, (long)(na + 1), sca, 0, wid, ref,
                YYO + 10);

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine CLOUDR, get the surface
 *     albedos (SALB).
 *-----------------------------------------------------------------------
 */
    if(albd >= 0)
    { get_trans(YYO + 11, &sca, &wid, &ref);

      for (j = 0; j<=albd; j++)
        fscanf(fp, "%12e", (data + j));

      group_bytes(n4, n3, data, (long)(albd + 1), sca, 1, wid, ref,
                  YYO + 11);
    }

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine CLOUDR, get the
 *     extinction coefficients (REXT).
 *-----------------------------------------------------------------------
 */
    get_trans(YYO + 12, &sca, &wid, &ref);

    for (j = 0; j<itn1; j++)
      fscanf(fp, "%12e", (data + j));

    group_bytes(n4, n3, data, (long)itn1, sca, 0, wid, ref, YYO + 12);

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine CLOUDR, get the
 *     scattering coefficients (RSCT).
 *-----------------------------------------------------------------------
 */
    get_trans(YYO + 13, &sca, &wid, &ref);

    for (j = 0; j<itn1; j++)
      fscanf(fp, "%12e", (data + j));

    group_bytes(n4, n3, data, (long)itn1, sca, 0, wid, ref, YYO + 13);

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine CLOUDR, get the
 *     "unknown" coefficients (FDLT).
 *-----------------------------------------------------------------------
 */
    get_trans(YYO + 14, &sca, &wid, &ref);
```

```
    for (j = 0; j<itn1; j++)
      fscanf(fp, "%12e", (data + j));

    group_bytes(n4, n3, data, (long)itn1, sca, 0, wid, ref, YYO + 14);

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine CLOUDR, get the phase
 *     function angles (AGL).
 *-----------------------------------------------------------------------
 */
    get_trans(YYO + 15, &sca, &wid, &ref);

    for (incr = 0, j = 0; j<itn1; j++)
      for (i = 0; i<4; incr++, i++)
        fscanf(fp, "%12e", (data + incr));

    group_bytes(n4, n3, data, (long)(4 * itn1), sca, 0, wid, ref,
                YYO + 15);

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine CLOUDR, get the phase
 *     functions for different materials (PHF).
 *-----------------------------------------------------------------------
 */
    get_trans(YYO + 16, &sca, &wid, &ref);

    for (incr = 0, j = 0; j<itn1; j++)
      for (i = 0; i<4; incr++, i++)
        fscanf(fp, "%12e", (data + incr));

    group_bytes(n4, n3, data, (long)(4 * itn1), sca, 0, wid, ref,
                 YYO + 16);

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine CLOUDR, get the
 *     Legendre coefficients (RLEG).
 *-----------------------------------------------------------------------
 */
    get_trans(YYO + 17, &sca, &wid, &ref);

    for (incr = 0, j = 0; j<itn1; j++)
      for (i = 0; i<=(int)domd_isc; incr++, i++)
        fscanf(fp, "%12e", (data + incr));

    group_bytes(n4, n3, data, (long)((domd_isc + 1) * itn1), sca, 0,
                wid, ref, YYO + 17);

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine RFLX, get the direct
 *     solar flux, reflected solar flux, and 8 diffuse flux values at
 *     each (X,Y,Z) flux grid point.
 *-----------------------------------------------------------------------
 */
    for (l=0; l<10; l++)
    { get_trans(YYO + 18 + l, &sca, &wid, &ref);

      for (incr = 0, i = 0; i<out_imx[2]; i++)
        for (j = 0; j<out_imx[1]; j++)
          for (k = 0; k<out_imx[0]; incr++, k++)
            fscanf(fp, "%12e", (data + incr));

      group_bytes(n4, n3, data, (long)(out_imx[0] * out_imx[1] *
```

```
                    out_imx[2]), sca, 0, wid, ref, YYO + 18 + 1);
      }
    }

/*-----------------------------------------------------------------------
 * --- Finish off the Section 3 stream.
 *-----------------------------------------------------------------------
 */
  if(bit3 > 0)
  { loop = 0;
    *n3 <<= (32 - bit3);
    while (bit3 > 0)
    { c1 = (char)((*n3 << (8 * loop)) >> 24);
      fputc(c1, fpo3);
      out3++;
      bit3 -= 8;
      loop++;
    }
  }
  if(out3 % (long)2 == (long)1)
  { fputc((char)0, fpo3);
    out3++;
  }
  fclose(fpo3);

/*-----------------------------------------------------------------------
 * --- Finish off the Section 4 stream.
 *-----------------------------------------------------------------------
 */
  if(bit4 > 0)
  { loop = 0;
    *n4 <<= (32 - bit4);
    while (bit4 > 0)
    { c1 = (char)((*n4 << (8 * loop)) >> 24);
      fputc(c1, fpo4);
      out4++;
      bit4 -= 8;
      loop++;
    }
  }
  if(out4 % 2 == 1)
  { fputc((char)0, fpo4);
    out4++;
  }
  fclose(fpo4);

  free(data);
}       /* end readoutput() */

/***********************************************************************
 *                          VOID GET_TRANS
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  get_trans
 *                          Type:  C void
 *                      Filename:  blirb_en.c
 *                        Parent:  readoutput, readcards
 *=====================================================================
 *<Description>
 *    Gets the Scale, Data Width, and Reference Value for a variable.
 *=====================================================================
 *<Called routines>
```

```
 *    None.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void get_trans(int y0, int *scale, int *width, float *ref)
 *    Input:
 *       y0                  - the "y0" of Table B for the BLIRB variable
 *       *scale              - the standard (exponent of 10) scale factor
 *       *width              - the standard number of bits used by the
 *                             encoded data element
 *       *ref                - the reference value to be subtracted from
 *                             each data element
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    02/01/95  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void get_trans(int y0, int *scale, int *width, float *ref)
{
  int j, y1;

  for (j = 0; j < counter; j++)
  { if(tbl_b[j].descriptor == ((long)(1000) * (long)BB + (long)y0))
    { *scale = tbl_b[j].scale;
      *ref = (float) tbl_b[j].ref_val;
      *width = tbl_b[j].data_width;
      break;
    }
  }
}      /* end get_trans() */

/********************************************************************
 *                          VOID GROUP_BYTES
 ********************************************************************
 *<Begin>
 *<Identification>          Name:  group_bytes
 *                          Type:  C void
 *                      Filename:  blirb_en.c
 *                        Parent:  readoutput
 *=====================================================================
 *<Description>
 *    Collects the sequential BLIRB output data values together into
 *    commensurate groups and sends the group info and data to be written
 *    to the sections 3 & 4 streams.
 *=====================================================================
 *<Called routines>
 *    write_bytes           - Writes the group info and encoded data to
 *                            the sections 3 & 4 streams.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void group_bytes( unsigned long *n4, unsigned long *n3,
 *                         float *data, long nct, int scale, int nochg,
 *                         int width, float ref, int param)
 *    Input:
 *       *n4                - part of the section 4 output data stream
 *       *n3                - part of the section 3 output data stream
```

```
 *       *data                - pointer to the data to be encoded
 *       nct                  - number of BLIRB data elements
 *       scale                - the standard (exponent of 10) scale factor
 *       nochg                - flag to indicate if scale in unchangeable
 *       width                - the standard number of bits used by the
 *                              encoded data element
 *       ref                  - the reference value to be subtracted from
 *                              each data element
 *       param                - the Table B (Y0) number corresponding to
 *                              the BLIRB data
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    02/01/95  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void group_bytes(unsigned long *n4, unsigned long *n3, float *data,
                 long nct, int scale, int nochg, int width, float ref,
                 int param)
{
  static float min0 = (float)(1.0E+20);
  static float max0 = (float)(-1.0E+20);
  float dum, min1, max1, last;
  int k, flag, kk, tomuch, nochang;
  long i, j, start, zero;
  static long istart[1000], istop[1000];

/*-------------------------------------------------------------------
 * --- Find any identical data subgroups.
 *-------------------------------------------------------------------
 */
  i = (long)0;                          /* initialize data counter   */
  k = 0;                                /* init # of ident subgroups */
  flag = 0;                             /* init ident subgroup flag  */
  istart[0] = istop[0] = (long)(-1);    /* init start & stop pointers*/
  last = -1.0;                          /* initialize last data value*/

  while (i < nct)                       /* loop thru all the data    */
  { dum = (float)fabs((double)(*(data + i))); /* work with abs values */

    if(dum == last)                     /* if curr = last values     */
    { if(!flag)                         /* if no group in progress   */
      { istart[k] = i - (long)1;        /* set start pointer         */
        flag = 1;                       /* indicate group in progress*/
      }
      istop[k] = i;                     /* set stop pointer          */
    }
    else                                /* curr != last data values  */
    { last = dum;                       /* save current data value   */
      if(flag)                          /* if group in progress      */
      { flag = 0;                       /* terminate the group       */
        if((istop[k] - istart[k] + (long)1) * (long)width > (long)80)
        { k++;                          /* incr group counter        */
          if(k == 1000)                 /* if too many groups        */
          { k--;                        /* decr group counter        */
            break;                      /* stop checking for them    */
          }
        }
```

```
        istart[k] = istop[k] = (long)(-1); /* init start & stop pointr*/
      }
    }
    i++;                                  /* get next data value       */
  }

/*-----------------------------------------------------------------------
 * --- Check the data for commensurate subgroups and send the data groups
 *     (both commensurate and identical) to be processed.
 *-----------------------------------------------------------------------
 */
  min1 = min0;                            /* initialize minimum value  */
  max1 = max0;                            /* initialize maximum value  */

  tomuch = 0;                             /* initialize grp break index*/
  zero = j = (long)0;                     /* initialize counters       */
  start = (long)0;                        /* initialize pointer        */
  i = (long)0;                            /* initialize data counter   */
  kk = 0;                                 /* ident data group index    */

  while (i < nct)                         /* loop thru all the data    */
  { dum = (float)fabs((double)(*(data + i))); /* work with abs values */
    j++;                                  /* inc data subgroup index   */

/*-----------------------------------------------------------------------
 * --- Send the identical data groups (if any) for processing.
 *-----------------------------------------------------------------------
 */
    if(i == istart[kk])                   /* if start of ident data grp*/
    { j = istop[kk] - istart[kk] + (long)1; /* get data item count     */
      min1 = max1 = dum;                  /* get min and max values    */

      if(dum == 0.0)                      /* if data values are zero   */
      { zero = j;                         /* set zero counter to total */
        nochang = 0;                      /* allow change of scale     */
      }
      else                                /* if data values non-zero   */
      { zero = (long)0;                   /* set zero count to 0       */
        nochang = nochg;                  /* preserve "nochg" scale chg*/
      }

      write_bytes(n4, n3, (data + start), j, scale, nochang, width,
                  min1, max1, zero, ref, param);
      start += j;                         /* update pointer start point*/

      i += (j - (long)1);                 /* update data pointer       */
      min1 = min0;                        /* reset minimum value       */
      max1 = max0;                        /* reset maximum value       */
      j = zero = (long)0;                 /* reset counters            */
      tomuch = 0;                         /* reset group break index   */

      if(kk < k)                          /* if more ident data subgrps*/
        kk++;                             /* incr ident data group indx*/
    }
/*-----------------------------------------------------------------------
 * --- Get the commensurate data (if any) grouped for processing.
 *-----------------------------------------------------------------------
 */
    else                                  /* non-ident data checking   */
    { if(dum > 0.0)                       /* process non-zero data     */
      { if( dum < min1)                   /* find the minimum value    */
        { if(max1 / dum > 1000.0)
```

```
            tomuch = 1;                    /* range of data excessive   */
          else
            min1 = dum;
        }
        if( dum > max1)                    /* find the maximum value    */
        { if(dum / min1 > 1000.0)
            tomuch = 1;                    /* range of data excessive   */
          else
            max1 = dum;
        }
      }
      else                                 /* process zero data values  */
        zero++;                            /* incr zero data count      */

      if(tomuch || i == (istart[kk] - (long)1)) /* process data subgrp*/
      { if(tomuch)                         /* if not last non-ident val */
          j--;                             /* remove last value frm grp */
        if(j == zero)                      /* if group all zero values  */
          min1 = max1 = (float)0.0;        /* set max and min to zero   */
        write_bytes(n4, n3, (data + start), j, scale, nochg, width,
                    min1, max1, zero, ref, param);
        start += j;                        /* update pointer start point*/

        if(tomuch)                         /* if not last non-ident val */
          i--;                             /* reprocess last data value */
        min1 = min0;                       /* reset minimum value       */
        max1 = max0;                       /* reset maximum value       */
        j = zero = (long)0;                /* reset counters            */
        tomuch = 0;                        /* reset group break index   */
      }
    }

    i++;                                   /* inc pointer, get next data*/
  }

/*-----------------------------------------------------------------------
 * --- Send any remaining data for processing.
 *-----------------------------------------------------------------------
 */
  if(j > 0)                                /* process any remaining data*/
  { if(j == zero)                          /* if all data are zeros     */
      min1 = max1 = (float)0.0;            /* set min and max to zero   */
    write_bytes(n4, n3, (data + start), j, scale, nochg, width, min1,
                max1, zero, ref, param);
  }
}      /* end group_bytes() */

/***********************************************************************
 *                          VOID WRITE_BYTES
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  write_bytes
 *                          Type:  C void
 *                      Filename:  blirb_en.c
 *                        Parent:  group_bytes
 *=====================================================================
 *<Description>
 *    Writes the group info and encoded data to the sections 3 & 4
 *    streams.
 *=====================================================================
 *<Called routines>
 *    factor                - Factors an integer into up to 4 factors
```

```
 *                              (each less than 256)
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void write_bytes( unsigned long *n4, unsigned long *n3,
 *                         float *data, long nct, int scale, int nochg,
 *                         int width, float min1, float max1, long zero,
 *                         float ref, int param)
 *    Input:
 *       *n4                - part of the section 4 output data stream
 *       *n3                - part of the section 3 output data stream
 *       *data              - pointer to the data to be encoded
 *       nct                - number of BLIRB data elements
 *       scale              - the standard (exponent of 10) scale factor
 *       nochg              - flag to indicate if scale in unchangeable
 *       width              - the standard number of bits used by the
 *                            encoded data element
 *       min1               - the minimum value in the group
 *       max1               - the maximum value in the group
 *       zero               - number of zero data values
 *       ref                - the reference value to be subtracted from
 *                            each data element
 *       param              - the Table B (Y0) number corresponding to
 *                            the BLIRB data
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    02/01/95  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void write_bytes(unsigned long *n4, unsigned long *n3, float *data,
                 long nct, int scale, int nochg, int width, float min1,
                 float max1, long zero, float ref, int param)
{
  float lmax1, reff, reff1, sc10;
  double lmin1;
  int l, ll, lll;
  long nct1, nct2, nct3, nct4, nctx, wid, i;

/*-------------------------------------------------------------------
 * --- Determine a new reference value (if possible) assumming the
 *     scale does not change.
 *-------------------------------------------------------------------
 */
  if(ref != 0.0)
    ref = ref * (float) pow((double)10, (double)(-scale));

  if(zero == (long)0 && (nct * (long)width > (long)80))
  { if(fabs((double) (max1 - min1)) > 0.00001 * min1)
      reff = (float)0.0;
    else
    { if(fabs((double) (min1 - ref)) > 0.00001 * min1)
        reff = min1;
    }
  }
  else
  { if(zero == nct)
    { if(ref == 0.0)
```

```
        reff = (float)0.0;
      else
        reff = -ref;
    }
    else
      reff = (float)0.0;
  }

  ref += reff;

/*-----------------------------------------------------------------------
 * --- Determine the scale to apply to each data value.
 *-----------------------------------------------------------------------
 */
  if(nochg)
    ll = scale;
  else
  { if(min1 > ref)
    { lmin1 = log10( (double) (min1 - ref));
      ll = (int)((float)4 - (float)floor((double)lmin1));
    }
    else if(ref != 0.0)
    { lmin1 = log10( fabs((double)ref));
      ll = (int)((float)4 - (float)floor((double)lmin1));
    }
    else
      ll = scale;
  }

/*-----------------------------------------------------------------------
 * --- Determine the field width of the reference value.
 *-----------------------------------------------------------------------
 */
  if(ref != 0.0)
    reff1 = ref * (float) pow((double)10, (double)(ll));
  else
    reff1 = 0.0;

  if(reff != 0.0 || (ref != 0.0 && ll != scale))
  { lll = width;
    reff1 = (float)2.0 * (float)fabs((double)reff1);

    for (i=0, wid=1; i<31; wid *= 2, i++)
    { if(reff1 < (wid - 1))
      { lll = width;
        if(i > lll || (lll - i > 1))
          lll = (int)i;
        break;
      }
    }
  }
  else
    lll = 0;

/*-----------------------------------------------------------------------
 * --- Determine the field width of each data value.
 *-----------------------------------------------------------------------
 */
  if(max1 > ref)
    lmax1 = (float) pow((double)10, (log10((double) (max1 - ref))
            + (double)ll));
  else
```

```
    lmax1 = (float)1.0;
  lmax1 *= (float)2.0;

  l = width;
  for (i=0, wid=1; i<31; wid *= 2, i++)
  { if(lmax1 < (wid - 1))
    { l = width;
      if(i > l || (l - i > 1))
        l = (int)i;
      break;
    }
  }

/*-----------------------------------------------------------------------
 * --- If the data field width is different than standard tell the BUFR
 *     stream.
 *-----------------------------------------------------------------------
 */
  if(l != width)
  { putbits(n3, 2, &bit3, 2, fpo3, &out3);
    putbits(n3, 1, &bit3, 6, fpo3, &out3);
    putbits(n3, (long)(128 + l - width), &bit3, 8, fpo3, &out3);
  }

/*-----------------------------------------------------------------------
 * --- If the scale is different than standard tell the BUFR stream.
 *-----------------------------------------------------------------------
 */
  if(ll != scale)
  { putbits(n3, 2, &bit3, 2, fpo3, &out3);
    putbits(n3, 2, &bit3, 6, fpo3, &out3);
    putbits(n3, (long)(128 + ll - scale), &bit3, 8, fpo3, &out3);
    sc10 = (float) pow((double)10, (double)ll);
    ref *= sc10;
  }
  else
  { sc10 = (float) pow((double)10, (double)scale);
    ref *= sc10;
  }

/*-----------------------------------------------------------------------
 * --- If there is a reference value tell the BUFR stream.
 *-----------------------------------------------------------------------
 */
  if(lll != 0)
  { putbits(n3, 2, &bit3, 2, fpo3, &out3);
    putbits(n3, 3, &bit3, 6, fpo3, &out3);
    putbits(n3, (long)lll, &bit3, 8, fpo3, &out3);

    putbits(n3, 0, &bit3, 2, fpo3, &out3);
    putbits(n3, (long)BB, &bit3, 6, fpo3, &out3);
    putbits(n3, (long)param, &bit3, 8, fpo3, &out3);

    putbits(n3, 2, &bit3, 2, fpo3, &out3);
    putbits(n3, 3, &bit3, 6, fpo3, &out3);
    putbits(n3, 255, &bit3, 8, fpo3, &out3);

    putbits(n4, (long)ref, &bit4, lll, fpo4, &out4);
  }

/*-----------------------------------------------------------------------
 * --- Determine the replication factors and tell the BUFR stream.
```

```
 *-----------------------------------------------------------------
 */
  if(nct > (long)1)
  { nctx = nct;
    while (nctx > (long)0)
    { nct1 = nct2 = nct3 = nct4 = (long)1;

      if(nctx > (long)255)
      { factor(nctx, &nct1, &nct2, &nct3, &nct4);
        if(nct1 > (long)255)
          nct1 = (long)255;
      }
      else
        nct1 = nctx;

      nctx -= nct1 * nct2 * nct3 * nct4;

      if(nct4 > (long)1)
      { putbits(n3, 1, &bit3, 2, fpo3, &out3);
        putbits(n3, 4, &bit3, 6, fpo3, &out3);
        putbits(n3, nct4, &bit3, 8, fpo3, &out3);
      }
      if(nct3 > (long)1)
      { putbits(n3, 1, &bit3, 2, fpo3, &out3);
        putbits(n3, 3, &bit3, 6, fpo3, &out3);
        putbits(n3, nct3, &bit3, 8, fpo3, &out3);
      }
      if(nct2 > (long)1)
      { putbits(n3, 1, &bit3, 2, fpo3, &out3);
        putbits(n3, 2, &bit3, 6, fpo3, &out3);
        putbits(n3, nct2, &bit3, 8, fpo3, &out3);
      }
      if(nct1 > (long)1)
      { putbits(n3, 1, &bit3, 2, fpo3, &out3);
        putbits(n3, 1, &bit3, 6, fpo3, &out3);
        putbits(n3, nct1, &bit3, 8, fpo3, &out3);
      }
      if(nctx > (long)0)
      { putbits(n3, 0, &bit3, 2, fpo3, &out3);
        putbits(n3, (long)BB, &bit3, 6, fpo3, &out3);
        putbits(n3, (long)param, &bit3, 8, fpo3, &out3);
      }
    }
  }

/*-----------------------------------------------------------------
 * --- Tell the BUFR stream the Descriptor for the data.
 *-----------------------------------------------------------------
 */
  putbits(n3, 0, &bit3, 2, fpo3, &out3);
  putbits(n3, (long)BB, &bit3, 6, fpo3, &out3);
  putbits(n3, (long)param, &bit3, 8, fpo3, &out3);

/*-----------------------------------------------------------------
 * --- Transform the data and put it into the BUFR stream.
 *-----------------------------------------------------------------
 */
  for (i = 0; i<nct; i++)
  { reff = (sc10 * (*(data + i))) - ref;
    putbits(n4, (long)reff, &bit4, 1, fpo4, &out4);
  }
```

```
/*-----------------------------------------------------------------------
 * --- Tell the BUFR stream to kill any reference value changes.
 *-----------------------------------------------------------------------
 */
  if(lll != 0)
  { putbits(n3, 2, &bit3, 2, fpo3, &out3);
    putbits(n3, 3, &bit3, 6, fpo3, &out3);
    putbits(n3, 0, &bit3, 8, fpo3, &out3);
  }

/*-----------------------------------------------------------------------
 * --- Tell the BUFR stream to kill any scale changes.
 *-----------------------------------------------------------------------
 */
  if(ll != scale)
  { putbits(n3, 2, &bit3, 2, fpo3, &out3);
    putbits(n3, 2, &bit3, 6, fpo3, &out3);
    putbits(n3, 0, &bit3, 8, fpo3, &out3);
  }

/*-----------------------------------------------------------------------
 * --- Tell the BUFR stream to kill any field width changes.
 *-----------------------------------------------------------------------
 */
  if(l != width)
  { putbits(n3, 2, &bit3, 2, fpo3, &out3);
    putbits(n3, 1, &bit3, 6, fpo3, &out3);
    putbits(n3, 0, &bit3, 8, fpo3, &out3);
  }
}      /* end write_bytes() */

/***********************************************************************
 *                          VOID FACTOR
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  factor
 *                          Type:  C void
 *                      Filename:  blirb_en.c
 *                        Parent:  write_bytes
 *=====================================================================
 *<Description>
 *    Factors an integer into up to 4 factors (each less than 256)
 *=====================================================================
 *<Called routines>
 *    None.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void factor( long nct, long *nct1, long *nct2, long *nct3,
 *                    long *nct4)
 *    Input:
 *       nct                - number of BLIRB data elements
 *       *nct1              - pointer to the first factor
 *       *nct2              - pointer to the second factor
 *       *nct3              - pointer to the third factor
 *       *nct4              - pointer to the fourth factor
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    02/01/95  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
```

```
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void factor( long nct, long *nct1, long *nct2, long *nct3, long *nct4)
{
  static int num[26] = { 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41,
                         43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 91,
                         97 };
  static int mpy[26] = { 127, 85, 51, 36, 23, 19, 15, 13, 11, 8, 8, 6,
                         6, 5, 5, 4, 4, 4, 3, 3, 3, 3, 3, 2, 2, 2 };
  int i;

  *nct2 = *nct3 = *nct4 = (long)1;
  *nct1 = nct;

  while (*nct1 > (long)255)
  { for (i=0; i<26; i++)
    { if(*nct1 % (long)num[i] == 0)
      { while (*nct1 % (long)num[i] == (long)0 && *nct1 > (long)255)
        { *nct1 /= (long)num[i];
          if(*nct2 < (long)mpy[i])
            *nct2 *= (long)num[i];
          else if(*nct3 < (long)mpy[i])
            *nct3 *= (long)num[i];
          else if(*nct4 < (long)mpy[i])
            *nct4 *= (long)num[i];
        }
      }
    }
    break;
  }
}      /* end factor() */
```

Appendix D

Listing of BLIRB_DE.C

```
#include <stdio.h>
#include <fcntl.h>
#include <ctype.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
#include "bufr.h"                /* BUFR encoder/decoder include file */

/*-----------------------------------------------------------------------
 * --- Array declarations and assignments
 *-----------------------------------------------------------------------
 */

struct table_b *tbl_b;      /* pointer to array of structures Table B */
struct table_d *tbl_d;      /* pointer to array of structures Table D */

char *file_name;                 /* BUFR input filename                */
FILE *fp;                        /* Pointer to BUFR input file         */
FILE *fpo;                       /* Pointer to BLIRB output file       */
FILE *fpo30;                     /* Pointer to 1st temp Sect 3 file    */
static char *temp30 = "temp30.tmp"; /* Sect 3 1st temporary filename   */
static char *temp31 = "temp31.tmp"; /* Sect 3 2nd temporary filename   */
int counter;                     /* number of records found in Table B*/
int dt_sets;                     /* number of subsets in the message   */
long count_desc;                 /* number of descriptors sent         */
int seq_count;                   /* number of sequences in Table D     */
long seq_desc[SEQ_SIZE];         /* pointer to array of desc expanded */
unsigned long lMessageSize;      /* length of current Section in bytes*/
float bufr_data;                 /* pointer to data value              */
int compressed;                  /* flag for compressed data:
                                         1-compressed, 0-uncompressed */
long sect4_bits;                 /* number of remaining bits in sect 4*/
long mess_desc;                  /* descriptor from message            */

/***********************************************************************
 *                          VOID MAIN
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  main
 *                          Type:  C Main Program
 *                      Filename:  blirb_de.c
 *                        Parent:  None
 *=====================================================================
 *<Description>
 *    This program decodes a BLIRB BUFR binary output file into a ascii
 *    BLIRB output file.
 *=====================================================================
 *<Called routines>
 *    ParseInput            - Parses the command line inputs
 *    read_table_b          - reads the structure records from Table_B
 *    read_table_d          - reads the structure sequences from Table_D
 *    gbyte                 - extracts specified bits from BUFR stream
 *    bufridenti            - gets the BUFR section 1 data
 *    bufrgetind            - expands the input descriptors
 *    get_data              - interprets individual descriptors and gets
 *                            section 4 data processed
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
```

```
 *       void main(int argc, char **argv)
 *    Input:
 *       argv                - (char pointer) the array of command lines
 *       argc                - (int) the number of command lines inputs
 *                             inputs
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    02/02/94  CSC Monterey, CA   Mugur Georgescu
 *              Developed the original source code.
 *    05/31/94  PL Hanscom AFB, MA   Rodger Biasca
 *              Modified the original.
 *    02/01/95  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Adapted the modified code to BLIRB BUFR decode use.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void main(int argc, char **argv)
{
  unsigned long edition;          /* BUFR edition number                */
  struct identif id;              /* structure identification section   */
  unsigned long lMessageSizeT;    /* length of BUFR message in bytes    */
  int observed;                   /* observed data indicator            */
  unsigned long lByte1, lByte2, lByte3, status4, something;
  unsigned int dumm;
  char dum[5];
  long ii;
  int status, ft, xt, yt, i, fin, flip;

/*---------------------------------------------------------------------
 * --- Get descriptors from Table B.
 *---------------------------------------------------------------------
 */
  read_table_b();

/*---------------------------------------------------------------------
 * --- Get sequences from Table D.
 *---------------------------------------------------------------------
 */
  read_table_d();

/*---------------------------------------------------------------------
 * --- Get the filenames from the command line.
 *---------------------------------------------------------------------
 */
  ParseInput(argc, argv);

/*---------------------------------------------------------------------
 * --- Begin processing the encoded data.
 *---------------------------------------------------------------------
 */
  for (i = 0; i < 4; i++)         /* search for start of BUFR message  */
    dum[i] = fgetc(fp);
  while (strncmp(dum, "BUFR", 4) != 0)
  { for (i = 1; i < 4; i++)
      dum[i-1] = dum[i];
    if((dum[3] = fgetc(fp)) == EOF)
    { printf("The characters BUFR were never found.\n");
      printf("Program Aborting!\n");
      fclose(fp);
```

```
      fclose(fpo);
      fclose(fpo30);
      remove(temp30);
      exit(-1);
    }
  }
  status = 0;

/*-----------------------------------------------------------------------
 * --- Process Section 0 information.
 *-----------------------------------------------------------------------
 */
  gbyte(&lMessageSizeT, 24);      /* get the BUFR message length       */

  printf("\n  Total Message length      = %lu\n", lMessageSizeT);

  gbyte(&edition, 8);             /* get edition                       */
  if(edition != (unsigned long)2)
  { status = 1;
    printf("The edition of the BUFR message is not 2\n");
  }

/*-----------------------------------------------------------------------
 * --- Process Section 1 information.
 *-----------------------------------------------------------------------
 */
  if(!status)
  { bufridenti(&id);              /* get the identification section    */

    printf("  Master Table ID           = %d\n", id.master_tbl);
    printf("  Originating Center        = %u\n", id.orig_cntr);
    printf("  Local Originating Center  = %d\n", id.locc);
    printf("  Update Sequence           = %d\n", id.updt_seq);
    printf("  Optional Section Present? = %d\n", id.opt_sec);
    printf("  Message Type              = %d\n", id.msg_type);
    printf("  Message SubType           = %d\n", id.msg_subtype);
    printf("  Master Table Version      = %d\n", id.mstr_vrsn);
    printf("  Local Table Version       = %d\n", id.locl_vrsn);
    printf("  Year                      = %d\n", id.year);
    printf("  Month                     = %d\n", id.month);
    printf("  Day                       = %d\n", id.day);
    printf("  Hour                      = %d\n", id.hour);
    printf("  Minute                    = %d\n", id.minute);
    printf("  Second                    = %d\n", id.second);
    printf("  Model Identifier          = %d\n", id.model);
    printf("  Database Sequence No.     = %u\n", id.dbsn);

    if(id.master_tbl != 0)        /* chk std WMO FM 94-IX Ext BUFR tbls*/
    { status = 1;
      printf("The BUFR tables are not WMO standard.\n");
    }

    if(id.mstr_vrsn != 3)         /* chk version num BUFR tables used  */
    { status = 1;
      printf("Version # %d of the BUFR tables is used\n", id.mstr_vrsn);
    }

    if(id.updt_seq != 0)          /* check update sequence number      */
    { status = 1;
      printf("This is an update of the BUFR format.\n");
    }
  }
```

```
/*-----------------------------------------------------------------------
 * --- Process Section 2 information.
 *-----------------------------------------------------------------------
 */
  if(!status)
  { if(id.opt_sec)
    { gbyte(&lMessageSize, 24);  /* get section size                    */

      lMessageSize -= (unsigned long)3;   /* skip the info              */
      for (i = 0; i < (int)lMessageSize; i++)
        gbyte(&something, 8);
    }

/*-----------------------------------------------------------------------
 * --- Process Section 3 information.
 *-----------------------------------------------------------------------
 */
    gbyte(&lMessageSize, 24);     /* get section size                   */
    gbyte(&something, 8);         /* reserved 0                         */
    gbyte(&something, 16);
    dt_sets = (unsigned)something;   /* Number data subsets             */
    gbyte(&something, 8);         /* data type info                     */

    dumm = (unsigned)something % 256;
    observed = (int)(dumm << 8) >> 15;  /* observed or other data flag*/
    compressed = (int)(dumm << 9) >> 15;/* compressed or uncompressed */

    printf("  Data subsets              = %d\n", dt_sets);
    printf("  Observed bit              = %d\n", observed);
    printf("  Compressed bit            = %d\n\n", compressed);

    count_desc = ((long)lMessageSize - (long)7) / (long)2;

    for (ii = 0; ii < count_desc; ii++) /* get element descriptors    */
    { gbyte(&something, 8);
      lByte1 = something >> 6;
      lByte2 = (something << 26) >> 26;
      gbyte(&something, 8);
      lByte3 = something;
      mess_desc = (long)lByte1 * (long)100000 +
                  (long)lByte2 * (long)1000 + (long)lByte3;

      fwrite(&mess_desc, sizeof(long), 1, fpo30);    /* put into file */
    }
    fclose(fpo30);
    gbyte(&something, 8);         /* get the trailing zero byte        */

    printf("  Expanding Section 3 Descriptors:");
    i = 1;
    fin = 1;
    flip = 0;
    while (fin != 0)              /* get elements descriptors indices  */
    { printf("\n%2d:", i);
      i++;

      flip++;
      bufrgetind((flip % 2), &fin, &status4);

      if(status4 != (unsigned long)(-1))  /* error processing          */
      { printf("\nCould not find descriptor\n");
        ft = (int)((long)status4/(long)100000);
        xt = (int)(((long)status4 - (long)ft * (long)100000) /
```

```
                    (long)1000);
        yt = (int)((long)status4 - (long)ft * (long)100000 -
                    (long)xt * (long)1000);
        printf(" %d %d %d\n", ft, xt, yt);

        printf("\nProgram Aborting!\n");
        remove(temp30);
        remove(temp31);
        fclose(fpo);
        fclose(fp);
        exit(-1);
      }
    }

/*-----------------------------------------------------------------------
 * --- Process Section 4 information.
 *-----------------------------------------------------------------------
 */
    free(tbl_d);

    gbyte(&lMessageSize, 24);     /* get section size                  */
    gbyte(&something, 8);         /* reserved 0                        */
    lMessageSize -= (unsigned long)4;

    printf("\n\n  Processing Section 4 data:\n   ");
    get_data();                   /* decode the encoded BLIRB data     */
    free(tbl_b);
  }
}
       /* end main() */

/***********************************************************************
 *                           VOID PARSEINPUT
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  ParseInput
 *                          Type:  C void
 *                      Filename:  blirb_de.c
 *                        Parent:  main
 *=====================================================================
 *<Description>
 *    Parses the command line inputs.
 *=====================================================================
 *<Called routines>
 *    gbyte                 - extracts specified bits from BUFR stream
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void ParseInput( int argc, char **argv)
 *    Input:
 *       argc               - (int) the number of command lines inputs
 *       argv               - (char pointer) the array of command lines
 *                            inputs
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    02/02/94  CSC Monterey, CA   Mugur Georgescu
 *              Developed the original source code.
 *    05/31/94  PL Hanscom AFB, MA   Rodger Biasca
 *              Modified the original.
 *    02/01/95  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
```

```
 *              Adapted the modified code to BLIRB BUFR decode use.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void ParseInput(int argc, char **argv)
{
  int i, status, indx;

  status = indx = 0;
  for (i = 1; i < argc; i++)
  { if(argv[i][0] == '-')
    { switch(argv[i][1])
      { case 'o':
          fpo = fopen(argv[i+1], "w");
          fpo30 = fopen(temp30, "wb");
          indx++;
          break;
        case 'i':
          file_name = argv[i+1];
          fp = fopen(file_name, "rb");
        break;
        default:
          status = 1;
        break;
      }
    }
  }

  if(status == 1 || fpo == NULL || fp == NULL)
  { printf("Usage: bufr_enc -o outfile -i infile\n");
    if(indx)
    { fclose(fpo);
      fclose(fpo30);
      remove(temp30);
    }
    exit(-1);
  }
}      /* end ParseInput() */

/***********************************************************************
 *                          VOID READ_TABLE_B
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  read_table_b
 *                          Type:  C void
 *                      Filename:  blirb_de.c
 *                        Parent:  main
 *=====================================================================
 *<Description>
 *    Reads the descriptors from Table_B.
 *=====================================================================
 *<Called routines>
 *    None.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void read_table_b( void)
 *    Input:
 *       None.
 *    Output:
 *       None
```

```
 *=====================================================================
 *<History>
 *    02/02/94  CSC Monterey, CA   Mugur Georgescu
 *              Developed the original source code.
 *    05/31/94  PL Hanscom AFB, MA   Rodger Biasca
 *              Modified the original.
 *    02/01/95  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Adapted the modified code to BLIRB BUFR decode use.
 *=====================================================================
 *<End>
 **********************************************************************
 */
void read_table_b(void)
{
  FILE *ftb;
  long f0, x0, y0, scale, ref, width;
  int size, num;
  char card[RLEN], name[200], unit[16];

  size = 50;
  num = 0;

  if((ftb = fopen("table_b", "r")) == 0)
  { printf("Table_B file not found.  Program Aborting!\n");
    exit(-1);
  }
  else
  { tbl_b = (struct table_b *) malloc(sizeof(struct table_b) * size);

    while(!feof(ftb))
    { fgets(card, RLEN, ftb);
      sscanf(card, "%ld%ld%ld%ld%ld%ld%s%s", &f0, &x0, &y0,
             &scale, &ref, &width, unit, name);

      tbl_b[num].descriptor = f0*(long)100000 + x0*(long)1000 + y0;
      tbl_b[num].scale = (int)scale;
      tbl_b[num].ref_val = ref;
      tbl_b[num].data_width = (int)width;
      strcpy(tbl_b[num].units, unit);

      num++;
      if(num >= size)
      { size += 50;
        tbl_b = (struct table_b *) realloc(tbl_b,
                 sizeof(struct table_b) * size);
      }
    }

    counter = num - 1;
    fclose(ftb);
  }
}      /* end read_table_b() */

/**********************************************************************
 *                          VOID READ_TABLE_D
 **********************************************************************
 *<Begin>
 *<Identification>          Name:  read_table_d
 *                          Type:  C void
 *                      Filename:  blirb_de.c
 *                        Parent:  main
 *=====================================================================
```

```
 *<Description>
 *    Reads the descriptor sequences from Table_D.
 *=====================================================================
 *<Called routines>
 *    None.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void read_table_d( void)
 *    Input:
 *       None.
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    02/02/94  CSC Monterey, CA   Mugur Georgescu
 *              Developed the original source code.
 *    05/31/94  PL Hanscom AFB, MA   Rodger Biasca
 *              Modified the original.
 *    02/01/95  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Adapted the modified code to BLIRB BUFR decode use.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void read_table_d(void)
{
  short IsDes;
  int size, num, num_seq, seq_tot;
  long f0, x0, y0;
  char card[RLEN];
  FILE *fpd;

  size = 50;
  num = num_seq = seq_tot = 0;

  if((fpd = fopen("table_d", "r")) == 0)
  { printf("Table_D file not found.  Program Aborting!\n");
    exit(-1);
  }
  else
  { tbl_d = (struct table_d *) malloc(sizeof(struct table_d) * size);

    IsDes = 1;
    while(!feof(fpd))
    { fgets(card, RLEN, fpd);
      sscanf(card, "%ld%ld%ld", &f0, &x0, &y0);

      if(IsDes)
      { tbl_d[num].sequence =  f0*(long)100000 + x0*(long)1000 + y0;
        tbl_d[num].seq_index = (long)seq_tot;
        IsDes = 0;
      }

      if(f0 == (long)(-1) || x0 == (long)(-1) || y0 == (long)(-1))
      { IsDes = 1;
        tbl_d[num].seq_expand = (long)num_seq;
        num_seq = 0;
        num++;
        if(num >= size - 1)
        { size += 50;
        tbl_d = (struct table_d *) realloc(tbl_d,
```

```
                    sizeof(struct table_d) * size);
          }
        }
        else
        { IsDes = 0;
          num_seq++;
          seq_desc[seq_tot] = f0*(long)100000 + x0*(long)1000 + y0;
          seq_tot++;
          if(seq_tot >= SEQ_SIZE - 1)
          { printf("The Pointer Array of Sequences (seq_desc) is not\n");
            printf("large enough.  Please increase the value of\n");
            printf("SEQ_SIZE in <bufr.h> and recompile the program.\n");
            printf("\nProgram Aborting!\n");
          }
        }
      }

      seq_count = num - 1;
      fclose(fpd);
    }
  }      /* end read_table_d() */

/*********************************************************************
 *                            VOID GBYTE
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  gbyte
 *                          Type:  C void
 *                      Filename:  blirb_de.c
 *                        Parent:  main
 *====================================================================
 *<Description>
 *    Extracts a few bits from the BUFR binary stream.
 *====================================================================
 *<Called routines>
 *    None
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void gbyte(unsigned long *iout, int nbyte)
 *    Input:
 *       *iout                - the extracted bits
 *       nbyte                - number of bits to extract
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    02/01/95  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void gbyte(unsigned long *iout, int nbyte)
{
  static unsigned long nn = (unsigned long)0;
  static unsigned long remainder = (long)0;
  static int nbits = 0;
  static int first = 1;
  int i, cnt, addbytes;

  if(first)                        /* read the first 4 bytes from file  */
```

```
    { for (i = 0; i < 4; i++)
        nn = (nn << 8) + (unsigned long)fgetc(fp);
      first = 0;
    }

    if(nbyte > 32)                  /* "nbyte" must be <= 32 bits        */
    { printf("More than 32 bits were asked for in <gbyte>.\n");
      printf("This is unacceptable.  Program Aborting!\n");
      fclose(fp);
      fclose(fpo);
      fclose(fpo30);
      remove(temp30);
      exit(-1);
    }
    else
    { *iout = (unsigned long) (nn >> (32 - nbyte));  /* get return value*/

      cnt = nbyte;                  /* now remove bits from temp buffer  */
      if(nbits)                     /* if still have bits in temp buffer */
      { if(nbits <= nbyte)          /* bits in buffer <= bits removed    */
        { nn = (nn << nbits) + remainder;
          remainder = (long)0;
          cnt -= nbits;
          nbits = 0;
        }
        else                        /* bits in temp buffer > bits removed*/
        { nn = (nn << nbyte) + (remainder >> (nbits - nbyte));
          remainder = (remainder << (32 - nbits + nbyte)) >>
                      (32 - nbits + nbyte);
          cnt = 0;
          nbits -= nbyte;
        }
      }

      addbytes = cnt / 8;           /* get new bytes from file to buffer */
      if(addbytes > 0)
        for (i = 0; i < addbytes; i++)
        { nn = (nn << 8) + (unsigned long)fgetc(fp);
          cnt -= 8;
        }

      if(cnt > 0)                   /* if necessary, get part of another */
      { remainder = (unsigned long) fgetc(fp);
        nn = (nn << cnt) + (remainder >> (8 - cnt));
        remainder = (remainder << (24 + cnt)) >> (24 + cnt);
        nbits = 8 - cnt;
      }
    }
}
       /* end gbyte() */

/*********************************************************************
 *                          VOID BUFRIDENTI
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  bufridenti
 *                          Type:  C void
 *                      Filename:  blirb_de.c
 *                        Parent:  main
 *====================================================================
 *<Description>
 *    Extracts the values in BUFR Section 1.
```

```
 *=====================================================================
 *<Called routines>
 *    gbyte                    - extracts specified bits from BUFR stream
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void bufridenti(struct identif *id)
 *    Input:
 *       *id                   - structure to hold the Section 1 data.
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    02/02/94  CSC Monterey, CA   Mugur Georgescu
 *              Developed the original source code.
 *    05/31/94  PL Hanscom AFB, MA   Rodger Biasca
 *              Modified the original.
 *    02/01/95  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Adapted the modified code to BLIRB BUFR decode use.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void bufridenti(struct identif *id)
{
  unsigned long something;       /* value extracted from message      */
  int i;

  gbyte(&something, 24);
  id->length = (long)something;  /* get the section length            */

  gbyte(&something, 8);
  id->master_tbl = (int)something; /* get BUFR master table 0-standard*/

  gbyte(&something, 16);
  id->orig_cntr = (unsigned)something;   /* get originating center id */

  gbyte(&something, 8);
  id->updt_seq = (int)something; /* get update sequence # 0 for orig  */

  gbyte(&something, 8);
  id->opt_sec = (int)something;  /* get the optional section flag     */

  gbyte(&something, 8);
  id->msg_type = (int)something; /* get the message type (table A)    */

  gbyte(&something, 8);
  id->msg_subtype = (int)something; /* get the message sub-type       */

  gbyte(&something, 8);
  id->mstr_vrsn = (int)something;   /* get master table version cur 2 */

  gbyte(&something, 8);
  id->locl_vrsn = (int)something;   /* get the local table version    */

  gbyte(&something, 8);
  id->year = (int)something;     /* get the year                      */

  gbyte(&something, 8);
  id->month = (int)something;    /* get the month                     */

  gbyte(&something, 8);
```

```
  id->day = (int)something;       /* get the day                          */

  gbyte(&something, 8);
  id->hour = (int)something;      /* get the hour                         */

  gbyte(&something, 8);
  id->minute = (int)something;    /* get the minute                       */

  gbyte(&something, 8);
  id->locc = (int)something;      /* get the Local Originating Center     */

  gbyte(&something, 8);
  id->model = (int)something;     /* get the model identifier             */

  gbyte(&something, 8);
  id->second = (int)something;    /* get the second                       */

  gbyte(&something, 16);
  id->dbsn = (unsigned)something; /* get the database sequence number */

  if(id->length > 22)             /* get any remaining bytes in Sect 1 */
    for (i = 22; i < (int)id->length; i++)
      gbyte(&something, 8);
}
       /* end bufridenti() */

/*********************************************************************
 *                          VOID BUFRGETIND
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  bufrgetind
 *                          Type:  C void
 *                      Filename:  blirb_de.c
 *                        Parent:  main
 *====================================================================
 *<Description>
 *    Extracts the data in BUFR Section 3 and expands the sequences.
 *====================================================================
 *<Called routines>
 *    None
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void bufrgetind(int file, int *fin, unsigned long *status)
 *    Input:
 *       file                - indicates which file contains descriptors
 *       *fin                - indicator of how many expansions were done
 *       *status             - error return info
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    02/02/94  CSC Monterey, CA   Mugur Georgescu
 *              Developed the original source code.
 *    05/31/94  PL Hanscom AFB, MA   Rodger Biasca
 *              Modified the original.
 *    02/01/95  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Adapted the modified code to BLIRB BUFR decode use.
 *====================================================================
 *<End>
 *********************************************************************
 */
```

```
void bufrgetind(int file, int *fin, unsigned long *status)
{
  long i;                          /* index in mess_desc array          */
  int j;                           /* index in descriptors array        */
  int k;                           /* index in data sets                */
  long a;                          /* for loop counter                  */
  int found;                       /* flag if element descriptor found  */
  int f;                           /* descriptor type:
                                        0 - element descriptor,
                                        1 - replication descriptor,
                                        2 - operator descriptor,
                                        3 - sequence descriptor.        */
  int x;                           /* last 6 bits of first byte in descr*/
  unsigned long y;                 /* the second byte in a descriptor   */
  long newcounter;                 /* the new counter after expansion   */
  FILE *fp1, *fp2;                 /* pointers to temp descriptor files */
  long rep[20];                    /* temp store replicated descriptors */
  int ii, ft, xt, yt;

  i = (long)0;
  j = k = 0;

  *status = (unsigned long)(-1); /* set status to OK                    */
  *fin = 0;                        /* initialize chg of descriptor count*/

  if(file == 1)
  { fp1 = fopen(temp30, "rb");     /* get temp descriptor input file    */
    fp2 = fopen(temp31, "wb");     /*   and temp descriptor output file */
  }
  else if(file == 0)
  { fp1 = fopen(temp31, "rb");
    fp2 = fopen(temp30, "wb");
  }
  else
  { printf("file = %d (an unacceptable value).\n", file);
    printf("Program Aborting!\n");
    fclose(fpo);
    fclose(fp);
    remove(temp30);
    remove(temp31);
    exit(-1);
  }

  newcounter = (long)0;            /* set temp counter to initial value */

  while(k < dt_sets)               /* loop through all the records      */
  { while(i < count_desc)          /* get thru all message descriptors  */
    { fread(&mess_desc, sizeof(long), 1, fp1);
      i++;
      if(i % (long)100 == (long)0)
      { printf("*");
        if(i % (long)7000 == (long)0)
          printf("\n   ");
        else if(i % (long)1000 == (long)0)
          printf(" ");
      }

      f = (int)(mess_desc / (long)100000);   /* type descriptor        */

      switch (f)                   /* switch on descriptor type         */
      { case 0:                    /* element descriptor                */
          j = 0;                   /* set index in descriptors to 0     */
```

```
          found = 0;              /* set found to not true              */

/*-----------------------------------------------------------------------
 * --- search descriptors array till find a match for the descriptor
 *     that came in the message or exceed the size of descriptors array
 *-----------------------------------------------------------------------
 */
          while(!found)
          { if(j > counter)       /* descriptor not found               */
            { ft = (int)(mess_desc / (long)100000);
              xt = (int)((mess_desc - (long)ft * (long)100000)
                   / (long)1000);
              yt = (int)(mess_desc - (long)ft * (long)100000 -
                   (long)xt * (long)1000);

              printf("Descriptor %d %d %d is UNRECOGNIZED\n", ft, xt,
                     yt);
              *status = (unsigned long) mess_desc;
              return;
            }

            if(mess_desc == tbl_b[j].descriptor)
            { if(compressed)      /* Compressed data error processing  */
              { printf("Error - the data are compressed.\n");
                printf("Program Aborting!\n");
                fclose(fp);
                fclose(fpo);
                fclose(fp1);
                fclose(fp2);
                remove(temp30);
                remove(temp31);
                exit(-1);
              }
              else                /* Descriptor found!                  */
              { fwrite(&mess_desc, sizeof(long), 1, fp2);
                newcounter++;
              }
              found = 1;          /* indicate descriptor found          */
            }
            j++;                  /* increment index in descriptors     */
          }                       /* end while loop                     */
          break;
        case 1:                   /* replication descriptor             */
/*
          printf("\nDescriptor %ld[i=%ld] is a replication\n",
                  mess_desc, i);
*/
          mess_desc -= (long)100000;  /* remove F see the BUFR guide  */
          x = (int)(mess_desc / (long)1000); /* # desc to be replicat */
          y = mess_desc - (long)x * (long)1000; /* # times to replicat*/

          for (j = 0; j < x; j++)     /* get next "x" descriptors     */
          { fread(&rep[j], sizeof(long), 1, fp1);
            i++;                  /* increment the index in this array */
            if(i % (long)100 == (long)0)
            { printf("*");
              if(i % (long)7000 == (long)0)
                printf("\n   ");
              else if(i % (long)1000 == (long)0)
                printf(" ");
            }
          }
```

```
          for (ii = 0; ii < (int)y; ii++)     /* replicate y times      */
          { for (j = 0; j < x; j++)    /* the "x" descriptors          */
            { fwrite(&rep[j], sizeof(long), 1, fp2);
              newcounter++;
            }
          }
          break;
        case 2:                    /* operator descriptor              */
/*
          printf("\n Descriptor %ld[i=%ld] is an OPERAND\n",
                  mess_desc, i);
*/
          fwrite(&mess_desc, sizeof(long), 1, fp2);
          newcounter++;
          break;
        case 3:                    /* sequence descriptor              */
/*
          printf("\n Descriptor  %ld[i=%ld] is a sequence\n",
                  mess_desc, i);
*/
          j = 0;                   /* set index in sequences to 0      */
          found = 0;               /* set found to not true            */

/*-----------------------------------------------------------------------
 * --- Search in tbl_d array till find a match for the sequence that
 *     came in the message or exceed the size of the tbl_d array
 *-----------------------------------------------------------------------
 */
          while(!found)
          { if(j > seq_count)      /* descriptor not found             */
/*-----------------------------------------------------------------------
 * --- if the size of the sequences array was exceeded and no match was
 *     found return the message sequence as not being implemented
 *-----------------------------------------------------------------------
 */
            { printf( "\nDescriptor %ld is UNRECOGNIZED\n", mess_desc);
              *status = (unsigned long) mess_desc;
              return;
            }

            if(mess_desc == tbl_d[j].sequence)     /* match found      */
            { for (a = (long)1; a < tbl_d[j].seq_expand; a++)
              { fwrite(&seq_desc[tbl_d[j].seq_index + a], sizeof(long),
                       1, fp2);
                newcounter++;      /* expand sequence                  */
              }
              found = 1;           /* set found flag to true           */
            }                      /* end if match found               */
            j++;                   /* increment the sequences index    */
          }                        /* end while loop                   */
          break;
      }                            /* end switch on descriptor type    */
    }                              /* end while loop on message descrip */

    *fin += (int)(newcounter - count_desc);  /* update descr count chg*/
    count_desc = newcounter;       /* update # descriptors sent to file */

    k++;                           /* stay in loop till last record    */
    i = (long)0;                   /* reset pointer to first descriptor */
  }                                /* end while loop on data sets      */
  fclose(fp1);                     /* close the temp descriptor files  */
  fclose(fp2);
```

```
}
      /* end bufrgetind() */

/*********************************************************************
 *                          VOID GET_DATA
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  get_data
 *                          Type:  C void
 *                      Filename:  blirb_de.c
 *                        Parent:  main
 *====================================================================
 *<Description>
 *    Interprets the individual descriptors and gets the BUFR section 4
 *    data processed
 *====================================================================
 *<Called routines>
 *    bufruncomp            - gets the section 4 data for processing
 *    gbyte                 - extracts specified bits from BUFR stream
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void get_data(void)
 *    Input:
 *       None
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    02/01/95  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Adapted the "bufrgetind" function for a new purpose.
 *====================================================================
 *<End>
 *********************************************************************
 */
void get_data(void)
{
  long i;                        /* index in mess_desc array          */
  int k;                         /* index in data sets                */
  int found;                     /* flag if element descriptor found  */
  int f;                         /* descriptor type:
                                      0 - element descriptor,
                                      1 - replication descriptor,
                                      2 - operator descriptor,
                                      3 - sequence descriptor.        */
  int x;                         /* last 6 bits of first byte in descr*/
  unsigned long y;               /* the second byte in a descriptor   */
  static struct operation oper;  /* struct operation                  */
  long operand;                  /* operand descriptor                */
  FILE *fp3;                     /* pointer for descriptor file       */
  unsigned long something;
  long ref_chg_val[100], ref_chg_desc[100];
  char dummy[5];
  int desc_cnt, ref_chg_flag, j;

  i = (long)0;
  k = 0;
  sect4_bits = (long)8 * (long)lMessageSize;  /* # bits BLIRB Sec 4   */
  desc_cnt = ref_chg_flag = 0;

  remove(temp31);                /* remove 2nd temp descriptor file   */
  fp3 = fopen(temp30, "rb");     /* open primary temp descriptor file */
```

```
while(k < dt_sets)              /* loop through all the records      */
{ oper.dt_width_op = 0;         /* initialize operator fields        */
  oper.scale_op = 0;
  oper.ref_val_op = (long)0;
  oper.assoc_fld = (long)0;
  oper.assoc_width = (long)0;

  while(i < count_desc)         /* get thru all message descriptors  */
  { fread(&mess_desc, sizeof(long), 1, fp3);
    f = (int)(mess_desc / (long)100000);   /* type descriptor       */
    i++;
    if(i % (long)100 == (long)0)
    { printf("*");
      if(i % (long)7000 == (long)0)
        printf("\n   ");
      else if(i % (long)1000 == (long)0)
        printf(" ");
    }

    switch (f)                  /* switch on descriptor type         */
    { case 0:                   /* element descriptor                */
        if(!ref_chg_flag)       /* if ref_chg_flag is false          */
        { if(desc_cnt)          /* check for "ref_val_op" value      */
          { for(j = 0; j < desc_cnt; j++)
            { if(ref_chg_desc[j] == mess_desc)
              { oper.ref_val_op = ref_chg_val[j];
                break;
              }
            }
          }
          bufruncomp(&oper);    /* decode the data                   */
        }
        else                    /* if ref_chg_flag is true           */
        { ref_chg_desc[desc_cnt] = mess_desc;
          gbyte(&something, (int)oper.ref_val_op); /* get modify ref*/
          sect4_bits -= oper.ref_val_op;  /* decrement Sec 4 bit cnt*/
          ref_chg_val[desc_cnt] = (long)something; /* ref chg value */
/*
          printf("     New Reference Value = %ld\n",
                 ref_chg_val[desc_cnt]);
*/
          oper.ref_val_op = (long)0;      /* reset ref_val_op       */
          desc_cnt++;
        }
        break;
      case 1:                   /* replication descriptor            */
        printf("\nDescriptor %ld[i=%ld] is a replication\n",
                mess_desc, i);
        printf("This should not happen now.\n");
        break;
      case 2:                   /* operator descriptor               */
/*
        printf("\n Descriptor %ld[i=%ld] is an OPERAND\n", mess_desc,
                i);
*/
        operand = mess_desc - (long)200000;  /* remove F            */
        x = (int)(operand / (long)1000);     /* get type of operand */
        y = operand - (long)(x * 1000);      /* value of operation  */

        switch (x)              /* switch on operation type          */
        { case 1:               /* change of data width              */
            if(!y)              /* y = 0                             */
```

```
                oper.dt_width_op = 0;      /* cancel change data width */
              else                  /* y different than 0                */
                oper.dt_width_op = (int)y - 128;  /* change data width*/
/*
              printf("     Data Width operator = %d\n",
                     oper.dt_width_op);
*/
              break;
            case 2:                 /* change scale                      */
              if(!y)                /* y = 0                             */
                oper.scale_op = 0;        /* cancel change scale         */
              else                  /* y different than 0                */
                oper.scale_op = (int)y - 128;     /* change scale        */
/*
              printf("     Data Scale operator = %d\n", oper.scale_op);
*/
              break;
            case 3:                 /* change reference value            */
              if(!y)                /* y = 0                             */
              { desc_cnt = 0;       /* kill all reference changes        */
                ref_chg_flag = 0;
                oper.ref_val_op = (long)0;    /* cancel chg ref value */
              }
              else if(y == (long)255)  /* terminate ref chg definition*/
                ref_chg_flag = 0;
              else                  /* initiate change of reference value*/
              { ref_chg_flag = 1;       /* set ref_chg_flag to true    */
                oper.ref_val_op = y;    /* get data width of new ref   */
/*
                printf("     Reference Value Width = %ld\n",
                       oper.ref_val_op);
*/
              }
              break;
            case 4:                 /* associated fields                 */
/*-----------------------------------------------------------------------
 * --- Do not store it yet. Only read it and then skip to the data
 *     itself.  Later will decide on whether need to store it or not.
 *-----------------------------------------------------------------------
 */
              if(!y)                /* y = 0                             */
                oper.assoc_fld = (long)0; /* cancel associated fields */
              else                  /* y not 0                           */
              { fread(&mess_desc, sizeof(long), 1, fp3);
                i++;
                if(i % (long)100 == (long)0)
                { printf("*");
                  if(i % (long)7000 == (long)0)
                    printf("\n   ");
                  else if(i % (long)1000 == (long)0)
                    printf(" ");
                }

                oper.assoc_width = (long)y; /* set width of assoc fld */
                gbyte(&something, 6);
                sect4_bits -= (long)6; /* decrement Sect 4 bit count  */
                oper.assoc_fld = (long)something;  /* signif assoc fld*/
              }
              break;
            case 5:                 /* included character data           */
              for (j = 0; j < (int)y; j++)
              { gbyte(&something, 8);  /* skip over it                   */
```

```
            sect4_bits -= (long)8; /* decrement Sect 4 bit count  */
          }
          break;
        case 6:                  /* local descriptor follows          */
          gbyte(&something, (int)y); /* skip over it                  */
          sect4_bits -= (long)y;   /* decrement Sect 4 bit count  */
          fread(&mess_desc, sizeof(long), 1, fp3);
          i++;
          if(i % (long)100 == (long)0)
          { printf("*");
            if(i % (long)7000 == (long)0)
              printf("\n   ");
            else if(i % (long)1000 == (long)0)
              printf(" ");
          }
          break;
      }                          /* end switch on operation type      */
      break;
    case 3:                      /* sequence descriptor               */
      printf("\n Descriptor  %ld[i=%ld] is a sequence\n",
              mess_desc, i);
      printf("This should not happen now.\n");
      break;
    }                            /* end switch on descriptor type     */
  }                              /* end while loop on message descrip */

  k++;                           /* stay in loop till last record     */
  i = (long)0;                   /* reset pointer to first descriptor */
}                                /* end while loop on data sets       */
fclose(fp3);                     /* close the temp descriptor file    */
remove(temp30);                  /* remove it                         */

write_file(dummy, (long)(-1)); /* indicate all descriptors used     */
}
      /* end get_data() */

/*********************************************************************
 *                          VOID BUFRUNCOMP
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  bufruncomp
 *                          Type:  C void
 *                      Filename:  blirb_de.c
 *                        Parent:  get_data
 *====================================================================
 *<Description>
 *    Extracts the uncompressed data in BUFR Section 4.
 *====================================================================
 *<Called routines>
 *    gbyte                 - extracts specified bits from BUFR stream
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void bufruncomp(struct operation *oper)
 *    Input:
 *       *oper              - current BUFR operation structure
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    02/02/94  CSC Monterey, CA   Mugur Georgescu
 *              Developed the original source code.
```

```
 *    05/31/94  PL Hanscom AFB, MA   Rodger Biasca
 *              Modified the original.
 *    02/01/95  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Adapted the modified code to BLIRB BUFR decode use.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void bufruncomp(struct operation *oper)
{
  unsigned long something;        /* value extracted from message      */
  double denominator;             /* denominator                       */
  long nominator;                 /* numerator                         */
  int temppow;                    /* power of 10 for scale factor      */
  unsigned long assoc_val;        /* the value in the associated field */
  char tempascii[81];             /* temp char array store ASCII data  */
  int i, sign, width, maxlen, indice;

  assoc_val = (unsigned long)0;  /* initialize associated field       */

  for (indice = 0; indice < counter; indice++)
    if(mess_desc == tbl_b[indice].descriptor)
      break;

  if(strcmp(tbl_b[indice].units, "CCITT_IA5") == 0)  /* it is ASCII   */
  { maxlen = tbl_b[indice].data_width / 8;
    for (i = 0; i < maxlen; i++)
    { gbyte(&something, 8);
      sect4_bits -= (long)8;      /* decrement Sect 4 bit count        */
      if(i < 80)
        tempascii[i] = (char)something;
    }
/*
    printf("  %8lu  %8ld  %4d  =  %4s %9s\n", something,
            tbl_b[indice].ref_val,
            tbl_b[indice].scale + oper->scale_op, tempascii,
            tbl_b[indice].units);
*/
    write_file(tempascii, mess_desc);  /* write data to output file   */
  }
  else                            /* it is not ASCII                   */
  { if(oper->assoc_fld)           /* there is assoc field before data  */
    { gbyte(&assoc_val, (int)oper->assoc_width); /* get assoc field   */
      sect4_bits -= oper->assoc_width;  /* decrement Sect 4 bit count */
    }

    something = (unsigned long)0;
    width = tbl_b[indice].data_width + oper->dt_width_op;
    gbyte(&something, width);     /* get the data from Sect 4          */
    sect4_bits -= (long) width;   /* decrement Sect 4 bit count        */

    sign = (int)(something >> (width - 1));  /* get any neg sign       */
    something = (something << (32 - width + 1)) >> (32 - width + 1);

    if(something != (unsigned long)MISSING)  /* not missing data       */
    { if(oper->ref_val_op)        /* if reference value is modified    */
        nominator = (long)something + oper->ref_val_op;  /* new ref    */
      else                        /* if reference value is unchanged   */
        nominator = (long)something + tbl_b[indice].ref_val;

      temppow = tbl_b[indice].scale + oper->scale_op; /* get tot scale*/
      if(temppow != 0)            /* if total scale is not 0           */
```

```
      { denominator = pow((double)10, (double)temppow);
        bufr_data = (float)((double)nominator / denominator); /* apply*/
      }
      else                          /* if total scale is 0, forget it    */
        bufr_data = (float)nominator;

      if(sign)                      /* if negative sign present          */
        bufr_data = -bufr_data;     /* apply it to the data              */
    }
    else                            /* if data is missing                */
      bufr_data = (float)0.0;       /* set BLIRB data to 0               */
/*
    printf("  %8lu  %8ld  %4d  =  %12.5e %9s\n", something,
            tbl_b[indice].ref_val,
            tbl_b[indice].scale + oper->scale_op, bufr_data,
            tbl_b[indice].units);

    if(assoc_val == (unsigned long)0)
      printf("\n");
    else
      printf("   Associated field = %lu\n", assoc_val);
*/
    write_file(tempascii, mess_desc);   /* write data to output file  */
  }

  if(sect4_bits <= (long)0)       /* check for remaining Sect 4 bits   */
    write_file(tempascii, (long)(-2));  /* if so, indicate no data    */
}
       /* end bufruncomp() */

/*********************************************************************
 *                          VOID WRITE_FILE
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  write_file
 *                          Type:  C void
 *                      Filename:  blirb_de.c
 *                        Parent:  bufruncomp, get_data
 *====================================================================
 *<Description>
 *    Writes the BLIRB data to an ASCII file.
 *====================================================================
 *<Called routines>
 *    None.
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void write_file(char *string, long index)
 *    Input:
 *       *string            - pointer to a string of ASCII characters
 *       index              - data element Table B descriptor
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    02/01/95  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void write_file(char *string, long index)
```

```
{
  static int ch_desc[16] = {  0,  6, 11, 25, 31, 39, 42, 45, 48, 52, 59,
                             63, 69, 75, 79, 81 }; /* char data indice*/
  static int ch_cnt = 16;          /* num of types of BLIRB input cards */
  static int ch_last[16] = {  5, 10, 17, 30, 38, 41, 44, 47, 51, 58, 62,
                             68, 74, 78, 80, 82 }; /*end BLIRB inp crd*/
  static int in_desc[7] = { 18, 19, 20, 21, 22, 23, 24 }; /* int input*/
  static int in_cnt = 7;           /* number of int input data indices  */
  static int out_desc[5] = { 0, 1, 2, 3, 4 };  /* integer output data */
  static int out_cnt = 5;          /* number of int output data indices */
  static int max_desc = YYM;       /* maximum BLIRB index               */
  static long x0 = (long)(BB * 1000); /* x0 from Table B for BLIRB     */
  static long cnt = (long)0;       /* data element count for each group */
  static long item = (long)0;      /* current group data element count  */
  static long total = (long)0;     /* total data element count          */
  static int itmod = 0;            /* index to indicate print/no print  */
  static int rep[7];               /* array for integer BLIRB data      */
  static float data[7];            /* array for floating BLIRB data     */
  static char ident[5];            /* array for BLIRB char data         */
  static int out_wave, out_imx[3], na1, itn1, albd1, isc1;
  int i, j, desc;

  if(index < (long)0)              /* if index is negative              */
  { desc = (int)index;             /* use it as is to indicate no data  */
    printf("\n  There were %ld data elements processed.\n\n", total);
  }
  else                             /* if index is >= 0                  */
  { desc = (int)(index - x0);      /* get the y0 portion                */
    if(desc < YYO)
      desc -= YYI;                 /* adjust BLIRB input for Table B    */
    total++;                       /* increment total data element count*/
  }

  for (j = 0; j < ch_cnt; j++)     /* check for char input data index   */
    if(desc == ch_desc[j])
    { for (i = 0; i < 4; i++)
        ident[i] = string[i];      /* if found, save it for later       */
      if(desc == 69)               /* if "SUN", add a blank & get "SUN "*/
        ident[3] = (char)32;
      return;
    }

  for (j = 0; j < in_cnt; j++)     /* check for integer input data index*/
    if(desc == in_desc[j])
    { rep[item] = (int)(bufr_data + 0.001);   /* if found, save it    */
      item++;                      /* increment group data count        */
      itmod++;                     /* increment the print index         */

      if(desc == 23)               /* get the number of ALBD cards      */
        albd1 = rep[item - (long)1];
      else if(desc == 24)          /* if last of multiple input crd cnts*/
      { fprintf(fpo, "%6d%6d%6d%6d%6d%6d%6d\n", rep[0], rep[1],
                rep[2], rep[3], rep[4], rep[5], rep[6]); /* print data*/
        item = (long)0;            /* reinitialize group data count     */
        itmod = 0;                 /* reinitialize the print index      */
      }
      return;
    }

  if(desc < 0 && !itmod)           /* if no more data and all printed   */
  { fclose(fpo);                   /* close output file                 */
    fclose(fp);                    /* close input file                  */
```

```
    exit(0);                        /* exit program                        */
  }

  data[itmod] = bufr_data;          /* save the data in a temp array       */
  item++;                           /* increment group data count          */

  if(desc > 0 && desc <= ch_last[ch_cnt - 1]) /* if BLIRB input card  */
  { itmod++;                        /* increment print index               */
    if(desc == 64)                  /* value of "ISC" + 1 from DOMD card */
      isc1 = (int)(data[itmod - 1] + 1.0);

    for (j = 0; j < ch_cnt; j++)
    { if(desc == ch_last[j])        /* check for last data on input card */
      { if(desc == 78)              /* get the number of wavelengths       */
        { out_wave = (int)data[2];
          data[2] = (data[1] - data[0]) / data[2];
        }

        switch (itmod)              /* if last data on input card, print */
        { case 1:
            fprintf(fpo, "%s      %10.3e\n", ident, data[0]);
            break;
          case 2:
            fprintf(fpo, "%s      %10.3e%10.3e\n", ident, data[0],
                 data[1]);
            break;
          case 3:
            fprintf(fpo, "%s      %10.3e%10.3e%10.3e\n", ident, data[0],
                 data[1], data[2]);
            break;
          case 4:
            fprintf(fpo, "%s      %10.3e%10.3e%10.3e%10.3e\n", ident,
                 data[0], data[1], data[2], data[3]);
            break;
          case 5:
            fprintf(fpo, "%s      %10.3e%10.3e%10.3e%10.3e%10.3e\n",
                 ident, data[0], data[1], data[2], data[3], data[4]);
            break;
          case 6:
            fprintf(fpo,
                 "%s      %10.3e%10.3e%10.3e%10.3e%10.3e%10.3e\n",
                 ident, data[0], data[1], data[2], data[3], data[4],
                 data[5]);
            break;
          case 7:
            fprintf(fpo,
                 "%s      %10.3e%10.3e%10.3e%10.3e%10.3e%10.3e%10.3e\n",
                 ident, data[0], data[1], data[2], data[3], data[4],
                 data[5], data[6]);
            break;
        }
        item = (long)0;             /* reinitialize group data count       */
        itmod = 0;                  /* reinitialize print index            */
        return;
      }
    }
  }
  else if(desc <= max_desc)         /* BLIRB data, not BLIRB input card  */
  { if(desc >= 0)
      itmod = (int)(item % (long)6); /* get print index                    */
    desc -= YYO;
```

```
for (j = 0; j < out_cnt; j++)  /* check for integer output data   */
  if(desc == out_desc[j])
  { rep[item - (long)1] = (int)(bufr_data + 0.001); /* save it    */

    if(desc == 0)              /* get value of "NA + 1"            */
      na1 = rep[item - (long)1] + 1;
    else if(desc == 1)         /* get value of "ITN + 1"           */
    { itn1 = rep[item - (long)1] + 1;
      fprintf(fpo, "%12d%12d\n", rep[0], rep[1]);
      item = (long)0;          /* reinitialize group data count    */
      itmod = 0;               /* reinitialize print index         */
    }
    else if(desc == 2)         /* get number of X grid sections    */
      out_imx[0] = rep[item - (long)1];
    else if(desc == 3)         /* get number of Y grid sections    */
      out_imx[1] = rep[item - (long)1];
    else if(desc == 4)         /* get number of Z grid sections    */
    { out_imx[2] = rep[item - (long)1];
      fprintf(fpo, "%12d%12d%12d\n", rep[0], rep[1], rep[2]);
      item= (long)0;           /* reinitialize group data count    */
      itmod = 0;               /* reinitialize print index         */
    }
    return;
  }

if(cnt == (long)0)             /* determine expected # of data     */
{ if(desc > 4 && desc < 8)                  /* elements in group   */
    cnt = (long)(out_imx[desc - 5] + 1);
  else if(desc == 8)
    cnt = (long)(out_imx[0] * out_imx[1]);
  else if(desc == 9)
    cnt = (long)(out_imx[0] * out_imx[1] * out_imx[2]);
  else if(desc == 10)
    cnt = (long)na1;
  else if(desc == 11)
    cnt = (long)albd1;
  else if(desc > 11 && desc < 15)
    cnt = (long)itn1;
  else if(desc > 14 && desc < 17)
    cnt = (long)(4 * itn1);
  else if(desc == 17)
    cnt = (long)(isc1 * itn1);
  else if(desc > 17)
    cnt = (long)(out_imx[0] * out_imx[1] * out_imx[2]);
}

if(!itmod || ((item == cnt) && (desc < 20))
          || ((item == (long)8 * cnt) && (desc == YYM - YYO))
          || desc < 0)         /* determine whether to print or not */
{ switch (itmod)
  { case 0:
      fprintf(fpo, "%12.4e%12.4e%12.4e%12.4e%12.4e%12.4e\n",
              data[0], data[1], data[2], data[3], data[4],
              data[5]);
      break;
    case 1:
      fprintf(fpo, "%12.4e\n", data[0]);
      break;
    case 2:
      fprintf(fpo, "%12.4e%12.4e\n", data[0], data[1]);
      break;
    case 3:
```

```
            fprintf(fpo, "%12.4e%12.4e%12.4e\n", data[0], data[1],
                    data[2]);
            break;
          case 4:
            fprintf(fpo, "%12.4e%12.4e%12.4e%12.4e\n", data[0], data[1],
                    data[2], data[3]);
            break;
          case 5:
            fprintf(fpo, "%12.4e%12.4e%12.4e%12.4e%12.4e\n", data[0],
                    data[1], data[2], data[3], data[4]);
            break;
        }
        itmod = 0;                  /* reinitialize print index          */

        if(((item == cnt) && (desc < 20)) ||
           ((item == (long)8 * cnt) && (desc == (YYM - YYO)))) /*1st dat*/
        { item = (long)0;           /* reinitialize group data count      */
          cnt = (long)0;            /* reinitialize expected grp data cnt */
        }

        if(desc < 0)                /* if no more data available         */
        { fclose(fpo);              /* close output file                 */
          fclose(fp);               /* close input file                  */
          exit(0);                  /* exit program                      */
        }
      }
    }
    else                            /* descriptor not BLIRB data         */
      printf("Descriptor %ld not recognized!\n", index);
}
      /* end write_file() */
```

Appendix E

Listing of BLIRB_CM.C

```
/*=====================================================================
 * --- Definitions
 *=====================================================================
 */
#define RLEN    190                          /* Input Record Length bytes*/

/*=====================================================================
 * --- Procedure (function) Prototypes.
 *=====================================================================
 */
void main(int argc, char **argv);
void readcards(void);
void readoutput(void);

/*=====================================================================
 * --- Includes and declarations
 *=====================================================================
 */
#include <string.h>
#include <stdio.h>
#include <math.h>
#include <fcntl.h>
#include <ctype.h>
#include <time.h>
#include <stdlib.h>

/*---------------------------------------------------------------------
 * --- Array declarations and assignments
 *---------------------------------------------------------------------
 */
                                      /* Filename Variables            */
FILE *fp1;                                 /* Pointer to Input File 1  */
FILE *fp2;                                 /* Pointer to Input File 2  */

                                      /* BLIRB Input/Output Variables  */
int albd;                                  /* Indicator of ALBD cards  */
float mdl1_model;                          /* The BLIRB Temp model     */
float domd_isc;                            /* Order Spherical Harmonic */
float wavn_v1, wavn_v2, wavn_dv;           /* Wavenumber info          */
int out_imx[3];                            /* Num of X, Y, & Z grid pt */
int out_nwave;                             /* Number of Waves          */

/*********************************************************************
 *                          VOID MAIN
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  main
 *                          Type:  C Main Program
 *                      Filename:  blirb_cm.c
 *                        Parent:  None
 *=====================================================================
 *<Description>
 *    This program compares the original BLIRB output file with the
 *    BUFR decoded version of the file.
 *=====================================================================
 *<Called routines>
 *    readcards             - reads the BLIRB input records from both
 *                            BLIRB output files and compares them
 *=====================================================================
```

```
 *<Parameters>
 *    Formal declaration:
 *       void main(int argc, char **argv)
 *    Input:
 *       argc                - (int) the number of command lines inputs
 *       argv                - (char pointer) the array of command lines
 *                             inputs
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    02/01/95  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 *********************************************************************
 */
void main(int argc, char **argv)
{
  fp1 = fopen(argv[1], "r");
  fp2 = fopen(argv[2], "r");

  readcards();
}      /* end main() */

/*********************************************************************
 *                          VOID READCARDS
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  readcards
 *                          Type:  C void
 *                      Filename:  blirb_cm.c
 *                        Parent:  main
 *=====================================================================
 *<Description>
 *    Reads the BLIRB input records from the two BLIRB output files
 *    and compares the corresponding values.
 *=====================================================================
 *<Called routines>
 *    readoutput            - reads BLIRB output from the output files
 *                            and compares them
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void readcards( void)
 *    Input:
 *       None
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    02/01/95  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 *********************************************************************
 */
void readcards(void)
{
  char cardlabel[5], card[RLEN], c[5];
  char cardlabel1[5], card1[RLEN], c1[5];
  float x[10], x1[10], avg, dif;
```

```
  int y[10], y1[10], i, recl;

  albd = recl = -1;

  do                                          /* From VIEW.RJOB Subroutine*/
/*-----------------------------------------------------------------------
 * --- Read a BLIRB input card from the file.
 *-----------------------------------------------------------------------
 */
  { fgets(card, RLEN, fp1);
    fgets(card1, RLEN, fp2);

/*-----------------------------------------------------------------------
 * --- Get the first 5 characters (card identifier) from the card.
 *-----------------------------------------------------------------------
 */
    sscanf(card, "%s", cardlabel);
    sscanf(card1, "%s", cardlabel1);

/*-----------------------------------------------------------------------
 * --- Process the rest of the card depending upon the identifier.
 *-----------------------------------------------------------------------
 */
    if(strncmp(cardlabel,"MDL1",4) == 0)
    { sscanf(card, "%s%10e%10e%10e%10e%10e", c, &x[0], &x[1], &x[2],
             &x[3], &x[4]);
      mdl1_model = x[1];

      if(strncmp(cardlabel1,"MDL1",4) == 0)
        sscanf(card1, "%s%10e%10e%10e%10e%10e", c1, &x1[0], &x1[1],
               &x1[2], &x1[3], &x1[4]);

      for (i = 0; i < 5; i++)
      { avg = 0.5 * (x1[i] + x[i]);
        avg = (float)fabs((double)avg);
        dif = x1[i] - x[i];
        dif = (float)fabs((double)dif);
        if(fabs((double) dif > (double) (0.001 * avg)))
          printf("MDL1: x1[%d] = %12e, x2[%d] = %12e\n", i, x[i], i,
                 x1[i]);
      }
    }

    else if(strncmp(cardlabel,"MDL2",4) == 0)
    { sscanf(card, "%s%10e%10e%10e%10e", c, &x[0], &x[1], &x[2], &x[3]);

      if(strncmp(cardlabel1,"MDL2",4) == 0)
        sscanf(card1, "%s%10e%10e%10e%10e", c1, &x1[0], &x1[1], &x1[2],
               &x1[3]);

      for (i = 0; i < 4; i++)
      { avg = 0.5 * (x1[i] + x[i]);
        avg = (float)fabs((double)avg);
        dif = x1[i] - x[i];
        dif = (float)fabs((double)dif);
        if(fabs((double) dif > (double) (0.001 * avg)))
          printf("MDL2: x1[%d] = %12e, x2[%d] = %12e\n", i, x[i], i,
                 x1[i]);
      }

      if((int)mdl1_model != 7)
      { fgets(card, RLEN, fp1);
```

```
      sscanf(card, "%ld%ld%ld%ld%ld%ld%ld", &y[0], &y[1], &y[2],
             &y[3], &y[4], &y[5], &y[6]);

      fgets(card1, RLEN, fp2);
      sscanf(card1, "%ld%ld%ld%ld%ld%ld%ld", &y1[0], &y1[1], &y1[2],
             &y1[3], &y1[4], &y1[5], &y1[6]);

      for (i = 0; i < 7; i++)
        if(abs(y1[i] - y[i]) > 0)
          printf("cnts: y1[%d] = %6d, y2[%d] = %6d\n", i, y[i], i,
                 y1[i]);
    }
  }

  else if(strncmp(cardlabel,"MDL3",4) == 0)
  { sscanf(card, "%s%10e%10e%10e%10e%10e%10e", c, &x[0], &x[1], &x[2],
           &x[3], &x[4], &x[5]);

    if(strncmp(cardlabel1,"MDL3",4) == 0)
      sscanf(card1, "%s%10e%10e%10e%10e%10e%10e", c1, &x1[0], &x1[1],
             &x1[2], &x1[3], &x1[4], &x1[5]);

    for (i = 0; i < 6; i++)
    { avg = 0.5 * (x1[i] + x[i]);
      avg = (float)fabs((double)avg);
      dif = x1[i] - x[i];
      dif = (float)fabs((double)dif);
      if(fabs((double) dif > (double) (0.001 * avg)))
        printf("MDL3: x1[%d] = %12e, x2[%d] = %12e\n", i, x[i], i,
               x1[i]);
    }

    fgets(card, RLEN, fp1);
    sscanf(card, "%ld%ld%ld%ld%ld%ld%ld", &y[0], &y[1], &y[2], &y[3],
           &y[4], &y[5], &y[6]);

    fgets(card1, RLEN, fp2);
    sscanf(card1, "%ld%ld%ld%ld%ld%ld%ld", &y1[0], &y1[1], &y1[2],
           &y1[3], &y1[4], &y1[5], &y1[6]);

    for (i = 0; i < 7; i++)
      if(abs(y1[i] - y[i]) > 0)
        printf("cnts: y1[%d] = %6d, y2[%d] = %6d\n", i, y[i], i,
               y1[i]);
  }

  else if(strncmp(cardlabel,"AREA",4) == 0)
  { sscanf(card, "%s%10e%10e%10e%10e%10e", c, &x[0], &x[1], &x[2],
           &x[3], &x[4]);

    if(strncmp(cardlabel1,"AREA",4) == 0)
      sscanf(card1, "%s%10e%10e%10e%10e%10e", c1, &x1[0], &x1[1],
             &x1[2], &x1[3], &x1[4]);

    for (i = 0; i < 5; i++)
    { avg = 0.5 * (x1[i] + x[i]);
      avg = (float)fabs((double)avg);
      dif = x1[i] - x[i];
      dif = (float)fabs((double)dif);
      if(fabs((double) dif > (double) (0.001 * avg)))
        printf("AREA: x1[%d] = %12e, x2[%d] = %12e\n", i, x[i], i,
               x1[i]);
```

```
    }
  }

  else if(strncmp(cardlabel,"REGN",4) == 0)
  { sscanf(card, "%s%10e%10e%10e%10e%10e%10e%10e", c, &x[0], &x[1],
           &x[2], &x[3], &x[4], &x[5], &x[6]);

    if(strncmp(cardlabel1,"REGN",4) == 0)
      sscanf(card1, "%s%10e%10e%10e%10e%10e%10e%10e", c1, &x1[0],
             &x1[1], &x1[2], &x1[3], &x1[4], &x1[5], &x1[6]);

    for (i = 0; i < 7; i++)
    { avg = 0.5 * (x1[i] + x[i]);
      avg = (float)fabs((double)avg);
      dif = x1[i] - x[i];
      dif = (float)fabs((double)dif);
      if(fabs((double) dif > (double) (0.001 * avg)))
        printf("REGN: x1[%d] = %12e, x2[%d] = %12e\n", i, x[i], i,
                x1[i]);
    }
  }

  else if(strncmp(cardlabel,"MESX",4) == 0)
  { sscanf(card, "%s%10e%10e", c, &x[0], &x[1]);

    if(strncmp(cardlabel1,"MESX",4) == 0)
      sscanf(card1, "%s%10e%10e", c1, &x1[0], &x1[1]);

    for (i = 0; i < 2; i++)
    { avg = 0.5 * (x1[i] + x[i]);
      avg = (float)fabs((double)avg);
      dif = x1[i] - x[i];
      dif = (float)fabs((double)dif);
      if(fabs((double) dif > (double) (0.001 * avg)))
        printf("MESX: x1[%d] = %12e, x2[%d] = %12e\n", i, x[i], i,
                x1[i]);
    }
  }

  else if(strncmp(cardlabel,"MESY",4) == 0)
  { sscanf(card, "%s%10e%10e", c, &x[0], &x[1]);

    if(strncmp(cardlabel1,"MESY",4) == 0)
      sscanf(card1, "%s%10e%10e", c1, &x1[0], &x1[1]);

    for (i = 0; i < 2; i++)
    { avg = 0.5 * (x1[i] + x[i]);
      avg = (float)fabs((double)avg);
      dif = x1[i] - x[i];
      dif = (float)fabs((double)dif);
      if(fabs((double) dif > (double) (0.001 * avg)))
        printf("MESY: x1[%d] = %12e, x2[%d] = %12e\n", i, x[i], i,
                x1[i]);
    }
  }

  else if(strncmp(cardlabel,"MESZ",4) == 0)
  { sscanf(card, "%s%10e%10e", c, &x[0], &x[1]);

    if(strncmp(cardlabel1,"MESZ",4) == 0)
      sscanf(card1, "%s%10e%10e", c1, &x1[0], &x1[1]);
```

```
    for (i = 0; i < 2; i++)
    { avg = 0.5 * (x1[i] + x[i]);
      avg = (float)fabs((double)avg);
      dif = x1[i] - x[i];
      dif = (float)fabs((double)dif);
      if(fabs((double) dif > (double) (0.001 * avg)))
        printf("MESZ: x1[%d] = %12e, x2[%d] = %12e\n", i, x[i], i,
                x1[i]);
    }
  }

  else if(strncmp(cardlabel,"ALBD",4) == 0)
  { sscanf(card, "%s%10e%10e%10e", c, &x[0], &x[1], &x[2]);
    albd++;

    if(strncmp(cardlabel1,"ALBD",4) == 0)
      sscanf(card1, "%s%10e%10e%10e", c1, &x1[0], &x1[1], &x1[2]);

    for (i = 0; i < 3; i++)
    { avg = 0.5 * (x1[i] + x[i]);
      avg = (float)fabs((double)avg);
      dif = x1[i] - x[i];
      dif = (float)fabs((double)dif);
      if(fabs((double) dif > (double) (0.001 * avg)))
        printf("ALBD: x1[%d] = %12e, x2[%d] = %12e\n", i, x[i], i,
                x1[i]);
    }
  }

  else if(strncmp(cardlabel,"MTRL",4) == 0)
  { sscanf(card,"%s%10e%10e%10e%10e%10e%10e", c, &x[0], &x[1],
           &x[2], &x[3], &x[4], &x[5]);

    if(strncmp(cardlabel1,"MTRL",4) == 0)
      sscanf(card1, "%s%10e%10e%10e%10e%10e%10e", c1, &x1[0], &x1[1],
             &x1[2], &x1[3], &x1[4], &x1[5]);

    for (i = 0; i < 6; i++)
    { avg = 0.5 * (x1[i] + x[i]);
      avg = (float)fabs((double)avg);
      dif = x1[i] - x[i];
      dif = (float)fabs((double)dif);
      if(fabs((double) dif > (double) (0.001 * avg)))
        printf("MTRL: x1[%d] = %12e, x2[%d] = %12e\n", i, x[i], i,
                x1[i]);
    }
  }

  else if(strncmp(cardlabel,"CLDS",4) == 0)
  { sscanf(card, "%s%10e%10e%10e%10e", c, &x[0], &x[1], &x[2], &x[3]);

    if(strncmp(cardlabel1,"CLDS",4) == 0)
      sscanf(card1, "%s%10e%10e%10e%10e", c1, &x1[0], &x1[1], &x1[2],
             &x1[3]);

    for (i = 0; i < 4; i++)
    { avg = 0.5 * (x1[i] + x[i]);
      avg = (float)fabs((double)avg);
      dif = x1[i] - x[i];
      dif = (float)fabs((double)dif);
      if(fabs((double) dif > (double) (0.001 * avg)))
        printf("CLDS: x1[%d] = %12e, x2[%d] = %12e\n", i, x[i], i,
```

```
                x1[i]);
    }
  }

  else if(strncmp(cardlabel,"DOMD",4) == 0)
  { sscanf(card, "%s%10e%10e%10e%10e%10e", c, &x[0], &x[1], &x[2],
           &x[3], &x[4]);
    domd_isc = x[0];

    if(strncmp(cardlabel1,"DOMD",4) == 0)
      sscanf(card1, "%s%10e%10e%10e%10e%10e", c1, &x1[0], &x1[1],
             &x1[2], &x1[3], &x1[4]);

    for (i = 0; i < 5; i++)
    { avg = 0.5 * (x1[i] + x[i]);
      avg = (float)fabs((double)avg);
      dif = x1[i] - x[i];
      dif = (float)fabs((double)dif);
      if(fabs((double) dif > (double) (0.001 * avg)))
        printf("DOMD: x1[%d] = %12e, x2[%d] = %12e\n", i, x[i], i,
                x1[i]);
    }
  }

  else if(strncmp(cardlabel,"SUN",3) == 0)
  { sscanf(card, "%s%10e%10e%10e%10e%10e", c, &x[0], &x[1], &x[2],
           &x[3], &x[4]);

    if(strncmp(cardlabel1,"SUN",3) == 0)
      sscanf(card1, "%s%10e%10e%10e%10e%10e", c1, &x1[0], &x1[1],
             &x1[2], &x1[3], &x1[4]);

    for (i = 0; i < 5; i++)
    { avg = 0.5 * (x1[i] + x[i]);
      avg = (float)fabs((double)avg);
      dif = x1[i] - x[i];
      dif = (float)fabs((double)dif);
      if(fabs((double) dif > (double) (0.001 * avg)))
        printf("SUN : x1[%d] = %12e, x2[%d] = %12e\n", i, x[i], i,
                x1[i]);
    }
  }

  else if(strncmp(cardlabel,"WAVN",4) == 0)
  { sscanf(card, "%s%10e%10e%10e", c, &x[0], &x[1], &x[2]);
    x[2] = (x[1] - x[0]) / x[2];
    wavn_v1 = x[0];
    wavn_v2 = x[1];
    wavn_dv = x[2];

    if(strncmp(cardlabel1,"WAVN",4) == 0)
      sscanf(card1, "%s%10e%10e%10e", c1, &x1[0], &x1[1], &x1[2]);
    x1[2] = (x1[1] - x1[0]) / x1[2];

    for (i = 0; i < 3; i++)
    { avg = 0.5 * (x1[i] + x[i]);
      avg = (float)fabs((double)avg);
      dif = x1[i] - x[i];
      dif = (float)fabs((double)dif);
      if(fabs((double) dif > (double) (0.001 * avg)))
        printf("WAVN: x1[%d] = %12e, x2[%d] = %12e\n", i, x[i], i,
                x1[i]);
```

```
      }
    }

    else if(strncmp(cardlabel,"ASCI",4) == 0)
    { sscanf(card, "%s%10e", c, &x[0]);

      if(strncmp(cardlabel1,"ASCI",4) == 0)
        sscanf(card1, "%s%10e", c1, &x1[0]);

      avg = 0.5 * (x1[0] + x[0]);
      avg = (float)fabs((double)avg);
      dif = x1[0] - x[0];
      dif = (float)fabs((double)dif);
      if(fabs((double) dif > (double) (0.001 * avg)))
        printf("ASCI: x1 = %12e, x2 = %12e\n", x[0], x1[0]);
    }

    else if(strncmp(cardlabel,"RECL",4) == 0)
    { recl = 0;
      sscanf(card, "%s%10e", c, &x[0]);

      if(strncmp(cardlabel1,"RECL",4) == 0)
        sscanf(card1, "%s%10e", c1, &x1[0]);

      avg = 0.5 * (x1[0] + x[0]);
      avg = (float)fabs((double)avg);
      dif = x1[0] - x[0];
      dif = (float)fabs((double)dif);
      if(fabs((double) dif > (double) (0.001 * avg)))
        printf("RECL: x1 = %12e, x2 = %12e\n", x[0], x1[0]);
    }

    else
      printf(" Record ID %s not identified.\n", cardlabel); /* Unknown*/

  } while (recl != 0);

  readoutput();

/*-----------------------------------------------------------------------
 * --- When finished reading all the data, close the files.
 *-----------------------------------------------------------------------
 */
  fclose(fp1);
  fclose(fp2);
}      /* end readcards() */

/*********************************************************************
 *                          VOID READOUTPUT
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  readoutput
 *                          Type:  C void
 *                      Filename:  blirb_cm.c
 *                        Parent:  readcards
 *=====================================================================
 *<Description>
 *    Reads the BLIRB output data from the BLIRB output files and
 *    compares the corresponding values.
 *=====================================================================
 *<Called routines>
 *    None
```

```
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void readoutput(void)
 *    Input:
 *       None
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    02/01/95  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void readoutput(void)
{
  float dum, *data, *data1, avg, dif;
  int i, k, m, l, na, itn, itn1;
  long j, incr, y[3];

/*---------------------------------------------------------------------
 * --- Determine the number of wavenumbers.
 *---------------------------------------------------------------------
 */
  out_nwave = wavn_dv;

/*---------------------------------------------------------------------
 * --- Read the values of NA and ITN.
 *---------------------------------------------------------------------
 */
  fscanf(fp1, "%d%d", &na, &itn);
  fscanf(fp2, "%d%d", &y[0], &y[1]);

  if(abs(y[0] - na) > 0)
    printf("NA: NA[1] = %d, NA[2] = %d\n", na, y[0]);
  if(abs(y[1] - itn) > 0)
    printf("ITN: ITN[1] = %d, ITN[2] = %d\n", itn, y[1]);
  itn1 = itn + 1;

/*---------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine RJOB, get the X, Y, and
 *     Z BLIRB main region grid points and calculate the flux grid
 *     points from them.
 *---------------------------------------------------------------------
 */
  fscanf(fp1, "%d%d%d", &out_imx[0], &out_imx[1], &out_imx[2]);
  fscanf(fp2, "%d%d%d", &y[0], &y[1], &y[2]);

  for (i = 0; i < 3; i++)
  { if(abs(y[i] - out_imx[i]) > 0)
      printf("NXYZ: y1[%d] = %d, y2[%d] = %d\n", i, out_imx[i], i,
              y[i]);
  }

  dum = (float)(out_imx[0] * out_imx[1] * out_imx[2] + 1);
  if(dum < (na + 1))
    dum = (float)(na + 1);
  if(dum < 4 * itn1)
    dum = (float)(4 * itn1);
  if(dum < (domd_isc + (float)1.0) * (float)itn1)
```

```
    dum = (domd_isc + (float)1.0) * (float)itn1;

  data = (float *) malloc((long)sizeof(float) * (long)dum);
  data1 = (float *) malloc((long)sizeof(float) * (long)dum);

  for (i=0; i<3; i++)
  { if(out_imx[i] > 0)
    { for (j = 0; j<=out_imx[i]; j++)
        fscanf(fp1, "%12e", (data + j));  /* Read X,Y,Z grid points   */

      for (j = 0; j<=out_imx[i]; j++)
        fscanf(fp2, "%12e", (data1 + j));

      for (j = 0; j<=out_imx[i]; j++)
      { avg = 0.5 * (*(data1 + j) + *(data + j));
        avg = (float)fabs((double)avg);
        dif = *(data1 + j) - *(data + j);
        dif = (float)fabs((double)dif);
        if(fabs((double) dif > (double) (0.0001 * avg)))
          printf("IMX%d: data[%ld] = %12e, data1[%ld] = %12e\n", i, j,
                  *(data + j), j, *(data1 + j));
      }
    }
  }

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine RJOB, get the surface
 *     albedo indices at each (X,Y) grid point (ISURF).
 *-----------------------------------------------------------------------
 */
  if(out_imx[0] > 0 && out_imx[1] > 0)
  { for (incr = 0, i = 0; i<out_imx[1]; i++)
      for (j = 0; j<out_imx[0]; incr++, j++)
        fscanf(fp1, "%12e", (data + incr));

    for (incr = 0, i = 0; i<out_imx[1]; i++)
      for (j = 0; j<out_imx[0]; incr++, j++)
        fscanf(fp2, "%12e", (data1 + incr));

    for (j = 0; j < incr; j++)
    { avg = 0.5 * (*(data1 + j) + *(data + j));
      avg = (float)fabs((double)avg);
      dif = *(data1 + j) - *(data + j);
      dif = (float)fabs((double)dif);
      if(fabs((double) dif > (double) (0.0001 * avg)))
        printf("ISURF: data[%ld] = %12e, data1[%ld] = %12e\n", j,
                *(data + j), j, *(data1 + j));
    }
  }

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine RJOB, get the region
 *     material indices at each (X,Y,Z) grid point (IVOLM).
 *-----------------------------------------------------------------------
 */
  if(out_imx[0] > 0 && out_imx[1] > 0 && out_imx[2] > 0)
  { for (incr = 0, k = 0; k<out_imx[2]; k++)
      for (i = 0; i<out_imx[1]; i++)
        for (j = 0; j<out_imx[0]; incr++, j++)
          fscanf(fp1, "%12e", (data + incr));

    for (incr = 0, k = 0; k<out_imx[2]; k++)
```

```
      for (i = 0; i<out_imx[1]; i++)
        for (j = 0; j<out_imx[0]; incr++, j++)
          fscanf(fp2, "%12e", (data1 + incr));

    for (j = 0; j < incr; j++)
    { avg = 0.5 * (*(data1 + j) + *(data + j));
      avg = (float)fabs((double)avg);
      dif = *(data1 + j) - *(data + j);
      dif = (float)fabs((double)dif);
      if(fabs((double) dif > (double) (0.0001 * avg)))
        printf("IVOLM: data[%ld] = %12e, data1[%ld] = %12e\n", j,
                *(data + j), j, *(data1 + j));
    }
  }

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine RFLX, get the LOWTRAN
 *     molecular transmission (TRLW).
 *-----------------------------------------------------------------------
 */
  for (m = 0; m<out_nwave; m++)
  { for (j = 0; j<=na; j++)
      fscanf(fp1, "%12e", (data + j));

    for (j = 0; j<=na; j++)
      fscanf(fp2, "%12e", (data1 + j));

    for (j = 0; j<=na; j++)
    { avg = 0.5 * (*(data1 + j) + *(data + j));
      avg = (float)fabs((double)avg);
      dif = *(data1 + j) - *(data + j);
      dif = (float)fabs((double)dif);
      if(fabs((double) dif > (double) (0.0001 * avg)))
        printf("TRLW: data[%d,%ld] = %12e, data1[%d,%ld] = %12e\n", m,
               j, *(data + j), m, j, *(data1 + j));
    }

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine CLOUDR, get the surface
 *     albedos (SALB).
 *-----------------------------------------------------------------------
 */
    if(albd >= 0)
    { for (j = 0; j<=albd; j++)
        fscanf(fp1, "%12e", (data + j));

      for (j = 0; j<=albd; j++)
        fscanf(fp2, "%12e", (data1 + j));

      for (j = 0; j<=albd; j++)
      { avg = 0.5 * (*(data1 + j) + *(data + j));
        avg = (float)fabs((double)avg);
        dif = *(data1 + j) - *(data + j);
        dif = (float)fabs((double)dif);
        if(fabs((double) dif > (double) (0.0001 * avg)))
          printf("SALB: data[%d,%ld] = %12e, data1[%d,%ld] = %12e\n", m,
                 j, *(data + j), m, j, *(data1 + j));
      }
    }

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine CLOUDR, get the
```

```
 *     extinction coefficients (REXT).
 *-----------------------------------------------------------------------
 */
    for (j = 0; j<itn1; j++)
      fscanf(fp1, "%12e", (data + j));

    for (j = 0; j<itn1; j++)
      fscanf(fp2, "%12e", (data1 + j));

    for (j = 0; j<itn1; j++)
    { avg = 0.5 * (*(data1 + j) + *(data + j));
      avg = (float)fabs((double)avg);
      dif = *(data1 + j) - *(data + j);
      dif = (float)fabs((double)dif);
      if(fabs((double) dif > (double) (0.0001 * avg)))
        printf("REXT: data[%d,%ld] = %12e, data1[%d,%ld] = %12e\n", m,
                j, *(data + j), m, j, *(data1 + j));
    }

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine CLOUDR, get the
 *     scattering coefficients (RSCT).
 *-----------------------------------------------------------------------
 */
    for (j = 0; j<itn1; j++)
      fscanf(fp1, "%12e", (data + j));

    for (j = 0; j<itn1; j++)
      fscanf(fp2, "%12e", (data1 + j));

    for (j = 0; j<itn1; j++)
    { avg = 0.5 * (*(data1 + j) + *(data + j));
      avg = (float)fabs((double)avg);
      dif = *(data1 + j) - *(data + j);
      dif = (float)fabs((double)dif);
      if(fabs((double) dif > (double) (0.0001 * avg)))
        printf("RSCT: data[%d,%ld] = %12e, data1[%d,%ld] = %12e\n", m,
                j, *(data + j), m, j, *(data1 + j));
    }

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine CLOUDR, get the
 *     "unknown" coefficients (FDLT).
 *-----------------------------------------------------------------------
 */
    for (j = 0; j<itn1; j++)
      fscanf(fp1, "%12e", (data + j));

    for (j = 0; j<itn1; j++)
      fscanf(fp2, "%12e", (data1 + j));

    for (j = 0; j<itn1; j++)
    { avg = 0.5 * (*(data1 + j) + *(data + j));
      avg = (float)fabs((double)avg);
      dif = *(data1 + j) - *(data + j);
      dif = (float)fabs((double)dif);
      if(fabs((double) dif > (double) (0.0001 * avg)))
        printf("FDLT: data[%d,%ld] = %12e, data1[%d,%ld] = %12e\n", m,
                j, *(data + j), m, j, *(data1 + j));
    }

/*-----------------------------------------------------------------------
```

```
 * --- Referencing the VIEW program subroutine CLOUDR, get the phase
 *     function angles (AGL).
 *-----------------------------------------------------------------------
 */
    for (incr = 0, j = 0; j<itn1; j++)
      for (i = 0; i<4; incr++, i++)
        fscanf(fp1, "%12e", (data + incr));

    for (incr = 0, j = 0; j<itn1; j++)
      for (i = 0; i<4; incr++, i++)
        fscanf(fp2, "%12e", (data1 + incr));

    for (j = 0; j < incr; j++)
    { avg = 0.5 * (*(data1 + j) + *(data + j));
      avg = (float)fabs((double)avg);
      dif = *(data1 + j) - *(data + j);
      dif = (float)fabs((double)dif);
      if(fabs((double) dif > (double) (0.0001 * avg)))
        printf("AGL : data[%d,%ld] = %12e, data1[%d,%ld] = %12e\n", m,
                j, *(data + j), m, j, *(data1 + j));
    }

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine CLOUDR, get the phase
 *     functions for different materials (PHF).
 *-----------------------------------------------------------------------
 */
    for (incr = 0, j = 0; j<itn1; j++)
      for (i = 0; i<4; incr++, i++)
        fscanf(fp1, "%12e", (data + incr));

    for (incr = 0, j = 0; j<itn1; j++)
      for (i = 0; i<4; incr++, i++)
        fscanf(fp2, "%12e", (data1 + incr));

    for (j = 0; j < incr; j++)
    { avg = 0.5 * (*(data1 + j) + *(data + j));
      avg = (float)fabs((double)avg);
      dif = *(data1 + j) - *(data + j);
      dif = (float)fabs((double)dif);
      if(fabs((double) dif > (double) (0.0001 * avg)))
        printf("PHF : data[%d,%ld] = %12e, data1[%d,%ld] = %12e\n", m,
                j, *(data + j), m, j, *(data1 + j));
    }

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine CLOUDR, get the
 *     Legendre coefficients (RLEG).
 *-----------------------------------------------------------------------
 */
    for (incr = 0, j = 0; j<itn1; j++)
      for (i = 0; i<=(int)domd_isc; incr++, i++)
        fscanf(fp1, "%12e", (data + incr));

    for (incr = 0, j = 0; j<itn1; j++)
      for (i = 0; i<=(int)domd_isc; incr++, i++)
        fscanf(fp2, "%12e", (data1 + incr));

    for (j = 0; j < incr; j++)
    { avg = 0.5 * (*(data1 + j) + *(data + j));
      avg = (float)fabs((double)avg);
      dif = *(data1 + j) - *(data + j);
```

```
      dif = (float)fabs((double)dif);
      if(fabs((double) dif > (double) (0.0001 * avg)))
        printf("RLEG: data[%d,%ld] = %12e, data1[%d,%ld] = %12e\n", m,
                j, *(data + j), m, j, *(data1 + j));
    }

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine RFLX, get the direct
 *     solar flux, reflected solar flux, and 8 diffuse flux values at
 *     each (X,Y,Z) flux grid point.
 *-----------------------------------------------------------------------
 */
    for (l=0; l<10; l++)
    { for (incr = 0, i = 0; i<out_imx[2]; i++)
        for (j = 0; j<out_imx[1]; j++)
          for (k = 0; k<out_imx[0]; incr++, k++)
            fscanf(fp1, "%12e", (data + incr));

      for (incr = 0, i = 0; i<out_imx[2]; i++)
        for (j = 0; j<out_imx[1]; j++)
          for (k = 0; k<out_imx[0]; incr++, k++)
            fscanf(fp2, "%12e", (data1 + incr));

      for (j = 0; j < incr; j++)
      { avg = 0.5 * (*(data1 + j) + *(data + j));
        avg = (float)fabs((double)avg);
        dif = *(data1 + j) - *(data + j);
        dif = (float)fabs((double)dif);
        if(fabs((double) dif > (double) (0.0001 * avg)))
          printf("FLX%d: data[%d,%ld] = %12e, data1[%d,%ld] = %12e\n",
                  l, m, j, *(data + j), m, j, *(data1 + j));
      }
    }
  }

  free(data);
  free(data1);
}      /* end readoutput() */
```

Appendix F

Listing of Table B

F	X	Y	SCALE VALUE	REF VALUE	DATA WIDTH (BITS)	UNITS	ELEMENT NAME
0	0	1	0	0	24	CCITT_IA5	Table_A:_entry
0	0	2	0	0	256	CCITT_IA5	Table_A:_data_category_description,_line_1
0	0	3	0	0	256	CCITT_IA5	Table_A:_data_category_description,_line_2
0	0	5	0	0	24	CCITT_IA5	BUFR_edition_number
0	0	10	0	0	8	CCITT_IA5	F_descriptor_to_be_added_or_defined
0	0	11	0	0	16	CCITT_IA5	X_descriptor_to_be_added_or_defined
0	0	12	0	0	24	CCITT_IA5	Y_descriptor_to_be_added_or_defined
0	0	13	0	0	256	CCITT_IA5	Element_name,_line_1
0	0	14	0	0	256	CCITT_IA5	Element_name,_line_2
0	0	15	0	0	192	CCITT_IA5	Units_name
0	0	16	0	0	8	CCITT_IA5	Units_scale_sign
0	0	17	0	0	24	CCITT_IA5	Units_scale
0	0	18	0	0	8	CCITT_IA5	Units_reference_sign
0	0	19	0	0	80	CCITT_IA5	Units_reference_value
0	0	20	0	0	24	CCITT_IA5	Element_data_width
0	0	30	0	0	40	CCITT_IA5	Descriptor_defining_sequence
0	1	1	0	0	7	numeric	WMO_block_number
0	1	2	0	0	10	numeric	WMO_station_number
0	1	3	0	0	3	numeric	WMO_region_number
0	1	4	0	0	3	numeric	WMO_region_sub-area
0	1	5	0	0	17	numeric	Buoy/platform_identifier
0	1	6	0	0	64	CCITT_IA5	Aircraft_identifier_(flight_number)
0	1	7	0	0	10	code_table	Satellite_identifier
0	1	8	0	0	64	CCITT_IA5	Aircraft_registration_number_(tail_number)
0	1	9	0	0	64	CCITT_IA5	Type_of_commerical_aircraft
0	1	10	0	0	64	CCITT_IA5	Stationary_buoy_platform_identifier
0	1	11	0	0	72	CCITT_IA5	Ship's_call_sign
0	1	12	0	0	9	deg_true	Direction_of_motion_of_moving_observing_platform
0	1	13	0	0	10	m/s	Speed_of_motion_of_moving_observing_platform
0	1	14	2	0	10	m/s	Platform_drift_speed_(high_precision)
0	1	21	0	0	14	numeric	Synoptic_feature_identifier
0	1	25	0	0	24	CCITT_IA5	Storm_identifier
0	1	26	0	0	64	CCITT_IA5	WMO_storm_name
0	1	31	0	0	16	numeric	Generating_center
0	1	50	0	0	48	CCITT_IA5	NMC_report_identifier
0	1	62	0	0	40	CCITT_IA5	National_assigned_station_identifier
0	1	63	0	0	64	CCITT_IA5	ICAO_location_indicator
0	2	1	0	0	2	code_table	Type_of_station
0	2	2	0	0	4	flag_table	Type_of_instrumentation_for_wind_measurement
0	2	3	0	0	4	code_table	Type_of_measuring_instrumentation_used
0	2	4	0	0	4	code_table	Type_of_instrumentation_for_evaporation_measurement_or_ type_of_crop_for_which_evapotranspiration_is_reported
0	2	5	2	0	7	deg	Precision_of_temperature_observed
0	2	11	0	0	8	code_table	Radiosonde_type
0	2	12	0	0	4	code_table	Radiosonde_computational_method
0	2	13	0	0	4	code_table	Solar_and_infrared_radiation_correction
0	2	14	0	0	7	code_table	Tracking_technique/status_of_system
0	2	15	0	0	4	code_table	Radiosonde_completeness
0	2	21	0	0	9	flag_table	Satellite_instrumentation_data_used_in_processing
0	2	22	0	0	8	flag_table	Satellite_data_processing_technique_used
0	2	23	0	0	4	code_table	Cloud_motion_computational_method
0	2	24	0	0	4	code_table	Integrated_mean_humidity_computational_method
0	2	25	0	0	25	flag_table	Satellite_channel(s)_used_in_computation
0	2	26	2	0	12	m	Cross_track_resolution
0	2	27	2	0	12	m	Along_track_resolution
0	2	28	0	0	32	flag_table	Geostationary_sounder_satellite_channels_used
0	2	29	0	0	8	flag_table	Geostationary_sounder_satellite_channels_used
0	2	30	0	0	3	flag_table	GOES_I/M_parameter_calculation_data_source
0	2	31	0	0	5	code_table	Method_of_current_measurement
0	2	32	0	0	2	code_table	Indicator_for_digitization
0	2	33	0	0	3	code_table	Method_of_salinity/depth_measurement

0	2	34	0	0	5	code_table	Drogue_type
0	2	35	0	0	9	m	Cable_length
0	2	36	0	0	2	code_table	Buoy_type
0	2	41	0	0	6	code_table	Method_for_estimating_reports_related_to_synoptic_features
0	2	42	0	0	6	code_table	Method_of_report_construction
0	2	43	0	0	6	code_table	Precision_of_latitude/longitude_report
0	2	61	0	0	3	code_table	Aircraft_navigation_system
0	2	62	0	0	4	code_table	Type_of_aircraft_data_relay_system
0	2	63	2	-18000	16	deg	Aircraft_roll_angle
0	2	70	0	0	4	code_table	Original_specification_of_latitude/longitude
0	2	101	0	0	4	code_table	Type_of_antenna
0	2	102	0	0	8	m	Antenna_height_above_tower
0	2	103	0	0	2	flag_table	Radome
0	2	104	0	0	4	code_table	Antenna_polarisation
0	2	105	0	0	6	dB	Maximum_antenna_gain
0	2	106	1	0	6	deg	3-dB_bandwidth
0	2	107	0	0	6	dB	Sidelobe_suppression
0	2	108	0	0	6	dB	Crosspol_discrimination_(on_axis)
0	2	109	2	0	12	deg/s	Antenna_speed_(azimuth)
0	2	110	2	0	12	deg/s	Antenna_speed_(elevation)
0	2	111	1	0	10	deg	Radar_incidence_angle
0	2	112	1	0	12	deg	Radar_azimuth_angle
0	2	113	0	0	4	numeric	Number_of_azimuth_looks
0	2	114	0	0	15	m**2	Antenna_effective_surface_area
0	2	121	-8	0	7	Hz	Mean_frequency
0	2	122	-6	-128	8	Hz	Frequency_agility_range
0	2	123	-4	0	7	W	Peak_power
0	2	124	-1	0	7	W	Average_power
0	2	125	-1	0	8	Hz	Pulse_repetition_frequency
0	2	126	7	0	6	s	Pulse_width
0	2	127	-6	0	7	Hz	Receiver_intermediate_frequency
0	2	128	-5	0	6	Hz	Intermediate_frequency_bandwidth
0	2	129	0	-150	5	dB	Minimum_detectable_signal
0	2	130	0	0	7	dB	Dynamic_range
0	2	131	0	0	2	flag_table	Sensitivity_time_control
0	2	132	2	0	6	deg	Azimuth_pointing_accuracy
0	2	133	2	0	6	deg	Elevation_pointing_accuracy
0	2	134	2	0	16	deg	Antenna_beam_azimuth
0	2	135	2	-9000	15	deg	Antenna_elevation
0	2	190	0	0	8	code_table	U,_v_quality_indicator
0	2	191	0	0	4	code_table	W_quality_indicator
0	4	1	0	0	12	yr	Year
0	4	2	0	0	4	mo	Month
0	4	3	0	0	6	day	Day
0	4	4	0	0	5	hr	Hour
0	4	5	0	0	6	min	Minute
0	4	6	0	0	6	s	Second
0	4	11	0	-1024	11	yr	Time_increment
0	4	12	0	-1024	11	mo	Time_increment
0	4	13	0	-1024	11	days	Time_increment
0	4	14	0	-1024	11	hr	Time_increment
0	4	15	0	-2048	12	min	Time_increment
0	4	16	0	-4096	13	s	Time_increment
0	4	21	0	-1024	11	yr	Time_period_or_displacement
0	4	22	0	-1024	11	mo	Time_period_or_displacement
0	4	23	0	-1024	11	days	Time_period_or_displacement
0	4	24	0	-2048	12	hr	Time_period_or_displacement
0	4	25	0	-2048	12	min	Time_period_or_displacement
0	4	26	0	-4096	13	s	Time_period_or_displacement
0	4	31	0	0	8	hr	Duration_of_time_relative_to_followinG_value
0	4	43	0	0	9	day	Day_of_the_year
0	4	44	0	0	3	code_table	Day_of_the_week
0	5	1	5	-9000000	25	deg	Latitude_(high_accuracy)
0	5	2	2	-9000	15	deg	Latitude_(coarse_accuracy)
0	5	3	2	-9000	15	deg	Alternate_latitude
0	5	11	5	-9000000	25	deg	Latitude_increment_(high_accuracy)
0	5	12	2	-9000	15	deg	Latitude_increment_(coarse_accuracy)
0	5	21	2	0	16	deg_true	Bearing_or_azimuth
0	5	22	2	0	16	deg_true	Solar_azimuth

0	5	30	0	0	12	deg	Direction_(spectral)
0	5	31	0	0	12	numeric	Row_number
0	5	33	-1	0	16	m	Pixel_size_on_horizontal_-_1
0	5	40	0	0	24	numeric	Orbit_number
0	5	41	0	0	8	numeric	Scan_line_number
0	5	42	0	0	6	numeric	Channel_number
0	5	43	0	0	8	numeric	Field_of_view_number
0	5	50	5	0	17	numeric	Sigma_level
0	5	52	0	0	5	numeric	Channel_number_increment
0	5	53	0	0	5	numeric	Field_of_view_number_increment
0	6	1	5	-18000000	26	deg	Longtiude_(high_accuracy)
0	6	2	2	-18000	16	deg	Longtiude_(coarse_accuracy)
0	6	3	2	-18000	16	deg	Alternate_longtiude
0	6	11	5	-18000000	26	deg	Longtiude_increment_(high_accuracy)
0	6	12	2	-18000	16	deg	Longtiude_increment_(coarse_accuracy)
0	6	21	-1	0	13	m	Distance
0	6	30	5	0	13	rad/m	Wavenumber_(spectral)
0	6	31	0	0	12	numeric	Column_number
0	6	33	-1	0	16	m	Pixel_size_on_horizontal_-_2
0	7	1	0	-400	15	m	Height_of_station
0	7	2	-1	-40	16	m	Height_or_altitude
0	7	3	-1	-400	17	m**2/s**2	Geopotential
0	7	4	-1	0	14	Pa	Pressure
0	7	5	0	-400	12	m	Height_increment
0	7	6	0	0	15	m	Height_above_station
0	7	21	2	-9000	15	deg	Elevation
0	7	22	2	-9000	15	deg	Solar_elevation
0	7	61	2	0	14	m	Depth_below_land_surface
0	7	62	1	0	17	m	Depth_below_sea_surface
0	7	190	0	0	12	m	Height_increment
0	8	1	0	0	7	flag_table	Vertical_sounding_significance
0	8	2	0	0	6	code_table	Vertical_significance_(surface_observations)
0	8	3	0	0	6	code_table	Vertical_significance_(satellite_observations)
0	8	4	0	0	3	code_table	Phase_of_aircraft_flight
0	8	5	0	0	4	code_table	Surface_synoptic_features_significance
0	8	0	0	0	4	code_table	Vertical_significance_(forecast_soundings)
0	8	11	0	0	6	code_table	Horizontal_significance
0	8	12	0	0	2	code_table	land/sea_qualifier
0	8	13	0	0	2	code_table	Day/night_qualifier
0	8	21	0	0	5	code_table	Time_significance
0	8	22	0	0	16	numeric	Total_number_(with_respect_to_accumulation_or_average)
0	8	31	0	0	8	BUFR_Table_A	Data_category
0	8	32	0	0	14	code_table	Data_significance_for_simulated_forecast
0	8	33	0	0	3	code_table	Data_significance_for_corrected_forecast
0	8	34	0	0	3	code_table	Data_significance_for_simulated_retrievals
0	8	35	0	0	3	code_table	Data_significance_for_corrected_retrievals
0	8	36	0	0	3	code_table	Data_significance_for_simulated_analysis
0	8	37	0	0	3	code_table	Data_significance_for_corrected_analysis
0	8	38	0	0	3	code_table	Data_significance_for_sigma_level_data
0	10	1	0	-400	15	m	Height_of_land_surface
0	10	2	-1	-40	16	m	Height
0	10	3	-1	-400	17	m**2/s**2	Geopotential
0	10	4	-1	0	14	Pa	Pressure
0	10	50	2	0	16	m	Standard_deviation_altitude
0	10	51	-1	0	14	Pa	Pressure_reduced_to_mean_sea_level
0	10	52	-1	0	14	Pa	Altimeter_setting_(QNH)
0	10	60	-1	-1024	11	Pa	Pressure_change
0	10	61	-1	-500	10	Pa	3_hour_pressure_change
0	10	62	-1	-1000	11	Pa	24_hour_pressure_change
0	10	63	0	0	4	code_table	Characteristic_of_pressure_tendency
0	11	1	0	0	9	deg_true	Wind_direction
0	11	2	1	0	12	m/s	Wind_speed
0	11	3	1	-4096	13	m/s	U-component
0	11	4	1	-4096	13	m/s	V-component
0	11	5	1	-512	10	Pa/s	W-component
0	11	6	2	-4096	13	m/s	W-component
0	11	11	0	0	9	deg_true	Wind_direction_at_10_m
0	11	12	1	0	12	m/s	Wind_speed_at_10_m
0	11	13	0	0	9	deg_true	Wind_direction_at_5_m

0	11	14	1	0	12	m/s	Wind_speed_at_5_m
0	11	21	9	-65536	17	s**-1	Relative_vorticity
0	11	22	9	-65536	17	s**-1	Divergence
0	11	23	-2	-65536	17	m**2/s	Velocity_potential
0	11	31	0	0	4	code_table	Degree_of_turbulence
0	11	32	-1	-40	16	m	Height_of_base_of_turbulence
0	11	33	-1	-40	16	m	Height_of_top_of_turbulence
0	11	34	1	-1024	11	m/s	Vertical_gust_velocity
0	11	35	2	-8192	14	m/s**2	Vertical_gust_acceleration
0	11	36	1	0	10	m/si	Maximum_derived_equivalent_vertical_gust
0	11	41	1	0	12	m/s	Maximum_wind_speed_(gusts)
0	11	42	1	0	12	m/s	Maximum_wind_speed_(10_minute_mean_wind)
0	11	43	1	0	12	m/s	Surface_-_5000_feet_mean_LYR_wind_speed
0	11	44	0	0	9	deg_true	Surface_-_5000_feet_mean_LYR_wind_direction
0	11	45	0	0	9	deg_true	Maximum_wind_(gust)_direction
0	11	50	1	0	12	m/s	Standard_deviation_horizontal_wind_speed
0	11	51	1	0	8	m/s	Standard_deviation_vertical_wind_component
0	11	61	1	0	12	m/s	Absolute_wind_shear_(1_kilometer_layer_below)
0	11	62	1	0	12	m/s	Absolute_wind_shear_(1_kilometer_layer_above)
0	12	1	1	0	12	deg_K	Temperature/dry_bulb_temperature
0	12	2	1	0	12	deg_K	Wet_bulb_temperature
0	12	3	1	0	12	deg_K	Dewpoint_temperature
0	12	4	1	0	12	deg_K	Dry_bulb_temperature_at_2_meters
0	12	5	1	0	12	deg_K	Wet_bulb_temperature_at_2_meters
0	12	6	1	0	12	deg_K	Dewpoint_temperature_at_2_meters
0	12	7	1	0	12	deg_K	Virtual_temperature
0	12	11	1	0	12	deg_K	Maximum_temperature
0	12	12	1	0	12	deg_K	Minimum_temperature
0	12	13	1	0	12	deg_K	Ground_minimum_temperature_past_12_hours
0	12	14	1	0	12	deg_K	Maximum_temperature_at_2_meters,_past_12_hours
0	12	15	1	0	12	deg_K	Minimum_temperature_at_2_meters,_past_12_hours
0	12	16	1	0	12	deg_K	Maximum_temperature_at_2_meters,_past_24_hours
0	12	17	1	0	12	deg_K	Minimum_temperature_at_2_meters,_past_24_hours
0	12	20	2	-2000	12	deg_K/day	Radiative_heating_profile
0	12	30	1	0	12	deg_K	Soil_temperature
0	12	40	-3	-2048	12	W/m**2	Sensible_heat_flux
0	12	61	1	0	12	deg_K	Skin_temperature
0	12	62	1	0	12	deg_K	Equivalent_black_body_temperature
0	12	63	1	0	12	deg_K	Brightness_temperature
0	13	1	5	0	14	kg/kg	Specific_humidity
0	13	2	5	0	14	kg/kg	Mixing_ratio
0	13	3	0	0	7	percent	Relative_humidity
0	13	4	-1	0	10	Pa	Vapor_pressure
0	13	5	3	0	7	kg/m**3	Vapor_density
0	13	6	-1	-40	16	m	Mixing_heights
0	13	11	4	-1	14	kg/m**2	Total_precipitation/total_water_equivalent
0	13	12	2	-2	12	m	Depth_of_fresh_snow
0	13	13	2	-2	16	m	Total_snow_depth
0	13	14	4	0	12	kg/m**2/s	Rainfall/water_equivalent_of_snow_(average_rate)
0	13	15	7	0	12	m/s	Snowfall_(averaged_rate)
0	13	16	0	0	7	kg/m**2	Precipitable_water
0	13	19	1	-1	14	kg/m**2	Total_precipitation_past_1_hour
0	13	20	1	-1	14	kg/m**2	Total_precipitation_past_3_hours
0	13	21	1	-1	14	kg/m**2	Total_precipitation_past_6_hours
0	13	22	1	-1	14	kg/m**2	Total_precipitation_past_12_hours
0	13	23	1	-1	14	kg/m**2	Total_precipitation_past_24_hours
0	13	31	0	0	7	kg/m**2	Evapotranspiration
0	13	32	1	0	8	kg/m**2	Evaporation/evapotranspiration
0	13	40	-3	-2048	12	W/m**2	Latent_heat_flux
0	13	41	0	0	4	code_table	Pasquill-Gifford_stability_category
0	13	42	0	-20	6	deg_K	US_NWS_lifted_index
0	14	1	-3	-2048	12	Joules/m**2	Longwave_radiation,_integrated_over_24_hours
0	14	3	-3	-2048	12	Joules/m**2	Shortwave_radiation,_integrated_over_24_hours
0	14	4	-3	-2048	12	Joules/m**2	Shortwave_radiation,_integrated_over_period_specified
0	14	11	-3	-2048	12	Joules/m**2	Net_longwave_radiation,_integrated_over_24_hours
0	14	12	-3	-2048	12	Joules/m**2	Net_longwave_radiation,_integrated_over_period_specified
0	14	13	-3	-2048	12	Joules/m**2	Net_shortwave_radiation,_integrated_over_24_hours
0	14	14	-3	-2048	12	Joules/m**2	Net_shortwave_radiation,_integrated_over_period_specified
0	14	15	-4	-16384	15	Joules/m**2	Net_radiation,_integrated_over_24_hours

0	14	16	-4	-16384	15	Joules/m**2	Net_radiation,_integrated_over_period_specified
0	14	17	-3	-2048	12	W/m**2	Instantaneous_long_wave_radiation
0	14	18	-3	-2048	12	W/m**2	Instantaneous_short_wave_radiation
0	14	19	0	0	7	percent	Surface_albedo
0	14	20	-4	0	15	Joules/m**2	Global_solar_radiation,_integrated_over_24_hours
0	14	21	-4	0	15	Joules/m**2	Global_radiation,_integrated_over_period_specified
0	14	22	-4	0	15	Joules/m**2	Diffuse_solar_radiation,_integrated_over_24_hours
0	14	23	-4	0	15	Joules/m**2	Diffuse_solar_radiation,_integratedi_over_period_specified
0	14	24	-4	0	15	Joules/m**2	Direct_solar_radiation,_integrated_over_24_hours
0	14	25	-4	0	15	Joules/m**2	Direct_solar_radiation,_integrated_over_period_specified
0	14	31	0	0	11	min	Total_sunshine
0	14	32	0	0	10	hr	Total_sunshine
0-	14	41	0	0	7	percent	Short_wave_albedo
0	14	42	0	0	7	percent	Bi-directional_reflectance
0	15	1	0	0	10	Dobson_units	Ozone
0	19	1	0	0	6	code_table	Type_of_synoptic_features
0	19	2	-2	0	12	m	Effective_radius_of_feature
0	19	3	0	0	8	m/s	Wind_speed_threshold
0	19	4	-2	0	12	m	Effective_radius_of_feature_with_respect_to_wind_ speeds_above_threshold
0	19	5	0	0	9	deg_true	Direction_of_motion_of_features
0	19	6	2	0	14	m/s	Speed_of_motion_of_features
0	19	7	-3	0	12	m	Effective_radius_of_features
0	19	8	0	0	3	code_table	Vertical_extent_of_circulation
0	19	9	-3	0	12	m	Effective_radiation_with_respect_to_FFF_above_threshold
0	20	1	-1	0	13	m	Horizontal_visibility
0	20	2	-1	0	7	m	Vertical_visibility
0	20	3	0	0	9	code_table	Present_weather
0	20	4	0	0	5	code_table	Past_weather_(1)
0	20	5	0	0	5	code_table	Past_weather_(2)
0	20	9	-1	-40	11	m	Ceiling
0	20	10	0	0	7	percent	Cloud_cover_(total)
0	20	11	0	0	4	code_table	Cloud_amount
0	20	12	0	0	6	code_table	Cloud_type
0	20	13	-1	-40	11	m	Height_of_base_of_cloud
0	20	14	-1	-40	11	m	Height_of_top_of_cloud
0	20	15	-1	0	14	Pa	Pressure_at_the_base_of_cloud
0	20	16	-1	0	14	Pa	Pressure_at_the_top_of_cloud
0	20	17	0	0	4	code_table	Cloud_top_description
0	20	18	-1	40	11	m	Low_estimated_height_of_cloud_top
0	20	19	-1	40	11	m	High_estimated_height_of_cloud_top
0	20	20	0	0	4	code_table	Low_cloud_amount
0	20	21	0	0	4	code_table	Middle_cloud_amount
0	20	22	0	0	4	code_table	High_cloud_amount
0	20	23	0	0	7	percent	Satellite_sensed_effective_cloud_amount
0	20	31	2	0	7	m	Ice_deposit_(thickness)
0	20	32	0	0	3	code_table	Rate_of_ice_accretion
0	20	33	0	0	4	flag_table	Cause_of_ice_accretion
0	20	34	0	0	5	code_table	Sea_ice_concentration
0	20	35	0	0	4	code_table	Amount_and_type_of_ice
0	20	36	0	0	5	code_table	Ice_situation
0	20	37	0	0	5	code_table	Ice_development
0	20	38	0	0	12	deg_true	Bearing_of_ice_edge
0	20	39	-1	0	13	m	Ice_distance
0	20	41	0	0	4	code_table	Airframe_icing
0	20	49	0	0	10	code_table	Type_of_stability_measure
0	20	50	0	0	10	numeric	Value_of_stability
0	20	51	0	0	7	percent	Amount_of_low_clouds
0	20	52	0	0	7	percent	Amount_of_middle_clouds
0	20	53	0	0	7	percent	Amount_of_high_clouds
0	20	61	0	0	12	m	Runway_visual_range_(RVR)
0	20	62	0	0	5	code_table	State_of_the_ground(with_or_without_snow)
0	20	63	0	0	10	code_table	Special_phenomena
0	21	1	0	-64	7	dB	Horizontal_reflectivity
0	21	2	0	-64	7	dB	Vertical_reflectivity
0	21	3	1	-5	7	dB	Differential_reflectivity
0	21	5	0	-65	6	dB	Linear_depolarization_ratio
0	21	6	0	-65	6	dB	Circular_depolarization_ratio
0	21	11	0	-128	8	m/s	Doppler_mean_velocity_in_X_direction

0	21	12	0	-128	8	m/s	Doppler_mean_velocity_in_Y_direction
0	21	13	0	-128	8	m/s	Doppler_mean_velocity_in_Z_direction
0	21	14	1	-4096	13	m/s	Doppler_mean_velocity_(radiational)
0	21	17	1	-4096	8	m/s	Doppler_velocity_spectral_width
0	21	21	-3	0	4	m	Echo_tops
0	21	30	0	-32	8	dB	Signal_to_noise_ratio
0	21	31	0	0	7	kg/m**2	Vertical_integrated_liquid_water_content
0	21	36	7	0	12	m/s	Radar_rainfall_intensity
0	21	41	-2	0	8	m	Bright_band_height
0	21	51	0	-256	8	dB	Signal_power_above_1_MW
0	22	1	0	0	9	deg_true	Direction_of_waves
0	22	2	0	0	9	deg_true	Direction_of_wind_waves
0	22	3	0	0	9	deg_true	Direction_of_swell_waves
0	22	4	0	0	9	deg_true	Direction_of_current_waves
0	22	11	0	0	6	s	Period_of_waves
0	22	12	0	0	6	s	Period_of_wind_waves
0	22	13	0	0	6	s	Period_of_swell_waves
0	22	21	1	0	10	m	Height_of_waves
0	22	22	1	0	10	m	Height_of_wind_waves
0	22	23	1	0	10	m	Height_of_swell_waves
0	22	25	2	0	10	m	Standard_deviation_wave
0	22	31	2	0	13	m/s	Speed_of_current
0	22	41	1	0	12	deg_K	Sea_surface_temperature_(15_day_running_mean)
0	22	42	1	0	12	deg_K	Sea_temperature
0	22	43	2	0	15	deg_K	Sea_temperature
0	22	44	1	0	14	m/s	Sound_velocity
0	22	50	2	0	8	K	Standard_deviation_sea_surface_temperature
0	22	61	0	0	4	code_table	State_of_sea
0	22	62	2	0	12	ppt	Salinity
0	22	63	0	0	14	m	Total_water_depth
0	22	105	-1	0	6	deg_true	Direction_of_waves
0	22	143	2	2650	13	deg_K	Sea_temperature
0	23	1	0	0	3	code_table	Accident_early_notification_-_article_applicable
0	23	2	0	0	5	code_table	Activity_or_facility_involved_in_incident
0	23	3	0	0	3	code_table	Type_of_release
0	23	4	0	0	3	code_table	Countermeasures_taken_near_border
0	23	5	0	0	2	code_table	Cause_of_incident
0	23	6	0	0	3	code_table	Incident_situation
0	23	7	0	0	3	code_table	Characteristic_of_release
0	23	8	0	0	2	code_table	State_of_current_release
0	23	9	0	0	2	code_table	State_of_expected_release
0	23	16	0	0	2	code_table	Possibility_of_significant_chemical_toxic_health_effect
0	23	17	6	0	20	m**3/s	Flow_discharge_of_major_recipient
0	23	18	0	0	3	code_table	release_behavior_over_time
0	23	19	0	-15000	17	m	Actual_release_height
0	23	21	0	-15000	17	m	Effective_release_height
0	23	22	0	0	24	m	Distance_of_relative_point_or_site_of_incident
0	23	23	1	0	12	m/s	Main_transport_speed_in_atmosphere
0	23	24	2	0	13	m/s	Main_transport_speed_in_water
0	23	25	2	0	13	m/s	Main_transport_speed_in_ground_water
0	23	27	0	0	9	deg_true	Main_transport_direction_in_atmosphere
0	23	28	0	0	9	deg_true	Main_transport_direction_in_water
0	23	29	0	0	9	deg_true	Main_transport_direction_in_ground_water
0	23	31	0	0	2	code_table	Possibility_that_plume_will_encounter_precipitation_cwin_ state_in_which_incident_occurred
0	23	32	0	0	2	code_table	Plume_will_encounter_ichange_in_wind_direction_and/or_ speed_flag
0	24	1	-1	0	28	Bq	Estimate_of_radiatioactivity_released_up_to_specified_time
0	24	2	-1	0	28	Bq	Estimated_maximum_potential_release
0	24	3	0	0	5	code_table	Composition_of_release
0	24	4	0	0	16	CCITT_IA5	Element_nam
0	24	5	0	0	9	numeric	Isotope_mass
0	24	11	2	0	32	mSv	Dose
0	24	12	2	0	32	mSv	Trajectory_dose_(defined_location_and_expected_ time_of_arrival)
0	24	13	2	0	32	mSv	Gamma_dose_in_air_along_the_main_transport_path_ (defined_location_and_time_period
0	24	21	2	0	32	Bq/m**3	Air_concentration_(named_isotope_type_including_gross_beta)
0	24	22	2	0	32	Bq/l	Concentration_in_precipitation_(of_named_isotope_type)

0	25	1	-1	0	6	m	Range_gate_length
0	25	2	0	0	4	numeric	Number_of_gates_averaged
0	25	3	0	0	8	numeric	Number_of_integrated_pulses
0	25	4	0	0	2	code_table	Echo_processing
0	25	5	0	0	2	code_table	Echo_integration
0	25	6	0	0	3	code_table	Z_to_R_conversion
0	25	7	0	0	12	numeric	Z_to_R_conversion_factor
0	25	8	2	0	9	numeric	Z_to_R_conversion_exponent
0	25	9	0	0	4	flag_table	Calibration_method
0	25	10	0	0	4	code_table	Clutter_treatment
0	25	11	0	0	2	code_table	Ground_occultation_correction_(screening)
0	25	12	0	0	2	code_table	Range_attenuation_correction
0	25	13	0	0	2	flag_table	Bright_band_correction
0	25	15	0	0	2	flag_table	Radome_attenuation_correction
0	25	16	5	0	6	dB/m	Clear_air_attenuation_correction
0	25	17	0	0	2	flag_table	Precipitation_attenuation_correction
0	25	18	7	0	6	numeric	A_to_Z_law_for_attenuation_factor
0	25	19	2	0	7	numeric	A_to_Z_law_for_attenuation_exponent
0	25	20	0	0	2	code_table	Mean_speed_estimation
0	25	21	0	0	8	flag_table	Wind_computation_enhancement
0	25	30	0	0	2	code_table	SST_usage
0	25	31	0	0	2	code_table	15_day_SST_availability
0	25	32	0	0	2	code_table	NOAA_wind_profiler_high/low_mode_data
0	25	33	0	0	2	code_table	NOAA_wind_profiler_submode
0	25	34	0	0	4	code_table	NOAA_wind_profiler_Q/C_test_results
0	25	35	0	0	3	code_table	First_guess_adjustment
0	25	36	3	-5000	14	numeric	NSTAR_average_value
0	25	37	0	0	2	code_table	HIRS-8_surface_air_temperature
0	25	38	0	0	3	code_table	TOVS_filter_flags
0	27	1	5	-9000000	25	deg	Latitude_(high_accuracy)
0	27	2	2	-9000	15	deg	Latitude_(coarse_accuracy)
0	27	3	2	-9000	15	deg	Alternate_latitude
0	27	20	0	0	16	numeric	Satellite_locationon_counter
0	27	21	0	0	16	numeric	Satellite_sub-locationon_dimensions
0	28	1	5	-18000000	26	deg	Longtiude_(high_accuracy)
0	28	2	2	-18000	16	deg	Longtiude_(coarse_accuracy)
0	28	3	2	-18000	16	deg	Alternate_longtiude
0	29	1	0	0	3	code_table	Projection_type
0	29	2	0	0	2	code_table	Coordinate_grid_type
0	30	1	0	0	4	numeric	Pixel_value_(4_bits)
0	30	21	0	0	12	numeric	Number_of_pixels_per_row
0	30	22	0	0	12	numeric	Number_of_pixels_per_column
0	30	31	0	0	4	code_table	Picture_type
0	30	32	0	0	16	flag_table	Combined_picture
0	31	0	0	0	1	flag_table	Delayed_descriptor_replication_switch
0	31	1	0	0	8	numeric	Delayed_descriptor_replication_factor
0	31	2	0	0	16	numeric	Extended_delayed_descriptor_replication_factor
0	31	11	0	0	8	numeric	Delayed_descriptor_and_data_repition_factor
0	31	12	0	0	16	numeric	Extended_delayed_descriptor_and_data_repitition_factor
0	31	21	0	0	6	code_table	Associated_field_significance
0	0	0	0	0	0	undef	undefined_used_for_SSMI_-_temporary
0	50	200	2	636700	24	km	Altitude(km)
0	50	201	6	0	24	Rad(Si)/sec	Dosage
0	50	202	6	0	24	Gauss	B-Field
0	51	200	-2	0	24	KeV/(cm2-s-Sr)	e-Energy_Flux
0	51	201	-2	0	24	#/(cm2-s-Sr)	e-Number_Flux
0	51	202	-2	0	24	KeV/(cm2-s-Sr)	i-Energy_Flux
0	51	203	-2	0	24	#/(cm2-s-Sr)	i-Number_Flux
0	51	204	6	-10	24	mhos	Ped_Conduct.
0	51	205	6	-10	24	mhos	Hall_Conduct.
0	52	200	5	0	26	-	S4
0	52	201	5	0	25	-	SI
0	52	202	5	0	24	-	PCT
0	52	203	5	0	24	-	PRMS
0	53	200	-1	0	25	cm**-3	Ne
0	53	201	3	0	24	MHz	FE
0	53	202	3	0	24	MHz	FoF2
0	53	203	3	0	20	km	HE
0	53	204	3	0	20	km	HF2

0	55	1	0	0	32	CCITT_IA5	<MDL1>_BLIRB
0	55	2	0	0	8	numeric	Value_of_IAERSL_BLIRB
0	55	3	0	0	4	numeric	Value_of_MODEL_BLIRB
0	55	4	0	0	4	numeric	Value_of_IVIS_BLIRB
0	55	5	0	0	4	numeric	Value_of_ISEASN_BLIRB
0	55	6	0	0	4	numeric	Value_of_IVULCN_BLIRB
0	55	7	0	0	32	CCITT_IA5	<MDL2>_BLIRB
0	55	8	4	0	16	numeric	Value_of_SN_BLIRB
0	55	9	1	0	16	deg_K	Value_of_TBOUND_BLIRB
0	55	10	0	-1	4	numeric	Value_of_IALB_BLIRB
0	55	11	0	0	4	numeric	Value_of_IP_BLIRB
0	55	12	0	0	32	CCITT_IA5	<MDL3>_BLIRB
0	55	13	1	0	13	deg_K	Value_of_T(0 km)_BLIRB
0	55	14	1	0	13	deg_K	Value_of_T(1 km)_BLIRB
0	55	15	1	0	13	deg_K	Value_of_T(2 km)_BLIRB
0	55	16	1	0	13	deg_K	Value_of_T(3 km)_BLIRB
0	55	17	1	0	14	deg_K	Value_of_T(4 km)_BLIRB
0	55	18	1	0	14	deg_K	Value_of_T(5 km)_BLIRB
0	55	19	0	0	8	numeric	Number_of_AREA_records_BLIRB
0	55	20	0	0	8	numeric	Number_of_REGN_records_BLIRB
0	55	21	0	0	8	numeric	Number_of_MESX_records_BLIRB
0	55	22	0	0	8	numeric	Number_of_MESY_records_BLIRB
0	55	23	0	0	8	numeric	Number_of_MESZ_records_BLIRB
0	55	24	0	0	8	numeric	Number_of_ALBD_records_BLIRB
0	55	25	0	0	8	numeric	Number_of_MTRL_records_BLIRB
0	55	26	0	0	32	CCITT_IA5	<AREA>_BLIRB
0	55	27	2	0	16	km	Value_of_ALX_BLIRB
0	55	28	2	0	16	km	Value_of_AHX_BLIRB
0	55	29	2	0	16	km	Value_of_ALY_BLIRB
0	55	30	2	0	16	km	Value_of_AHY_BLIRB
0	55	31	0	0	8	numeric	Value_of_IAMTL_BLIRB
0	55	32	0	0	32	CCITT_IA5	<REGN>_BLIRB
0	55	33	2	0	16	km	Value_of_RLX_BLIRB
0	55	34	2	0	16	km	Value_of_RHX_BLIRB
0	55	35	2	0	16	km	Value_of_RLY_BLIRB
0	55	36	2	0	16	km	Value_of_RHY_BLIRB
0	55	37	2	0	16	km	Value_of_RLZ_BLIRB
0	55	38	2	0	16	km	Value_of_RHZ_BLIRB
0	55	39	0	0	8	numeric	Value_of_IZMTL_BLIRB
0	55	40	0	0	32	CCITT_IA5	<MESX>_BLIRB
0	55	41	0	0	8	numeric	Value_of_MHX_BLIRB
0	55	42	2	0	16	km	Value_of_XMS_BLIRB
0	55	43	0	0	32	CCITT_IA5	<MESY>_BLIRB
0	55	44	0	0	8	numeric	Value_of_MHY_BLIRB
0	55	45	2	0	16	km	Value_of_YMS_BLIRB
0	55	46	0	0	32	CCITT_IA5	<MESZ>_BLIRB
0	55	47	0	0	8	numeric	Value_of_MHZ_BLIRB
0	55	48	2	0	16	km	Value_of_ZMS_BLIRB
0	55	49	0	0	32	CCITT_IA5	<ALBD>_BLIRB
0	55	50	0	0	8	numeric	Value_of_LALB_BLIRB
0	55	51	4	0	16	numeric	Value_of_FLAB_BLIRB
0	55	52	1	0	16	deg_K	Value_of_TALB_BLIRB
0	55	53	0	0	32	CCITT_IA5	<MTRL>_BLIRB
0	55	54	0	-3	11	numeric	Value_of_LMTL(1)_BLIRB
0	55	55	4	0	19	numeric	Value_of_WMTL(1)_BLIRB
0	55	56	0	-3	11	numeric	Value_of_LMTL(2)_BLIRB
0	55	57	4	0	19	numeric	Value_of_WMTL(2)_BLIRB
0	55	58	0	-3	10	numeric	Value_of_LMTL(3)_BLIRB
0	55	59	4	0	18	numeric	Value_of_WMTL(3)_BLIRB
0	55	60	0	0	32	CCITT_IA5	<CLDS>_BLIRB
0	55	61	0	0	6	numeric	Value_of_ICLD_BLIRB
0	55	62	0	0	4	numeric	Value_of_IBND_BLIRB
0	55	63	1	0	14	m/s	Value_of_WIND_BLIRB
0	55	64	0	0	32	CCITT_IA5	<DOMD>_BLIRB
0	55	65	0	0	6	numeric	Value_of_ISC_BLIRB
0	55	66	0	0	10	numeric	Value_of_IITL_BLIRB
0	55	67	4	0	16	numeric	Value_of_EPSI_BLIRB
0	55	68	0	0	6	numeric	Value_of_IDELTA_BLIRB
0	55	69	0	0	10	numeric	Value_of_NPTS_BLIRB

0	55	70	0	0	32	CCITT_IA5	<SUN >_BLIRB
0	55	71	2	0	17	deg	Value_of_THSUN_BLIRB
0	55	72	2	0	17	deg	Value_of_PHSUN_BLIRB
0	55	73	0	0	2	numeric	Value_of_IFSUN_BLIRB
0	55	74	0	0	2	numeric	Value_of_ISKY_BLIRB
0	55	75	0	0	2	numeric	Value_of_IFTRN_BLIRB
0	55	76	0	0	32	CCITT_IA5	<WAVN>_BLIRB
0	55	77	0	0	16	per cm	Value_of_V1_BLIRB
0	55	78	0	0	16	per cm	Value_of_V2_BLIRB
0	55	79	0	0	8	numeric	Value_of_NV_BLIRB
0	55	80	0	0	32	CCITT_IA5	<ASCI>_BLIRB
0	55	81	0	0	8	numeric	Value_of_IRITE_BLIRB
0	55	82	0	0	32	CCITT_IA5	<RECL>_BLIRB
0	55	83	0	0	8	numeric	Value_of_IRPT_BLIRB
0	55	120	0	0	8	numeric	Value_of_NA_BLIRB
0	55	121	0	0	8	numeric	Value_of_ITN_BLIRB
0	55	122	0	0	8	numeric	Number_of_X_mesh_subblocks_BLIRB
0	55	123	0	0	8	numeric	Number_of_Y_mesh_subblocks_BLIRB
0	55	124	0	0	8	numeric	Number_of_Z_mesh_subblocks_BLIRB
0	55	125	2	0	12	km	X_Grid_Point_BLIRB
0	55	126	2	0	12	km	Y_Grid_Point_BLIRB
0	55	127	2	0	12	km	Z_Grid_Point_BLIRB
0	55	128	0	0	6	numeric	Value_of_ISURF_BLIRB
0	55	129	0	0	8	numeric	Value_of_IVOLM_BLIRB
0	55	130	7	0	25	numeric	Value_of_TRLW_BLIRB
0	55	131	3	0	11	numeric	Value_of_SALB_BLIRB
0	55	132	4	0	18	numeric	Value_of_REXT_BLIRB
0	55	133	5	0	25	numeric	Value_of_RSCT_BLIRB
0	55	134	5	0	25	numeric	Value_of_FDLT_BLIRB
0	55	135	6	0	22	numeric	Value_of_AGL_BLIRB
0	55	136	5	0	25	numeric	Value_of_PHF_BLIRB
0	55	137	5	0	25	numeric	Value_of_RLEG_BLIRB
0	55	138	5	0	18	numeric	Value_of_QSOL_BLIRB
0	55	139	5	0	18	numeric	Value_of_QRFL_BLIRB
0	55	140	5	0	18	numeric	Value_of_FLX-1_BLIRB
0	55	141	5	0	18	numeric	Value_of_FLX-2_BLIRB
0	55	142	5	0	18	numeric	Value_of_FLX-3_BLIRB
0	55	143	5	0	18	numeric	Value_of_FLX-4_BLIRB
0	55	144	5	0	18	numeric	Value_of_FLX-5_BLIRB
0	55	145	5	0	18	numeric	Value_of_FLX-6_BLIRB
0	55	146	5	0	18	numeric	Value_of_FLX-7_BLIRB
0	55	147	5	0	18	numeric	Value_of_FLX-8_BLIRB

Appendix G

Listing of Table D

F	X	Y
3	0	2
0	0	2
0	0	3
-1	-1	-1
3	0	3
0	0	10
0	0	11
0	0	12
-1	-1	-1
3	0	4
3	0	3
0	0	13
0	0	14
0	0	15
0	0	16
0	0	17
0	0	18
0	0	19
0	0	20
-1	-1	-1
3	0	10
3	0	3
1	1	0
0	31	1
0	0	30
-1	-1	-1
3	1	1
0	1	1
0	1	2
-1	-1	-1
3	1	2
0	1	3
0	1	4
0	1	5
-1	-1	-1
3	1	3
0	1	11
0	1	12
0	1	13
-1	-1	-1
3	1	11
0	4	1
0	4	2
0	4	3
-1	-1	-1
3	1	12
0	4	4
0	4	5
-1	-1	-1
3	1	13
0	4	4
0	4	5
0	4	6
-1	-1	-1
3	1	21
0	5	1
0	6	1

0	7	1
-1	-1	-1
3	1	23
0	5	2
0	6	2
-1	-1	-1
3	1	24
0	5	2
0	6	2
0	7	2
-1	-1	-1
3	1	25
3	1	23
0	4	3
3	1	12
-1	-1	-1
3	1	26
3	1	21
0	4	3
0	4	3
0	4	4
0	4	4
0	4	5
0	4	5
-1	-1	-1
3	1	31
3	1	1
0	2	1
3	1	11
3	1	12
3	1	22
-1	-1	-1
3	1	32
3	1	1
0	2	1
3	1	11
3	1	12
3	1	24
-1	-1	-1
3	1	33
0	1	5
0	2	1
3	1	11
3	1	12
3	1	21
-1	-1	-1
3	1	34
0	1	5
0	2	1
3	1	11
3	1	12
3	1	23
-1	-1	-1
3	1	35
0	1	5
0	1	12
0	1	13
0	2	1
3	1	11
3	1	12
3	1	23
-1	-1	-1

3	1	36
3	1	3
0	2	1
3	1	11
3	1	12
3	1	23
-1	-1	-1
3	1	37
3	1	1
0	2	11
0	2	12
3	1	11
3	1	12
3	1	22
-1	-1	-1
3	1	38
3	1	1
0	2	11
0	2	12
3	1	11
3	1	12
3	1	24
-1	-1	-1
3	1	39
3	1	3
0	2	11
0	2	12
3	1	11
3	1	12
3	1	23
-1	-1	-1
3	1	40
3	1	3
0	2	11
0	2	12
3	1	11
3	1	12
3	1	24
-1	-1	-1
3	1	41
0	1	7
0	2	21
0	2	22
3	1	11
3	1	12
-1	-1	-1
3	1	42
3	1	41
3	1	21
-1	-1	-1
3	1	43
0	1	7
0	2	23
3	1	11
3	1	13
3	1	21
-1	-1	-1
3	1	44
0	1	7
0	2	24
3	1	11
3	1	13

3	1	21
-1	-1	-1
3	1	51
0	1	6
0	2	61
3	1	11
3	1	12
3	1	21
0	8	4
-1	-1	-1
3	1	62
1	1	0
0	31	1
3	1	1
-1	-1	-1
3	2	1
0	10	4
0	10	51
0	10	61
0	10	63
-1	-1	-1
3	2	2
0	10	4
0	7	4
0	10	3
0	10	61
0	10	63
-1	-1	-1
3	2	3
0	11	11
0	11	12
0	12	4
0	12	6
0	13	3
0	20	1
0	20	3
0	20	4
0	20	5
-1	-1	-1
3	2	4
0	20	10
0	8	2
0	20	11
0	20	13
0	20	12
0	20	12
0	20	12
-1	-1	-1
3	2	5
0	8	2
0	20	11
0	20	12
0	20	13
-1	-1	-1
3	2	11
3	2	1
3	2	3
3	2	4
-1	-1	-1
3	2	12
3	2	2
3	2	3

3	2	4
-1	-1	-1
3	2	21
0	22	1
0	22	11
0	22	21
-1	-1	-1
3	2	22
0	22	2
0	22	12
0	22	22
-1	-1	-1
3	2	23
0	22	3
0	22	13
0	22	23
-1	-1	-1
3	2	24
3	2	22
1	1	2
3	2	23
-1	-1	-1
3	3	1
0	7	3
0	11	1
0	11	22
-1	-1	-1
3	3	2
0	7	4
0	11	1
0	11	2
-1	-1	-1
3	3	3
0	7	4
0	10	3
0	12	1
0	12	3
-1	-1	-1
3	3	4
0	7	4
0	10	3
0	12	1
0	12	3
0	11	1
0	11	2
-1	-1	-1
3	3	11
0	7	3
0	8	1
0	11	1
0	11	2
-1	-1	-1
3	3	12
0	7	4
0	8	1
0	11	1
0	11	2
-1	-1	-1
3	3	13
0	7	4
0	8	1
0	10	3

0	12	1
0	13	3
0	11	1
0	11	2
-1	-1	-1
3	3	14
0	7	4
0	8	1
0	10	3
0	12	1
0	12	3
0	11	1
0	11	2
-1	-1	-1
3	3	21
0	7	4
0	7	4
2	4	7
0	31	21
-1	-1	-1
3	3	22
3	3	21
0	10	3
2	4	0
-1	-1	-1
3	3	23
3	3	21
0	12	1
2	4	0
-1	-1	-1
3	3	24
3	3	21
0	13	16
2	4	0
-1	-1	-1
3	3	25
0	2	25
2	4	7
0	31	21
0	12	63
2	4	0
-1	-1	-1
3	3	26
0	7	4
0	8	3
2	4	7
0	31	21
0	12	1
2	4	0
-1	-1	-1
3	3	27
0	7	4
2	4	7
0	31	21
0	10	3
2	4	0
-1	-1	-1
3	3	31
0	7	4
0	8	3
0	7	21
0	7	22

0	8	12
0	12	61
-1	-1	-1
3	3	32
0	20	11
0	20	16
-1	-1	-1
3	4	1
0	8	3
0	10	4
0	12	1
0	11	1
0	11	2
-1	-1	-1
3	4	2
0	8	3
0	10	4
0	11	1
0	11	2
-1	-1	-1
3	4	3
0	8	3
0	12	1
-1	-1	-1
3	4	4
0	8	3
0	10	4
0	20	10
0	12	1
0	2	24
0	7	4
0	7	4
0	13	3
-1	-1	-1
3	4	6
0	14	1
0	14	1
0	14	3
-1	-1	-1
3	6	1
0	2	32
1	2	0
0	31	1
0	7	62
0	22	42
-1	-1	-1
3	6	2
0	2	31
0	22	4
0	22	31
-1	-1	-1
3	6	3
0	2	2
0	11	11
0	11	12
0	12	4
-1	-1	-1
3	6	4
0	2	32
0	2	33
1	3	0
0	31	1

0	7	62
0	22	43
0	22	62
-1	-1	-1
3	6	5
0	2	31
1	3	0
0	31	1
0	7	62
0	22	4
0	22	31
-1	-1	-1
3	6	6
3	6	3
3	6	2
0	22	63
-1	-1	-1
3	6	7
0	1	12
0	1	14
3	6	8
0	4	24
0	27	3
0	28	3
-1	-1	-1
3	6	8
0	2	34
0	2	35
0	2	36
-1	-1	-1
3	7	1
3	1	31
3	2	11
-1	-1	-1
3	7	2
3	1	32
3	2	11
-1	-1	-1
3	7	3
3	7	1
1	1	0
0	31	1
3	2	5
-1	-1	-1
3	7	4
3	7	2
1	1	0
0	31	1
3	2	5
-1	-1	-1
3	7	5
3	7	1
1	1	4
3	2	5
-1	-1	-1
3	7	6
3	7	2
1	1	4
3	2	5
-1	-1	-1
3	7	7
3	1	31

3	2	12
-1	-1	-1
3	7	8
3	1	32
3	2	12
-1	-1	-1
3	8	1
3	1	33
3	2	11
0	22	42
-1	-1	-1
3	8	2
3	1	34
3	2	11
0	22	42
-1	-1	-1
3	8	3
3	1	35
3	2	11
0	22	42
-1	-1	-1
3	8	4
3	1	36
3	2	11
0	22	42
-1	-1	-1
3	8	5
3	8	4
3	2	24
-1	-1	-1
3	8	6
0	10	4
0	10	61
0	10	63
0	11	1
0	11	2
0	12	4
0	13	3
0	22	42
-1	-1	-1
3	9	1
3	1	37
1	1	0
0	31	1
3	3	11
-1	-1	-1
3	9	2
3	1	38
1	1	0
0	31	1
3	3	11
-1	-1	-1
3	9	3
3	1	37
1	1	0
0	31	1
3	3	12
-1	-1	-1
3	9	4
3	1	38
1	1	0
0	31	1

3	3	12
-1	-1	-1
3	9	5
3	1	37
3	2	4
1	1	0
0	31	1
3	3	13
-1	-1	-1
3	9	6
3	1	38
3	2	4
1	1	0
0	31	1
3	3	13
-1	-1	-1
3	9	7
3	1	37
3	2	4
1	1	0
0	31	1
3	3	14
-1	-1	-1
3	9	8
3	1	38
3	2	4
1	1	0
0	31	1
3	3	14
-1	-1	-1
3	9	11
3	1	39
1	1	0
0	31	1
3	3	11
-1	-1	-1
3	9	12
3	1	39
1	1	0
0	31	1
3	3	12
-1	-1	-1
3	9	13
3	1	39
3	2	4
1	1	0
0	31	1
3	3	13
-1	-1	-1
3	9	14
3	1	39
3	2	4
1	1	0
0	31	1
3	3	14
-1	-1	-1
3	9	15
3	1	40
1	1	0
0	31	1
3	3	11
-1	-1	-1

3	9	16
3	1	40
1	1	0
0	31	1
3	3	12
-1	-1	-1
3	9	17
3	1	40
3	2	4
1	1	0
0	31	1
3	3	13
-1	-1	-1
3	9	18
3	1	40
3	2	4
1	1	0
0	31	1
3	3	14
-1	-1	-1
3	9	19
3	1	31
0	2	3
1	1	0
0	31	1
3	3	11
-1	-1	-1
3	9	20
3	1	31
0	2	3
1	4	0
0	31	1
0	7	3
0	11	3
0	11	4
0	11	5
-1	-1	-1
3	10	1
3	1	42
3	3	31
3	3	32
1	1	26
3	3	25
-1	-1	-1
3	10	2
3	1	42
3	3	31
3	3	32
1	1	9
3	3	23
-1	-1	-1
3	10	3
3	1	42
3	3	31
3	3	32
1	1	6
3	3	23
-1	-1	-1
3	10	4
3	1	42
3	3	31
3	3	32

1	1	3
3	3	24
-1	-1	-1
3	11	1
3	1	51
0	7	2
0	12	1
0	11	1
0	11	2
0	11	31
0	11	32
0	11	33
0	20	41
-1	-1	-1
3	12	1
3	1	43
3	4	1
-1	-1	-1
3	12	2
3	1	43
3	4	2
-1	-1	-1
3	12	3
3	1	42
3	4	3
-1	-1	-1
3	12	4
3	1	42
3	4	4
-1	-1	-1
3	12	5
3	1	42
0	20	14
-1	-1	-1
3	12	6
3	1	44
3	4	5
-1	-1	-1
3	12	7
3	1	42
3	4	6
-1	-1	-1
3	12	10
0	1	7
0	5	40
0	2	21
0	5	41
0	4	1
0	4	43
-1	-1	-1
3	12	11
2	2	131
2	1	149
0	4	6
2	1	0
1	2	32
0	10	2
2	2	0
0	5	43
0	5	53
-1	-1	-1
3	12	12

2	2	129
2	1	132
1	1	19
0	12	63
2	1	0
2	2	0
-1	-1	-1
3	12	13
0	5	42
2	2	129
2	1	135
0	12	63
2	1	0
2	2	0
-1	-1	-1
3	12	14
3	12	10
3	12	11
1	5	56
3	1	23
0	5	42
0	5	52
3	12	12
3	12	13
-1	-1	-1
3	12	15
1	9	11
3	1	23
0	5	42
0	5	52
2	2	129
2	1	132
1	1	4
0	12	63
2	2	0
2	1	0
-1	-1	-1
3	12	16
3	12	10
3	12	11
3	12	15
-1	-1	-1
3	12	17
1	9	8
3	1	23
0	5	42
0	5	52
2	2	129
2	1	132
1	1	3
0	12	63
2	2	0
2	1	0
-1	-1	-1
3	12	18
3	12	10
3	12	11
3	12	17
-1	-1	-1
3	13	9
0	21	1
1	1	0

0	31	1
0	21	1
-1	-1	-1
3	13	10
0	21	36
1	1	0
0	31	1
0	21	36
-1	-1	-1
3	13	31
0	6	2
0	6	12
1	1	0
0	31	2
0	30	1
-1	-1	-1
3	13	32
0	5	2
0	5	12
1	1	0
0	31	2
3	13	31
-1	-1	-1
3	13	41
0	6	2
0	6	12
1	6	0
0	31	1
1	1	0
0	31	11
0	30	1
1	1	0
0	31	1
0	30	1
-1	-1	-1
3	13	42
0	5	2
0	5	12
1	1	0
0	31	2
3	13	41
-1	-1	-1
3	15	1
0	1	11
3	1	11
3	1	12
3	1	23
3	6	1
-1	-1	-1
3	15	2
0	1	11
3	1	11
3	1	12
3	1	23
3	6	4
-1	-1	-1
3	16	1
3	1	11
0	4	4
3	1	23
0	1	21
0	2	41

0	19	1
0	10	51
0	19	2
0	19	3
0	19	4
-1	-1	-1
3	18	1
3	1	25
0	24	11
-1	-1	-1
3	18	3
3	1	26
0	24	5
0	24	4
0	24	21
-1	-1	-1
3	18	4
3	1	25
0	4	23
0	13	11
0	24	5
0	24	4
0	24	22
-1	-1	-1
3	21	1
0	2	101
0	2	114
0	2	105
0	2	106
0	2	107
0	2	121
-1	-1	-1
3	21	3
0	21	51
0	21	14
0	21	17
0	21	30
-1	-1	-1
3	21	4
3	1	31
0	2	3
1	1	0
0	31	1
3	21	3
-1	-1	-1
3	21	5
0	25	4
0	2	121
0	2	122
0	2	123
0	2	124
0	2	125
0	2	126
0	2	127
0	2	128
0	2	129
0	2	130
0	2	131
-1	-1	-1
3	21	6
0	25	1
0	25	2

0	25	3
0	25	5
-1	-1	-1
3	21	7
0	25	9
0	25	10
0	25	11
0	25	12
0	25	13
0	25	15
0	25	16
0	25	17
-1	-1	-1
3	21	8
0	25	6
0	25	7
0	25	8
-1	-1	-1
3	21	9
0	25	18
0	25	19
-1	-1	-1
3	21	10
0	2	101
0	7	2
0	2	102
0	2	103
0	2	104
0	2	105
0	2	106
0	2	107
0	2	108
0	2	109
0	2	110
0	2	132
0	2	133
-1	-1	-1
3	21	11
0	30	31
0	30	32
0	29	2
-1	-1	-1
3	21	12
1	1	0
0	31	1
0	2	135
0	2	135
-1	-1	-1
3	25	1
0	55	1
0	55	2
0	55	3
0	55	4
0	55	5
0	55	6
0	55	7
0	55	8
0	55	9
0	55	10
0	55	11
0	55	12
0	55	13

0	55	14
0	55	15
0	55	16
0	55	17
0	55	18
0	55	19
0	55	20
0	55	21
0	55	22
0	55	23
0	55	24
0	55	25
-1	-1	-1
3	25	2
0	55	1
0	55	2
0	55	3
0	55	4
0	55	5
0	55	6
0	55	7
0	55	8
0	55	9
0	55	10
0	55	11
0	55	19
0	55	20
0	55	21
0	55	22
0	55	23
0	55	24
0	55	25
-1	-1	-1
3	25	3
0	55	26
0	55	27
0	55	28
0	55	29
0	55	30
0	55	31
-1	-1	-1
3	25	4
0	55	32
0	55	33
0	55	34
0	55	35
0	55	36
0	55	37
0	55	38
0	55	39
-1	-1	-1
3	25	5
0	55	40
0	55	41
0	55	42
-1	-1	-1
3	25	6
0	55	43
0	55	44
0	55	45
-1	-1	-1
3	25	7

0	55	46
0	55	47
0	55	48
-1	-1	-1
3	25	8
0	55	49
0	55	50
0	55	51
0	55	52
-1	-1	-1
3	25	9
0	55	53
0	55	54
0	55	55
0	55	56
0	55	57
0	55	58
0	55	59
-1	-1	-1
3	25	10
0	55	60
0	55	61
0	55	62
0	55	63
0	55	64
0	55	65
0	55	66
0	55	67
0	55	68
0	55	69
0	55	70
0	55	71
0	55	72
0	55	73
0	55	74
0	55	75
0	55	76
0	55	77
0	55	78
0	55	79
0	55	80
0	55	81
0	55	82
0	55	83
0	55	120
0	55	121
0	55	122
0	55	123
0	55	124
-1	-1	-1

Appendix H

Listing of VISUAL.C

```
/*********************************************************************
 *                          PROGRAM VISUAL
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  visual
 *                          Type:  C Program
 *                      Filename:  visual.c
 *                        Parent:  None
 *====================================================================
 *<Description>
 *    This program displays:
 *       the entire BLIRB grid space,
 *       surface albedo areas (in shades of green),
 *       regions of aerosol concentrations (various transparent colors),
 *       flare positions (in red),
 *       searchlight positions (in white), and
 *       BLIRB output flux fields (red and white).
 *
 *    Optionally, text widgets may be displayed depicting:
 *       the current BLIRB filename,
 *       the surface albedo areas information,
 *       the aerosols regions information, and
 *       the output flux field information.
 *
 *    All BLIRB inputs can be selected through a Graphical User
 *    Interface (GUI) using only the mouse.  The keyboard may be used
 *    for "hot-key" use and must be used for inputting a "savefile
 *    name".
 *
 *    This program uses the SGI IRIS Graphics Library for the display
 *    graphics and X-Windows/Motif for the menus and message boxes of
 *    the GUI.
 *
 *    The user can EXIT the program by either:
 *       pressing the F12 key or
 *       selecting the "Exit" option under the "File" menubar option.
 *
 *    The user can ABORT the program by either:
 *       pressing the ESC key or
 *       through the window manager (upper left corner).
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
/*--------------------------------------------------------------------
 * --- Include Files
 *--------------------------------------------------------------------
 */
#include <Xm/Xm.h>
#include <Xm/Frame.h>
#include <Xm/Form.h>
#include <Xm/List.h>
#include <Xm/Text.h>
#include <Xm/RowColumn.h>
#include <Xm/CascadeB.h>
```

```
#include <Xm/ToggleB.h>
#include <Xm/PushB.h>
#include <Xm/MessageB.h>
#include <Xm/Separator.h>
#include <Xm/Label.h>
#include <Xm/FileSB.h>
#include <Xm/Scale.h>
#include <X11/Xirisw/GlxMDraw.h>
#include <X11/StringDefs.h>
#include <X11/keysym.h>
#include <gl/gl.h>
#include <gl/device.h>
#include <gl/get.h>
#include <string.h>
#include <stdio.h>
#include <math.h>
#include "visual0.h"                /* Prototypes & Definitions        */

/*-----------------------------------------------------------------------
 * --- Array declarations and assignments
 *-----------------------------------------------------------------------
 */
                                    /* Main Widgets & Structures       */
PLOTPTR mov;                            /* Eye Position Information */
static Widget file_dialog = (Widget)0;  /* File Select B Board Wid  */
Widget menu;                            /* Menubar Widget           */
Widget form;                            /* Form Widget              */
static Widget file_form = (Widget)0;    /* Form for Filename        */
static Widget mtrl_form = (Widget)0;    /* Form for Materials List  */
static Widget albd_form = (Widget)0;    /* Form for Albedo List     */
static Widget flux_form = (Widget)0;    /* form for Flux Info       */
static XtAppContext app_context;
static XmStringCharSet charset = (XmStringCharSet)
              XmSTRING_DEFAULT_CHARSET; /* used to set up XmStrings */

                                    /* Filename and I/O Variables      */
/* static char filename_filter[] =
      {"Scenarios/Visible/*.BLIRB8"}; */  /* BLIRB data filename filtr*/
static char filename_filter[] =
                         {"grid*.*"};   /* BLIRB data filename filtr*/
static Boolean nofile = TRUE;           /* No file currently avail  */
static Boolean file_found = FALSE;      /* No Filename Found        */
static char *file_name = NULL;          /* BLIRB Data Filename      */
static Boolean blirb_in = TRUE;         /* BLIRB Input File used    */
static Boolean badfile = FALSE;         /* Neither Input or Output  */
static Boolean new_file = FALSE;        /* Data file not modified   */
static Boolean def_file = FALSE;        /* Default datafile not used*/
int filefctn;                           /* Index newfile loss source*/

                                    /* BLIRB Input Card Variables      */
float wavl_wavl;
float vis_vis;
float mdl1_iaersl, mdl1_model, mdl1_ivis, mdl1_iseasn, mdl1_ivulcn;
float mdl2_sn, mdl2_tbound, mdl2_ialb, mdl2_ip;
float mdl3_t[6];
float area_alx[IAM], area_ahx[IAM], area_aly[IAM], area_ahy[IAM],
      area_iamtl[IAM];
float regn_rlx[IRM], regn_rhx[IRM], regn_rly[IRM], regn_rhy[IRM],
      regn_rlz[IRM], regn_rhz[IRM], regn_izmtl[IRM];
float mesx_mhx[ISM], mesx_xms[ISM];
float mesy_mhy[ISM], mesy_yms[ISM];
float mesz_mhz[ISM], mesz_zms[ISM];
```

```
float albd_lalb[IAM], albd_falb[IAM];
float mtrl_lmtl[IRM][MXMTR], mtrl_wmtl[IRM][MXMTR];
float clds_icld, clds_ibnd, clds_wind;
float domd_isc, domd_iitl, domd_epsi, domd_idelta, domd_npts;
float sun_thsun, sun_phsun, sun_ifsun, sun_isky, sun_iftrn;
float flar_idflr[NEST], flar_itflr[NEST], flar_xflar[NEST],
      flar_yflar[NEST], flar_zflar[NEST], flar_qflar[NEST],
      flar_tflar[NEST], flar_frrfup[NEST][4], flar_frrfdn[NEST][4];
float flen_idflr[NEST], flen_qflar[NEST], flen_tflar[NEST];
float flup_idflr[NEST], flup_frrfup[NEST][4];
float fldn_idflr[NEST], fldn_frrfdn[NEST][4];
float srch_xsrch, srch_ysrch, srch_zsrch, srch_thsrch, srch_azsrch,
      srch_psrch, srch_tmsrch, srch_sdiam;
float sren_psrch, sren_tmsrch, sren_sdiam;
float wavn_v1, wavn_v2, wavn_dv;
float asci_irite;
float recl_irpt;

static Boolean in_changea = FALSE;          /* No change in area inputs */
static Boolean in_changef = FALSE;          /* No change in Flare input */
static Boolean in_changesl = FALSE;         /* No change in Slite inputs*/
static Boolean in_change = FALSE;           /* No change in other inputs*/
static Boolean area_order = FALSE;          /* iamtl values not ordered */
static Boolean regn_order = FALSE;          /* MTRL cards individualized*/
static int metrng_indx = -1;                /* Met Range Class Index    */
static int wavn_indx = -1;                  /* Spectral Interval Index  */

                                      /* Input Card Counts              */
int ilcl, mdl1, mdl2, mdl3, area, regn, mesx, mesy, mesz, albd, mtrl,
    clds, domd, sun, flar, flen, flup, fldn, srch, sren, wavn, asci,
    recl, wavl, vis, blirb, done;

                                      /* Substitute Input Card Varbls   */
float area_iamt[IAM];                       /* consolidated albedos     */
float regn_izmt[IRM];                       /* consolidated aerosols    */
float mes_mh[3][ISM], mes_ms[3][ISM];       /* mes$_mh$ and mes$_$ms    */
int mes[3];                                 /* mesx, mesy, and mesz     */
static int mes_del_cnt[3] = { 0, 0, 0 };    /* Number Meshes to Delete  */
int mes_del[3][ISM];                        /* Meshes to be Deleted     */
static float mes_add[3][2] = { -1, -1, -1, -1, -1, -1 };  /* Add Mesh */
float regn_rl[3][IRM], regn_rh[3][IRM];     /* regn_rl$ and regn_rh$    */
                                            /* Maximum dim of Region 1  */
static int regn_max_dim[3] = {MAX_LEN_X, MAX_LEN_Y, MAX_LEN_Z};

                                      /* More Input Variables           */
static float background_albedo = 0.2;       /* Default Backgrnd Albedo  */
static float background_aerosol = 3.0;      /* Default Backgrnd Aerosol */
static Boolean spect_albedo = FALSE;        /* No spectral Albedo       */

static char broad_type[31][15] = { "Default       ",
        "Dark soil     ",          "Light soil    ",
        "Dark-ploughed ",          "Light-ploughed",
        "Clay          ",          "Sandy soil    ",
        "Sand          ",          "White sand    ",
        "Asphalt       ",          "Lava          ",
        "Tundra        ",          "Steppe        ",
        "Concrete      ",          "Stone         ",
        "Desert        ",          "Rock          ",
        "Dirt road     ",          "Clay road     ",
        "Grass         ",          "Mowed grass   ",
        "Desiduous     ",          "Coniferous    ",
        "Rice          ",          "Beet, wheat   ",
```

```
        "Potato          ",          "Rye             ",
        "Cotton          ",          "Lettuce         ",
        "Snow            ",          "Ice             " };

static char broad_band[56][21] = { "Default",
        "Dark soil (dry)",           "Dark soil (wet)",
        "Light soil (dry)",          "Light soil (wet)",
        "Dark-ploughed (dry)",       "Dark-ploughed (wet)",
        "Light-ploughed (dry)",      "Light-ploughed (wet)",
        "Clay (dry)",                "Clay (wet)",
        "Sandy soil (dry)",          "Sandy soil (wet)",
        "Sand (dry)",                "Sand (wet)",
        "White sand",                "Asphalt",
        "Lava",                      "Tundra",
        "Steppe",                    "Concrete",
        "Stone",                     "Desert",
        "Rock (dry)",                "Rock (wet)",
        "Dirt road (dry)",           "Dirt road (wet)",
        "Clay road (dry)",           "Clay road (wet)",
        "Grass (growing)",           "Grass (dormant)",
        "Grass (unspecified)",       "Mowed (growing)",
        "Mowed (dormant)",           "Mowed (unspecified)",
        "Desiduous (growing)",       "Desidious (dormant)",
        "Desidious (unspec.)",       "Coniferous (growing)",
        "Coniferous (dormant)",      "Coniferous (unspec.)",
        "Rice",                      "Beet, wheat",
        "Potato",                    "Rye",
        "Cotton",                    "Lettuce",
        "Snow (fresh)",              "Snow (dense)",
        "Snow (moist)",              "Snow (old)",
        "Snow (melting)",            "Ice (white)",
        "Ice (grey)",                "Ice (snow and ice)",
        "Ice (dark glass)" };

static float broad_albedo[56] = { 0.20, 0.13, 0.08, 0.18, 0.10, 0.08,
    0.06, 0.16, 0.08, 0.23, 0.16, 0.25, 0.18, 0.40, 0.20, 0.55, 0.10,
    0.10, 0.20, 0.20, 0.30, 0.30, 0.30, 0.35, 0.20, 0.25, 0.18, 0.30,
    0.20, 0.18, 0.13, 0.16, 0.26, 0.19, 0.22, 0.18, 0.12, 0.15, 0.14,
    0.12, 0.13, 0.12, 0.18, 0.19, 0.20, 0.21, 0.22, 0.85, 0.75, 0.65,
    0.55, 0.35, 0.75, 0.60, 0.65, 0.10 }; /* Broadband Albedos        */

static float spectral_albedo[16][7] = {
     { 0.20, 0.13, 0.43, 0.15, 0.12, 0.44, 0.11 },
     { 0.20, 0.15, 0.41, 0.15, 0.16, 0.40, 0.41 },
     { 0.20, 0.17, 0.39, 0.15, 0.20, 0.36, 0.23 },
     { 0.20, 0.19, 0.37, 0.15, 0.24, 0.32, 0.14 },
     { 0.20, 0.21, 0.35, 0.15, 0.28, 0.28, 0.12 },
     { 0.20, 0.23, 0.33, 0.15, 0.32, 0.24, 0.33 },
     { 0.20, 0.25, 0.31, 0.15, 0.36, 0.20, 0.26 },
     { 0.20, 0.27, 0.29, 0.15, 0.40, 0.16, 0.29 },
     { 0.20, 0.29, 0.27, 0.15, 0.36, 0.12, 0.32 },
     { 0.20, 0.31, 0.25, 0.15, 0.32, 0.16, 0.25 },
     { 0.20, 0.33, 0.23, 0.15, 0.28, 0.20, 0.27 },
     { 0.20, 0.35, 0.21, 0.15, 0.24, 0.24, 0.10 },
     { 0.20, 0.37, 0.19, 0.15, 0.20, 0.28, 0.15 },
     { 0.20, 0.39, 0.17, 0.15, 0.16, 0.32, 0.17 },
     { 0.20, 0.41, 0.15, 0.15, 0.12, 0.36, 0.35 },
     { 0.20, 0.43, 0.13, 0.15, 0.10, 0.40, 0.48 }
    };                                  /* Spectral Albedos         */

static char mtl_obsc[101][19] =
   {"Dirt",                          "Deirmendjian C",
```

```
    "Default Material",
    "LO Rural  0% RH",
    "LO Rural 80% RH",
    "LO Mtime  0% RH",
    "LO Mtime 90% RH",
    "LO Urban  0% RH",
    "LO Urban 80% RH",
    "",
    "",
    "LO Tropo  0% RH",
    "LO Tropo 80% RH",
    "LO Stratospheric",
    "LO Fresh Volcanic",
    "LO Advective Fog",
    "LO Cumulus Cloud",
    "LO Stratus Cloud",
    "LO Nimbostratus",
    "LO Subvis Cirrus",
    "LO Desert, 10 mps",
    "LO Desert, 30 mps",
    "",
    "EO Mtime  0% RH",
    "EO Mtime 70% RH",
    "EO Mtime 90% RH",
    "EO Mtime 98% RH",
    "EO Urban  0% RH",
    "EO Urban 70% RH",
    "EO Urban 90% RH",
    "EO Urban 98% RH",
    "EO Rural  0% RH",
    "EO Rural 70% RH",
    "EO Rural 90% RH",
    "EO Rural 98% RH",
    "EO Fog (hvy adv)",
    "EO Rain (drizzle)",
    "EO Rain (Tstorm)",
    "",
    "",
    "",
    "",
    "",
    "EO Cu Congestus",
    "",
    "",
    "",
    "",
    "EO Dust (hvy load)",
    "EO WP Smoke 17% RH",
    "EO WP Smoke 90% RH",
    "EO HC Smoke 85% RH"
  };

int num_grid_main_pts[3];
float axis_main_pts[3][MAXXYZ+1];

int num_grid_pts[3];
float axis_pts[3][MAXXYZ+1];
static Boolean minor_grid = FALSE;
Boolean cur_minor_grid;
```

```
    "No Cloud",
    "LO Rural 70% RH",
    "LO Rural 99% RH",
    "LO Mtime 70% RH",
    "LO Mtime 99% RH",
    "LO Urban 70% RH",
    "LO Urban 99% RH",
    "",
    "",
    "LO Tropo 70% RH",
    "LO Tropo 99% RH",
    "LO Aged Volcanic",
    "LO Radiative Fog",
    "LO Meteoric Dust",
    "LO Altostratus",
    "LO Stratus/Strato",
    "LO Std Cirrus",
    "LO Desert,  0 mps",
    "LO Desert, 20 mps",
    "",
    "",
    "EO Mtime 50% RH",
    "EO Mtime 80% RH",
    "EO Mtime 95% RH",
    "EO Mtime 99% RH",
    "EO Urban 50% RH",
    "EO Urban 80% RH",
    "EO Urban 95% RH",
    "EO Urban 99% RH",
    "EO Rural 50% RH",
    "EO Rural 80% RH",
    "EO Rural 95% RH",
    "EO Rural 99% RH",
    "EO Fog (mod rad)",
    "EO Rain (wide)",
    "EO Snow",
    "",
    "",
    "",
    "",
    "EO Fairweather Cu",
    "",
    "",
    "",
    "",
    "EO Dust (lgt load)",
    "EO HE Dust",
    "EO WP Smoke 50% RH",
    "EO Fog Oil",

        /* Material Definitions     */

  /* Grid Point Variables          */
        /* Num Main X, Y, Z Grid Pts*/
        /* X, Y, Z Major Grid Points*/

        /* Number of X, Y, Z Grid Pt*/
        /* X, Y, Z Minor Grid Points*/
        /* No Minor Grid Lines      */
        /* Temp value of minor_grid */

  /* Viewpoint Axis Flags          */
```

```
static Boolean view_axis[] = {FALSE,
                              TRUE, FALSE};   /* +Z Axis Viewpoint        */
Boolean temp_axis[3];                         /* Temp storage of Axis Flgs*/

                                            /* Miscellaneous Aerosol Vrbls   */
static Boolean label_obsc =  TRUE;            /* Aerosol Labels Displayed */
Boolean cur_lab_obsc;                         /* Temp value of label_obsc */
long mtrl_color[101][3];                      /* Aerosol Material Colors  */
Boolean mtl_color_set = FALSE;                /* "mtrl_color" not yet set */

                                            /* Transparent Color Variables   */
static int trans_index =  55;                 /* Tranparency Indx (0-255) */
static Boolean transparency =  TRUE;          /* Transparent Colors On    */

                                            /* Sun Display Variables         */
static Boolean sun_plot = TRUE;               /* Plot the Sun Location    */
static float sun_radius = 0.15;               /* Radius of the Sun        */
float sun_distance[3];                        /* X, Y, Z Comp of Sun-Orgn */
float sun_dist;                               /* Sun's distance from Orgn */
static float sun_earth[3] ={0.0,0.0,0.0}; /* Locn of Sun Ray on Grnd  */
float sun_sun[3];                             /* Location of Sun          */
static Boolean move_sun = FALSE;              /* No Sun movement          */

                                            /* Location change flags         */
static Boolean move_area = FALSE;             /* No Albedo Area movement  */
static Boolean move_regnh = FALSE;            /* No horizontal region move*/
static Boolean move_regnv = FALSE;            /* No vertical region move  */
static Boolean move_flarh = FALSE;            /* No horizontal Flare move */
static Boolean move_flarv = FALSE;            /* No vertical Flare move   */
static Boolean move_srchh = FALSE;            /* No horizontal Slite move */
static Boolean move_srchv = FALSE;            /* No vertical Slite move   */

                                            /* Current parameter indices     */
int cur_area;                                 /* Current Albedo Area      */
int cur_regn;                                 /* Current BLIRB Region     */
int cur_mtl;                                  /* Current Region Material  */
int cur_ialb;                                 /* Current value mdl2_ialb  */
int cur_flar;                                 /* Current Flare            */

                                            /* Temporary Storage Variables   */
int nndx, nndy, ttdx, ttdy;                   /* Temp storage of "mov"    */
float fac;                                    /* Temp storage of "mov"    */

                                            /* Original Viewing Display Vrbls*/
long xsize, ysize;                            /* The X,Y Window Pixel Cnt */
long ysize0;                                  /* Original value of ysize  */
float sfac;                                   /* Ratio of ysize to ysize0 */
static float org_magfactor = 1.0;             /* Orig Plot Magnification  */
static float org_center[] = {0.0, 0.0,
                             0.0};            /* Orig BLIRB Region Center */
static float org_offset[] = {0.0, 0.0};       /* Orig X,Y transln offset  */
static Boolean reset_flag = TRUE;             /* Reinitial the plot param */

                                            /* BLIRB Output Variables        */
int out_imx[3];                               /* Num of X, Y, & Z grid pt */
float out_m0[3][MAXXYZ];                      /* X, Y, & Z flux grid pts  */

int out_nwave;                                /* Number of Waves          */
float out_waveno[NV];                         /* Wavenumbers              */

float out_flux[10][NV][MAXMX][MAXMY][MAXMZ]; /* Log Sol Dir, Ref, 8Dif*/
Boolean flux_zero[10][NV];                    /* Flag for Flux Val <= 0.0 */
```

```
float mini_flux[10][NV];                    /* Minimum Flux Value        */
float maxi_flux[10][NV];                    /* Maximum Flux Value        */
float mini1_flux[10][NV];                   /* Minimum Log Flux Value    */
float maxi1_flux[10][NV];                   /* Maximum Log Flux Value    */

                                      /* Output Display Variables       */
static Boolean flux_flag[] = { FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,
               FALSE, FALSE, FALSE, FALSE}; /* No Flux Field Display   */
static Boolean noflux = TRUE;               /* "No Flux" Flag            */
static Boolean cross_axis[] = {FALSE,
               FALSE, TRUE};                /* Default Z Cross_section   */
static int cur_nwave = 0;                   /* Cross-sect Wave Number    */
static int cross_value[] = {0, 0, 0};       /* Cross-section plane       */
float log_range;                            /* Range of Log Flux Values */
int flux_index_low;                         /* Lowest Log Flux Val Indx */
int flux_index_high;                        /* Highest Log Flx Val Indx */

/*********************************************************************
 *                          VOID MAIN
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  main
 *                          Type:  C Main Program
 *                      Filename:  visual.c
 *                        Parent:  None
 *====================================================================
 *<Description>
 *    Sets up the toplevel window, the menubar, and the SGI graphics
 *    window as Motif widgets.  In addition, it checks for command
 *    line filename input.
 *====================================================================
 *<Called routines>
 *    reset                  - resets the viewing and plot parameters to
 *                             the original values
 *    create_menubar         - creates the menubar for selecting the
 *                             various options
 *    getdata                - gets the data from a BLIRB input or output
 *                             file
 *    newfile                - resets the input parameters to initial
 *                             configuration
 *    initCB                 - GL graphics mode initialization callback
 *    exposeCB               - GL window re-expose callback
 *    resizeCB               - GL window resize callback
 *    inputCB                - GL window inputs callback
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void main(int argc, char **argv)
 *    Input:
 *       argc                - (int) the number of command lines inputs
 *       argv                - (char pointer) the array of command lines
 *                             inputs
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
```

```
void main(int argc, char **argv)
{
  Arg wargs[20];
  int n;
  Widget toplevel, frame, glw;

/*-----------------------------------------------------------------------
 * --- Setup some required structures and their default values
 *-----------------------------------------------------------------------
 */
  static GLXconfig glxConfig [] = {
      {GLX_NORMAL, GLX_DOUBLE, TRUE},
      {GLX_NORMAL, GLX_RGB, TRUE},
      {GLX_NORMAL, GLX_ZSIZE, GLX_NOCONFIG},
      {0, 0, 0}
    };                              /* GL graphics defaults            */

  static String fallback_resources[] = {
    "*frame*shadowType: SHADOW_IN",
    "*glwidget*width: 1280",        /* Window width (pixels)           */
    "*glwidget*height: 1024",       /* Window height (pixels)          */
    NULL };

/*-----------------------------------------------------------------------
 * --- Initialize the toplevel Widget, "toplevel".
 *-----------------------------------------------------------------------
 */
  toplevel = XtAppInitialize(
    &app_context,                   /* Application context             */
    "BLIRB",                        /* Application class               */
    NULL, 0,                        /* command line option list        */
    &argc, argv,                    /* command line arguments          */
    fallback_resources,             /* fallback resources              */
    NULL,                           /* argument list                   */
    0);                              /* number of arguments             */

/*-----------------------------------------------------------------------
 * --- Create main data structure, the "mov" pointer
 *-----------------------------------------------------------------------
 */
  mov = (PLOTPTR) malloc (sizeof(MOVEPLOT));
  reset();                          /* Reset the view & plot variables */

/*-----------------------------------------------------------------------
 * --- Create the overall "form" widget to hold frame widget and menubar
 *     widget
 *-----------------------------------------------------------------------
 */
  n = 0;
  form = XmCreateForm(toplevel, "form", wargs, n);
  XtManageChild(form);

/*-----------------------------------------------------------------------
 * --- Create the "menubar" widget
 *-----------------------------------------------------------------------
 */
  menu = create_menubar(form);

/*-----------------------------------------------------------------------
 * --- Create the "frame" widget to hold the GL widget for plotting
 *-----------------------------------------------------------------------
 */
```

```
  n = 0;
  XtSetArg(wargs[n], XtNx, 30); n++;
  XtSetArg(wargs[n], XtNy, 30); n++;
  XtSetArg(wargs[n], XmNbottomAttachment, XmATTACH_FORM); n++;
  XtSetArg(wargs[n], XmNleftAttachment, XmATTACH_FORM); n++;
  XtSetArg(wargs[n], XmNrightAttachment, XmATTACH_FORM); n++;
  XtSetArg(wargs[n], XmNtopAttachment, XmATTACH_WIDGET); n++;
  XtSetArg(wargs[n], XmNtopWidget, menu); n++;
  XtSetArg(wargs[n], XmNleftOffset, 15); n++;
  XtSetArg(wargs[n], XmNbottomOffset, 15); n++;
  XtSetArg(wargs[n], XmNrightOffset, 15); n++;
  frame = XmCreateFrame (form, "frame", wargs, n);
  XtManageChild (frame);

/*-----------------------------------------------------------------------
 * --- Create the "GL graphics" widget over the "frame" widget
 *-----------------------------------------------------------------------
 */
  n = 0;
  XtSetArg(wargs[n], GlxNglxConfig, glxConfig); n++;
  glw = GlxCreateMDraw(frame, "glwidget", wargs, n);
  XtAddCallback(glw, GlxNginitCallback, initCB, NULL);
  XtAddCallback(glw, GlxNexposeCallback, exposeCB, NULL);
  XtAddCallback(glw, GlxNresizeCallback, resizeCB, NULL);
  XtAddCallback(glw, GlxNinputCallback, inputCB, NULL);
  XtManageChild (glw);

/*-----------------------------------------------------------------------
 * --- Realize the created widgets so you can see them on the screen
 *-----------------------------------------------------------------------
 */
  XtRealizeWidget(toplevel);

/*-----------------------------------------------------------------------
 * --- Check for command line input and process it
 *-----------------------------------------------------------------------
 */
  if(argc > 1)
  { file_name = argv[1];          /* Get data filename from Cmd Line */
    new_file = FALSE;             /* New data file not created       */
    def_file = FALSE;             /* Default data file not used      */
    area_order = FALSE;           /* "iamtl" values not ordered      */
    regn_order = FALSE;           /* MTRL cards not individualized   */
    getdata();                    /* Read the data & process it      */
  }
  else
    newfile();                    /* Setup the default inputs        */

/*-----------------------------------------------------------------------
 * --- Continuously loop looking for input events
 *-----------------------------------------------------------------------
 */
  XtAppMainLoop(app_context);
}     /*   end main()   */

/***********************************************************************
 *                          WIDGET CREATE_MENUBAR
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  create_menubar
 *                          Type:  C Widget
 *                      Filename:  visual.c
```

```
 *                        Parent:  main, getdata, regnCB, regn_delCB,
 *                                 areaCB, area_delCB, cross_sectionCB,
 *                                 albedo_chg, flarinCB, flar_delCB,
 *                                 srchinCB, srch_delCB
 *=====================================================================
 *<Description>
 *    Creates the pull-down menus, rollover menus, and menubar to
 *    control them.
 *=====================================================================
 *<Called routines>
 *    create_filemenu       - creates the "File" menupane
 *    create_viewmenu       - creates the "View" menupane
 *    create_outputsmenu    - creates the "Flux" menupane
 *    create_modifymenu     - creates the "Modify" menupane
 *    resetCB               - resets the magnification factor and eye
 *                            position to the original values
 *    create_helpmenu       - creates the "Help" menupane
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       Widget create_menubar(Widget form)
 *    Input:
 *       form                - the "form" widget to hold the menubar
 *    Output:
 *       menubar             - the main menubar widget
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
Widget create_menubar (Widget form)
{
  Widget menubar, cascade, helpcas;
  Arg wargs[10];
  int n;
  static int hv, sd;

/*-------------------------------------------------------------------
 * --- Create Menubar and attach it to the Top, Left & Right of the
 *     "form" widget
 *-------------------------------------------------------------------
 */
  n = 0;
  XtSetArg (wargs[n], XmNtopAttachment, XmATTACH_FORM); n++;
  XtSetArg (wargs[n], XmNleftAttachment, XmATTACH_FORM); n++;
  XtSetArg (wargs[n], XmNrightAttachment, XmATTACH_FORM); n++;
  menubar = XmCreateMenuBar (form, "menubar", wargs, n);

/*-------------------------------------------------------------------
 * --- Create the other buttons and the pulldown menupanes
 *-------------------------------------------------------------------
 */
  hv = 1;
  sd = 2;

  create_filemenu(menubar);         /* Create the "File" menupane      */
  create_viewmenu(menubar);         /* Create the "View" menupane      */
  create_modifymenu(menubar);       /* Create the "Modify" menupane    */
```

```
  if(!blirb_in)                          /* If datafile is Output, then */
    create_outputsmenu(menubar);    /* Create the "Flux" menupane      */

  n = 0;                             /* Create the "Reset" button       */
  XtSetArg(wargs[n], XmNlabelString, XmStringCreateLtoR ("  Reset  ",
           charset)); n++;
  XtSetArg(wargs[n], XmNmnemonic, 'R'); n++;
  cascade = XmCreateCascadeButton (menubar, "  Reset  ", wargs, n);
  XtAddCallback (cascade, XmNactivateCallback, resetCB, NULL);
  XtManageChild (cascade);

  create_helpmenu(menubar, helpcas); /* Create the "Help" menupane    */

  n = 0;                             /* Realize the Menubar             */
  XtSetArg (wargs[n], XmNmenuHelpWidget, helpcas); n++;
  XtSetValues (menubar, wargs, n);
  XtManageChild (menubar);

  return (menubar);                  /* Return the Menubar ID           */
}     /*   end create_menubar()   */

/*********************************************************************
 *                          VOID CREATE_FILEMENU
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  create_filemenu
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_menubar
 *====================================================================
 *<Description>
 *    Creates the "File" options selection pulldown menu widget.
 *====================================================================
 *<Called routines>
 *    create_pushbuttonfn   - create pushbutton widget with
 *                            accelerator keys and NULL client
 *    newfCB                - Decides whether or not to call "newfile".
 *    fileCB                - Decides whether or not to call "fileopen".
 *    savefileCB            - Saves the current inputs to a file with
 *                            the current input filename
 *    savefileasCB          - Creates a "Save Filename" Text Widget.
 *    exitCB                - Closes the windows and exits the program.
 *    create_cascadebutton  - create the cascade button for the menupane
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void create_filemenu(Widget menubar)
 *    Input:
 *       menubar            - (Widget) the "menubar" widget to hold the
 *                            file options selection menu
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void create_filemenu(Widget menubar)
{
```

```
  Widget menupane;
  Arg wargs[5];
  int n;
  static char *fctn_key[] = { "Shift-F1", "Shift-F2", "Shift-F3",
                              "Shift-F4", "F12" };
  static char *fc_key[] = { "Shift <Key>F1:", "Shift <Key>F2:",
                            "Shift <Key>F3:", "Shift <Key>F4:",
                            "<Key>F12:" };

  n = 0;                            /* Create the "File" menupane      */
  menupane = XmCreatePulldownMenu (menubar, "filepane", wargs, n);

                                    /* Create "Create New File" button */
  create_pushbuttonfn(menupane, fctn_key[0], fc_key[0],
                      "Create New File", 'N', newfCB);

                                    /* Create the "Open File" button   */
  create_pushbuttonfn(menupane, fctn_key[1], fc_key[1], "Open File",
                      'O', fileCB);

                                    /* Create the "Save File" button   */
  create_pushbuttonfn(menupane, fctn_key[2], fc_key[2], "Save File",
                      'S', savefileCB);

                                    /* Create "Save File As" button    */
  create_pushbuttonfn(menupane, fctn_key[3], fc_key[3], "Save File As",
                      'A', savefileasCB);

                                    /* Create "Exit Program" button    */
  create_pushbuttonfn(menupane, fctn_key[4], fc_key[4], "Exit Program",
                      'x', exitCB);

                                    /* Attach "File" pane to menubar   */
  create_cascadebutton(menubar, menupane, "  File  ", 'F');
}     /*   end create_filemenu()    */

/*********************************************************************
 *                          VOID CREATE_VIEWMENU
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  create_viewmenu
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_menubar
 *====================================================================
 *<Description>
 *    Creates the "View" viewing options selection pulldown menu widget.
 *====================================================================
 *<Called routines>
 *    create_axismenu       - creates the viewing axis options menupane
 *    create_sunmenu        - creates the sun display options menupane
 *    create_zoommenu       - creates the zoom options menupane
 *    create_pushbuttonfn   - create pushbutton widget with NULL
 *                            client and accelerator key
 *    minor_gridCB          - turns on/off the minor grid lines and
 *                            redraws the scene.
 *    transCB               - turns on/off the transparent color mode
 *                            for the aerosol regions
 *    obscCB                - turns on/off the obscuration and albedo
 *                            region labels
 *    create_cascadebutton  - create the cascade button for the menupane
 *====================================================================
```

```
 *<Parameters>
 *    Formal declaration:
 *       void create_viewmenu(Widget menubar)
 *    Input:
 *       menubar              - (Widget) the "menubar" widget to hold the
 *                              viewing options selection menu
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 *********************************************************************
 */
void create_viewmenu(Widget menubar)
{
  Widget menupane;
  Arg wargs[5];
  int n;

  n = 0;                              /* Create the "View" menupane      */
  menupane = XmCreatePulldownMenu (menubar, "viewpane", wargs, n);

  create_axismenu(menupane);          /* "Viewing Axis Options" menupane */
  create_sunmenu(menupane);           /* Create "Sun Options" menupane   */
  create_zoommenu(menupane);          /* Create "Zoom Options" menupane  */

                                      /* "Minor Grid Lines On/Off" button*/
  create_pushbuttonfn(menupane, "F8", "<Key>F8:",
                      "Minor Grid Lines On/Off", 'M', minor_gridCB);

                                      /* "Transparent Colors On/Off"     */
  create_pushbuttonfn(menupane, "F9", "<Key>F9:",
                      "Transparent Colors On/Off", 'T', transCB);

                                      /* "Region Definitions On/Off"     */
  create_pushbuttonfn(menupane, "F11", "<Key>F11:",
                      "Region Definitions On/Off", 'D', obscCB);

                                      /* Attach "View" pane to menubar   */
  create_cascadebutton(menubar, menupane, "  View  ", 'V');
}     /*   end create_viewmenu()    */

/*********************************************************************
 *                          VOID CREATE_AXISMENU
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  create_axismenu
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_viewmenu
 *=====================================================================
 *<Description>
 *    Creates the "Viewing Axis Options" selection pulldown menu widget.
 *=====================================================================
 *<Called routines>
 *    create_buttonsf       - creates a series of related pushbuttons
 *                            with accelerator keys and clients
 *    axisCB                - switches to the selected Axis coming out
 *                            of the screen in the plot
```

```
 *    create_cascadebutton  - create the cascade button for the menupane
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void create_axismenu(Widget viewpane)
 *    Input:
 *       viewpane            - (Widget) the "View" menupane widget to
 *                             hold the viewing axis options selection
 *                             menu
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 *********************************************************************
 */
void create_axismenu(Widget viewpane)
{
  Widget menupane;
  Arg wargs[5];
  int n;
  static int index[3] = { 1, 2, 3 };
  static char *fctn_key[] = { "F1", "F2", "F3" };
  static char *fc_key[] = { "<Key>F1:", "<Key>F2:", "<Key>F3:" };
  static char *axis_label[] = { "Positive X-axis", "Negative Y-axis",
                                "Positive Z-axis" };
  static int nm_key[3] = { 'X', 'Y', 'Z' };

  n = 0;                            /* Create "View Axis Options" menu */
  menupane = XmCreatePulldownMenu (viewpane, "axispane", wargs, n);

                                    /* Create "Positive X-axis" button */
                                    /*   "    "Negative Y-axis"    "   */
                                    /*   "    "Positive Z-axis"    "   */
  create_buttonsf(menupane, 3, index, fctn_key, fc_key, axis_label,
                  nm_key, axisCB);

                                    /* Attach "View Axis Opt" - "View" */
  create_cascadebutton(viewpane, menupane, "Viewing Axis Options", 'A');
}     /*   end create_axismenu()   */

/*********************************************************************
 *                          VOID CREATE_SUNMENU
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  create_sunmenu
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_viewmenu
 *=====================================================================
 *<Description>
 *    Creates the "Sun Options" selection pulldown menu widget.
 *=====================================================================
 *<Called routines>
 *    create_pushbuttonfn   - create pushbutton widget with NULL
 *                            client and accelerator key
 *    sunCB                 - turns on/off the plotting of the Sun
 *    sun_posCB             - sets a flag to initiate moving the point
 *                            where the line from the Sun intersects
```

```
 *                              the ground.
 *    create_cascadebutton  - create the cascade button for the menupane
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void create_sunmenu(Widget viewpane)
 *    Input:
 *       viewpane            - (Widget) the "View" menupane widget to
 *                             hold the sun options selection menu
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void create_sunmenu(Widget viewpane)
{
  Widget menupane;
  Arg wargs[5];
  int n;

  n = 0;                              /* Create "Sun Options" menupane   */
  menupane = XmCreatePulldownMenu (viewpane, "sunpane", wargs, n);

                                      /* Create "Sun Plot On/Off" button */
  create_pushbuttonfn(menupane, "F4", "<Key>F4:", "Sun Plot On/Off",
                      'P', sunCB);

                                      /* "Select New Sun Position"       */
  create_pushbuttonfn(menupane, "F5", "<Key>F5:",
                      "Select New Sun Position", 'N', sun_posCB);

                                      /* Attach "Sun Options" to "View"  */
  create_cascadebutton(viewpane, menupane, "Sun Options", 'S');
}     /*   end create_sunmenu()   */

/********************************************************************
 *                          VOID CREATE_ZOOMMENU
 ********************************************************************
 *<Begin>
 *<Identification>          Name:  create_zoommenu
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_viewmenu
 *=====================================================================
 *<Description>
 *    Creates the "Zoom Options" selection pulldown menu widget.
 *=====================================================================
 *<Called routines>
 *    create_buttonsf       - creates a series of related pushbuttons
 *                            with accelerator keys and clients
 *    zoomCB                - changes the magnification factor by 5%
 *                            and redraws the scene.
 *    create_cascadebutton  - create the cascade button for the menupane
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void create_zoommenu(Widget viewpane)
```

```
 *    Input:
 *       viewpane            - (Widget) the "View" menupane widget to
 *                             hold the zoom options selection menu
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void create_zoommenu(Widget viewpane)
{
  Widget menupane;
  Arg wargs[5];
  int n;
  static int index[2] = { 1, 2 };
  static char *fctn_key[] = { "F6", "F7" };
  static char *fc_key[] = { "<Key>F6:>", "<Key>F7:" };
  static char *zoom_label[] = { "Zoom In", "Zoom Out" };
  static int nm_key[2] = { 'I', 'O' };

  n = 0;                            /* Create "Zoom Options" menupane  */
  menupane = XmCreatePulldownMenu (viewpane, "zoompane", wargs, n);

                                    /* Create "Zoom In" button         */
                                    /*    "   "Zoom Out"   "           */
  create_buttonsf(menupane, 2, index, fctn_key, fc_key, zoom_label,
                 nm_key, zoomCB);

                                    /* Attach "Zoom Options" to "View" */
  create_cascadebutton(viewpane, menupane, "Zoom Options", 'Z');
}       /*   end create_zoommenu()   */

/********************************************************************
 *                          VOID CREATE_OUTPUTSMENU
 ********************************************************************
 *<Begin>
 *<Identification>          Name:  create_outputsmenu
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_menubar
 *=====================================================================
 *<Description>
 *    Creates the "Flux" options selection pulldown menu widget.
 *=====================================================================
 *<Called routines>
 *    create_fluxmenu       - creates the BLIRB flux options menupane
 *    create_csectmenu      - creates the cross-section orientation
 *                            options menupane
 *    create_cvaluemenu     - creates the cross-section value options
 *                            menupane
 *    create_wavemenu       - creates the wavenumber options menupane
 *    create_cascadebutton  - create the cascade button for the menupane
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void create_outputsmenu(Widget menubar)
 *    Input:
 *       menubar             - (Widget) the "menubar" widget to hold the
```

```
 *                              output options selection menu
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void create_outputsmenu(Widget menubar)
{
  Widget menupane;
  Arg wargs[5];
  int n;

  n = 0;                              /* Create "Flux" options menupane  */
  menupane = XmCreatePulldownMenu (menubar, "outpane", wargs, n);

  create_fluxmenu(menupane);          /* Create "Flux" menupane          */
  create_csectmenu(menupane);         /* Create "Cross-section" menupane */
  create_cvaluemenu(menupane);        /* Create "Cross-section Value"    */
  create_wavemenu(menupane);          /* Create "Wave Number" menupane   */

                                      /* Attach "Flux" menu to menubar   */
  create_cascadebutton(menubar, menupane, "  Flux  ", 'l');
}       /*   end create_outputsmenu()   */

/********************************************************************
 *                          VOID CREATE_FLUXMENU
 ********************************************************************
 *<Begin>
 *<Identification>          Name:  create_fluxmenu
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_outputsmenu
 *=====================================================================
 *<Description>
 *    Creates the "Flux Options" selection pulldown menu widget.
 *=====================================================================
 *<Called routines>
 *    create_pushbuttons     - creates a series of related pushbuttons
 *                             with non-NULL clients
 *    fluxCB                 - Selects the particular flux field for
 *                             display and redraws the scene.
 *    create_cascadebutton   - create the cascade button for the menupane
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void create_fluxmenu(Widget outputpane)
 *    Input:
 *       outputpane          - (Widget) the "Flux" menupane to hold
 *                             the flux options selection menu
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
```

```
 ***********************************************************************
 */
void create_fluxmenu(Widget outputpane)
{
  Widget menupane, button;
  Arg wargs[10];
  int n, i;
  static int index[11] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 0 };
  static char *flux_label[] = { "Solar Direct Flux",
                                "Solar Reflected Flux",
                                "Diffuse Flux - 1",
                                "Diffuse Flux - 2",
                                "Diffuse Flux - 3",
                                "Diffuse Flux - 4",
                                "Diffuse Flux - 5",
                                "Diffuse Flux - 6",
                                "Diffuse Flux - 7",
                                "Diffuse Flux - 8",
                                "No Flux" };
  static int nm_key[11] = { 'D', 'R', '1', '2', '3', '4', '5', '6', '7',
                            '8', 'N' };
  static int indexx[2] = { -2, -1 };
  static char *inc_label[] = { "Dec by 1 Button",
                               "Inc by 1 Button" };
  static char *fctn_key[] = { "Ctrl-F1", "Ctrl-F2" };
  static char *fc_key[] = { "Ctrl <Key>F1:", "Ctrl <Key>F2:" };

  n = 0;                            /* Create "Flux Options" menupane  */
  menupane = XmCreatePulldownMenu (outputpane, "fluxpane", wargs, n);

                                    /* "Solar Direct Flux" button      */
                                    /* "Solar Reflected Flux" button   */
                                    /* 8 "Solar Diffuse Flux" buttons  */
                                    /* "No Flux" button                */
  create_pushbuttons(menupane, 11, index, flux_label, nm_key, fluxCB);

  for (i=0; i<2; i++)               /* Add "Decrement/Increment" button*/
  { n = 0;
    XtSetArg(wargs[n], XmNlabelString,
             XmStringCreateLtoR(inc_label[i], charset)); n++;
    XtSetArg(wargs[n], XmNacceleratorText,
             XmStringCreateLtoR(fctn_key[i], charset)); n++;
    XtSetArg(wargs[n], XmNaccelerator, fc_key[i]); n++;
    button = XmCreatePushButton(menupane, inc_label[i], wargs, n);
    XtAddCallback (button, XmNactivateCallback, fluxCB, &indexx[i]);
    XtManageChild (button);
  }

                                    /* Attach "Flux Options" to "Flux" */
  create_cascadebutton(outputpane, menupane, "Flux Options", 'F');
}       /*   end create_fluxmenu()   */

/***********************************************************************
 *                          VOID CREATE_CSECTMENU
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  create_csectmenu
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_outputsmenu
 *=====================================================================
 *<Description>
```

```
 *    Creates the "Cross-section Plane Orientation" options selection
 *    pulldown menu widget.
 *=====================================================================
 *<Called routines>
 *    create_buttonsf        - creates a series of related pushbuttons
 *                             with accelerator keys and clients
 *    cross_sectionCB        - the BLIRB flux cross-section plane
 *                             orientation option callback
 *    create_cascadebutton   - create the cascade button for the menupane
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void create_csectmenu(Widget outputpane)
 *    Input:
 *       outputpane          - (Widget) the "Flux" menupane to hold
 *                             the cross-section plane orientation
 *                             options
 *                             selection menu
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void create_csectmenu(Widget outputpane)
{
  Widget menupane;
  Arg wargs[5];
  int n;
  static int index[3] = { 1, 2, 3 };
  static char *fctn_key[] = { "Ctrl-F3", "Ctrl-F4", "Ctrl-F5" };
  static char *fc_key[] = { "Ctrl <Key>F3:", "Ctrl <Key>F4:",
                            "Ctrl <Key>F5:" };
  static char *cross_label[] = { "X-Plane Cross-Section",
                                 "Y-Plane Cross-Section",
                                 "Z-Plane Cross-Section" };
  static int nm_key[3] = { 'X', 'Y', 'Z' };

  n = 0;                            /* "Cross-section Plane Orient."   */
  menupane = XmCreatePulldownMenu (outputpane, "Xpane", wargs, n);

                                    /* "X-Plane Cross-Section" button  */
                                    /* "Y-Plane Cross-Section" button  */
                                    /* "Z-Plane Cross-Section" button  */
  create_buttonsf(menupane, 3, index, fctn_key, fc_key, cross_label,
                  nm_key, cross_sectionCB);

                                    /* Attach "X-Sect Plane Orientation*/
  create_cascadebutton(outputpane, menupane,
                       "Cross-section Plane Orientation", 'O');
}     /*   end create_csectmenu()   */

/********************************************************************
 *                          VOID CREATE_CVALUEMENU
 ********************************************************************
 *<Begin>
 *<Identification>          Name:  create_cvaluemenu
 *                          Type:  C void
```

```
 *                        Filename:  visual.c
 *                          Parent:  create_outputsmenu
 *=====================================================================
 *<Description>
 *    Creates the "Cross-section Plane Value Selection" pulldown menu
 *    widget.
 *=====================================================================
 *<Called routines>
 *    planeCB                - the BLIRB flux cross-section plane
 *                             value selection callback
 *    create_cascadebutton   - create the cascade button for the menupane
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void create_cvaluemenu(Widget outputpane)
 *    Input:
 *       outputpane          - (Widget) the "Flux" menupane to hold
 *                             the cross-section plane value options
 *                             selection menu
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void create_cvaluemenu(Widget outputpane)
{
  Widget menupane, button;
  Arg wargs[10];
  int n, i, j;
  char plane_label[11];
  static int index[MAXXYZ];
  static int indexx[2] = { -2, -1 };
  static char *inc_label[] = { "Dec by 1 Button",
                               "Inc by 1 Button" };
  static char *fctn_key[] = { "Ctrl-F6", "Ctrl-F7" };
  static char *fc_key[] = { "Ctrl <Key>F6:", "Ctrl <Key>F7:" };

  n = 0;                            /* "Cross-sect Plane Value Select" */
  menupane = XmCreatePulldownMenu (outputpane, "crossval", wargs, n);

  for (j=0; j<3; j++)               /* Create "Plane Value" buttons    */
  { if(cross_axis[j])
    { for (i=0; i<out_imx[j]; i++)
      { sprintf(plane_label, " %6.3f Km", out_m0[j][i]);
        index[i] = i+1;

        n = 0;
        XtSetArg(wargs[n], XmNlabelString,
                 XmStringCreateLtoR(plane_label, charset)); n++;
        button = XmCreatePushButton(menupane, plane_label, wargs, n);
        XtAddCallback (button, XmNactivateCallback, planeCB, &index[i]);
        XtManageChild (button);
      }

      for (i=0; i<2; i++)           /* Add "Decrement/Increment" button*/
      { n = 0;
        XtSetArg(wargs[n], XmNlabelString,
```

```
                XmStringCreateLtoR(inc_label[i], charset)); n++;
        XtSetArg(wargs[n], XmNacceleratorText,
                XmStringCreateLtoR(fctn_key[i], charset)); n++;
        XtSetArg(wargs[n], XmNaccelerator, fc_key[i]); n++;
        button = XmCreatePushButton(menupane, inc_label[i], wargs, n);
        XtAddCallback (button, XmNactivateCallback, planeCB,&indexx[i]);
        XtManageChild (button);
      }
    }
  }
                                    /* Attach "X-Sect Plane Val Select"*/
  create_cascadebutton(outputpane, menupane,
                       "Cross-section Plane Value Selection", 'V');
}     /*   end create_cvaluemenu()   */

/*********************************************************************
 *                          VOID CREATE_WAVEMENU
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  create_wavemenu
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_outputsmenu
 *====================================================================
 *<Description>
 *    Creates the "Wave Number Selection" pulldown menu widget.
 *====================================================================
 *<Called routines>
 *    waveCB                - callback to change the wavenumber
 *    create_cascadebutton  - create the cascade button for the menupane
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void create_wavemenu(Widget outputpane)
 *    Input:
 *       outputpane         - (Widget) the "Flux" menupane to hold
 *                            the wavenumber value selection menu
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void create_wavemenu(Widget outputpane)
{
  Widget menupane, button;
  Arg wargs[10];
  int n, i;
  static int index[NV];
  static char wave_label[20];
  static int indexx[2] = { -2, -1 };
  static char *inc_label[] = { "Dec by 1 Button",
                               "Inc by 1 Button" };
  static char *fctn_key[] = { "Ctrl-F8", "Ctrl-F9" };
  static char *fc_key[] = { "Ctrl <Key>F8:", "Ctrl <Key>F9:" };

  n = 0;                            /* "Wave Number Selection" menu    */
  menupane = XmCreatePulldownMenu (outputpane, "waveval", wargs, n);
```

```
                                        /* "Wave Number Value" buttons     */
  for (i=0; i<out_nwave; i++)
  { sprintf(wave_label, " %10.4f per cm", out_waveno[i]);
    index[i] = i+1;

    n = 0;
    XtSetArg(wargs[n], XmNlabelString,
             XmStringCreateLtoR(wave_label, charset)); n++;
    button = XmCreatePushButton(menupane, wave_label, wargs, n);
    XtAddCallback (button, XmNactivateCallback, waveCB, &index[i]);
    XtManageChild (button);
  }

  for (i=0; i<2; i++)                   /* Add "Decrement/Increment" button*/
  { n = 0;
    XtSetArg(wargs[n], XmNlabelString,
             XmStringCreateLtoR(inc_label[i], charset)); n++;
    XtSetArg(wargs[n], XmNacceleratorText,
             XmStringCreateLtoR(fctn_key[i], charset)); n++;
    XtSetArg(wargs[n], XmNaccelerator, fc_key[i]); n++;
    button = XmCreatePushButton(menupane, inc_label[i], wargs, n);
    XtAddCallback (button, XmNactivateCallback, waveCB, &indexx[i]);
    XtManageChild (button);
  }

  create_cascadebutton(outputpane, menupane, "Wave Number Selection",
                       'W');        /* Attach "Wave # Select" - "Flux" */
}     /*   end create_wavemenu()    */

/*********************************************************************
 *                          VOID CREATE_HELPMENU
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  create_helpmenu
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_menubar
 *====================================================================
 *<Description>
 *    Creates the "Help" options selection pulldown menu widget.
 *====================================================================
 *<Called routines>
 *    create_pushbuttonss    - creates a pushbutton widget with a
 *                             non-NULL client
 *    help_generalCB         - displays general VISUAL information
 *    help_fileCB            - displays File Option information
 *    help_viewCB            - displays Viewing Options menu
 *    help_fluxCB            - displays Flux Display Options menu
 *    help_modifyCB          - displays Input Modifications Options menu
 *    help_resetCB           - displays Reset information
 *    help_rotationCB        - displays BLIRB8 Space Rotation information
 *    help_translationCB     - displays BLIRB8 Space Translation info
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void create_helpmenu(Widget menubar, Widget helpcas)
 *    Input:
 *       menubar             - (Widget) the "menubar" widget to hold the
 *                             help options selection menu
 *       helpcas             - (Widget) the "Help" cascade widget
 *    Output:
 *       None
```

```
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void create_helpmenu(Widget menubar, Widget helpcas)
{
  Widget menupane;
  Arg wargs[10];
  int n;
  static int *indx[8] = { 0, 1, 2, 3, 4, 5, 6, 7 };
  static char *button_label[] = { "General Information",
                                  "File Options",
                                  "Viewing Options",
                                  "Flux Display Options",
                                  "Input Modification Options",
                                  "Reset",
                                  "BLIRB8 Space Rotations",
                                  "BLIRB8 Space Translations" };
  static int nm_key[8] = { 'G', 'F', 'V', 'x', 'M', 's', 'R', 'T' };

  n = 0;                            /* Create the "Help" menupane      */
  menupane = XmCreatePulldownMenu (menubar, "helppane", wargs, n);

                                    /* Create "General Help" button    */
  create_pushbuttonss( menupane, indx[0], button_label[0], nm_key[0],
                       help_generalCB);

                                    /* Create "File Help" button       */
  create_pushbuttonss( menupane, indx[1], button_label[1], nm_key[1],
                       help_fileCB);

                                    /* Create "View Help" button       */
  create_pushbuttonss( menupane, indx[2], button_label[2], nm_key[2],
                       help_viewCB);

                                    /* Create "Flux Help" button       */
  create_pushbuttonss( menupane, indx[3], button_label[3], nm_key[3],
                       help_fluxCB);

                                    /* Create "Modify Help" button     */
  create_pushbuttonss( menupane, indx[4], button_label[4], nm_key[4],
                       help_modifyCB);

                                    /* Create "Reset Help" button      */
  create_pushbuttonss( menupane, indx[5], button_label[5], nm_key[5],
                       help_resetCB);

                                    /* Create "Rotation Help" button   */
  create_pushbuttonss( menupane, indx[6], button_label[6], nm_key[6],
                       help_rotationCB);

                                    /* Create "Translation Help" button*/
  create_pushbuttonss( menupane, indx[7], button_label[7], nm_key[7],
                       help_translationCB);

  n = 0;                            /* Attach "Help" pane to menubar   */
  XtSetArg(wargs[n], XmNlabelString, XmStringCreateLtoR("  Help  ",
           charset)); n++;
```

```
  XtSetArg(wargs[n], XmNmnemonic, 'H'); n++;
  XtSetArg (wargs[n], XmNsubMenuId, menupane);  n++;
  helpcas = XmCreateCascadeButton (menubar, "  Help  ", wargs, n);
  XtManageChild (helpcas);
/*
  XtAddCallback (helpcas, XmNactivateCallback, helpCB, NULL);
*/
}     /*   end create_helpmenu()   */

/*********************************************************************
 *                          VOID CREATE_MODIFYMENU
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  create_modifymenu
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_menubar
 *====================================================================
 *<Description>
 *    Creates the "Modify" options selection pulldown menu widget.
 *====================================================================
 *<Called routines>
 *    create_regionmenu     - creates the "BLIRB Region Selection"
 *                            pulldown menu widget.
 *    create_areamenu       - creates the "Albedo Areas Selection"
 *                            pulldown menu widget.
 *    create_meshmenu       - creates the "X, Y, Z Grid Mesh Selection"
 *                            pulldown menu widget.
 *    create_flarmenu       - creates the "Flare Selection" pulldown
 *                            menu widget.
 *    create_srchmenu       - creates the "SearchLight Selection"
 *                            pulldown menu widget.
 *    create_spectmenu      - creates the "Spectral Range Units Choice"
 *                            options pulldown menu.
 *    create_pushbuttonfn   - create pushbutton widget with
 *                            accelerator keys and NULL client
 *    model_optCB           - sets the various Model Options for BLIRB
 *                            on MDL1, MDL2, and MDL3 cards.
 *    cloud_optCB           - selects various Cloud Options for BLIRB
 *    sun_optCB             - selects Sun Input Options for BLIRB
 *    comp_optCB            - selects Computation Options for BLIRB
 *    output_optCB          - selects BLIRB Output File Format Option
 *    create_cascadebutton  - create the cascade button for the menupane
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void create_modifymenu(Widget menubar)
 *    Input:
 *       menubar            - (Widget) the "menubar" widget to hold the
 *                            modifying options selection menu
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void create_modifymenu(Widget menubar)
{
```

```
  Widget menupane;
  Arg wargs[5];
  int n;
  static char *fctn_key[] = { "Shift-F5", "Shift-F9", "Shift-F10",
                              "Shift-F11", "Shift-F12" };
  static char *fc_key[] = { "Shift <Key>F5:", "Shift <Key>F9:",
                            "Shift <Key>F10:", "Shift <Key>F11:",
                            "Shift <Key>F12:" };

  n = 0;                            /* Create the "Modify" menupane    */
  menupane = XmCreatePulldownMenu (menubar, "modifypane", wargs, n);

                                    /* Create "Model Changes" button   */
  create_pushbuttonfn(menupane, fctn_key[0], fc_key[0], "Model Changes",
                      'M', model_optCB);

  create_regionmenu(menupane);      /* Create the BLIRB Regions menu   */

  create_areamenu(menupane);        /* Create the Albedo Areas menu    */

  create_meshmenu(menupane);        /* Create the Grid Mesh menu       */

                                    /* Create "Cloud Changes" button   */
  create_pushbuttonfn(menupane, fctn_key[1], fc_key[1], "Cloud Changes",
                      'C', cloud_optCB);

                                    /* Create "Sun Changes" button     */
  create_pushbuttonfn(menupane, fctn_key[2], fc_key[2], "Sun Changes",
                      'S', sun_optCB);

  create_flarmenu(menupane);        /* Create the Flares menu          */

  create_srchmenu(menupane);        /* Create the SearchLight menu     */

  create_spectmenu(menupane);       /* Create the Spectral Range menu  */

                                    /* Create "Computation Changes"    */
  create_pushbuttonfn(menupane, fctn_key[3], fc_key[3],
                      "Computation Changes", 'p', comp_optCB);

                                    /* Create "Output File Changes"    */
  create_pushbuttonfn(menupane, fctn_key[4], fc_key[4],
                      "Output File Changes", 'O', output_optCB);

                                    /* Attach "Modify" menu to menubar */
  create_cascadebutton(menubar, menupane, "  Modify  ", 'M');
}     /*   end create_modifymenu()   */

/*********************************************************************
 *                          VOID CREATE_SPECTMENU
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  create_spectmenu
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_modifymenu
 *====================================================================
 *<Description>
 *    Creates the "Spectral Range Units Choice" options pulldown menu.
 *====================================================================
 *<Called routines>
 *    create_buttonsf       - creates a series of related pushbuttons
```

```
 *                              with accelerator keys and clients
 *    spectlmenuCB            - creates the Wavenumber/Wavelength Menu
 *    create_cascadebutton    - create the cascade button for the menupane
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void create_spectmenu(Widget modifypane)
 *    Input:
 *       modifypane           - (Widget) the "modifypane" widget to hold
 *                              the spectral range interval options
 *                              selection menu
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 *********************************************************************
 */
void create_spectmenu(Widget modifypane)
{
  Widget menupane;
  Arg wargs[10];
  int i, n;
  static int index[2] = { 0, 1 };
  static char *fctn_key[] = { "Ctrl-F11", "Ctrl-F12" };
  static char *fc_key[] = { "Ctrl <Key>F11:", "Ctrl <Key>F12:" };
  static char *spect_label[] = { "Wavenumber", "Wavelength" };
  static int nm_key[2] = { 'n', 'l' };

  n = 0;                              /* Create the "Spectral" menupane  */
  menupane = XmCreatePulldownMenu (modifypane, "spectpane", wargs, n);

                                      /* Create "Wavenumber/length" keys */
  create_buttonsf(menupane, 2, index, fctn_key, fc_key, spect_label,
                  nm_key, spectlmenuCB);

                                      /* Attach "Spectral" menu to menubr*/
  create_cascadebutton(modifypane, menupane, "Spectral Range Changes",
                       't');
}     /*   end create_spectmenu()    */

/*********************************************************************
 *                           VOID CREATE_AREAMENU
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  create_areamenu
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_modifymenu
 *=====================================================================
 *<Description>
 *    Creates the "Albedo Areas Selection" pulldown menu widget.
 *=====================================================================
 *<Called routines>
 *    create_arealmenu        - create the "Albedo Area Options" menu
 *    create_cascadebutton    - create the cascade button for the menupane
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
```

```
 *       void create_areamenu(Widget modifypane)
 *    Input:
 *       modifypane          - (Widget) the "modifypane" widget to hold
 *                             the Albedo Area options selection menu
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void create_areamenu(Widget modifypane)
{
  Widget menupane, areapane[IAM], cascade;
  Arg wargs[10];
  int n, nct, i;
  static int index[IAM];
  static char area_label[IAM][20];

  if(area > 0)
  { for (nct=0; nct<(area+1); nct++)
      sprintf(area_label[nct], "Albedo Area - %d", nct+1);
  }
  else if(area == 0)
  { sprintf(area_label[0], "Albedo Area - 1");
    nct = 1;
  }
  else
    nct = 0;

  if(area < (IAM - 1))
  { sprintf(area_label[nct], "Add New Albedo Area");
    nct++;
  }

  n = 0;                              /* Create the "Albedo Areas" pane  */
  menupane = XmCreatePulldownMenu (modifypane, "areapane", wargs, n);

                                      /* "Albedo Areas Selection" menu   */
  for (i=0; i<nct; i++)
  { n = 0;
    areapane[i] = XmCreatePulldownMenu (menupane, "areapane1", wargs,n);

    index[i] = i;
    create_area1menu( areapane[i], &index[i]);

    n = 0;
    XtSetArg(wargs[n], XmNlabelString,
             XmStringCreateLtoR (area_label[i], charset)); n++;
    XtSetArg (wargs[n], XmNsubMenuId, areapane[i]);  n++;
    cascade = XmCreateCascadeButton (menupane, area_label[i], wargs, n);
    XtManageChild (cascade);
  }

                                      /* Attach "Area" menu to menubar   */
  create_cascadebutton(modifypane, menupane, "Albedo Area Selection",
                       'A');
}     /*   end create_areamenu()    */
```

```
/*********************************************************************
 *                          VOID CREATE_AREA1MENU
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  create_area1menu
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_areamenu
 *====================================================================
 *<Description>
 *    Creates the "Albedo Areas Options" selection pulldown menu widget.
 *====================================================================
 *<Called routines>
 *    create_pushbuttonss   - creates a pushbutton widget with a
 *                            non-NULL client
 *    area_optCB            - sets up the scales for the area dimensions
 *    area_albCB            - calls the appropriate function for the
 *                            area albedo
 *    area_locCB            - moves the area to the desired location
 *    area_delCB            - removes the area from the BLIRB inputs
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void create_area1menu(Widget areapane, int *indx)
 *    Input:
 *       areapane           - the ID of the widget for which the
 *                            callback is registered
 *       *indx              - the pushbutton client data (passthrough)
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void create_area1menu(Widget areapane, int *indx)
{
  static char *button_label[] = { "Dimensions", "Albedo", "Location",
                                  "Delete Area" };
  static int nm_key[4] = { 'D', 'b', 'L', 'A' };

  if((*indx) != 0)
    create_pushbuttonss( areapane, indx, button_label[0], nm_key[0],
                         area_optCB);
  if((*indx) <= area)
  { create_pushbuttonss( areapane, indx, button_label[1], nm_key[1],
                         area_albCB);
    if((*indx) != 0)
    { create_pushbuttonss( areapane, indx, button_label[2], nm_key[2],
                           area_locCB);
      create_pushbuttonss( areapane, indx, button_label[3], nm_key[3],
                           area_delCB);
    }
  }
}     /*   end create_area1menu()   */

/*********************************************************************
 *                          VOID CREATE_REGIONMENU
 *********************************************************************
```

```
 *<Begin>
 *<Identification>          Name:  create_regionmenu
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_modifymenu
 *=====================================================================
 *<Description>
 *    Creates the "BLIRB Region Selection" pulldown menu widget.
 *=====================================================================
 *<Called routines>
 *    create_region1menu     - create the "Region Options" menu
 *    create_cascadebutton   - create the cascade button for the menupane
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void create_regionmenu(Widget modifypane)
 *    Input:
 *       modifypane          - (Widget) the "modifypane" widget to hold
 *                             the BLIRB Region options selection menu
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void create_regionmenu(Widget modifypane)
{
  Widget menupane, regnpane[IRM], cascade;
  Arg wargs[10];
  int n, nct, i;
  static int index[IRM];
  static char regn_label[IRM][21];

  if(regn > 0)
  { for (nct=0; nct<(regn+1); nct++)
      sprintf(regn_label[nct], "BLIRB Region - %d", nct+1);
  }
  else if(regn == 0)
  { sprintf(regn_label[0], "BLIRB Region - 1");
    nct = 1;
  }
  else
    nct = 0;

  if(regn < (IRM - 1))
  { sprintf(regn_label[nct], "Add New BLIRB Region");
    nct++;
  }

  n = 0;                              /* Create the "BLIRB Region" pane  */
  menupane = XmCreatePulldownMenu (modifypane, "regnpane", wargs, n);

                                      /* "BLIRB Regions Selection" menu  */
  for (i=0; i<nct; i++)
  { n = 0;
    regnpane[i] = XmCreatePulldownMenu (menupane, "regnpane1", wargs,n);

    index[i] = i;
```

```
      create_region1menu( regnpane[i], &index[i]);

      n = 0;
      XtSetArg(wargs[n], XmNlabelString,
               XmStringCreateLtoR (regn_label[i], charset)); n++;
      XtSetArg (wargs[n], XmNsubMenuId, regnpane[i]);  n++;
      cascade = XmCreateCascadeButton (menupane, regn_label[i], wargs, n);
      XtManageChild (cascade);
    }

                                        /* Attach "Region" menu to menubar */
  create_cascadebutton(modifypane, menupane, "Region Selection", 'R');
}     /*   end create_regionmenu()    */

/*********************************************************************
 *                          VOID CREATE_REGION1MENU
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  create_region1menu
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_regionmenu
 *====================================================================
 *<Description>
 *    Creates the "BLIRB Regions Options" selection pulldown menu
 *====================================================================
 *<Called routines>
 *    create_pushbuttonss    - creates a pushbutton widget with a
 *                             non-NULL client
 *    regn_optCB             - sets up the scales for region dimensions
 *    regn_mtlCB             - sets the region material density
 *    regn_locCB             - moves the region to the desired location
 *    regn_delCB             - removes the region from the BLIRB inputs
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void create_region1menu(Widget regnpane, int *indx)
 *    Input:
 *       regnpane            - the ID of the widget for which the
 *                             callback is registered
 *       *indx               - the pushbutton client data (passthrough)
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void create_region1menu(Widget regnpane, int *indx)
{
  static char *button_label[] = { "Dimensions", "Material - 1",
                                  "Material - 2", "Material - 3",
                                  "Location", "Delete Region" };
  static int nm_key[6] = { 'D', '1', '2', '3', 'L', 'R' };
  static int mtx[IRM][MXMTR];

  create_pushbuttonss( regnpane, indx, button_label[0], nm_key[0],
                       regn_optCB);
  if((*indx) <= regn)
```

```
  { mtx[*indx][0] = 10 * (*indx);
    create_pushbuttonss( regnpane, &mtx[*indx][0], button_label[1],
                         nm_key[1], regn_mtlCB);
    mtx[*indx][1] = 10 * (*indx) + 1;
    create_pushbuttonss( regnpane, &mtx[*indx][1], button_label[2],
                         nm_key[2], regn_mtlCB);
    mtx[*indx][2] = 10 * (*indx) + 2;
    create_pushbuttonss( regnpane, &mtx[*indx][2], button_label[3],
                         nm_key[3], regn_mtlCB);
    if((*indx) != 0)
    { create_pushbuttonss( regnpane, indx, button_label[4], nm_key[4],
                           regn_locCB);
      create_pushbuttonss( regnpane, indx, button_label[5], nm_key[5],
                           regn_delCB);
    }
  }
}     /*   end create_region1menu()    */

/*********************************************************************
 *                          VOID CREATE_FLARMENU
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  create_flarmenu
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_modifymenu
 *====================================================================
 *<Description>
 *    Creates the "Flare Selection" pulldown menu widget.
 *====================================================================
 *<Called routines>
 *    create_flar1menu       - create the "Flare Options" menu
 *    create_cascadebutton   - create the cascade button for the menupane
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void create_flarmenu(Widget modifypane)
 *    Input:
 *       modifypane          - (Widget) the "modifypane" widget to hold
 *                             the Flare options selection menu
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void create_flarmenu(Widget modifypane)
{
  Widget menupane, flarpane[NEST], cascade;
  Arg wargs[10];
  int n, nct, i;
  static int index[NEST];
  static char flar_label[NEST][14];

  if(flar > 0)
  { for (nct=0; nct<(flar+1); nct++)
      sprintf(flar_label[nct], "Flare - %d", nct+1);
  }
```

```
  else if(flar == 0)
  { sprintf(flar_label[0], "Flare - 1");
    nct = 1;
  }
  else
    nct = 0;

  if(flar < (NEST - 1))
  { sprintf(flar_label[nct], "Add New Flare");
    nct++;
  }

  n = 0;                              /* Create the "Flare" pane          */
  menupane = XmCreatePulldownMenu (modifypane, "flarpane", wargs, n);

                                      /* "Flare Selection" menu           */
  for (i=0; i<nct; i++)
  { n = 0;
    flarpane[i] = XmCreatePulldownMenu (menupane, "flarpane1", wargs,n);

    index[i] = i;
    create_flar1menu( flarpane[i], &index[i]);

    n = 0;
    XtSetArg(wargs[n], XmNlabelString,
             XmStringCreateLtoR (flar_label[i], charset)); n++;
    XtSetArg (wargs[n], XmNsubMenuId, flarpane[i]);  n++;
    cascade = XmCreateCascadeButton (menupane, flar_label[i], wargs, n);
    XtManageChild (cascade);
  }

                                      /* Attach "Flare" menu to menubar   */
  create_cascadebutton(modifypane, menupane, "Flare Selection", 'F');
}     /*   end create_flarmenu()    */

/*********************************************************************
 *                          VOID CREATE_FLAR1MENU
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  create_flar1menu
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_flarmenu
 *====================================================================
 *<Description>
 *    Creates the "Flare Options" selection pulldown menu widget.
 *====================================================================
 *<Called routines>
 *    create_pushbuttonss    - creates a pushbutton widget with a
 *                             non-NULL client
 *    flar_optCB             - sets up scales for the Flare parameters
 *    flar_locCB             - moves the Flare to the desired location
 *    flar_delCB             - removes the Flare from the BLIRB inputs
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void create_flar1menu(Widget flarpane, int *indx)
 *    Input:
 *       flarpane            - the ID of the widget for which the
 *                             callback is registered
 *       *indx               - the pushbutton client data (passthrough)
 *    Output:
```

```
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void create_flar1menu(Widget flarpane, int *indx)
{
  static char *button_label[] = { "Parameter Options", "Location",
                                  "Delete Flare" };
  static int nm_key[3] = { 'P', 'L', 'D' };

  create_pushbuttonss( flarpane, indx, button_label[0], nm_key[0],
                       flar_optCB);
  if((*indx) <= flar)
  { create_pushbuttonss( flarpane, indx, button_label[1], nm_key[1],
                         flar_locCB);
    create_pushbuttonss( flarpane, indx, button_label[2], nm_key[2],
                         flar_delCB);
  }
}     /*   end create_flar1menu()   */

/***********************************************************************
 *                          VOID CREATE_SRCHMENU
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  create_srchmenu
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_modifymenu
 *=====================================================================
 *<Description>
 *    Creates the "SearchLight Selection" pulldown menu widget.
 *=====================================================================
 *<Called routines>
 *    create_pushbuttonss   - creates a pushbutton widget with a
 *                            non-NULL client
 *    srch_optCB            - sets up the scales for the Slite params
 *    srch_locCB            - moves the Slite to the desired location
 *    srch_delCB            - removes the Slite from the BLIRB inputs
 *    create_cascadebutton  - create the cascade button for the menupane
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void create_srchmenu(Widget modifypane)
 *    Input:
 *       modifypane         - (Widget) the "modifypane" widget to hold
 *                            the Slite options selection menu
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void create_srchmenu(Widget modifypane)
```

```
{
  Widget menupane;
  Arg wargs[10];
  int n;
  static int index[3] = { 0, 1, 2 };
  static char *srch_label[] = { "Add/Modify Searchlight",
                                "Location",
                                "Delete SearchLight" };
  static int nm_key[3] = { 'A', 'L', 'D' };

  n = 0;                            /* Create the "Searchlight" pane   */
  menupane = XmCreatePulldownMenu (modifypane, "srchpane", wargs, n);

                                    /* "Slite Selection" menu          */
  create_pushbuttonss( menupane, &index[0], srch_label[0], nm_key[0],
                       srch_optCB);
  if(srch == 0)
  { create_pushbuttonss( menupane, &index[1], srch_label[1], nm_key[1],
                         srch_locCB);
    create_pushbuttonss( menupane, &index[2], srch_label[2], nm_key[2],
                         srch_delCB);
  }

                                    /* Attach "Slite" menu to menubar  */
  create_cascadebutton(modifypane, menupane, "SearchLight Selection",
                       'L');
}     /*   end create_srchmenu()    */

/*********************************************************************
 *                          VOID CREATE_MESHMENU
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  create_meshmenu
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_modifymenu
 *====================================================================
 *<Description>
 *    Creates the "X, Y, Z Grid Mesh Selection" pulldown menu widget.
 *====================================================================
 *<Called routines>
 *    create_pushbuttonfn    - create pushbutton widget with
 *                             accelerator keys and NULL client
 *    meshxmenuCB            - create the "X Grid Mesh Selection" menu
 *    meshymenuCB            - create the "Y Grid Mesh Selection" menu
 *    meshzmenuCB            - create the "Z Grid Mesh Selection" menu
 *    create_cascadebutton   - create the cascade button for the menupane
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void create_meshmenu(Widget modifypane)
 *    Input:
 *       modifypane          - (Widget) the "modifypane" widget to hold
 *                             the Grid Mesh options selection menu
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
```

```
 ***********************************************************************
 */
void create_meshmenu(Widget modifypane)
{
  Widget menupane;
  Arg wargs[5];
  int n;
  static char *fctn_key[] = { "Shift-F6", "Shift-F7", "Shift-F8" };
  static char *fc_key[] = { "Shift <Key>F6:", "Shift <Key>F7:",
                            "Shift <Key>F8:" };

  n = 0;                              /* Create the "Grid Mesh" pane      */
  menupane = XmCreatePulldownMenu (modifypane, "meshpane", wargs, n);

                                      /* "X Grid Mesh Selection" menu     */
  create_pushbuttonfn(menupane, fctn_key[0], fc_key[0], "X Grid Mesh",
                      'X', meshxmenuCB);

                                      /* "Y Grid Mesh Selection" menu     */
  create_pushbuttonfn(menupane, fctn_key[1], fc_key[1], "Y Grid Mesh",
                      'Y', meshymenuCB);

                                      /* "Z Grid Mesh Selection" menu     */
  create_pushbuttonfn(menupane, fctn_key[2], fc_key[2], "Z Grid Mesh",
                      'Z', meshzmenuCB);

                                      /* Attach "Mesh" menu to menubar    */
  create_cascadebutton(modifypane, menupane, "Grid Mesh Selection",'G');
}     /*   end create_meshmenu()    */

/***********************************************************************
 *                          VOID CREATE_MESSAGE
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  create_message
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  checkfiletypeCB, helpCB, savefileCB,
 *                                 getfilenameCB, regnCB, getdata,
 *                                 readcards, set_albedo, set_aerosol,
 *                                 writecards, help_generalCB,
 *                                 help_fileCB, help_resetCB,
 *                                 help_rotationCB, help_translationCB,
 *                                 help_axisCB, help_sunoCB,
 *                                 help_zoomCB, help_togCB, help_optCB,
 *                                 help_orCB, help_valCB, help_waveCB,
 *                                 help_modCB, help_regCB, help_areaCB,
 *                                 help_meshCB, help_cldCB, help_sunCB,
 *                                 help_flareCB, help_sliteCB,
 *                                 help_spectCB, help_compCB, help_outCB
 *=====================================================================
 *<Description>
 *    Draws a message in a dialog message box.  The cancel and help
 *    buttons are removed.
 *=====================================================================
 *<Called routines>
 *    xstr2xmstr              - converts char strings to XmString strings
 *    okCB                    - removes the message box widget
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void create_message(Widget w, char *string[],
```

```
 *                              unsigned char dialogtype)
 *    Input:
 *       w                    - (Widget) the widget to which to attach the
 *                              message box
 *       *string[]            - pointer to char array containing the
 *                              message to be displayed
 *       dialogtype           - unsigned char type of message index
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Modified and adapted the original SGI graphics course
 *              source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void create_message (Widget w, char *string[], unsigned char dialogtype)
{
  int i, n;
  Widget dialog, label;
  XmString xmstr;
  Arg wargs[10];

/*-------------------------------------------------------------------
 * --- Count the text characters up to the first NULL string.
 *-------------------------------------------------------------------
 */
  for(i=0; string[i][0] != '\0'; i++);

/*-------------------------------------------------------------------
 * --- Convert the string array to an XmString array and set it as the
 *     "label" text.
 *-------------------------------------------------------------------
 */
  xmstr = xstr2xmstr(string, i);

/*-------------------------------------------------------------------
 * --- Create the "dialog" dialog widget to display the input message.
 *-------------------------------------------------------------------
 */
  n = 0;
  XtSetArg(wargs[n], XmNmessageString, xmstr);n++;
  XtSetArg(wargs[n], XmNalignment, XmALIGNMENT_BEGINNING);n++;
  XtSetArg(wargs[n], XmNautoUnmanage, FALSE); n++;
  XtSetArg(wargs[n], XmNdialogType, dialogtype); n++;

  switch (dialogtype)
  { case XmDIALOG_MESSAGE:
      dialog = XmCreateMessageDialog(w, "Message", wargs, n);
      break;
    case XmDIALOG_ERROR:
      dialog = XmCreateErrorDialog(w, "Error", wargs, n);
      break;
    case XmDIALOG_INFORMATION:
      dialog = XmCreateInformationDialog(w, "Information", wargs, n);
      break;
    case XmDIALOG_WARNING:
      dialog = XmCreateWarningDialog(w, "Warning", wargs, n);
      break;
    case XmDIALOG_WORKING:
```

```
      dialog = XmCreateWorkingDialog(w, "Working", wargs, n);
      break;
    default:
      break;
  }

/*-----------------------------------------------------------------------
 * --- Retrieve the "label" widget and make the text left justified.
 *-----------------------------------------------------------------------
 */
  label = XmMessageBoxGetChild (dialog, XmDIALOG_MESSAGE_LABEL);

/*-----------------------------------------------------------------------
 * --- Add an "OK" callback to get rid of the message box.
 *-----------------------------------------------------------------------
 */
  XtAddCallback(dialog, XmNokCallback, okCB, NULL);

/*-----------------------------------------------------------------------
 * --- Remove the "Cancel" and "Help" buttons.
 *-----------------------------------------------------------------------
 */
  XtUnmanageChild(XmMessageBoxGetChild (dialog,XmDIALOG_CANCEL_BUTTON));
  XtUnmanageChild(XmMessageBoxGetChild (dialog,XmDIALOG_HELP_BUTTON));

/*-----------------------------------------------------------------------
 * --- Realize the message dialog box (display the message).
 *-----------------------------------------------------------------------
 */
  XtManageChild(dialog);
}     /* end create_message()  */

/***********************************************************************
 *                          VOID CREATE_MESSAGEF
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  create_messagef
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  newfCB, filefCB, exitCB
 *=====================================================================
 *<Description>
 *    Draws a message in a dialog message box.  The help button is
 *    removed.
 *=====================================================================
 *<Called routines>
 *    xstr2xmstr              - converts char strings to XmString strings
 *    okfCB                   - removes the message box widget and
 *                              continues with the ongoing operation.
 *    canceloCB               - removes the message box and calls a file
 *                              save function.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void create_messagef(Widget w, char *string[],
 *                            unsigned char dialogtype)
 *    Input:
 *       w                    - (Widget) the widget to which to attach the
 *                              message box
 *       *string[]            - pointer to char array containing the
 *                              message to be displayed
 *       dialogtype           - unsigned char type of message index
```

```
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Modified and adapted the original SGI graphics course
 *              source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void create_messagef(Widget w, char *string[], unsigned char dialogtype)
{
  int i, n;
  Widget dialog, label;
  XmString xmstr;
  Arg wargs[10];

/*-------------------------------------------------------------------
 * --- Count the text characters up to the first NULL string.
 *-------------------------------------------------------------------
 */
  for(i=0; string[i][0] != '\0'; i++);

/*-------------------------------------------------------------------
 * --- Convert the string array to an XmString array and set it as the
 *     "label" text.
 *-------------------------------------------------------------------
 */
  xmstr = xstr2xmstr(string, i);

/*-------------------------------------------------------------------
 * --- Create the "dialog" dialog widget to display the input message.
 *-------------------------------------------------------------------
 */
  n = 0;
  XtSetArg(wargs[n], XmNmessageString, xmstr);n++;
  XtSetArg(wargs[n], XmNalignment, XmALIGNMENT_BEGINNING);n++;
  XtSetArg(wargs[n], XmNautoUnmanage, FALSE); n++;
  XtSetArg(wargs[n], XmNdialogType, dialogtype); n++;
  XtSetArg(wargs[n], XmNokLabelString,
           XmStringLtoRCreate( "Continue", charset)); n++;

  switch (dialogtype)
  { case XmDIALOG_MESSAGE:
      dialog = XmCreateMessageDialog(w, "Message", wargs, n);
      break;
    case XmDIALOG_ERROR:
      dialog = XmCreateErrorDialog(w, "Error", wargs, n);
      break;
    case XmDIALOG_INFORMATION:
      dialog = XmCreateInformationDialog(w, "Information", wargs, n);
      break;
    case XmDIALOG_WARNING:
      dialog = XmCreateWarningDialog(w, "Warning", wargs, n);
      break;
    case XmDIALOG_WORKING:
      dialog = XmCreateWorkingDialog(w, "Working", wargs, n);
      break;
    default:
      break;
  }
```

```
/*-----------------------------------------------------------------------
 * --- Retrieve the "label" widget and make the text left justified.
 *-----------------------------------------------------------------------
 */
  label = XmMessageBoxGetChild (dialog, XmDIALOG_MESSAGE_LABEL);

/*-----------------------------------------------------------------------
 * --- Add an "Continue" callback to get rid of the message box and
 *     continue with the ongoing operation.
 *-----------------------------------------------------------------------
 */
  XtAddCallback(dialog, XmNokCallback, okfCB, NULL);

/*-----------------------------------------------------------------------
 * --- Add a "Cancel" callback to get rid of the message box and stop
 *     the ongoing operation.
 *-----------------------------------------------------------------------
 */
  XtAddCallback(dialog, XmNcancelCallback, canceloCB, NULL);

/*-----------------------------------------------------------------------
 * --- Remove the "Help" button.
 *-----------------------------------------------------------------------
 */
  XtUnmanageChild(XmMessageBoxGetChild (dialog,XmDIALOG_HELP_BUTTON));

/*-----------------------------------------------------------------------
 * --- Realize the message dialog box (display the message).
 *-----------------------------------------------------------------------
 */
  XtManageChild(dialog);

  XmStringFree(xmstr);
}     /* end create_messagef()  */

/***********************************************************************
 *                          VOID CREATE_PUSHBUTTONS
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  create_pushbuttons
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  Numerous Functions
 *=====================================================================
 *<Description>
 *    Creates a series of related pushbuttons with non_NULL clients.
 *=====================================================================
 *<Called routines>
 *    cbfctn                - option callback (obtained from calling
 *                            procedure)
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void create_pushbuttons(Widget w, int nct, int index[],
 *                               char *button_label[], int nm_key[],
 *                               FctnPointer cbfctn);
 *    Input:
 *       w                  - the widget to which the new button is
 *                            attached
 *       nct                - the number of buttons to create
 *       index[]            - the array of client data inputs
 *       *button_label[]    - the button label string array
```

```
 *       nm_key[]            - the mnemonic character array
 *       cbfctn              - the callback function
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570  Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void create_pushbuttons(Widget w, int nct, int index[],
                        char *button_label[], int nm_key[],
                        FctnPointer cbfctn)
{
  Widget button;
  Arg wargs[10];
  int n, i;

/*-------------------------------------------------------------------
 * --- Create the pushbuttons, attach them to the parent, and realize
 *     them.
 *-------------------------------------------------------------------
 */
  for(i=0; i<nct; i++)
  { n = 0;
    XtSetArg(wargs[n], XmNlabelString,
             XmStringCreateLtoR(button_label[i], charset)); n++;
    XtSetArg(wargs[n], XmNmnemonic, nm_key[i]); n++;
    button = XmCreatePushButton(w, button_label[i], wargs, n);
    XtAddCallback (button, XmNactivateCallback, cbfctn, &index[i]);
    XtManageChild (button);
  }
}     /*   end create_pushbuttons()   */

/*******************************************************************
 *                          VOID CREATE_PUSHBUTTONSS
 *******************************************************************
 *<Begin>
 *<Identification>          Name:  create_pushbuttonss
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  Numerous Functions
 *=====================================================================
 *<Description>
 *    Creates a pushbutton widget with a non-NULL client
 *=====================================================================
 *<Called routines>
 *    cbfctn                - option callback (obtained from calling
 *                            procedure)
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void create_pushbuttonss(Widget w, int *index,
 *                                char *button_label, int nm_key,
 *                                FctnPointer cbfctn);
 *    Input:
 *       w                  - the widget to which the new button is
 *                            attached
 *       *index             - the client data input
 *       *button_label      - the button label string
```

```
 *       nm_key              - mnemonic key for button
 *       cbfctn              - the callback function
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570  Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 *********************************************************************
 */
void create_pushbuttonss(Widget w, int *index, char *button_label,
                         int nm_key, FctnPointer cbfctn)
{
  Widget button;
  Arg wargs[10];
  int n;

/*-------------------------------------------------------------------
 * --- Create the pushbutton, attach it to the parent, and realize it.
 *-------------------------------------------------------------------
 */
  n = 0;
  XtSetArg(wargs[n], XmNlabelString,
           XmStringCreateLtoR(button_label, charset)); n++;
  XtSetArg(wargs[n], XmNmnemonic, nm_key); n++;
  button = XmCreatePushButton(w, button_label, wargs, n);
  XtAddCallback (button, XmNactivateCallback, cbfctn, index);
  XtManageChild (button);
}     /*   end create_pushbuttonss()   */

/*********************************************************************
 *                          VOID CREATE_ROWCOL
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  create_rowcol
 *                          Type:  C Widget
 *                      Filename:  visual.c
 *                        Parent:  Numerous Functions
 *=====================================================================
 *<Description>
 *    Creates a BulletinBoard Widget, Rowcol Widget, and "Finish" Button
 *=====================================================================
 *<Called routines>
 *    none
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       Widget create_rowcol( Widget w, char *label,
 *                             FctnPointer cbfctn);
 *    Input:
 *       w                  - the widget to which the new button is
 *                            attached
 *       *label             - the label for the bulletinboard & rowcol
 *       cbfctn             - the callback for the "Finish" button
 *    Output:
 *       rowcol             - the returned Rowcol Widget
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570  Elton P. Avara
 *              Developed the original source code.
```

```
 *=====================================================================
 *<End>
 ***********************************************************************
 */
Widget create_rowcol(Widget w, char *label, FctnPointer cbfctn)
{
  Widget bboard, rowcol, button;
  Arg wargs[5];
  int n;

/*---------------------------------------------------------------------
 * --- Create a bulletinboard, attach it to the parent, and realize it.
 *---------------------------------------------------------------------
 */
  n = 0;
  XtSetArg(wargs[n], XmNdialogStyle, XmDIALOG_MODELESS); n++;
  XtSetArg(wargs[n], XmNwidth, 200); n++;
  XtSetArg(wargs[n], XmNheight, 200); n++;
  bboard = XmCreateBulletinBoardDialog(w, label, wargs, n);
  XtManageChild(bboard);

/*---------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *---------------------------------------------------------------------
 */
  n = 0;
  XtSetArg(wargs[n], XmNorientation, XmVERTICAL); n++;
  rowcol = XtCreateManagedWidget(label, xmRowColumnWidgetClass, bboard,
           wargs, n);

/*---------------------------------------------------------------------
 * --- Create a finishbutton, attach it to "rowcol", and realize it.
 *---------------------------------------------------------------------
 */
  n = 0;
  button = XmCreatePushButton(rowcol, "Finished with Selections", wargs,
           n);
  XtAddCallback (button, XmNactivateCallback, cbfctn, bboard);
  XtManageChild (button);

  return (rowcol);
}     /*   end create_rowcol()    */

/***********************************************************************
 *                          VOID CREATE_BBOARD
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  create_bboard
 *                          Type:  C Widget
 *                      Filename:  visual.c
 *                        Parent:  model_optCB, meshxmenuCB,
 *                                 meshymenuCB, meshzmenuCB
 *=====================================================================
 *<Description>
 *    Creates a BulletinBoard Widget.
 *=====================================================================
 *<Called routines>
 *    none
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       Widget create_bboard(Widget w, char *label);
```

```
 *    Input:
 *       w                    - the widget to which the new button is
 *                              attached
 *       *label               - the label for the bulletinboard
 *    Output:
 *       bboard               - the returned BulletinBoard Widget
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570  Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
Widget create_bboard(Widget w, char *label)
{
  Widget bboard;
  Arg wargs[5];
  int n;

/*-------------------------------------------------------------------
 * --- Create a bulletinboard, attach it to the parent, and realize it.
 *-------------------------------------------------------------------
 */
  n = 0;
  XtSetArg(wargs[n], XmNdialogStyle, XmDIALOG_MODELESS); n++;
  XtSetArg(wargs[n], XmNwidth, 200); n++;
  XtSetArg(wargs[n], XmNheight, 200); n++;
  bboard = XmCreateBulletinBoardDialog(w, label, wargs, n);
  XtManageChild(bboard);

  return (bboard);
}     /*   end create_bboard()    */

/*******************************************************************
 *                          VOID CREATE_SEPARATOR
 *******************************************************************
 *<Begin>
 *<Identification>          Name:  create_separator
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  Numerous Functions
 *=====================================================================
 *<Description>
 *    Creates a Separator Widget.
 *=====================================================================
 *<Called routines>
 *    none
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void create_separator(Widget w, int *hv, int *sd);
 *    Input:
 *       w                    - the widget to which the new button is
 *                              attached
 *       *hv                  - the "orientation" indicator for the
 *                              widget:  0 --> Horizontal
 *                                       1 --> Vertical
 *       *sd                  - the "line character" indicator for the
 *                              widget:  1 --> Single Line
 *                                       2 --> Double Line
 *    Output:
```

```
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570  Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void create_separator(Widget w, int *hv, int *sd)
{
  Widget sep;
  Arg wargs[5];
  int n;

/*---------------------------------------------------------------------
 * --- Create a separator, attach it to the parent, and realize it.
 *---------------------------------------------------------------------
 */
  n = 0;
  if(*hv == 0)
  { XtSetArg(wargs[n], XmNorientation, XmHORIZONTAL); n++; }
  else
  { XtSetArg(wargs[n], XmNorientation, XmVERTICAL); n++; }

  if(*sd == 1)
  { XtSetArg(wargs[n], XmNseparatorType, XmSINGLE_LINE); n++; }
  else
  { XtSetArg(wargs[n], XmNseparatorType, XmDOUBLE_LINE); n++; }

  sep = XmCreateSeparator( w, "Separator", wargs, n);
  XtManageChild(sep);
}     /*   end create_separator()    */

/***********************************************************************
 *                          WIDGET CREATE_TOGGLEBUTTON
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  create_togglebutton
 *                          Type:  C Widget
 *                      Filename:  visual.c
 *                        Parent:  Numerous Functions
 *=====================================================================
 *<Description>
 *    Creates a Togglebutton Widget.
 *=====================================================================
 *<Called routines>
 *    none
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       Widget create_togglebutton(Widget w, char *tog_label,
 *                                  int *index, FctnPointer cbfctn);
 *    Input:
 *       w                  - the widget to which the new button is
 *                            attached
 *       *tog_label         - the button label string
 *       *index             - the client data input
 *       cbfctn             - the callback function
 *    Output:
 *       toggle             - the returned Togglebutton Widget
 *=====================================================================
```

```
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570  Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
Widget create_togglebutton(Widget w, char *tog_label, int *index,
                           FctnPointer cbfctn)
{
  Widget toggle;
  Arg wargs[5];
  int n;

/*---------------------------------------------------------------------
 * --- Create a toggle, attach it to the parent, and realize it.
 *---------------------------------------------------------------------
 */
  n = 0;
  XtSetArg(wargs[n], XmNshadowThickness, 3); n++;
  XtSetArg(wargs[n], XmNlabelString,
           XmStringCreateLtoR(tog_label, charset)); n++;
  toggle = XtCreateManagedWidget(tog_label, xmToggleButtonWidgetClass,
           w, wargs, n);
  XtAddCallback(toggle, XmNarmCallback, cbfctn, index);

  return (toggle);
}     /*   end create_togglebutton()   */

/********************************************************************
 *                          WIDGET CREATE_RADIOBOX
 ********************************************************************
 *<Begin>
 *<Identification>          Name:  create_radiobox
 *                          Type:  C Widget
 *                      Filename:  visual.c
 *                        Parent:  Numerous Functions
 *=====================================================================
 *<Description>
 *    Creates a Radiobox Widget.
 *=====================================================================
 *<Called routines>
 *    none
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       Widget create_radiobox(Widget w, int *nc, char *cat_label)
 *    Input:
 *       w                  - the parent widget
 *       *nc                - the number of columns
 *       *cat_label         - the button label string
 *    Output:
 *       radio              - the returned Radiobox Widget
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570  Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
Widget create_radiobox(Widget w, int *nc, char *cat_label)
```

```
{
  Widget radioform, radiolabel, radio;
  Arg wargs[15];
  int n;

/*-----------------------------------------------------------------------
 * --- Create a radiobox, attach it to the parent, and realize it.
 *-----------------------------------------------------------------------
 */
  radioform = XtCreateManagedWidget("radio", xmFormWidgetClass, w, NULL,
              0);
  radiolabel = XtCreateManagedWidget(cat_label, xmLabelWidgetClass,
               radioform, NULL, 0);
  n = 0;
  XtSetArg(wargs[n], XmNentryClass, xmToggleButtonWidgetClass); n++;
  XtSetArg(wargs[n], XmNtopAttachment, XmATTACH_WIDGET); n++;
  XtSetArg(wargs[n], XmNtopWidget, radiolabel); n++;
  XtSetArg(wargs[n], XmNradioBehavior, TRUE); n++;
  XtSetArg(wargs[n], XmNradioAlwaysOne, TRUE); n++;
  XtSetArg(wargs[n], XmNnumColumns, *nc); n++;
  XtSetArg(wargs[n], XmNpacking, XmPACK_COLUMN); n++;
  XtSetArg(wargs[n], XmNspacing, 5); n++;
  radio = XmCreateRadioBox( radioform, cat_label, wargs, n);
  XtManageChild(radio);

  return (radio);
}     /*   end create_radiobox()    */

/***********************************************************************
 *                          WIDGET CREATE_SCALE
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  create_scale
 *                          Type:  C Widget
 *                      Filename:  visual.c
 *                        Parent:  Numerous Functions
 *=====================================================================
 *<Description>
 *    Creates a Scale Widget.
 *=====================================================================
 *<Called routines>
 *    none
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       Widget create_scale(Widget w, char *scale_label, int *wid,
 *                           int *min, int *max, int *inc, int *dec,
 *                           int *val, int *swid, int *index,
 *                           FctnPointer cbfctn)
 *    Input:
 *       w                  - the parent widget
 *       *scale_label       - the scale label string
 *       *wid               - the width
 *       *min               - the minimum value
 *       *max               - the maximum value
 *       *inc               - the increment value
 *       *dec               - the number of digits to right of decimal
 *                            point
 *       *val               - the current value of scale variable
 *       *swid              - the scale width
 *       *index             - the client value for the callback
 *       cbfctn             - the callback function
```

```
 *    Output:
 *       scale                - the returned Scale Widget
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570  Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
Widget create_scale(Widget w, char *scale_label, int *wid, int *min,
                    int *max, int *inc, int *dec, int *val, int *swid,
                    int *index, FctnPointer cbfctn)
{
  Widget scaleform, scale;
  Arg wargs[20];
  int n;

/*-------------------------------------------------------------------
 * --- Create a scale, attach it to the parent, and realize it.
 *-------------------------------------------------------------------
 */
  scaleform = XtCreateManagedWidget("scale", xmFormWidgetClass, w, NULL,
              0);
  n = 0;
  XtSetArg (wargs[n], XmNtitleString,
            XmStringCreateLtoR(scale_label, charset)); n++;
  XtSetArg (wargs[n], XmNwidth, *wid); n++;
  XtSetArg (wargs[n], XmNheight, 80); n++;
  XtSetArg (wargs[n], XmNorientation, XmHORIZONTAL); n++;
  XtSetArg (wargs[n], XmNprocessingDirection, XmMAX_ON_RIGHT); n++;
  XtSetArg (wargs[n], XmNshowValue, TRUE); n++;
  XtSetArg (wargs[n], XmNminimum, *min); n++;
  XtSetArg (wargs[n], XmNmaximum, *max); n++;
  if(*inc > 0)
  { XtSetArg (wargs[n], XmNincrement, *inc); n++; }
  if(*dec > 0)
  { XtSetArg (wargs[n], XmNdecimalPoints, *dec); n++; }
  XtSetArg (wargs[n], XmNvalue, *val); n++;
  XtSetArg (wargs[n], XmNscaleWidth, *swid); n++;
  XtSetArg (wargs[n], XmNscaleHeight, 20); n++;
  scale = XmCreateScale (scaleform, scale_label, wargs, n);
  XtAddCallback(scale, XmNvalueChangedCallback, cbfctn, index);
  XtManageChild (scale);

  return (scale);
}     /*   end create_scale()    */

/********************************************************************
 *                          VOID CREATE_CASCADEBUTTON
 ********************************************************************
 *<Begin>
 *<Identification>          Name:  create_cascadebutton
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  Numerous Functions
 *=====================================================================
 *<Description>
 *    Creates a cascadebutton widget.
 *=====================================================================
 *<Called routines>
 *    none
```

```
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void create_cascadebutton(Widget parent, widget menupane,
 *                                 char button_label[], int nm_key)
 *    Input:
 *       parent              - the widget to which "menupane" is
 *                             attached
 *       menupane            - the widget to which the cascadebutton is
 *                             attached
 *       button_label[]      - the button label string
 *       nm_key              - the mnemonic character
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570  Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 **********************************************************************
 */
void create_cascadebutton(Widget parent, Widget menupane,
                          char button_label[], int nm_key)
{
  Widget cascade;
  Arg wargs[10];
  int n;

/*---------------------------------------------------------------------
 * --- Create the cascadebutton, attach it to the parent, and
 *     realize it.
 *---------------------------------------------------------------------
 */
  n = 0;
  XtSetArg(wargs[n], XmNlabelString, XmStringCreateLtoR (button_label,
           charset)); n++;
  XtSetArg(wargs[n], XmNmnemonic, nm_key); n++;
  XtSetArg (wargs[n], XmNsubMenuId, menupane);  n++;
  cascade = XmCreateCascadeButton (parent, button_label, wargs, n);
  XtManageChild (cascade);
}     /*   end create_cascadebutton()    */

/**********************************************************************
 *                          VOID CREATE_PUSHBUTTONFN
 **********************************************************************
 *<Begin>
 *<Identification>          Name:  create_pushbuttonfn
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  Numerous Functions
 *=====================================================================
 *<Description>
 *    Creates a pushbutton widget with accelerator key and NULL
 *    client.
 *=====================================================================
 *<Called routines>
 *    cbfctn                - option callback (obtained from calling
 *                            procedure)
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
```

```
 *       void create_pushbuttonfn(Widget w, char *fctn_key,
 *                                char *fc_key, char *button_label,
 *                                int nm_key, FctnPointer cbfctn);
 *    Input:
 *       w                    - the widget to which the new buttons are
 *                              attached
 *       *fctn_key            - the accelerator text string
 *       *fc_key              - the accelerator string
 *       *button_label        - the button label string
 *       nm_key               - the mnemonic character
 *       cbfctn               - the callback function
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570  Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void create_pushbuttonfn(Widget w, char *fctn_key, char *fc_key,
                    char *button_label, int nm_key, FctnPointer cbfctn)
{
  Widget button;
  Arg wargs[10];
  int n;

/*-------------------------------------------------------------------
 * --- Create the pushbutton, attach it to the parent, and realize it
 *-------------------------------------------------------------------
 */
  n = 0;
  XtSetArg(wargs[n], XmNlabelString,
           XmStringCreateLtoR(button_label, charset)); n++;
  XtSetArg(wargs[n], XmNmnemonic, nm_key); n++;
  XtSetArg(wargs[n], XmNacceleratorText,
           XmStringCreateLtoR(fctn_key, charset)); n++;
  XtSetArg(wargs[n], XmNaccelerator, fc_key); n++;
  button = XmCreatePushButton(w, button_label, wargs, n);
  XtAddCallback (button, XmNactivateCallback, cbfctn, NULL);
  XtManageChild (button);
}     /*   end create_pushbuttonfn()   */

/*******************************************************************
 *                         VOID CREATE_BUTTONSF
 *******************************************************************
 *<Begin>
 *<Identification>          Name:  create_buttonsf
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  Numerous Functions
 *=====================================================================
 *<Description>
 *    Creates several related pushbutton widgets with accelerator keys
 *    and clients.
 *=====================================================================
 *<Called routines>
 *    cbfctn                - option callback (obtained from calling
 *                            procedure)
 *=====================================================================
 *<Parameters>
```

```
 *    Formal declaration:
 *       void create_buttonsf(Widget w, int nct, int index[],
 *                            char *fctn_key[], char *fc_key[],
 *                            char *button_label[], int nm_key[],
 *                            FctnPointer cbfctn);
 *    Input:
 *       w                  - the widget to which the new buttons are
 *                            attached
 *       nct                - the number of new buttons
 *       index              - the array of integers which should be
 *                            passed to the callback
 *       *fctn_key          - the array of accelerator text strings
 *       *fc_key            - the array of accelerator strings
 *       *button_label      - the array of button label strings
 *       nm_key             - the array of mnemonic characters
 *       cbfctn             - the callback function
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570  Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 *********************************************************************
 */
void create_buttonsf(Widget w, int nct, int index[], char *fctn_key[],
                     char *fc_key[], char *button_label[],
                     int nm_key[], FctnPointer cbfctn)
{
  Widget button;
  Arg wargs[10];
  int n, i;

/*-------------------------------------------------------------------
 * --- Create the series of pushbuttons, attach them to the parent, and
 *     realize them
 *-------------------------------------------------------------------
 */
  for (i=0; i<nct; i++)
  { n = 0;
    XtSetArg(wargs[n], XmNlabelString,
             XmStringCreateLtoR(button_label[i], charset)); n++;
    XtSetArg(wargs[n], XmNmnemonic, nm_key[i]); n++;
    XtSetArg(wargs[n], XmNacceleratorText,
             XmStringCreateLtoR(fctn_key[i], charset)); n++;
    XtSetArg(wargs[n], XmNaccelerator, fc_key[i]); n++;
    button = XmCreatePushButton(w, button_label[i], wargs, n);
    XtAddCallback (button, XmNactivateCallback, cbfctn, &index[i]);
    XtManageChild (button);
  }
}     /*   end create_buttonsf()   */

/*********************************************************************
 *                          VOID FILECB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  fileCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_filemenu
 *=====================================================================
```

```
 *<Description>
 *    Decides whether or not to call "fileopen".
 *=====================================================================
 *<Called routines>
 *    create_messagef        - draws a message in a dialog message box
 *                             and continues or halts the current
 *                             operation.
 *    fileopen               - sets up the "File Select" menu for new
 *                             BLIRB input or output file
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void fileCB(Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                   - the ID of the widget for which the
 *                             callback is registered
 *       c                   - the data passed to the routine
 *       call_data           - a pointer to the callback structure which
 *                             contains information on why the callback
 *                             occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void fileCB (Widget w, XtPointer c, XtPointer call_data)
{
  static char *msg[] = {
                "The original BLIRB input data has been\n",
                "    modified and not saved to disk.\n",
                "You will destroy the current data if you proceed!\n",
                "==================================================\n",
                "      CONTINUE to proceed, CANCEL to stop.\n",
                "" };

  filefctn = 1;

  if(new_file)
    create_messagef( menu, msg, XmDIALOG_ERROR);
  else
    fileopen();
}     /*   end fileCB()    */

/********************************************************************
 *                          VOID FILEOPEN
 ********************************************************************
 *<Begin>
 *<Identification>          Name:  fileopen
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  fileCB, okfCB
 *=====================================================================
 *<Description>
 *    Gets the BLIRB input or output file selection.
 *=====================================================================
 *<Called routines>
 *    checkfiletypeCB        - gets the input BLIRB filename and checks
```

```
 *                              the file type for input or output
 *    cancelCB                - option cancellation callback
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void fileopen( void )
 *    Input:
 *       None
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void fileopen (void)
{
  Widget fileinfo, helpbutton;
  XmString blirb_filename_filter, filter_label, file_name;
  Arg wargs[10];
  int n;

/*-------------------------------------------------------------------
 * --- Use the Widget "file_dialog" to create a Bulletin Board Dialog
 *     Widget with that ID.
 *-------------------------------------------------------------------
 */
  n = 0;
  XtSetArg(wargs[n], XmNdialogStyle, XmDIALOG_MODELESS); n++;
  XtSetArg(wargs[n], XmNwidth, 350); n++;
  XtSetArg(wargs[n], XmNheight, 450); n++;
  file_dialog = XmCreateBulletinBoardDialog(menu, "Select File", wargs,
                n);

/*-------------------------------------------------------------------
 * --- Create a File Selection Box Widget attached to "file_dialog".
 *-------------------------------------------------------------------
 */
  blirb_filename_filter = XmStringLtoRCreate(filename_filter,charset);
  filter_label = XmStringLtoRCreate ("Filename Filter", charset);
  file_name = XmStringLtoRCreate ("Selected Filename", charset);

  n = 0;
  XtSetArg (wargs[n], XmNdirMask, blirb_filename_filter); n++;
  XtSetArg (wargs[n], XmNfilterLabelString, filter_label); n++;
  XtSetArg (wargs[n], XmNselectionLabelString, file_name); n++;
  fileinfo = XmCreateFileSelectionBox(file_dialog, "File Selection",
             wargs, n);
  XtAddCallback (fileinfo, XmNokCallback, checkfiletypeCB, file_dialog);
  XtAddCallback (fileinfo, XmNcancelCallback, cancelCB, file_dialog);
  XtManageChild (fileinfo);

/*-------------------------------------------------------------------
 * --- Free the space used for the XmStrings.
 *-------------------------------------------------------------------
 */
  XmStringFree(blirb_filename_filter);
  XmStringFree(filter_label);
  XmStringFree(file_name);
```

```
/*-----------------------------------------------------------------------
 * --- Remove the "Help" button from the file selection box.
 *-----------------------------------------------------------------------
 */
  helpbutton = XmFileSelectionBoxGetChild (fileinfo,
                                           XmDIALOG_HELP_BUTTON);
  XtUnmanageChild (helpbutton);

/*-----------------------------------------------------------------------
 * --- Realize the File Selection Box Widget.
 *-----------------------------------------------------------------------
 */
  XtManageChild(file_dialog);
}     /*   end fileopen()   */

/***********************************************************************
 *                          VOID CHECKFILETYPECB
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  checkfiletypeCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  fileopen
 *=====================================================================
 *<Description>
 *    Gets the input BLIRB filename and checks the file type for input
 *    or output.
 *=====================================================================
 *<Called routines>
 *    blirb_inout           - decides whether a BLIRB input or output
 *                            file was selected.  If neither, an error
 *                            flag is returned.
 *    create_message        - draws a message in a dialog message box
 *    getdata               - gets the data from a BLIRB input or output
 *                            file
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void checkfiletypeCB(Widget w, XtPointer c,XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - the ID of the widget to which to attach
 *                            the message box widget
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S  (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void checkfiletypeCB (Widget w, XtPointer c, XtPointer call_data)
{
  XmSelectionBoxCallbackStruct *call_value =
                              (XmSelectionBoxCallbackStruct *)call_data;
  static char *error_mesg[] = {
```

```
                "Selected Filename is neither Input nor Output.\n",
                "Please try another Filename.\n",
                "" };

/*-----------------------------------------------------------------------
 * --- Remove the File Selection Box Widget from the screen.
 *-----------------------------------------------------------------------
 */
  XtUnmanageChild(file_dialog);

/*-----------------------------------------------------------------------
 * --- Get the selected BLIRB data filename.
 *-----------------------------------------------------------------------
 */
  XmStringGetLtoR(call_value->value, charset, &file_name);

/*-----------------------------------------------------------------------
 * --- Check the datafile to find out if it is input, output, or
 *     neither.
 *-----------------------------------------------------------------------
 */
  blirb_inout();

/*-----------------------------------------------------------------------
 * --- If the datafile is neither input nor output, display an error
 *     message in a box.  If the datafile is either input or output,
 *     then get the data from the file.
 *-----------------------------------------------------------------------
 */
  if(badfile)
    create_message( (Widget)c, error_mesg, XmDIALOG_ERROR);
  else
  { new_file = FALSE;
    def_file = FALSE;              /* Default data file not used      */
    area_order = FALSE;
    regn_order = FALSE;
    getdata();
  }
}     /*   end checkfiletypeCB()   */

/***********************************************************************
 *                           VOID INITCB
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  initCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  main
 *=====================================================================
 *<Description>
 *    Initializes GL graphics modes.
 *=====================================================================
 *<Called routines>
 *    None
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void initCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                    - the ID of the widget for which the
 *                              callback is registered
 *       c                    - a pointer to any input data to be given to
```

```
 *                                the routine
 *       call_data              - a pointer to the callback structure which
 *                                contains information on why the callback
 *                                occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void initCB (Widget w, XtPointer c, XtPointer call_data)
{
  GLXwinset(XtDisplay(w), XtWindow(w));

  getsize( &xsize, &ysize );
  xsize -= 109;
  ysize -= 80;
  ysize0 = ysize;
  sfac = 1.0;

  if (getgdesc(GD_BITS_NORM_ZBUFFER) == 0)
  { printf("This machine does not have a hardware zbuffer\n");
    exit(0);
  }

  shademodel(FLAT);
  mmode(MVIEWING);
  backface(TRUE);
  zbuffer (FALSE);

  subpixel(TRUE);
  linesmooth(SML_SMOOTHER);
  pntsmooth(SMP_SMOOTHER);
  polysmooth(PYSM_ON);
}     /*   end initCB()    */

/********************************************************************
 *                          VOID EXITCB
 ********************************************************************
 *<Begin>
 *<Identification>          Name:  exitCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_filemenu
 *=====================================================================
 *<Description>
 *    Closes the windows and exits the program.
 *=====================================================================
 *<Called routines>
 *    create_messagef       - draws a message in a dialog message box
 *                            and continues or halts the current
 *                            operation.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void exitCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
```

```
 *                              callback is registered
 *       c                    - a pointer to any input data to be given to
 *                              the routine
 *       call_data            - a pointer to the callback structure which
 *                              contains information on why the callback
 *                              occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void exitCB (Widget w, XtPointer c, XtPointer call_data)
{
  static char *msg[] = {
                "The original BLIRB input data has been\n",
                "    modified and not saved to disk.\n",
                "Do you still wish to exit the program?\n",
                "=====================================\n",
                " CONTINUE to proceed, CANCEL to stop.\n",
                "" };

  filefctn = 2;

  if(new_file)
    create_messagef( menu, msg, XmDIALOG_ERROR);
  else
  { XtCloseDisplay(XtDisplay(w));
    exit(0);
  }
}     /*   end exitCB()   */

/********************************************************************
 *                          VOID EXPOSECB
 ********************************************************************
 *<Begin>
 *<Identification>          Name:  exposeCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  main
 *=====================================================================
 *<Description>
 *    Called when the window is uncovered or moved.
 *=====================================================================
 *<Called routines>
 *    drawscene            - Plots the 3-D BLIRB grid points, albedo
 *                           areas, aerosol regions, and the output
 *                           flux.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void exposeCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                    - the ID of the widget for which the
 *                              callback is registered
 *       c                    - a pointer to any input data to be given to
 *                              the routine
 *       call_data            - a pointer to the callback structure which
```

```
 *                              contains information on why the callback
 *                              occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void exposeCB (Widget w, XtPointer c, XtPointer call_data)
{
/*-------------------------------------------------------------------
 * --- Redraw the window and redraw the scene in it.
 *-------------------------------------------------------------------
 */
  GLXwinset (XtDisplay(w), XtWindow(w));
  drawscene();
}     /*   end exposeCB()   */

/********************************************************************
 *                          VOID RESIZECB
 ********************************************************************
 *<Begin>
 *<Identification>          Name:  resizeCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  main
 *=====================================================================
 *<Description>
 *    Called when the window is resized.
 *=====================================================================
 *<Called routines>
 *    drawscene             - Plots the 3-D BLIRB grid points, albedo
 *                            areas, aerosol regions, and the output
 *                            flux.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void resizeCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - a pointer to any input data to be given to
 *                            the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void resizeCB (Widget w, XtPointer c, XtPointer call_data)
{
```

```
  GlxDrawCallbackStruct *call_value =(GlxDrawCallbackStruct *)call_data;
  float ysize0, fac;

  ysize0 = ysize;

/*-----------------------------------------------------------------------
 * --- Resize the window and redraw it.
 *-----------------------------------------------------------------------
 */
  GLXwinset (XtDisplay(w), XtWindow(w));
  viewport (0, (Screencoord) call_value->width-1,
            0, (Screencoord) call_value->height-1);

/*-----------------------------------------------------------------------
 * --- Get the window size.
 *-----------------------------------------------------------------------
 */
  getsize( &xsize, &ysize );
  sfac = (float)ysize / (float)ysize0;

/*-----------------------------------------------------------------------
 * --- Redraw the scene in the window.
 *-----------------------------------------------------------------------
 */
  label_obsc = FALSE;
  obsc();
  drawscene();
}     /*   end resizeCB()    */

/***********************************************************************
 *                          VOID INPUTCB
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  inputCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  main
 *=====================================================================
 *<Description>
 *    Handles all types of input from the GL widget.  The KeyRelease
 *    handles the ESCape key, so that it exits the program.  When
 *    Button1 is pressed, the mode is set, and the mouse position is
 *    recorded.  As long as the mouse is in motion, the relative
 *    position of the mouse is stored, and the scene is redrawn.  When
 *    the button is released, this mode is reset, and motion is stopped.
 *=====================================================================
 *<Called routines>
 *    area_fix              - rectifies the Albedo area location
 *    regn_fix              - rectifies the BLIRB Region location
 *    reset                 - resets the values of "mov"
 *    drawscene             - Plots the 3-D BLIRB grid points, albedo
 *                            areas, aerosol regions, and the output
 *                            flux.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void inputCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - the input from the calling routine
 *       call_data          - a pointer to the callback structure which
```

```
 *                              contains information on why the callback
 *                              occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void inputCB(Widget w, XtPointer c, XtPointer call_data)
{
  GlxDrawCallbackStruct *call_value =(GlxDrawCallbackStruct *)call_data;
  char buffer[1];
  KeySym keysym;
  static Boolean mode1 = RELEASED;
  static Boolean mode3 = RELEASED;
  static int org_left_y, left_y, org_left_x, left_x;
  static int org_right_y, right_y, org_right_x, right_x;
  int i;
  float del, chg;

  switch(call_value->event->type)
  { case KeyRelease:
/*---------------------------------------------------------------------
 * --- If a key on the keyboard was released, check to see if it was the
 *     ESCape key.  It is necessary to convert the keycode to a keysym
 *     before it is possible to check if it is an ESCape.
 *---------------------------------------------------------------------
 */
      if(XLookupString((XKeyReleasedEvent *)
                       (call_value->event), buffer, 1, &keysym,NULL) == 1)
      { if(keysym == (KeySym)XK_Escape)
          exit(0);
      }
      break;

    case ButtonPress:
/*---------------------------------------------------------------------
 * --- If a mouse button was pressed, find out if it was the left or
 *     right button and record the current mouse position.
 *---------------------------------------------------------------------
 */
      switch(call_value->event->xbutton.button)
      { case Button1:
          mode1 = PRESSED;
          org_left_x = left_x = call_value->event->xbutton.x;
          org_left_y = left_y = call_value->event->xbutton.y;
          break;

        case Button3:
          mode3 = PRESSED;
          org_right_x = right_x = call_value->event->xbutton.x;
          org_right_y = right_y = call_value->event->xbutton.y;
          break;
      }
/*---------------------------------------------------------------------
 * --- Get the size of the GL window just in case it has changed.
 *---------------------------------------------------------------------
 */
```

```
      getsize( &xsize, &ysize );
      sfac = (float)ysize / (float)ysize0;
      break;

    case ButtonRelease:
/*-----------------------------------------------------------------------
 * --- If a mouse button was released, find out if it was the left or
 *     right button.  If it was the left button and the user is moving
 *     the position of the Sun-Earth intersection point or changing the
 *     position of an Albedo area, restore the scene parameters to
 *     those in effect before moving the Sun.
 *-----------------------------------------------------------------------
 */
      switch(call_value->event->xbutton.button)
        { case Button1:
            mode1 = RELEASED;
            if((move_sun || move_area) || (move_regnh || move_regnv) ||
               (move_flarh || move_flarv) || (move_srchh || move_srchv))
            { if(move_sun)
                move_sun = FALSE;
              else if(move_area)
              { move_area = FALSE;
                label_obsc = cur_lab_obsc;
                area_fix();
              }
              else if(move_regnh)
              { move_regnh = FALSE;
                view_axis[0] = view_axis[2] = FALSE;
                view_axis[1] = TRUE;
                reset();
              }
              else if(move_regnv)
              { move_regnv = FALSE;
                label_obsc = cur_lab_obsc;
                minor_grid = cur_minor_grid;
                regn_fix();
              }
              else if(move_flarh)
              { move_flarh = FALSE;
                view_axis[0] = view_axis[2] = FALSE;
                view_axis[1] = TRUE;
                reset();
              }
              else if(move_flarv)
              { move_flarv = FALSE;
                label_obsc = cur_lab_obsc;
                minor_grid = cur_minor_grid;
              }
              else if(move_srchh)
              { move_srchh = FALSE;
                view_axis[0] = view_axis[2] = FALSE;
                view_axis[1] = TRUE;
                reset();
              }
              else if(move_srchv)
              { move_srchv = FALSE;
                label_obsc = cur_lab_obsc;
                minor_grid = cur_minor_grid;
              }

              if((!move_sun && !move_area) &&
                 (!move_regnh && !move_regnv) &&
```

```
                (!move_flarh && !move_flarv) &&
                (!move_srchh && !move_srchv))
            { mov->magfactor = fac;
              mov->ndx = nndx;
              mov->ndy = nndy;
              mov->tdx = ttdx;
              mov->tdy = ttdy;

              for(i=0; i<3; i++)
                view_axis[i] = temp_axis[i];
            }

/*-----------------------------------------------------------------------
 * --- Redraw the scene in the window.
 *-----------------------------------------------------------------------
 */
            drawscene();
          }
        break;

        case Button3:
        mode3 = RELEASED;
        break;
      }
      break;

    case MotionNotify:
      if(mode1 == PRESSED && call_value->event->xmotion.state &
                                Button1Mask)
/*-----------------------------------------------------------------------
 * --- If the mouse is moving while the left button is pressed, either
 *     change the viewpoint or change the Sun_Earth intersection point
 *     by the relative change in the mouse position.
 *-----------------------------------------------------------------------
 */
      { org_left_x = left_x;
        left_x = call_value->event->xbutton.x;
        chg = 0.01067*(float)(left_x-org_left_x) / (sfac*org_magfactor);
        if(move_sun)
          sun_earth[0] += chg + axis_pts[0][0];
        else if(move_area)
        { area_alx[cur_area] += chg + axis_pts[0][0];
          area_ahx[cur_area] += chg + axis_pts[0][0];

          del = area_ahx[cur_area] - area_alx[cur_area];
          if(area_alx[cur_area] < 0.0)
          { area_alx[cur_area] = 0.0;
            area_ahx[cur_area] = del;
          }
          if(area_ahx[cur_area] > regn_rh[0][0])
          { area_ahx[cur_area] = regn_rh[0][0];
            area_alx[cur_area] = area_ahx[cur_area] - del;
          }
        }
        else if(move_regnh || move_regnv)
        { regn_rl[0][cur_regn] += chg + axis_pts[0][0];
          regn_rh[0][cur_regn] += chg + axis_pts[0][0];

          del = regn_rh[0][cur_regn] - regn_rl[0][cur_regn];
          if(regn_rl[0][cur_regn] < 0.0)
          { regn_rl[0][cur_regn] = 0.0;
            regn_rh[0][cur_regn] = del;
```

```
    }
    if(regn_rh[0][cur_regn] > regn_rh[0][0])
    { regn_rh[0][cur_regn] = regn_rh[0][0];
      regn_rl[0][cur_regn] = regn_rh[0][cur_regn] - del;
    }
  }
  else if(move_flarh || move_flarv)
    flar_xflar[cur_flar] += chg + axis_pts[0][0];
  else if(move_srchh || move_srchv)
    srch_xsrch += chg + axis_pts[0][0];
  else
    mov->ndx = mov->ndx - (left_x - org_left_x);

  org_left_y = left_y;
  left_y = call_value->event->xbutton.y;
  chg = 0.01067*(float)(left_y-org_left_y) / (sfac*org_magfactor);
  if(move_sun)
    sun_earth[1] -= chg + axis_pts[1][0];
  else if(move_area)
  { area_aly[cur_area] -= chg + axis_pts[1][0];
    area_ahy[cur_area] -= chg + axis_pts[1][0];

    del = area_ahy[cur_area] - area_aly[cur_area];
    if(area_aly[cur_area] < 0.0)
    { area_aly[cur_area] = 0.0;
      area_ahy[cur_area] = del;
    }
    if(area_ahy[cur_area] > regn_rh[1][0])
    { area_ahy[cur_area] = regn_rh[1][0];
      area_aly[cur_area] = area_ahy[cur_area] - del;
    }
  }
  else if(move_regnh)
  { regn_rl[1][cur_regn] -= chg + axis_pts[1][0];
    regn_rh[1][cur_regn] -= chg + axis_pts[1][0];

    del = regn_rh[1][cur_regn] - regn_rl[1][cur_regn];
    if(regn_rl[1][cur_regn] < 0.0)
    { regn_rl[1][cur_regn] = 0.0;
      regn_rh[1][cur_regn] = del;
    }
    if(regn_rh[1][cur_regn] > regn_rh[1][0])
    { regn_rh[1][cur_regn] = regn_rh[1][0];
      regn_rl[1][cur_regn] = regn_rh[1][cur_regn] - del;
    }
  }
  else if(move_regnv)
  { regn_rl[2][cur_regn] -= chg + axis_pts[2][0];
    regn_rh[2][cur_regn] -= chg + axis_pts[2][0];

    del = regn_rh[2][cur_regn] - regn_rl[2][cur_regn];
    if(regn_rl[2][cur_regn] < 0.0)
    { regn_rl[2][cur_regn] = 0.0;
      regn_rh[2][cur_regn] = del;
    }
    if(regn_rh[2][cur_regn] > regn_rh[2][0])
    { regn_rh[2][cur_regn] = regn_rh[2][0];
      regn_rl[2][cur_regn] = regn_rh[2][cur_regn] - del;
    }
  }
  else if(move_flarh)
    flar_yflar[cur_flar] -= chg + axis_pts[1][0];
```

```
        else if(move_flarv)
          flar_zflar[cur_flar] -= chg + axis_pts[2][0];
        else if(move_srchh)
          srch_ysrch -= chg + axis_pts[1][0];
        else if(move_srchv)
          srch_zsrch -= chg + axis_pts[2][0];
        else
        { mov->ndy = mov->ndy + (left_y - org_left_y);
          reset_flag = FALSE;
        }
      }
      else if(mode3 == PRESSED && call_value->event->xmotion.state &
                                  Button3Mask)
/*-----------------------------------------------------------------------
 * --- If the mouse is moving while the right button is pressed,
 *     translate the scene by the relative change in mouse position.
 *-----------------------------------------------------------------------
 */
      { org_right_x = right_x;
        right_x = call_value->event->xbutton.x;
        mov->tdx = mov->tdx - (right_x - org_right_x);

        org_right_y = right_y;
        right_y = call_value->event->xbutton.y;
        mov->tdy = mov->tdy + (right_y - org_right_y);

        reset_flag = FALSE;
      }

/*-----------------------------------------------------------------------
 * --- Redraw the scene in the window.
 *-----------------------------------------------------------------------
 */
      drawscene();
      break;
  }
}     /*   end inputCB()   */

/***********************************************************************
 *                          VOID RESETCB
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  resetCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_menubar
 *=====================================================================
 *<Description>
 *    Resets the magnification factor and eye position to the original
 *    values and redraws the scene.
 *=====================================================================
 *<Called routines>
 *    reset                 - resets the viewing and plot parameters to
 *                            the original values
 *    drawscene             - Plots the 3-D BLIRB grid points, albedo
 *                            areas, aerosol regions, and the output
 *                            flux.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void resetCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
```

```
 *       w                    - the ID of the widget for which the
 *                              callback is registered
 *       c                    - a pointer to any input data to be given to
 *                              the routine
 *       call_data            - a pointer to the callback structure which
 *                              contains information on why the callback
 *                              occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 *********************************************************************
 */
void resetCB (Widget w, XtPointer c, XtPointer call_data)
{
  reset();
  reset_flag = TRUE;

  drawscene();
}     /*   end resetCB()    */

/*********************************************************************
 *                          VOID SUN_POSCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  sun_posCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_sunmenu
 *=====================================================================
 *<Description>
 *    Sets a flag to initiate moving the point where the line from the
 *    sun intersects the ground.
 *=====================================================================
 *<Called routines>
 *    reset                - resets the viewing and plot parameters to
 *                           the original values
 *    drawscene            - Plots the 3-D BLIRB grid points, albedo
 *                           areas, aerosol regions, and the output
 *                           flux.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void sun_posCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                    - the ID of the widget for which the
 *                              callback is registered
 *       c                    - the input data from the calling routine
 *       call_data            - a pointer to the callback structure which
 *                              contains information on why the callback
 *                              occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
```

```
 *<End>
 ***********************************************************************
 */
void sun_posCB (Widget w, XtPointer c, XtPointer call_data)
{
  int i;

  move_sun = TRUE;

/*----------------------------------------------------------------------
 * --- Save the current viewing axis flags and choose a +Z axis option.
 *----------------------------------------------------------------------
 */
  for(i=0; i<3; i++)
    temp_axis[i] = view_axis[i];

  view_axis[0] = view_axis[1] = FALSE;
  view_axis[2] = TRUE;

/*----------------------------------------------------------------------
 * --- Save the current "mov" parameters.
 *----------------------------------------------------------------------
 */
  fac = mov->magfactor;
  nndx = mov->ndx;
  nndy = mov->ndy;
  ttdx = mov->tdx;
  ttdy = mov->tdy;

/*----------------------------------------------------------------------
 * --- Reset the viewpoint and redraw the scene in the window.
 *----------------------------------------------------------------------
 */
  reset();
  drawscene();
}     /*   end sun_posCB()    */

/***********************************************************************
 *                              VOID ZOOMCB
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  zoomCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_zoommenu
 *=====================================================================
 *<Description>
 *    Changes the magnification factor by 5% and redraws the scene.
 *=====================================================================
 *<Called routines>
 *    drawscene             - Plots the 3-D BLIRB grid points, albedo
 *                            areas, aerosol regions, and the output
 *                            flux.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void zoomCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - an index to indicate whether the zoom
 *                            direction is in (1) or out (2)
```

```
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 *********************************************************************
 */
void zoomCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *n = (int *)c;

  if(*n == 1)                              /* Zoom In                 */
    mov->magfactor = 1.05 * mov->magfactor;
  else if(*n == 2)                         /* Zoom Out                */
    mov->magfactor = mov->magfactor / 1.05;

  drawscene();
}     /*   end zoomCB()   */

/*********************************************************************
 *                          VOID MINOR_GRIDCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  minor_gridCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_viewmenu
 *=====================================================================
 *<Description>
 *    Turns on/off the minor grid lines and redraws the scene.
 *=====================================================================
 *<Called routines>
 *    drawscene             - Plots the 3-D BLIRB grid points, albedo
 *                            areas, aerosol regions, and the output
 *                            flux.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void minor_gridCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - a pointer to any input data to be given to
 *                            the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 *********************************************************************
```

```
 */
void minor_gridCB (Widget w, XtPointer c, XtPointer call_data)
{
  minor_grid = !minor_grid;                /* Minor Grid On/Off        */
  drawscene();
}     /*   end minor_gridCB()   */

/*********************************************************************
 *                          VOID TRANSCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  transCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_viewmenu
 *====================================================================
 *<Description>
 *    Turns on/off the transparent color mode for the aerosol regions
 *    and redraws the scene.
 *====================================================================
 *<Called routines>
 *    drawscene             - Plots the 3-D BLIRB grid points, albedo
 *                            areas, aerosol regions, and the output
 *                            flux.
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void transCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - a pointer to any input data to be given to
 *                            the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void transCB (Widget w, XtPointer c, XtPointer call_data)
{
  transparency = !transparency;            /* Transparency On/Off      */
  drawscene();
}     /*   end transCB()   */

/*********************************************************************
 *                          VOID AXISCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  axisCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_axismenu
 *====================================================================
 *<Description>
```

```
 *    Switches to the selected Axis coming out of the screen in the plot
 *    and redraws the scene.
 *=====================================================================
 *<Called routines>
 *    reset                    - resets the viewing and plot parameters to
 *                               the original values
 *    drawscene                - Plots the 3-D BLIRB grid points, albedo
 *                               areas, aerosol regions, and the output
 *                               flux.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void axisCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                     - the ID of the widget for which the
 *                               callback is registered
 *       c                     - an index to indicate the viewing axis
 *                               choice
 *       call_data             - a pointer to the callback structure which
 *                               contains information on why the callback
 *                               occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void axisCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *n = (int *)c;
  int i;

/*-------------------------------------------------------------------
 * --- Select the viewing axis.
 *-------------------------------------------------------------------
 */
  for (i=0; i<3; i++)
    if(*n == i+1)
      view_axis[i] = TRUE;
    else
      view_axis[i] = FALSE;

/*-------------------------------------------------------------------
 * --- If a "reset" is required, reset the viewing options.
 *-------------------------------------------------------------------
 */
  if(reset_flag)
    reset();

/*-------------------------------------------------------------------
 * --- Redraw the scene
 *-------------------------------------------------------------------
 */
  drawscene();
}     /*   end axisCB()   */

/********************************************************************
 *                          VOID SUNCB
```

```
*************************************************************************
*<Begin>
*<Identification>          Name:  sunCB
*                          Type:  C void
*                      Filename:  visual.c
*                        Parent:  create_sunmenu
*=========================================================================
*<Description>
*    Turns on/off the plotting of the Sun and redraws the scene.
*=========================================================================
*<Called routines>
*    drawscene              - Plots the 3-D BLIRB grid points, albedo
*                             areas, aerosol regions, and the output
*                             flux.
*=========================================================================
*<Parameters>
*    Formal declaration:
*       void sunCB( Widget w, XtPointer c, XtPointer call_data)
*    Input:
*       w                   - the ID of the widget for which the
*                             callback is registered
*       c                   - a pointer to any input data to be given to
*                             the routine
*       call_data           - a pointer to the callback structure which
*                             contains information on why the callback
*                             occurred
*    Output:
*       None
*=========================================================================
*<History>
*    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
*              Developed the original source code.
*=========================================================================
*<End>
*************************************************************************
*/
void sunCB (Widget w, XtPointer c, XtPointer call_data)
{
  sun_plot = !sun_plot;                    /* Plot the Sun -  On/Off   */
  drawscene();
}     /*   end sunCB()   */

/*************************************************************************
*                          VOID OBSCCB
*************************************************************************
*<Begin>
*<Identification>          Name:  obscCB
*                          Type:  C void
*                      Filename:  visual.c
*                        Parent:  create_viewmenu
*=========================================================================
*<Description>
*    Turns on/off the obscuration and albedo region labels and redraws
*    the scene.
*=========================================================================
*<Called routines>
*    obsc                   - sets up rowcolumn widget with names of
*                             materials.
*    drawscene              - Plots the 3-D BLIRB grid points, albedo
*                             areas, aerosol regions, and the output
*                             flux.
*=========================================================================
```

```
 *<Parameters>
 *    Formal declaration:
 *       void obscCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                    - the ID of the widget for which the
 *                              callback is registered
 *       c                    - a pointer to any input data to be given to
 *                              the routine
 *       call_data            - a pointer to the callback structure which
 *                              contains information on why the callback
 *                              occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 *********************************************************************
 */
void obscCB (Widget w, XtPointer c, XtPointer call_data)
{
  label_obsc = !label_obsc;                /* Aerosol Definition On/Off*/
  obsc();
  drawscene();
}     /*   end obscCB()   */

/*********************************************************************
 *                          VOID HELP_GENERALCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  help_generalCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_helpmenu
 *=====================================================================
 *<Description>
 *    Displays general VISUAL information.
 *=====================================================================
 *<Called routines>
 *    create_message        - draws a message in a dialog message box
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void help_generalCB( Widget w, XtPointer c,
 *                            XtPointer call_data)
 *    Input:
 *       w                    - the ID of the widget for which the
 *                              callback is registered
 *       c                    - a pointer to any input data to be given to
 *                              the routine
 *       call_data            - a pointer to the callback structure which
 *                              contains information on why the callback
 *                              occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
```

```
 *<End>
 ***********************************************************************
 */
void help_generalCB (Widget w, XtPointer c, XtPointer call_data)
{
  static char *help_str[] = {
      "VISUAL provides a graphical user interface (GUI) for\n",
      "selecting BLIRB8 inputs and graphically display the BLIRB8\n",
      "output radiative flux fields in an interactive mode.\n",
      "   ",
      "It graphically displays the entire physical space including:\n",
      "   * surface albedo areas (in shades of green),\n",
      "   * aerosol regions (various transparent colors),\n",
      "   * flare positions (in red),\n",
      "   * searchlight position (in white),\n",
      "   * relative position of Sun (in yellow), and\n",
      "   * BLIRB8 output radiative flux fields (red and white).\n",
      "   ",
      "Optionally, text widgets may be displayed depicting:\n",
      "   * the current BLIRB8 input/output filename,\n",
      "   * the surface albedo areas information,\n",
      "   * the aerosols regions information, and\n",
      "   * the output radiative flux field information.\n",
      "   ",
      "All input parameters (except a filename for saving the\n",
      "inputs) can be selected using only the mouse.  The keyboard\n",
      "may be used for hot-keys and must be used for specifying a\n",
      "Savefile Name.\n",
      "   ",
      "VISUAL uses the Silicon Graphics Inc. (SGI) IRIS Graphics\n",
      "Library for the display graphics and X-Windows/Motif for the\n",
      "menus and message boxes of the GUI. This software requires\n",
      "an SGI workstation with at least 128 Mbytes of RAM for\n",
      "program execution.\n",
      "   ",
      "NOTE:  The <Esc> key is the ABORT PROGRAM key.\n",
      "" };

  create_message (w, help_str, XmDIALOG_INFORMATION);
}     /*   end help_generalCB()   */

/***********************************************************************
 *                          VOID HELP_FILECB
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  help_fileCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_helpmenu
 *=====================================================================
 *<Description>
 *    Displays File Options information.
 *=====================================================================
 *<Called routines>
 *    create_message        - draws a message in a dialog message box
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void help_fileCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
```

```
 *       c                    - a pointer to any input data to be given to
 *                              the routine
 *       call_data            - a pointer to the callback structure which
 *                              contains information on why the callback
 *                              occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void help_fileCB (Widget w, XtPointer c, XtPointer call_data)
{
  static char *help_str[] = {
      "Create New File:\n",
      "This option checks for unsaved modifications to the initial\n",
      "input parameter values.  If false, default input values are\n",
      "loaded.  If true, a message will appear warning that\n",
      "continuing will delete all previous changes.  A choice of\n",
      "whether to continue or cancel is given.  If the user\n",
      "continues, the default BLIRB8 input values are loaded.\n",
      "    ",
      "Open File:\n",
      "This option checks for unsaved modifications to the initial\n",
      "input parameter values.  If true, a message will appear\n",
      "warning that continuing will delete all previous changes.  A\n",
      "choice of whether to continue or cancel is given.  This\n",
      "option creates a File Selection Box with the file names in\n",
      "the current directory filtered according to the \"Filename\n",
      "Filter\".  Use the scrollbar on the right side of the File\n",
      "Selection Box to scroll the list of file names.  Select a\n",
      "file to process and the inputs from the file are loaded.\n",
      "    ",
      "Save File:\n",
      "This option causes the current inputs to be saved to the\n",
      "currently selected file name, after renaming the original\n",
      "input file to a backup file with \".bak\" appended to the\n",
      "file name.\n",
      "    ",
      "Save File As:\n",
      "This option prompts for a file name for saving the current\n",
      "BLIRB8 input parameters.\n",
      "    ",
      "Exit Program:\n",
      "This option causes the VISUAL program to terminate.  If\n",
      "any BLIRB8 input parameters were modified and not saved a\n",
      "warning message will appear allowing the user to cancel\n",
      "or continue and loose the modifications.\n",
      "" };

  create_message (w, help_str, XmDIALOG_INFORMATION);
}     /*   end help_fileCB()    */

/********************************************************************
 *                          VOID HELP_RESETCB
 ********************************************************************
 *<Begin>
 *<Identification>          Name:  help_resetCB
```

```
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_helpmenu
 *=====================================================================
 *<Description>
 *    Displays Reset information.
 *=====================================================================
 *<Called routines>
 *    create_message        - draws a message in a dialog message box
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void help_resetCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - a pointer to any input data to be given to
 *                            the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void help_resetCB (Widget w, XtPointer c, XtPointer call_data)
{
  static char *help_str[] = {
      "Reset:\n",
      "   ",
      "Selecting this option causes all current rotation and/or\n",
      "translation of the BLIRB8 space display to be removed.  The\n",
      "display is reset to a view along an axis looking at the\n",
      "center of the BLIRB8 space.\n",
      "" };

  create_message (w, help_str, XmDIALOG_INFORMATION);
}     /*   end help_resetCB()   */

/********************************************************************
 *                          VOID HELP_ROTATIONCB
 ********************************************************************
 *<Begin>
 *<Identification>          Name:  help_rotationCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_helpmenu
 *=====================================================================
 *<Description>
 *    Displays BLIRB8 Space Rotation information.
 *=====================================================================
 *<Called routines>
 *    create_message        - draws a message in a dialog message box
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
```

```
 *       void help_rotationCB( Widget w, XtPointer c,
 *                             XtPointer call_data)
 *    Input:
 *       w                   - the ID of the widget for which the
 *                             callback is registered
 *       c                   - a pointer to any input data to be given to
 *                             the routine
 *       call_data           - a pointer to the callback structure which
 *                             contains information on why the callback
 *                             occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void help_rotationCB (Widget w, XtPointer c, XtPointer call_data)
{
  static char *help_str[] = {
      "BLIRB8 Space Rotation:\n",
      "   ",
      "All rotations of the BLIRB8 space are performed holding the\n",
      "left mouse button down while moving the mouse.  The display\n",
      "is an \"ortho\" projection such that everything in the BLIRB8\n",
      "space is projected using reasonably parallel projection lines\n",
      "onto an observer position plane.  This type of projection\n",
      "helps to conserve length and angular relationships between\n",
      "the objects in the display (near objects do not appear much\n",
      "larger than distant objects).  Unfortunately, with this type\n",
      "of projection the observer cannot select arbitrary positions\n",
      "around the BLIRB8 space from which to view the objects.  The\n",
      "projections are onto a plane not a point.\n",
      "   ",
      "There are three observer position planes to choose from --\n",
      "a YZ plane on the positive-X axis, an XZ plane on the\n",
      "negative-Y axis, and an XY plane on the positive-Z axis.  The\n",
      "observer position is moved on one of these planes as the left\n",
      "mouse button is held down while the mouse is moved.  The\n",
      "point \"looked at\" from all planes using all observer\n",
      "positions is the center of the BLIRB8 space.  If a severe\n",
      "rotation is performed on any observer plane, the BLIRB8 space\n",
      "may disappear from the screen.  If this happens, undo some of\n",
      "the rotation and select another observer plane from which to\n",
      "view the BLIRB8 space.  There is no need for severe\n",
      "rotations.\n",
      "" };

  create_message (w, help_str, XmDIALOG_INFORMATION);
}     /*   end help_rotationCB()    */

/***********************************************************************
 *                          VOID HELP_TRANSLATIONCB
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  help_translationCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_helpmenu
```

```
 *=====================================================================
 *<Description>
 *    Displays BLIRB8 Space Translation information.
 *=====================================================================
 *<Called routines>
 *    create_message        - draws a message in a dialog message box
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void help_translationCB( Widget w, XtPointer c,
 *                                XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - a pointer to any input data to be given to
 *                            the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void help_translationCB (Widget w, XtPointer c, XtPointer call_data)
{
  static char *help_str[] = {
      "BLIRB8 Space Translation:\n",
      "   ",
      "All translations of the BLIRB8 space are performed holding\n",
      "the right mouse button down while moving the mouse.  This\n",
      "causes the point looked at in the BLIRB8 space to change\n",
      "from the center of the BLIRB8 space.  Translations should be\n",
      "performed with caution as they can cause unexpected viewing\n",
      "problems when followed by rotations.\n",
      "" };

  create_message (w, help_str, XmDIALOG_INFORMATION);
}     /*   end help_translationCB()   */

/********************************************************************
 *                          VOID HELP_VIEWCB
 ********************************************************************
 *<Begin>
 *<Identification>          Name:  help_viewCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_helpmenu
 *=====================================================================
 *<Description>
 *    Creates the Viewing Options information menu
 *=====================================================================
 *<Called routines>
 *    create_rowcol         - creates a Rowcol Widget
 *    cancelCB              - removes the Rowcol Widget
 *    create_separator      - creates a Separator Widget
 *    create_radiobox       - creates a Radiobox Widget
```

```
 *    create_togglebutton    - creates a Togglebutton Widget
 *    help_axisCB            - displays the Viewing Axis Options info
 *    help_sunoCB            - displays the Sun Options information
 *    help_zoomCB            - displays the Zoom Options information
 *    help_togCB             - displays the Toggle Switch Options info
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void help_viewCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                   - the ID of the widget for which the
 *                             callback is registered
 *       c                   - the data passed to the routine
 *       call_data           - a pointer to the callback structure which
 *                             contains information on why the callback
 *                             occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570  Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void help_viewCB (Widget w, XtPointer c, XtPointer call_data)
{
  Widget radio, toggle[4], rowcol;
  static int index, hv, sd, nc;
  static char cat_label[13] = "View Options";
  static char tog_label[4][22] = { "Viewing Axis Options",
                                   "Sun Options",
                                   "Zoom Options",
                                   "Toggle Switch Options" };

/*-------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Help_View", cancelCB);

/*-------------------------------------------------------------------
 * --- Create the radioboxes and toggles
 *-------------------------------------------------------------------
 */
  hv = 0;
  sd = 2;
  nc = 1;
  index = 0;
  create_separator(rowcol, &hv, &sd);
  radio = create_radiobox(rowcol, &nc, cat_label);

  toggle[0] = create_togglebutton(radio, tog_label[0], &index,
              help_axisCB);

  toggle[1] = create_togglebutton(radio, tog_label[1], &index,
              help_sunoCB);

  toggle[2] = create_togglebutton(radio, tog_label[2], &index,
              help_zoomCB);
```

```
  toggle[3] = create_togglebutton(radio, tog_label[3], &index,
              help_togCB);
}     /*   end help_viewCB()   */

/*********************************************************************
 *                          VOID HELP_AXISCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  help_axisCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  help_viewCB
 *====================================================================
 *<Description>
 *    Displays Viewing Axis Options information.
 *====================================================================
 *<Called routines>
 *    create_message        - draws a message in a dialog message box
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void help_axisCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - a pointer to any input data to be given to
 *                            the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void help_axisCB (Widget w, XtPointer c, XtPointer call_data)
{
  static char *help_str[] = {
      "Viewing Axis Options:\n",
      "There are three options --\n",
      "   ",
      "Positive X-Axis:\n",
      "This option causes the observer position to be in a YZ plane\n",
      "on the positive X axis.\n",
      "   ",
      "Negative Y-Axis:\n",
      "This option causes the observer position to be in an XZ plane\n",
      "on the negative Y axis.  This is the initial viewing axis\n",
      "choice when starting VISUAL.\n",
      "   ",
      "Positive Z-Axis:\n",
      "This option causes the observer position to be in an XY plane\n",
      "on the positive Z axis.\n",
      "   ",
      "The point \"looked at\" is the center of the BLIRB8 space.\n",
      "The observer position on the observer plane can be changed\n",
      "by holding the left mouse button down while moving the\n",
```

```
      "mouse.  This creates a rotation effect on the BLIRB8 space.\n",
      "" };

  create_message (w, help_str, XmDIALOG_INFORMATION);
}     /*   end help_axisCB()   */

/*********************************************************************
 *                          VOID HELP_SUNOCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  help_sunoCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  help_viewCB
 *====================================================================
 *<Description>
 *    Displays Sun Options information.
 *====================================================================
 *<Called routines>
 *    create_message        - draws a message in a dialog message box
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void help_sunoCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - a pointer to any input data to be given to
 *                            the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void help_sunoCB (Widget w, XtPointer c, XtPointer call_data)
{
  static char *help_str[] = {
      "Sun Options:\n",
      "There are two options --\n",
      "   ",
      "Sun Plot On/Off:\n",
      "This option is a toggle switch.  It causes the relative Sun\n",
      "position to be displayed if it was not previously displayed.\n",
      "Similarly, it causes the relative Sun position not to be\n",
      "displayed if it was previously displayed.  Initially,\n",
      "\"Sun Plot\" is \"on\" when VISUAL is started.\n",
      "   ",
      "Select New Sun Position:\n",
      "This option causes the viewing axis to temporarily become\n",
      "the positive Z axis.  Holding the left mouse button down and\n",
      "moving the mouse causes the Sun position to change.  Move\n",
      "the mouse until the Sun is positioned at the desired\n",
      "location then release the mouse button.  Upon releasing the\n",
      "mouse button the display returns to the original viewing\n",
```

```
      "position and all objects are as they were except the\n",
      "position of the Sun has been changed.\n",
      "" };

  create_message (w, help_str, XmDIALOG_INFORMATION);
}     /*   end help_sunoCB()   */

/*********************************************************************
 *                          VOID HELP_ZOOMCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  help_zoomCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  help_viewCB
 *====================================================================
 *<Description>
 *    Displays Zoom Options information.
 *====================================================================
 *<Called routines>
 *    create_message        - draws a message in a dialog message box
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void help_zoomCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - a pointer to any input data to be given to
 *                            the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void help_zoomCB (Widget w, XtPointer c, XtPointer call_data)
{
  static char *help_str[] = {
      "Zoom Options:\n",
      "There are two options --\n",
      "   ",
      "Zoom In:\n",
      "This option causes the BLIRB8 space to be displayed 5%\n",
      "larger than it previously was displayed.\n",
      "   ",
      "Zoom Out:\n",
      "This option causes the BLIRB8 space to be displayed 5%\n",
      "smaller than it previously was displayed.\n",
      "" };

  create_message (w, help_str, XmDIALOG_INFORMATION);
}     /*   end help_zoomCB()   */

/*********************************************************************
```

```
*                           VOID HELP_TOGCB
*********************************************************************
*<Begin>
*<Identification>          Name:  help_togCB
*                          Type:  C void
*                      Filename:  visual.c
*                        Parent:  help_viewCB
*====================================================================
*<Description>
*    Displays View Toggle Switch information.
*====================================================================
*<Called routines>
*    create_message        - draws a message in a dialog message box
*====================================================================
*<Parameters>
*    Formal declaration:
*       void help_togCB( Widget w, XtPointer c, XtPointer call_data)
*    Input:
*       w                  - the ID of the widget for which the
*                            callback is registered
*       c                  - a pointer to any input data to be given to
*                            the routine
*       call_data          - a pointer to the callback structure which
*                            contains information on why the callback
*                            occurred
*    Output:
*       None
*====================================================================
*<History>
*    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
*              Developed the original source code.
*====================================================================
*<End>
*********************************************************************
*/
void help_togCB (Widget w, XtPointer c, XtPointer call_data)
{
  static char *help_str[] = {
      "Toggle Switch Options:\n",
      "There are three options --\n",
      "   ",
      "Minor Grid Lines On/Off:\n",
      "This option is a toggle switch.  It causes the minor grid\n",
      "lines to be displayed if they were not previously displayed.\n",
      "Similarly, it causes the minor grid lines not to be\n",
      "displayed if they were previously displayed.  Initially,\n",
      "\"Minor Grid Lines\" is \"off\" when VISUAL is started.\n",
      "   ",
      "Transparent Colors On/Off:\n",
      "This option is a toggle switch.  It causes the aerosol\n",
      "regions to be displayed in transparent colors if they were\n",
      "previously displayed as outlines.  Similarly, it causes the\n",
      "aerosol regions to be displayed as outlines if they were\n",
      "previously displayed in transparent colors.  Initially,\n",
      "\"Transparent Colors\" is \"on\" when VISUAL is started.\n",
      "   ",
      "Region Definitions On/Off:\n",
      "This option is a toggle switch.  It causes the text boxes to\n",
      "appear at the bottom of the BLIRB8 window containing the\n",
      "current filename, aerosol regions information, albedo areas\n",
      "information, and radiative flux information if they were not\n",
      "previously displayed.  Similarly, it causes the text boxes\n",
```

```
      "to disappear if they were previously displayed.  Initially,\n",
      "\"Region Definitions\" is \"on\" when VISUAL is started.\n",
      "" };

  create_message (w, help_str, XmDIALOG_INFORMATION);
}     /*   end help_togCB()   */

/*********************************************************************
 *                          VOID HELP_FLUXCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  help_fluxCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_helpmenu
 *====================================================================
 *<Description>
 *    Creates the Flux Display Options information menu
 *====================================================================
 *<Called routines>
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelCB               - removes the Rowcol Widget
 *    create_separator       - creates a Separator Widget
 *    create_radiobox        - creates a Radiobox Widget
 *    create_togglebutton    - creates a Togglebutton Widget
 *    help_optCB             - displays the Flux Options info
 *    help_orCB              - displays the Cross-section Orientation
 *                             Options information
 *    help_valCB             - displays the Cross-section Plane Value
 *                             Options information
 *    help_waveCB            - displays the Wave Number Options info
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void help_fluxCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                   - the ID of the widget for which the
 *                             callback is registered
 *       c                   - the data passed to the routine
 *       call_data           - a pointer to the callback structure which
 *                             contains information on why the callback
 *                             occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void help_fluxCB (Widget w, XtPointer c, XtPointer call_data)
{
  Widget radio, toggle[4], rowcol;
  static int index, hv, sd, nc;
  static char cat_label[21] = "Flux Display Options";
  static char tog_label[4][36] = { "Flux Options",
                                   "Cross-section Plane Orientation",
                                  "Cross-section Plane Value Selection",
                                   "Wave Number Selection" };
```

```
/*-----------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-----------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Help_Flux", cancelCB);

/*-----------------------------------------------------------------------
 * --- Create the radioboxes and toggles
 *-----------------------------------------------------------------------
 */
  hv = 0;
  sd = 2;
  nc = 1;
  index = 0;
  create_separator(rowcol, &hv, &sd);
  radio = create_radiobox(rowcol, &nc, cat_label);

  toggle[0] = create_togglebutton(radio, tog_label[0], &index,
              help_optCB);

  toggle[1] = create_togglebutton(radio, tog_label[1], &index,
              help_orCB);

  toggle[2] = create_togglebutton(radio, tog_label[2], &index,
              help_valCB);

  toggle[3] = create_togglebutton(radio, tog_label[3], &index,
              help_waveCB);
}     /*   end help_fluxCB()   */

/***********************************************************************
 *                          VOID HELP_OPTCB
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  help_optCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  help_fluxCB
 *=====================================================================
 *<Description>
 *    Displays Flux Options information.
 *=====================================================================
 *<Called routines>
 *    create_message        - draws a message in a dialog message box
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void help_optCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - a pointer to any input data to be given to
 *                            the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
```

```
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void help_optCB (Widget w, XtPointer c, XtPointer call_data)
{
  static char *help_str[] = {
      "Flux Options:\n",
      "There are thirteen options --\n",
      "  Solar Direct Flux\n",
      "  Solar Reflected Flux\n",
      "  Diffuse Flux - 1\n",
      "  Diffuse Flux - 2\n",
      "  Diffuse Flux - 3\n",
      "  Diffuse Flux - 4\n",
      "  Diffuse Flux - 5\n",
      "  Diffuse Flux - 6\n",
      "  Diffuse Flux - 7\n",
      "  Diffuse Flux - 8\n",
      "  No Flux\n",
      "  Dec by 1 Button\n",
      "  Inc by 1 Button\n",
      "   ",
      "Any of the first 10 options will display the corresponding\n",
      "flux magnitude within the BLIRB8 space.  The flux display\n",
      "is in the form of a red 3-dimensional grid wiremesh plot\n",
      "depicting the magnitude of the flux corresponding to the\n",
      "choices of flux cross-section plane, the distance of the\n",
      "cross-section plane from the BLIRB8 space origin, and the\n",
      "radiation wavenumber.  The current flux cross-section plane\n",
      "is depicted by a white grid mesh.  The distance between the\n",
      "red and white grid meshes is related to the magnitude of the\n",
      "flux.  The flux amplitude scale may be linear or logarithmic.\n",
      "   ",
      "No Flux:\n",
      "This option causes both the flux display and the flux text\n",
      "information box to be turned off.  It is the option used\n",
      "when VISUAL is started.\n",
      "   ",
      "Dec/Inc by 1 Button:\n",
      "This option causes the choice of flux fields to decrease/\n",
      "increase by one position in the ordered sequence of fluxes.\n",
      "" };

  create_message (w, help_str, XmDIALOG_INFORMATION);
}     /*   end help_optCB()   */

/***********************************************************************
 *                          VOID HELP_ORCB
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  help_orCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  help_fluxCB
 *=====================================================================
 *<Description>
 *    Displays Cross-section Plane Orientation Options information.
 *=====================================================================
 *<Called routines>
 *    create_message        - draws a message in a dialog message box
 *=====================================================================
```

```
 *<Parameters>
 *    Formal declaration:
 *       void help_orCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                    - the ID of the widget for which the
 *                              callback is registered
 *       c                    - a pointer to any input data to be given to
 *                              the routine
 *       call_data            - a pointer to the callback structure which
 *                              contains information on why the callback
 *                              occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 *********************************************************************
 */
void help_orCB (Widget w, XtPointer c, XtPointer call_data)
{
  static char *help_str[] = {
      "Cross-Section Plane Orientation:\n",
      "There are three options --\n",
      "   ",
      "X-Plane Cross-Section:\n",
      "This option causes the radiative flux field cross-section\n",
      "plane to be a YZ plane on the positive X axis.\n",
      "   ",
      "Y-Plane Cross-Section:\n",
      "This option causes the radiative flux field cross-section\n",
      "plane to be an XZ plane on the positive Y axis.\n",
      "   ",
      "Z-Plane Cross-Section:\n",
      "This option causes the radiative flux field cross-section\n",
      "plane to be an XY plane on the positive Z axis.  The Z-Plane\n",
      "Cross-Section is the default choice when starting VISUAL.\n",
      "   ",
      "The position of the plane on the axis is dependent upon the\n",
      "choice of \"Cross-section Plane Value\".\n",
      "" };

  create_message (w, help_str, XmDIALOG_INFORMATION);
}     /*   end help_orCB()   */

/*********************************************************************
 *                          VOID HELP_VALCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  help_valCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  help_fluxCB
 *=====================================================================
 *<Description>
 *    Displays Cross-section Plane Value Options information.
 *=====================================================================
 *<Called routines>
 *    create_message        - draws a message in a dialog message box
 *=====================================================================
```

```
 *<Parameters>
 *    Formal declaration:
 *       void help_valCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                    - the ID of the widget for which the
 *                              callback is registered
 *       c                    - a pointer to any input data to be given to
 *                              the routine
 *       call_data            - a pointer to the callback structure which
 *                              contains information on why the callback
 *                              occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 *********************************************************************
 */
void help_valCB (Widget w, XtPointer c, XtPointer call_data)
{
  static char *help_str[] = {
      "Cross-Section Plane Value Selection:\n",
      "There are an arbitrary number of options --\n",
      "   ",
      "All options but the last two are of the form \"xx.xxx Km\"\n",
      "where the xx.xxx is the value of one of the minor grid line\n",
      "values corresponding to the flux cross-section plane\n",
      "orientation choice.  Initially, the option values are 0.0 Km\n",
      "for all three cross-section plane orientations when VISUAL\n",
      "is started.\n",
      "   ",
      "Dec/Inc by 1 Button:\n",
      "This option has the effect of decreasing/increasing the flux\n",
      "cross-section value to the next lower/higher minor grid line\n",
      "value.\n",
      "" };

  create_message (w, help_str, XmDIALOG_INFORMATION);
}     /*   end help_valCB()   */

/*********************************************************************
 *                          VOID HELP_WAVECB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  help_waveCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  help_fluxCB
 *=====================================================================
 *<Description>
 *    Displays Wave Number Selection information.
 *=====================================================================
 *<Called routines>
 *    create_message        - draws a message in a dialog message box
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void help_waveCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
```

```
 *       w                    - the ID of the widget for which the
 *                              callback is registered
 *       c                    - a pointer to any input data to be given to
 *                              the routine
 *       call_data            - a pointer to the callback structure which
 *                              contains information on why the callback
 *                              occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void help_waveCB (Widget w, XtPointer c, XtPointer call_data)
{
  static char *help_str[] = {
      "Wave Number Selection:\n",
      "There are an arbitrary number of options --\n",
      "   ",
      "All options but the last two are of the form\n",
      "\"xxxx.xxxx per cm\" where the xxxx.xxxx is the value of one\n",
      "of the input wavenumbers.  Initially, the choice is the\n",
      "minimum value available when VISUAL is started.\n",
      "   ",
      "Dec/Inc by 1 Button:\n",
      "This option has the effect of decreasing/increasing the\n",
      "wavenumber value to the next lower/higher available value.\n",
      "" };

  create_message (w, help_str, XmDIALOG_INFORMATION);
}     /*   end help_waveCB()   */

/********************************************************************
 *                          VOID HELP_MODIFYCB
 ********************************************************************
 *<Begin>
 *<Identification>          Name:  help_modifyCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_helpmenu
 *=====================================================================
 *<Description>
 *    Creates the Input Modifications Options information menu
 *=====================================================================
 *<Called routines>
 *    create_rowcol         - creates a Rowcol Widget
 *    cancelCB              - removes the Rowcol Widget
 *    create_separator      - creates a Separator Widget
 *    create_radiobox       - creates a Radiobox Widget
 *    create_togglebutton   - creates a Togglebutton Widget
 *    help_modCB            - displays the Model Changes info
 *    help_regCB            - displays the Region Selection info
 *    help_areaCB           - displays the Albedo Area Selection info
 *    help_meshCB           - displays the Grid Mesh Selection info
 *    help_cldCB            - displays the Cloud Changes info
 *    help_sunCB            - displays the Sun Changes information
 *    help_flareCB          - displays the Flare Selection information
 *    help_sliteCB          - displays the Searchlight Selection info
```

```
 *    help_spectCB          - displays the Spectral Range Changes info
 *    help_compCB           - displays the Computation Changes info
 *    help_outCB            - displays the Output File Changes info
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void help_modifyCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - the data passed to the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void help_modifyCB (Widget w, XtPointer c, XtPointer call_data)
{
  Widget radio, toggle[11], rowcol;
  static int index, hv, sd, nc;
  static char cat_label[15] = "Modify Options";
  static char tog_label[11][23] = { "Model Changes",
                                    "Region Selection",
                                    "Albedo Area Selection",
                                    "Grid Mesh Selection",
                                    "Cloud Changes",
                                    "Sun Changes",
                                    "Flare Selection",
                                    "Searchlight Selection",
                                    "Spectral Range Changes",
                                    "Computation Changes",
                                    "Output File Changes" };

/*-------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Help_Modify", cancelCB);

/*-------------------------------------------------------------------
 * --- Create the radioboxes and toggles
 *-------------------------------------------------------------------
 */
  hv = 0;
  sd = 2;
  nc = 1;
  index = 0;
  create_separator(rowcol, &hv, &sd);
  radio = create_radiobox(rowcol, &nc, cat_label);

  toggle[0] = create_togglebutton(radio, tog_label[0], &index,
              help_modCB);

  toggle[1] = create_togglebutton(radio, tog_label[1], &index,
```

```
                help_regCB);

    toggle[2] = create_togglebutton(radio, tog_label[2], &index,
                help_areaCB);

    toggle[3] = create_togglebutton(radio, tog_label[3], &index,
                help_meshCB);

    toggle[4] = create_togglebutton(radio, tog_label[4], &index,
                help_cldCB);

    toggle[5] = create_togglebutton(radio, tog_label[5], &index,
                help_sunCB);

    toggle[6] = create_togglebutton(radio, tog_label[6], &index,
                help_flareCB);

    toggle[7] = create_togglebutton(radio, tog_label[7], &index,
                help_sliteCB);

    toggle[8] = create_togglebutton(radio, tog_label[8], &index,
                help_spectCB);

    toggle[9] = create_togglebutton(radio, tog_label[9], &index,
                help_compCB);

   toggle[10] = create_togglebutton(radio, tog_label[10], &index,
                help_outCB);
}     /*   end help_modifyCB()    */

/*************************************************************************
 *                          VOID HELP_MODCB
 *************************************************************************
 *<Begin>
 *<Identification>          Name:  help_modCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  help_modifyCB
 *=========================================================================
 *<Description>
 *    Displays Model Changes information.
 *=========================================================================
 *<Called routines>
 *    create_message        - draws a message in a dialog message box
 *=========================================================================
 *<Parameters>
 *    Formal declaration:
 *       void help_modCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - a pointer to any input data to be given to
 *                            the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *=========================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
```

```
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void help_modCB (Widget w, XtPointer c, XtPointer call_data)
{
  static char *help_str[] = {
      "Aerosol Selections:\n",
      "This option creates a popup menu with 12 aerosol options.\n",
      "Some of these create a secondary popup menu with aerosol\n",
      "refinements.  The default is \"No aerosols\".\n",
      "   ",
      "Temperature Profile Model:\n",
      "The first 6 options cause the corresponding temperature\n",
      "profile from the Standard Atmosphere to be used for the\n",
      "BLIRB8 temperature profile.  The 7th creates a popup menu\n",
      "with 6 temperature scales, one for each of the lowest 5 km\n",
      "of altitude.  The default option is the 1976 U.S. Standard.\n",
      "   ",
      "Meteorological Range:\n",
      "If the met range is less than 5 km, select the first option.\n",
      "Otherwise, use the second option.  Either creates a popup\n",
      "menu with a met range scale.  The default is 40 km.\n",
      "   ",
      "Tropospheric Profile:\n",
      "This option defines the aerosol profile above 2 km.  The\n",
      "default \"Set by Meteorological Range\".\n",
      "   ",
      "Albedo:\n",
      "Changing the option creates a series of nested popup menus\n",
      "which allow albedo value assignments to all albedo areas.\n",
      "The default option is \"Wave Independent, User-defined\" and\n",
      "the default albedo value is the background albedo (0.2).\n",
      "   ",
      "Aerosol Profile Printout:\n",
      "The default option is \"None\".\n",
      "   ",
      "Surface Temperature (K):\n",
      "The default surface temperature is 288.2 K.\n",
      "" };

  create_message (w, help_str, XmDIALOG_INFORMATION);
}     /*   end help_modCB()    */

/***********************************************************************
 *                          VOID HELP_REGCB
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  help_regCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  help_modifyCB
 *=====================================================================
 *<Description>
 *    Displays Region Selection information.
 *=====================================================================
 *<Called routines>
 *    create_message        - draws a message in a dialog message box
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void help_regCB( Widget w, XtPointer c, XtPointer call_data)
```

```
 *    Input:
 *       w                    - the ID of the widget for which the
 *                              callback is registered
 *       c                    - a pointer to any input data to be given to
 *                              the routine
 *       call_data            - a pointer to the callback structure which
 *                              contains information on why the callback
 *                              occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void help_regCB (Widget w, XtPointer c, XtPointer call_data)
{
  static char *help_str[] = {
      "Region Selection:\n",
      "This option creates a secondary pulldown menu listing all\n",
      "aerosol regions and possibly a \"Add New Region\" option.\n",
      "Any option causes another pulldown menu to appear with from\n",
      "1 - 6 new options.  These new options are --\n",
      "   ",
      "Dimensions:\n",
      "This option causes one or more popup menus to appear.  The\n",
      "first has 3 scales for setting the 3 dimensions of a region.\n",
      "Default dimensions of Region - 1 are (5, 4, 5).  If the\n",
      "dimensions of Region - 1 change, the corresponding grid mesh\n",
      "also changes.  This effect will change the positions and\n",
      "dimensions of the other aerosol regions and albedo areas. \n",
      "   ",
      "Material - 1 / Material - 2 / Material 3:\n",
      "This option creates a popup menu with 16 material options.\n",
      "Some of these can create a secondary menu with material\n",
      "refinements.  Aerosol regions are displayed as 3-D boxes\n",
      "with transparent color.  The color and shade correspond to\n",
      "the first material specified for a region.  A popup scale is\n",
      "created to input the visibility within the material.  The\n",
      "default is \"No cloud\" and infinite visibility.\n",
      "   ",
      "Location:\n",
      "This option causes the display to view downward from the\n",
      "positive Z axis and project the region onto the XY plane\n",
      "in transparent red with a red prompt to Move the Region.\n",
      "Change the region position in the XY plane with the mouse.\n",
      "The display will then view forward from the negative Y axis.\n",
      "Follow the same procedure to move the region in the XZ plane.\n",
      "The default position of a new region is at the origin.\n",
      "   ",
      "Delete Region:\n",
      "This option causes the region of interest and its associated\n",
      "materials to be removed from all inputs and the display.\n",
      "" };

  create_message (w, help_str, XmDIALOG_INFORMATION);
}     /*   end help_regCB()   */

/*********************************************************************
```

```
*                           VOID HELP_AREACB
*********************************************************************
*<Begin>
*<Identification>          Name:  help_areaCB
*                          Type:  C void
*                      Filename:  visual.c
*                        Parent:  help_modifyCB
*====================================================================
*<Description>
*    Displays Albedo Area Selection information.
*====================================================================
*<Called routines>
*    create_message        - draws a message in a dialog message box
*====================================================================
*<Parameters>
*    Formal declaration:
*       void help_areaCB( Widget w, XtPointer c, XtPointer call_data)
*    Input:
*       w                  - the ID of the widget for which the
*                            callback is registered
*       c                  - a pointer to any input data to be given to
*                            the routine
*       call_data          - a pointer to the callback structure which
*                            contains information on why the callback
*                            occurred
*    Output:
*       None
*====================================================================
*<History>
*    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
*              Developed the original source code.
*====================================================================
*<End>
*********************************************************************
*/
void help_areaCB (Widget w, XtPointer c, XtPointer call_data)
{
  static char *help_str[] = {
      "Albedo Area Selection:\n",
      "This option creates a secondary pulldown menu listing all\n",
      "albedo areas and possibly a \"Add New Albedo Area\" option.\n",
      "Any option causes another pulldown menu to appear with from\n",
      "1 - 4 new options.  These new options are --\n",
      "   ",
      "Dimensions:\n",
      "This option causes one or more popup menus to appear.  The\n",
      "first has 2 scales for setting the 2 dimensions of an area.\n",
      "Default dimensions of Albedo Area - 1 are (5, 4).  If the\n",
      "dimensions of Region - 1 are changed, the dimensions of\n",
      "Albedo Area - 1 are also changed to cover the entire XY\n",
      "projection of Region - 1 onto the Z = 0 plane.\n",
      "   ",
      "Albedo:\n",
      "This option creates a popup menu with several albedo options\n",
      "depending upon the type of albedo selected under \"Model\n",
      "Changes\".  Some of these options can create secondary menus\n",
      "with albedo refinements.  Albedo areas are displayed as\n",
      "shaded green rectangles.  The darker the green, the higher\n",
      "the albedo.  The default is the background albedo (0.2).\n",
      "   ",
      "Location:\n",
      "This option causes the display to view downward from the\n",
```

```
        "positive Z axis and present the area in red with a red\n",
        "prompt to Move the Area.  Change the area position in the XY\n",
        "plane with the mouse.  The default position of a new area is\n",
        "at the origin.\n",
        "   ",
        "Delete Area:\n",
        "This option causes the albedo area and its associated albedo\n",
        "to be removed from all inputs and the display.\n",
        "" };

  create_message (w, help_str, XmDIALOG_INFORMATION);
}     /*   end help_areaCB()   */

/*********************************************************************
 *                          VOID HELP_MESHCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  help_meshCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  help_modifyCB
 *====================================================================
 *<Description>
 *    Displays Grid Mesh Selection information.
 *====================================================================
 *<Called routines>
 *    create_message        - draws a message in a dialog message box
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void help_meshCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - a pointer to any input data to be given to
 *                            the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void help_meshCB (Widget w, XtPointer c, XtPointer call_data)
{
  static char *help_str[] = {
        "Grid Mesh Selection:\n",
        "There are 3 options --\n",
        "   ",
        "X Grid Mesh / Y Grid Mesh / Z Grid Mesh:\n",
        "This option creates a popup menu with two columns of two or\n",
        "more scales each in a one-to-one correspondence.  All the\n",
        "scales in the left column except possibly the last set the\n",
        "end point of a mesh interval.  The start point of the first\n",
        "mesh interval is 0 km, the origin of the BLIRB8 space.  The\n",
        "start point of each mesh interval after the first is the end\n",
```

```
        "point of the previous mesh interval.  To modify any of the\n",
        "current mesh intervals merely adjust the end points of the\n",
        "intervals of interest.  Remember to adjust the number of\n",
        "subintervals within the adjusted intervals or the regions\n",
        "and albedo areas within these intervals will be modified.\n",
        "   ",
        "To add a new mesh interval is the same as subdividing a\n",
        "current mesh interval and making the adjustments necessary\n",
        "to fit the model requirements.  One could add the new mesh\n",
        "interval by adjusting all the current interval end points\n",
        "such that the last scale in the left column depicted the\n",
        "greatest end point value.  This means one must also change\n",
        "all the corresponding Subinterval scales in the right column.\n",
        "Alternately, one could consider the last scale in the left\n",
        "column as out-of-sequence relative to the other interval end\n",
        "point scales and proceed to add the desired interval end\n",
        "point and the corresponding number of subintervals.  The\n",
        "software will insert the scale in the proper sequence after\n",
        "the menu is terminated.  Remember to examine all previous\n",
        "intervals to determine the start point of the new interval.\n",
        "   ",
        "The default values for the (X, Y, Z) grid mesh intervals are\n",
        "one end point each of (5, 4, 5) which gives (10, 8, 10)\n",
        "subintervals.\n",
        "" };

  create_message (w, help_str, XmDIALOG_INFORMATION);
}     /*   end help_meshCB()   */

/*********************************************************************
 *                          VOID HELP_CLDCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  help_cldCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  help_modifyCB
 *====================================================================
 *<Description>
 *    Displays Cloud Changes information.
 *====================================================================
 *<Called routines>
 *    create_message        - draws a message in a dialog message box
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void help_cldCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - a pointer to any input data to be given to
 *                            the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
```

```
 *<End>
 ***********************************************************************
 */
void help_cldCB (Widget w, XtPointer c, XtPointer call_data)
{
  static char *help_str[] = {
      "Cloud Changes:\n",
      "There are 3 options --\n",
      "   ",
      "Cloud Structure:\n",
      "   No Cloud -- no cloud in region.\n",
      "   Rectangular Structure -- aerosols uniformly distributed\n",
      "                            within the rectangular region.\n",
      "   CSS Model -- TASC Cloud Scene Simulation prototype model.\n",
      "The default option is \"No Cloud\".\n",
      "   ",
      "Aerosol Outside Physical Region:\n",
      "   Background Aerosol -- a uniform aerosol distribution\n",
      "                         outside the primary BLIRB8 region\n",
      "                         equal to the background aerosol.\n",
      "   Periodic Boundary Conditions -- BLIRB8 replicates the\n",
      "                         BLIRB8 space in all directions\n",
      "                         around and outside the BLIRB8 space.\n",
      "The default option is \"Periodic Boundary Conditions\".\n",
      "   ",
      "Wind Speed (mps):\n",
      "The scale is used to input the surface wind speed.  The\n",
      "default wind speed is 0.0 mps.\n",
      "" };

  create_message (w, help_str, XmDIALOG_INFORMATION);
}     /*   end help_cldCB()   */

/***********************************************************************
 *                          VOID HELP_SUNCB
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  help_sunCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  help_modifyCB
 *=====================================================================
 *<Description>
 *    Displays Sun Changes information.
 *=====================================================================
 *<Called routines>
 *    create_message        - draws a message in a dialog message box
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void help_sunCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - a pointer to any input data to be given to
 *                            the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *=====================================================================
```

```
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 *********************************************************************
 */
void help_sunCB (Widget w, XtPointer c, XtPointer call_data)
{
  static char *help_str[] = {
      "Sun Changes:\n",
      "There are 5 options --\n",
      "   ",
      "Solar Flux and Sky Radiance at 5 Km:\n",
      "   Parameterized\n",
      "   LOWTRAN\n",
      "The default option is \"LOWTRAN\".\n",
      "   ",
      "Sky Radiance Input:\n",
      "   No\n",
      "   Yes\n",
      "The default option is \"No\".\n",
      "   ",
      "Spectral Molecular Transmission:\n",
      "   No\n",
      "   Yes\n",
      "The default option is \"No\".\n",
      "   ",
      "Solar Zenith Angle (deg):\n",
      "This scale is used to input the solar zenith angle.  The\n",
      "default solar zenith angle is 0.0 deg.\n",
      "   ",
      "Solar Azimuth Angle (deg):\n",
      "This scale is used to input the solar azimuth angle.  The\n",
      "default solar azimuth angle is 0.0 deg.\n",
      "" };

  create_message (w, help_str, XmDIALOG_INFORMATION);
}     /*   end help_sunCB()   */

/*********************************************************************
 *                          VOID HELP_FLARECB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  help_flareCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  help_modifyCB
 *=====================================================================
 *<Description>
 *    Displays Flare Selection information.
 *=====================================================================
 *<Called routines>
 *    create_message        - draws a message in a dialog message box
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void help_flareCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - a pointer to any input data to be given to
```

```
 *                                the routine
 *       call_data              - a pointer to the callback structure which
 *                                contains information on why the callback
 *                                occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 *********************************************************************
 */
void help_flareCB (Widget w, XtPointer c, XtPointer call_data)
{
  static char *help_str[] = {
      "Flare Selection:\n",
      "This option creates a secondary pulldown menu listing all\n",
      "flares and possibly a \"Add New Flare\" option.  The default\n",
      "input contains no flare.  Any option causes another pulldown\n",
      "menu to appear with from 1 - 3 new options.  These new\n",
      "options are --\n",
      "   ",
      "Parameter Options:\n",
      "This option creates a popup menu to appear with 3 options --\n",
      "   Flare Type:\n",
      "      Select one of the 3 types of flare energy radiation.\n",
      "   Flare Intensity (watts):\n",
      "      This scale is used to input the flare intensity.\n",
      "   Flare Temperature (K)\n",
      "      This scale is used to input the flare temperature.\n",
      "   ",
      "Location:\n",
      "This option causes the display to view downward from the\n",
      "positive Z axis and project the flare onto the XY plane in\n",
      "red with a red prompt to Move the Flare.  Change the flare\n",
      "position in the XY plane with the mouse.  The display will\n",
      "then view forward from the negative Y axis.  Follow the same\n",
      "procedure to move the flare in the XZ plane.  The default\n",
      "position of a new flare is at the origin.  Flare positions\n",
      "are depicted as red stars.\n",
      "   ",
      "Delete Flare:\n",
      "This option causes the flare to be removed from all inputs\n",
      "and the display.\n",
      "" };

  create_message (w, help_str, XmDIALOG_INFORMATION);
}     /*   end help_flareCB()   */

/*********************************************************************
 *                          VOID HELP_SLITECB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  help_sliteCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  help_modifyCB
 *=====================================================================
 *<Description>
 *    Displays Searchlight Selection information.
```

```
 *=====================================================================
 *<Called routines>
 *    create_message        - draws a message in a dialog message box
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void help_sliteCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - a pointer to any input data to be given to
 *                            the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void help_sliteCB (Widget w, XtPointer c, XtPointer call_data)
{
  static char *help_str[] = {
      "Searchlight Selection:\n",
      "The default input contains no searchlight.  This option\n",
      "creates a secondary pulldown menu with 1 - 3 options.  These\n",
      "new options are --\n",
      "   ",
      "Add/Modify Searchlight:\n",
      "This option creates a popup menu with 5 input scales --\n",
      "   SearchLight Beam Zenith (deg)\n",
      "   Searchlight Beam Azimuth (deg)\n",
      "   SearchLight Intensity (watts)\n",
      "   SearchLight Temperature (K)\n",
      "   SearchLight Diameter (m)\n",
      "   ",
      "Location:\n",
      "This option causes the display to view downward from the\n",
      "positive Z axis and project the searchlight onto the XY\n",
      "plane in white with a red prompt to Move the Slite.  Change\n",
      "the searchlight position in the XY plane with the mouse.\n",
      "The display will then view forward from the negative Y axis.\n",
      "Follow the same procedure to move the searchlight in the XZ\n",
      "plane.  The default position of a new searchlight is at the\n",
      "origin.  Searchlight positions are depicted as white stars.\n",
      "   ",
      "Delete Searchlight:\n",
      "This option causes the searchlight to be removed from all\n",
      "inputs and the display.\n",
      "" };

  create_message (w, help_str, XmDIALOG_INFORMATION);
}     /*   end help_sliteCB()   */

/********************************************************************
 *                          VOID HELP_SPECTCB
 ********************************************************************
```

```
 *<Begin>
 *<Identification>          Name:  help_spectCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  help_modifyCB
 *=====================================================================
 *<Description>
 *    Displays Spectral Range Changes information.
 *=====================================================================
 *<Called routines>
 *    create_message        - draws a message in a dialog message box
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void help_spectCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - a pointer to any input data to be given to
 *                            the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void help_spectCB (Widget w, XtPointer c, XtPointer call_data)
{
  static char *help_str[] = {
      "Spectral Range Changes:\n",
      "This spectral range corresponds to the spectral range of the\n",
      "radiation energy wave bands for BLIRB8.  This option creates\n",
      "a pulldown submenu with 2 options to allow one to choose\n",
      "the units for the spectral range selection options.\n",
      "   ",
      "Wavenumber:\n",
      "This option creates a popup menu with 5 options --\n",
      "   Visible:   8000 - 28000 per cm\n",
      "   Near IR:   3000 - 13000 per cm\n",
      "   Mid IR:    1200 -  5200 per cm\n",
      "   Far IR:     500 -  1500 per cm\n",
      "   2 Color IR: 600 -  3600 per cm\n",
      "Choosing any of the 5 options will create another popup menu\n",
      "with 3 spectral range input scales --\n",
      "   Lowest Wavenumber (per cm)\n",
      "      The default value is 10000 per cm.\n",
      "   Highest Wavenumber (per cm)\n",
      "      The default value is 25000 per cm.\n",
      "   Number of Wavenumber Intervals\n",
      "      The default value is 15.\n",
      "   ",
      "Wavelength:\n",
      "This option creates a popup menu with 5 options --\n",
      "   Visible:    0.3 -  1.3 micrometers\n",
      "   Near IR:    0.7 -  3.2 micrometers\n",
```

```
        "   Mid IR:      2.0 -  7.0 micrometers\n",
        "   Far IR:      6.0 - 16.0 micrometers\n",
        "   2 Color IR: 3.0 - 13.0 micrometers\n",
        "Choosing any of the 5 options will create another popup menu\n",
        "with 3 spectral range input scales --\n",
        "   Lowest Wavelength (micrometers)\n",
        "   Highest Wavelength (micrometers)\n",
        "   Number of Wavelength Intervals\n",
        "" };

  create_message (w, help_str, XmDIALOG_INFORMATION);
}     /*   end help_spectCB()   */

/*********************************************************************
 *                          VOID HELP_COMPCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  help_compCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  help_modifyCB
 *====================================================================
 *<Description>
 *    Displays Computation Changes information.
 *====================================================================
 *<Called routines>
 *    create_message        - draws a message in a dialog message box
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void help_compCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - a pointer to any input data to be given to
 *                            the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void help_compCB (Widget w, XtPointer c, XtPointer call_data)
{
  static char *help_str[] = {
      "Computation Changes:\n",
      "This option creates a popup menu with 5 options --\n",
      "   ",
      "Delta Function Adjustment:\n",
      "   No\n",
      "   Yes\n",
      "The default choice is \"Yes\".\n",
      "   ",
      "Order of Spherical Harmonics:\n",
      "   Order 0\n",
```

```
        "   Order 1\n",
        "   Order 2\n",
        "   Order 3\n",
        "   Order 4\n",
        "   Order 5\n",
        "   Order 6\n",
        "The default choice is \"Order 2\".\n",
        "   ",
        "Maximum Number of Iterations:\n",
        "This scale is used to input the maximum number of iterations\n",
        "to be used in the computation.  The default value is 10.\n",
        "   ",
        "Convergence Criterion:\n",
        "This scale is used to input the convergence criterion to be\n",
        "used in the computation.  The default value is 0.002.\n",
        "   ",
        "Number of Convergence Fail Points:\n",
        "This scale is used to input the number of convergence fail\n",
        "points used in the computation.  The default value is 5.\n",
        "" };

  create_message (w, help_str, XmDIALOG_INFORMATION);
}     /*   end help_compCB()   */

/*********************************************************************
 *                          VOID HELP_OUTCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  help_outCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  help_modifyCB
 *====================================================================
 *<Description>
 *    Displays Output File Changes information.
 *====================================================================
 *<Called routines>
 *    create_message        - draws a message in a dialog message box
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void help_outCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - a pointer to any input data to be given to
 *                            the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void help_outCB (Widget w, XtPointer c, XtPointer call_data)
{
```

```
  static char *help_str[] = {
      "Output File Changes:\n",
      "These options govern the type of BLIRB8 radiant flux output\n",
      "file(s) created.  There are 4 options --\n",
      "   ",
      "No Output Flux -- no output file created\n",
      "Formatted Output File -- creates GRID.ASC\n",
      "Unformatted (binary) Output File -- creates GRID.BIN\n",
      "Both Formatted and Unformatted -- creates GRID.ASC and\n",
      "                                  GRID.BIN \n",
      "   ",
      "The default option is \"Both Formatted and Unformatted\".\n",
      "\n" };

  create_message (w, help_str, XmDIALOG_INFORMATION);
}     /*   end help_outCB()   */

/*********************************************************************
 *                          VOID CANCELCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  cancelCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  Numerous routines
 *====================================================================
 *<Description>
 *    Removes a widget.
 *====================================================================
 *<Called routines>
 *    none
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void cancelCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - the widget to be removed
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void cancelCB(Widget w, XtPointer c, XtPointer call_data)
{
  XtUnmanageChild((Widget)c);
}     /* end cancelCB() */

/*********************************************************************
 *                          VOID CANCELOBCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  cancelobCB
```

```
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  mtrl1CB
 *=====================================================================
 *<Description>
 *    Removes a widget and calls "obsc".
 *=====================================================================
 *<Called routines>
 *    obsc                  - sets up rowcolumn widget with names of
 *                            materials.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void cancelobCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - the widget to be removed
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 *********************************************************************
 */
void cancelobCB(Widget w, XtPointer c, XtPointer call_data)
{
  XtUnmanageChild((Widget)c);
  obsc();
}     /* end cancelobCB() */

/*********************************************************************
 *                          VOID CANCELOCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  canceloCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_messagef
 *=====================================================================
 *<Description>
 *    Removes the parent widget.
 *=====================================================================
 *<Called routines>
 *    none
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void canceloCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - the client data
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
```

```
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void canceloCB(Widget w, XtPointer c, XtPointer call_data)
{
  XtUnmanageChild(w);
  XtDestroyWidget(w);
}     /* end canceloCB() */

/********************************************************************
 *                          VOID CANCELSCB
 ********************************************************************
 *<Begin>
 *<Identification>          Name:  cancelsCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  sun_optCB
 *=====================================================================
 *<Description>
 *    Removes a widget and incorporates new Sun position data.
 *=====================================================================
 *<Called routines>
 *    drawscene             - Plots the 3-D BLIRB grid points, albedo
 *                            areas, aerosol regions, and the output
 *                            flux.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void cancelsCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - the widget to be removed
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void cancelsCB(Widget w, XtPointer c, XtPointer call_data)
{
  XtUnmanageChild((Widget)c);

  if(in_change)
    drawscene;

  in_change = FALSE;
}     /* end cancelsCB() */
```

```
/*********************************************************************
 *                          VOID CANCELACB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  cancelaCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  area_optCB
 *====================================================================
 *<Description>
 *    Removes a widget and locates the Albedo Area
 *====================================================================
 *<Called routines>
 *    area_albCB            - calls the appropriate function for the
 *                            area albedo
 *    area_locCB            - moves the area to the desired location
 *    set_albedo            - sets the Albedo values
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void cancelaCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - the widget to be removed
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void cancelaCB(Widget w, XtPointer c, XtPointer call_data)
{
  XtUnmanageChild((Widget)c);

  if(in_changea)
  { area_albCB(menu, &area, NULL);
    area_locCB(NULL, &area, NULL);
    set_albedo();
  }

  in_changea = FALSE;
}     /* end cancelaCB() */

/*********************************************************************
 *                          VOID CANCELBCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  cancelbCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  albedo1CB, albedo2CB, albedo3CB,
 *                                 albe0, albe1, albe2, albe3
 *====================================================================
 *<Description>
```

```
 *    Removes a widget and sets the Albedos
 *=====================================================================
 *<Called routines>
 *    set_albedo              - sets the Albedo values
 *    obsc                    - sets up rowcolumn widget with names of
 *                              materials.
 *    drawscene               - Plots the 3-D BLIRB grid points, albedo
 *                              areas, aerosol regions, and the output
 *                              flux.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void cancelbCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                    - the ID of the widget for which the
 *                              callback is registered
 *       c                    - the widget to be removed
 *       call_data            - a pointer to the callback structure which
 *                              contains information on why the callback
 *                              occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void cancelbCB(Widget w, XtPointer c, XtPointer call_data)
{
  XtUnmanageChild((Widget)c);

  if(in_change && (cur_ialb == mdl2_ialb))
  { set_albedo();
    obsc();
    drawscene();
  }

  in_change = FALSE;
}     /* end cancelbCB() */

/********************************************************************
 *                          VOID CANCELBBCB
 ********************************************************************
 *<Begin>
 *<Identification>          Name:  cancelbbCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  albedo_chg
 *=====================================================================
 *<Description>
 *    Removes a widget and sets the Albedos
 *=====================================================================
 *<Called routines>
 *    set_albedo              - sets the Albedo values
 *    obsc                    - sets up rowcolumn widget with names of
 *                              materials.
 *    drawscene               - Plots the 3-D BLIRB grid points, albedo
 *                              areas, aerosol regions, and the output
 *                              flux.
```

```
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void cancelbbCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                    - the ID of the widget for which the
 *                              callback is registered
 *       c                    - the widget to be removed
 *       call_data            - a pointer to the callback structure which
 *                              contains information on why the callback
 *                              occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void cancelbbCB(Widget w, XtPointer c, XtPointer call_data)
{
  XtUnmanageChild((Widget)c);

  cur_ialb = mdl2_ialb;
  set_albedo();
  obsc();
  drawscene();
}     /* end cancelbbCB() */

/********************************************************************
 *                          VOID CANCELRCB
 ********************************************************************
 *<Begin>
 *<Identification>          Name:  cancelrCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  regn_optCB
 *=====================================================================
 *<Description>
 *    Removes a widget and gets the Materials Info for a BLIRB Region
 *=====================================================================
 *<Called routines>
 *    regn_mtl1            - sets up the menu for selecting the
 *                           Materials and Densities
 *    regn_locCB           - moves the region to the desired location
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void cancelrCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                    - the ID of the widget for which the
 *                              callback is registered
 *       c                    - the widget to be removed
 *       call_data            - a pointer to the callback structure which
 *                              contains information on why the callback
 *                              occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
```

```
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void cancelrCB(Widget w, XtPointer c, XtPointer call_data)
{
  XtUnmanageChild((Widget)c);

  if(in_change)
  { regn_mtl1();
    regn_locCB(NULL, &regn, NULL);
  }

  in_change = FALSE;
}     /* end cancelrCB() */

/********************************************************************
 *                          VOID CANCELMCB
 ********************************************************************
 *<Begin>
 *<Identification>          Name:  cancelmCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  regn_mtlCB
 *=====================================================================
 *<Description>
 *    Removes a widget and incorporates new Region Material data.
 *=====================================================================
 *<Called routines>
 *    mtrl1CB                - Sets the Material Density (WMTL).
 *    set_aerosol            - sets the Aerosol Material values
 *    drawscene              - Plots the 3-D BLIRB grid points, albedo
 *                             areas, aerosol regions, and the output
 *                             flux.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void cancelmCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                   - the ID of the widget for which the
 *                             callback is registered
 *       c                   - the widget to be removed
 *       call_data           - a pointer to the callback structure which
 *                             contains information on why the callback
 *                             occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void cancelmCB(Widget w, XtPointer c, XtPointer call_data)
{
  static int index;

  XtUnmanageChild((Widget)c);
```

```
  if(in_change)
  { index = 10 * cur_regn + cur_mtl;
    mtrl1CB(menu, &index, NULL);
    if(cur_mtl == 0)
      set_aerosol();
    drawscene();
  }

  in_change = FALSE;
}     /* end cancelmCB() */

/*********************************************************************
 *                          VOID CANCELMECB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  cancelmeCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  meshxmenuCB, meshymenuCB,
 *                                 meshzmenuCB
 *====================================================================
 *<Description>
 *    Removes a widget and incorporates new Grid Mesh data.
 *====================================================================
 *<Called routines>
 *    set_axis_pts          - sets up the grid points for the X, Y, and
 *                            Z axes.
 *    regn_fix              - rectifies the BLIRB Region location
 *    area_fix              - rectifies the Albedo Area location
 *    drawscene             - Plots the 3-D BLIRB grid points, albedo
 *                            areas, aerosol regions, and the output
 *                            flux.
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void cancelmeCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - the widget to be removed
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void cancelmeCB(Widget w, XtPointer c, XtPointer call_data)
{
  int i, j, k, m;
  float sum;

  XtUnmanageChild((Widget)c);

  if(in_change)
/*-------------------------------------------------------------------
```

```
 * --- Delete any Mesh which was cataloged for deletion in "meshCB".
 *-----------------------------------------------------------------------
 */
  { for (i=0; i<3; i++)
    { if(mes_del_cnt[i] > 0)
      { if(mes_del_cnt[i] > 1)
        { for (j=0; j<(mes_del_cnt[i]-1); j++)
          { for (k=0; k<(mes_del_cnt[i]-1); k++)
            { if(mes_del[i][k+1] > mes_del[i][k])
              { m = mes_del[i][k+1];
                mes_del[i][k+1] = mes_del[i][k];
                mes_del[i][k] = m;
              }
            }
          }
        }

        for (j=0; j<mes_del_cnt[i]; j++)
        { for (k=mes_del[i][j]; k<mes[i]; k++)
          { mes_ms[i][k] = mes_ms[i][k+1];
            mes_mh[i][k] = mes_mh[i][k+1];
          }
          mes[i]--;
        }

        mes_del_cnt[i] = 0;
      }

/*-----------------------------------------------------------------------
 * --- Add the Mesh which was input in "meshCB".
 *-----------------------------------------------------------------------
 */
      if(mes_add[i][0] > 0 && mes_add[i][1] > 0)
      { for (sum=0.0, j=0; j<=mes[i]; sum += mes_mh[i][j], j++);
        if((sum + mes_add[i][1]) > MAXXYZ)
          mes_add[i][1] = MAXXYZ - sum;

        if(mes_add[i][1] > 0.1)
        { if(mes[i] == 0)
          { mes_ms[i][1] = mes_ms[i][0];
            mes_mh[i][1] = mes_mh[i][0];
            mes_ms[i][0] = mes_add[i][0];
            mes_mh[i][0] = mes_add[i][1];
            mes[i]++;
          }

          else
          { m = 0;
            for (j=0; j<mes[i]; j++)
            { if(mes_add[i][0] > mes_ms[i][j])
                m = j+1;
            }

            for (j=mes[i]; j>=m; j--)
            { mes_ms[i][j+1] = mes_ms[i][j];
              mes_mh[i][j+1] = mes_mh[i][j];
            }
            mes_ms[i][m] = mes_add[i][0];
            mes_mh[i][m] = mes_add[i][1];
            mes[i]++;
          }
        }
```

```
        mes_add[i][0] = mes_add[i][1] = -1;
      }
    }

/*-----------------------------------------------------------------------
 * --- Get the new major and minor grid points.
 *-----------------------------------------------------------------------
 */
    set_axis_pts();

/*-----------------------------------------------------------------------
 * --- Rectify the Regions with the new mesh.
 *-----------------------------------------------------------------------
 */
    if(regn > 0)
    { for (i=1; i<=regn; i++)
      { cur_regn = i;
        regn_fix();
      }
    }

/*-----------------------------------------------------------------------
 * --- Rectify the Areas with the new mesh.
 *-----------------------------------------------------------------------
 */
    if(area > 0)
    { for (i=1; i<=area; i++)
      { cur_area = i;
        area_fix();
      }
    }
  }

/*-----------------------------------------------------------------------
 * --- Draw the new scene and reset the "change" flag.
 *-----------------------------------------------------------------------
 */
  drawscene();

  in_change = FALSE;
}     /* end cancelmeCB() */

/***********************************************************************
 *                          VOID CANCELFCB
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  cancelfCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  flar_optCB
 *=====================================================================
 *<Description>
 *    Removes a widget and locates the Flare
 *=====================================================================
 *<Called routines>
 *    flar_locCB            - moves the flare to the desired location
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void cancelfCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
```

```
 *                                 callback is registered
 *       c                       - the widget to be removed
 *       call_data               - a pointer to the callback structure which
 *                                 contains information on why the callback
 *                                 occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void cancelfCB(Widget w, XtPointer c, XtPointer call_data)
{
  XtUnmanageChild((Widget)c);

  if(in_changef)
    flar_locCB(NULL, &flar, NULL);

  in_changef = FALSE;
}     /* end cancelfCB() */

/********************************************************************
 *                          VOID CANCELSLCB
 ********************************************************************
 *<Begin>
 *<Identification>          Name:  cancelslCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  srch_optCB
 *=====================================================================
 *<Description>
 *    Removes a widget and locates the SearchLight
 *=====================================================================
 *<Called routines>
 *    srch_locCB            - moves the Slite to the desired location
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void cancelslCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - the widget to be removed
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void cancelslCB(Widget w, XtPointer c, XtPointer call_data)
{
```

```
  XtUnmanageChild((Widget)c);

  if(in_changesl)
    srch_locCB(NULL, NULL, NULL);

  in_changesl = FALSE;
}     /* end cancelslCB() */

/*********************************************************************
 *                          VOID OKFCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  okfCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_messagef
 *====================================================================
 *<Description>
 *    Kills the widget which calls it then calls "newfile" or "fileopen"
 *====================================================================
 *<Called routines>
 *    newfile                - resets the inputs to initial configuration
 *    fileopen               - sets up the "File Select" menu for new
 *                             BLIRB input or output file
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void okfCB(Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                   - the ID of the widget for which the
 *                             callback is registered
 *       c                   - a pointer to any input data to be given to
 *                             the routine
 *       call_data           - a pointer to the callback structure which
 *                             contains information on why the callback
 *                             occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void okfCB(Widget w, XtPointer c, XtPointer call_data)
{
  if(filefctn == 2)
  { XtCloseDisplay(XtDisplay(w));
    exit(0);
  }

  XtUnmanageChild(w);
  XtDestroyWidget(w);

  if(filefctn == 0)
    newfile();
  else if(filefctn == 1)
    fileopen();
}     /* end okfCB() */
```

```
/*********************************************************************
 *                          VOID OKCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  okCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_message
 *====================================================================
 *<Description>
 *    Kills the widget which calls it.  Used to remove dialog boxes
 *    without any action.
 *====================================================================
 *<Called routines>
 *    None
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void okCB(Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - a pointer to any input data to be given to
 *                            the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void okCB(Widget w, XtPointer c, XtPointer call_data)
{
  XtUnmanageChild(w);
  XtDestroyWidget(w);
}     /* end okCB() */

/*********************************************************************
 *                          VOID FLUXCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  fluxCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_fluxmenu
 *====================================================================
 *<Description>
 *    Selects the particular flux field for display and redraws the
 *    scene.
 *====================================================================
 *<Called routines>
 *    plot_out_def1         - Creates a Widget telling the user what
 *                            flux info he is viewing.
 *    drawscene             - Plots the 3-D BLIRB grid points, albedo
 *                            areas, aerosol regions, and the output
 *                            flux.
```

```
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void fluxCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                    - the ID of the widget for which the
 *                              callback is registered
 *       c                    - an index to indicate the flux field choice
 *       call_data            - a pointer to the callback structure which
 *                              contains information on why the callback
 *                              occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void fluxCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *n = (int *)c;
  int i, ii;

  if(*n == -2)
  { ii = -1;
    for (i=0; i<10; i++)
    { if(flux_flag[i])
        ii = i;
    }
    if(ii == -1)
      ii = 0;
    else if(ii == 0)
      ii = 9;
    else if(ii > 0)
      ii--;
  }

  else if(*n == -1)
  { ii = -1;
    for (i=0; i<10; i++)
    { if(flux_flag[i])
        ii = i;
    }
    if(ii == -1)
      ii = 0;
    else if(ii < 9)
      ii++;
    else if(ii == 9)
      ii = 0;
  }

  else if(*n == 0)
  { ii = -1;
    noflux = TRUE;
  }
  if(*n > 0 && *n < 11)
    ii = *n - 1;

  for (i=0; i<10; i++)
```

```
  { if(i == ii)
    { flux_flag[i] = TRUE;
      noflux = FALSE;
    }
    else
      flux_flag[i] = FALSE;
  }

  plot_out_def1();
  drawscene();
}     /*   end fluxCB()   */

/*********************************************************************
 *                          VOID CROSS_SECTIONCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  cross_sectionCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_csectmenu
 *====================================================================
 *<Description>
 *    Selects the Cross-Section Plane for flux display, recreates the
 *    menubar, and redraws the scene.
 *====================================================================
 *<Called routines>
 *    create_menubar        - creates the menubar for selecting the
 *                            various options
 *    plot_out_def1         - Creates a Widget telling the user what
 *                            flux info he is viewing.
 *    drawscene             - Plots the 3-D BLIRB grid points, albedo
 *                            areas, aerosol regions, and the output
 *                            flux.
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void cross_sectionCB(Widget w, XtPointer c,XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - an index to indicate the flux cross-
 *                            section choice
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void cross_sectionCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *n = (int *)c;
  int i;

/*-------------------------------------------------------------------
 * --- Select the flux cross-section axis.
```

```
 *-----------------------------------------------------------------------
 */
  for (i=0; i<3; i++)
  { if(*n == i+1)
      cross_axis[i] = TRUE;
    else
      cross_axis[i] = FALSE;
  }

/*-----------------------------------------------------------------------
 * --- Destroy the current menubar widget and create a new one to
 *     account for the new cross-section plane value options.
 *-----------------------------------------------------------------------
 */
  XtUnmanageChild (menu);
  menu = create_menubar(form);

/*-----------------------------------------------------------------------
 * --- Redraw the scene.
 *-----------------------------------------------------------------------
 */
  plot_out_def1();
  drawscene();
}     /*   end cross_sectionCB()    */

/***********************************************************************
 *                          VOID PLANECB
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  planeCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_cvaluemenu
 *=====================================================================
 *<Description>
 *    Selects the Cross-Section Plane Value for flux display and redraws
 *    the scene.
 *=====================================================================
 *<Called routines>
 *    plot_out_def1         - Creates a Widget telling the user what
 *                            flux info he is viewing.
 *    drawscene             - Plots the 3-D BLIRB grid points, albedo
 *                            areas, aerosol regions, and the output
 *                            flux.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void planeCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - an index to indicate the flux cross-
 *                            section plane value choice
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
```

```
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void planeCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *n = (int *)c;
  int i, ii;

  for(i=0; i<3; i++)
  { if(cross_axis[i])
    { ii = cross_value[i];

      { if(*n == -2)
        { if(ii == 0)
            ii = out_imx[i] - 1;
          else if(ii > 0)
            ii--;
        }

        else if(*n == -1)
        { if(ii < out_imx[i] - 1)
            ii++;
          else if(ii == out_imx[i] - 1)
            ii = 0;
        }

        else if(*n >= 1 && *n <= out_imx[i])
          ii = *n - 1;
        else
          ii = out_imx[i] - 1;
      }

      cross_value[i] = ii;
    }
  }

  plot_out_def1();
  drawscene();
}     /*   end planeCB()    */

/***********************************************************************
 *                            VOID WAVECB
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  waveCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_wavemenu
 *=====================================================================
 *<Description>
 *    Selects the Wave Number Value for flux display and redraws the
 *    scene.
 *=====================================================================
 *<Called routines>
 *    plot_out_def1          - Creates a Widget telling the user what
 *                             flux info he is viewing.
 *    drawscene              - Plots the 3-D BLIRB grid points, albedo
 *                             areas, aerosol regions, and the output
 *                             flux.
 *=====================================================================
 *<Parameters>
```

```
 *    Formal declaration:
 *       void waveCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                    - the ID of the widget for which the
 *                              callback is registered
 *       c                    - an index to indicate the flux wavenumber
 *                              choice
 *       call_data            - a pointer to the callback structure which
 *                              contains information on why the callback
 *                              occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void waveCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *n = (int *)c;
  int i, ii;

  ii = cur_nwave;

  { if(*n == -2)
    { if(ii == 0)
        ii = out_nwave - 1;
      else if(ii > 0)
        ii--;
    }

    else if(*n == -1)
    { if(ii < out_nwave - 1)
        ii++;
      else if(ii == out_nwave - 1)
        ii = 0;
    }

    else if(*n >= 1 && *n <= out_nwave)
      ii = *n - 1;
    else
      ii = out_nwave - 1;
  }

  cur_nwave = ii;

  plot_out_def1();
  drawscene();
}     /*   end waveCB()    */

/********************************************************************
 *                          VOID NEWFCB
 ********************************************************************
 *<Begin>
 *<Identification>          Name:  newfCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_filemenu
 *=====================================================================
```

```
 *<Description>
 *    Decides whether or not to call "newfile".
 *=====================================================================
 *<Called routines>
 *    create_messagef        - draws a message in a dialog message box
 *                             and continues or halts the current
 *                             operation.
 *    newfile                - resets the inputs to initial configuration
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void newfCB(Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                   - the ID of the widget for which the
 *                             callback is registered
 *       c                   - the data passed to the routine
 *       call_data           - a pointer to the callback structure which
 *                             contains information on why the callback
 *                             occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 *********************************************************************
 */
void newfCB (Widget w, XtPointer c, XtPointer call_data)
{
  static char *msg[] = {
                "The original BLIRB input data has been\n",
                "    modified and not saved to disk.\n",
                "You will destroy the current data if you proceed!\n",
                "==============================================\n",
                "      CONTINUE to proceed, CANCEL to stop.\n",
                "" };

  filefctn = 0;

  if(new_file)
    create_messagef( menu, msg, XmDIALOG_ERROR);
  else
    newfile();
}     /*   end newfCB()    */

/*********************************************************************
 *                          VOID NEWFILE
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  newfile
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  main, newfCB, okfCB
 *=====================================================================
 *<Description>
 *    Resets the input parameters to initial configuration.
 *=====================================================================
 *<Called routines>
 *    getdata                - processes the data from a BLIRB input or
 *                             output file
```

```
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void newfile(void)
 *    Input:
 *       None
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void newfile (void)
{
  int i;
  static char *fname = "grid_newfile.i";

  def_file = TRUE;                /* Default data file is used       */
  new_file = FALSE;

  nofile = FALSE;
  area_order = FALSE;
  regn_order = FALSE;
  sun_plot = TRUE;

  file_name = fname;

  blirb_in = TRUE;
  badfile = FALSE;

  noflux = TRUE;
  for (i=0; i<10; i++)
    flux_flag[i] = FALSE;

  ilcl = 0;

  mdl1 = 0;
  mdl1_iaersl = 0;
  mdl1_model = 6;
  mdl1_ivis = 1;
  mdl1_iseasn = 0;
  mdl1_ivulcn = 1;

  mdl2 = 0;
  mdl2_sn = 0.8;
  mdl2_tbound = 288.2;
  mdl2_ialb = -1;
  mdl2_ip = 0;
  cur_ialb = mdl2_ialb;

  mdl3 = -1;
  mdl3_t[0] = 288.2;
  mdl3_t[1] = 281.8;
  mdl3_t[2] = 275.2;
  mdl3_t[3] = 268.8;
  mdl3_t[4] = 262.8;
  mdl3_t[5] = 255.8;
```

```
area = 0;
area_alx[0] = 0.0;
area_ahx[0] = 5.0;
area_aly[0] = 0.0;
area_ahy[0] = 4.0;
area_iamtl[0] = 1;

regn = 0;
regn_rlx[0] = 0.0;
regn_rhx[0] = 5.0;
regn_rly[0] = 0.0;
regn_rhy[0] = 4.0;
regn_rlz[0] = 0.0;
regn_rhz[0] = 5.0;
regn_izmtl[0] = 0;
regn_rl[0][0] = regn_rlx[0];
regn_rh[0][0] = regn_rhx[0];
regn_rl[1][0] = regn_rly[0];
regn_rh[1][0] = regn_rhy[0];
regn_rl[2][0] = regn_rlz[0];
regn_rh[2][0] = regn_rhz[0];

mesx = 0;
mesx_mhx[0] = 2.0 * regn_rhx[0];
mesx_xms[0] = regn_rhx[0];
mes[0] = mesx;
mes_mh[0][mesx] = mesx_mhx[mesx];
mes_ms[0][mesx] = mesx_xms[mesx];

mesy = 0;
mesy_mhy[0] =2.0 * regn_rhy[0];
mesy_yms[0] =regn_rhy[0];
mes[1] = mesy;
mes_mh[1][mesy] = mesy_mhy[mesy];
mes_ms[1][mesy] = mesy_yms[mesy];

mesz = 0;
mesz_mhz[0] =2.0 * regn_rhz[0];
mesz_zms[0] =regn_rhz[0];
mes[2] = mesz;
mes_mh[2][mesz] = mesz_mhz[mesz];
mes_ms[2][mesz] = mesz_zms[mesz];

albd = 0;
albd_lalb[0] = 1;
albd_falb[0] = 0.2;

mtrl = -1;

clds = 0;
clds_icld = 0;
clds_ibnd = 1;
clds_wind = 0;

domd = 0;
domd_isc = 2;
domd_iitl = 10;
domd_epsi = 0.002;
domd_idelta = 1;
domd_npts = 5;

sun = 0;
```

```
  sun_thsun = 0.0;
  sun_phsun = 0.0;
  sun_ifsun = 1;
  sun_isky = 0;
  sun_iftrn = 0;

  flar = -1;
  srch = -1;

  wavn = 0;
  wavn_v1 = 10000;
  wavn_v2 = 25000;
  wavn_dv = 15;
  wavn_indx = 0;

  asci = 0;
  asci_irite = 3;

  recl = 0;
  recl_irpt = 1;

  done = 0;

  getdata();
}     /*   end newfile()    */

/*********************************************************************
 *                          VOID SAVEFILECB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  savefileCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_filemenu
 *====================================================================
 *<Description>
 *    Saves the current inputs to a file with the current input filename
 *    after saving the current input file under the same filename with
 *    an extension ".bak".
 *====================================================================
 *<Called routines>
 *    writecards            - writes the input cards to a file
 *    create_message        - draws a message in a dialog message box
 *    drawscene             - Plots the 3-D BLIRB grid points, albedo
 *                            areas, aerosol regions, and the output
 *                            flux.
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void savefileCB(Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - the data passed to the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
```

```
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void savefileCB (Widget w, XtPointer c, XtPointer call_data)
{
  char command[200];
  static char *msg[] = {
                "                                                    ",
                "                                                    ",
                "                                                    ",
                "                                                    ",
                "" };

  if(!nofile)
  { if(blirb_in)
    { strcpy(command, "mv ");
      strcat(command, file_name);
      strcat(command, " ");
      strcat(command, file_name);
      strcat(command, ".bak\n");

      system(command);
      writecards();

      new_file = FALSE;
    }
    else
    { sprintf(msg[0], " Inputs were obtained from an Output File.\n");
      sprintf(msg[1], " Inputs must be saved in an Input File.\n");
      sprintf(msg[2], " Select <Save File As> option under <File>.\n");
      sprintf(msg[3], " Input cards NOT Saved.\n");

      create_message( menu, msg, XmDIALOG_ERROR);
    }
  }
  else
  { sprintf(msg[0], " No filename specified.\n");
    sprintf(msg[1], " Select <Save File As> option under <File>.\n");
    sprintf(msg[2], " Input cards NOT Saved.\n");
    sprintf(msg[3], "\n");

    create_message( menu, msg, XmDIALOG_ERROR);
  }

  drawscene();
}     /*   end savefileCB()   */

/***********************************************************************
 *                          VOID SAVEFILEASCB
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  savefileasCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_filemenu
 *=====================================================================
 *<Description>
 *    Creates a "Save Filename" Text Widget.
 *=====================================================================
 *<Called routines>
```

```
 *    getfilenameCB          - gets the specified filename and checks to
 *                             make sure it is an input filename
 *    drawscene              - Plots the 3-D BLIRB grid points, albedo
 *                             areas, aerosol regions, and the output
 *                             flux.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void savefileasCB(Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       filepane            - the ID of the widget for which the
 *                             callback is registered
 *       c                   - the data passed to the function
 *       call_data           - a pointer to the callback structure which
 *                             contains information on why the callback
 *                             occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void savefileasCB(Widget filepane, XtPointer c, XtPointer call_data)
{
  Widget form1, label1, text1;
  int n;
  Arg wargs[10];

/*-------------------------------------------------------------------
 * --- Use the Widget "file_dialog" to create a Bulletin Board Dialog
 *     Widget with that ID.
 *-------------------------------------------------------------------
 */
  n = 0;
  XtSetArg(wargs[n], XmNdialogStyle, XmDIALOG_MODELESS); n++;
  XtSetArg(wargs[n], XmNwidth, 400); n++;
  XtSetArg(wargs[n], XmNheight, 65); n++;
  file_dialog = XmCreateBulletinBoardDialog( filepane, "Save Filename",
                wargs, n);

  form1 = XtCreateManagedWidget("form1", xmFormWidgetClass,
          file_dialog, NULL, 0);
  label1 = XtCreateManagedWidget("Filename:", xmLabelWidgetClass,
           form1, NULL, 0);

  n = 0;
  XtSetArg (wargs[n], XmNleftAttachment, XmATTACH_WIDGET); n++;
  XtSetArg (wargs[n], XmNrightAttachment, XmATTACH_FORM); n++;
  XtSetArg (wargs[n], XmNleftWidget, label1); n++;
  text1 = XmCreateText(form1, "text1", wargs, n);
  XtAddCallback (text1, XmNactivateCallback, getfilenameCB, NULL);
  XtManageChild (text1);

/*-------------------------------------------------------------------
 * --- Realize the Filename Input Text Widget.
 *-------------------------------------------------------------------
 */
  XtManageChild(file_dialog);
```

```
  drawscene();
}     /*   end savefileasCB()    */

/*********************************************************************
 *                          VOID GETFILENAMECB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  getfilenameCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  savefileasCB
 *====================================================================
 *<Description>
 *    Gets the BLIRB "Save" filename and checks the file type for input
 *    or output.
 *====================================================================
 *<Called routines>
 *    create_message        - draws a message in a dialog message box
 *    writecards            - writes the BLIRB input cards to a file
 *    drawscene             - Plots the 3-D BLIRB grid points, albedo
 *                            areas, aerosol regions, and the output
 *                            flux.
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void getfilenameCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - the input from the calling routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void getfilenameCB (Widget w, XtPointer c, XtPointer call_data)
{
  char *filestring, *ptr;
  static char *exten = {".i"};
  Boolean inputfile;
  int i;
  static char *error_mesg[] = {
                "Specified Filename is not appropriate for Input.\n",
                "Please try a Filename with a <.i> extension.\n",
                "" };

/*-------------------------------------------------------------------
 * --- Remove the Filename Input Text Widget from the screen.
 *-------------------------------------------------------------------
 */
  XtUnmanageChild(file_dialog);

/*-------------------------------------------------------------------
 * --- Get the selected BLIRB data filename, then check the filename to
```

```
 *     find out whether or not it is input.
 *-----------------------------------------------------------------------
 */
  if( (filestring = XmTextGetString(w)) != NULL);
  { ptr = filestring;                        /* A filename was entered   */

    for (i=0; i<strlen(filestring); i++, ptr++)
    { if(ptr[0] == exten[0])
      { ptr++;
        if(ptr[0] == exten[1])
          inputfile = TRUE;
        else
          inputfile = FALSE;
      }
    }
/*-----------------------------------------------------------------------
 * --- If the datafile is not input, display an error message in a box.
 *     If the datafile is input, then write the cards to the file.
 *-----------------------------------------------------------------------
 */
    if(!inputfile)
      create_message( w, error_mesg, XmDIALOG_ERROR);
    else
    { file_name = filestring;
      writecards();
      new_file = FALSE;
    }
  }

  drawscene();
}     /*   end getfilenameCB()    */

/***********************************************************************
 *                           VOID CLOUD_OPTCB
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  cloud_optCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_modifymenu
 *=====================================================================
 *<Description>
 *    Selects the various Cloud Options for BLIRB
 *=====================================================================
 *<Called routines>
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelCB               - removes the Rowcol Widget
 *    create_separator       - creates a Separator Widget
 *    create_radiobox        - creates a Radiobox Widget
 *    create_togglebutton    - creates a Togglebutton Widget
 *    create_scale           - creates a Scale Widget
 *    cloudCB                - Set the input parameters of the CLDS card
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void cloud_optCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                   - the ID of the widget for which the
 *                             callback is registered
 *       c                   - the data passed to the routine
 *       call_data           - a pointer to the callback structure which
 *                             contains information on why the callback
```

```
 *                                occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void cloud_optCB (Widget w, XtPointer c, XtPointer call_data)
{
  Widget radio[2], toggle[5], rowcol, label[11], scale;
  Arg wargs[15];
  int n, i, j, k;
  static int index[6], hv, sd, nc, wid, min, max, inc, dec, val, swid;
  static int nct[2] = { 3, 2 };
  static char cat_label[2][32] = { "Cloud Structure",
                                   "Aerosol Outside Physical Region" };
  static char tog_label[5][29] = { "No Cloud", "Rectangular Structure",
                                   "CSS Model", "Background Aerosol",
                                   "Periodic Boundary Conditions"};
  static char scale_label[17] = "Wind Speed (mps)";
  static char numb[11][3] = { "0", "5", "10", "15", "20", "25", "30",
                              "35", "40", "45", "50" };

/*-------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-------------------------------------------------------------------
 */
  rowcol = create_rowcol(w, "Cloud_Options", cancelCB);

/*-------------------------------------------------------------------
 * --- Create the radioboxes and toggles
 *-------------------------------------------------------------------
 */
  hv = 0;
  sd = 2;
  nc = 1;
  k = 0;
  for (i=0; i<2; i++)
  { create_separator(rowcol, &hv, &sd);
    radio[i] = create_radiobox(rowcol, &nc, cat_label[i]);

    for (j=0; j<nct[i]; k++, j++)
    { index[k] = 10*i + j;
      toggle[k] = create_togglebutton(radio[i], tog_label[k], &index[k],
                  cloudCB);
    }

    if(clds == 0)
    { if(i == 0)
      { j = clds_icld;
        if(j > 0)
          j--;
        XmToggleButtonSetState(toggle[j], TRUE, FALSE);
      }
      else
        XmToggleButtonSetState(toggle[3+(int) clds_ibnd], TRUE, FALSE);
    }
  }
```

```
/*-----------------------------------------------------------------
 * --- Create the Scale
 *-----------------------------------------------------------------
 */
  create_separator(rowcol, &hv, &sd);

  k++;
  index[k] = 20;
  wid = 560;
  min = 0;
  max = 500;
  inc = 0;
  dec = 1;
  if(clds == 0)
    val = 10.0*clds_wind;
  else
    val = 0;
  swid = 538;

  scale = create_scale(rowcol, scale_label, &wid, &min, &max, &inc,
          &dec, &val, &swid, &index[k], cloudCB);

  for (j=0; j<11; j++)
  { n = 0;
    XtSetArg (wargs[n], XmNwidth, 45); n++;
    label[j] = XmCreateLabel(scale, numb[j], wargs, n);
  }
  XtManageChildren(label, 11);
}     /*   end cloud_optCB()   */

/*******************************************************************
 *                          VOID CLOUDCB
 *******************************************************************
 *<Begin>
 *<Identification>          Name:  cloudCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  cloud_optCB
 *===================================================================
 *<Description>
 *    Sets the input parameters of the CLDS card
 *===================================================================
 *<Called routines>
 *    None
 *===================================================================
 *<Parameters>
 *    Formal declaration:
 *       void cloudCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - pointer to the data passed to the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *===================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *===================================================================
```

```
 *<End>
 ***********************************************************************
 */
void cloudCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *n = (int *)c;
  int i, j, value;
  XmScaleCallbackStruct * call_value =
                              (XmScaleCallbackStruct *) call_data;

  value = call_value -> value;

  i = (*n)/10;                              /* RadioBox category        */
  j = (*n) % 10;                            /* Toggle button pushed     */

  if(clds < 0)
  { clds = 0;
    clds_icld = 0.0;
    clds_ibnd = 0.0;
    clds_wind = 0.0;
  }

  if(i == 0)
  { if(j == 0)
      clds_icld = j;
    else
      clds_icld = j+1;
  }
  else if(i == 1)
    clds_ibnd = j;
  else if(i == 2)
    clds_wind = 0.1 * (float)value;

  new_file = TRUE;
}     /*   end cloudCB()    */

/***********************************************************************
 *                          VOID MODEL_OPTCB
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  model_optCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_modifymenu
 *=====================================================================
 *<Description>
 *    Sets the various Model Options for BLIRB on MDL1, MDL2 and MDL3
 *    cards.
 *=====================================================================
 *<Called routines>
 *    create_bboard         - creates a BulletinBoard Widget
 *    cancelCB              - removes the BulletinBoard Widget
 *    create_separator      - creates a Separator Widget
 *    create_radiobox       - creates a Radiobox Widget
 *    create_togglebutton   - creates a Togglebutton Widget
 *    create_scale          - creates a Scale Widget
 *    aerosol_optCB         - sets the Aerosol Options for IAERSL
 *    modelCB               - sets some of the input parameters on the
 *                            MDL1 and MDL2 cards
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
```

```
 *       void model_optCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                    - the ID of the widget for which the
 *                              callback is registered
 *       c                    - the data passed to the routine
 *       call_data            - a pointer to the callback structure which
 *                              contains information on why the callback
 *                              occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void model_optCB (Widget w, XtPointer c, XtPointer call_data)
{
  Widget radio[5], toggle[20], button, bboard, rowcol, rowcol1, rowcol2,
         rowcol3, rowcol4, label[11], scale;
  Arg wargs[15];
  int n, i, j, k;
  static int index[21], hv, sd, nc, wid, inc, dec, val, swid;
  static int nct[5] = { 7, 2, 3, 3, 5 };
  static char cat_label[5][26] = { "Temperature Profile Model",
                                   "Meteorological Range",
                                   "Tropospheric Profile",
                                   "Albedo","Aerosol Profile Printout"};
  static char tog_label[20][33] = { "Tropical Atmosphere",
                                    "Midlatitude Summer",
                                    "Midlatitude Winter",
                                    "Subarctic Summer",
                                    "Subarctic Winter",
                                    "1976 U.S. Standard",
                                    "User Defined Temperature Profile",
                                    "Met Range < 5 km",
                                    "Met Range < 50 km",
                                    "Set by Meteorological Range",
                                    "Spring-Summer", "Fall-Winter",
                                    "Wave Independent, User-defined",
                                    "Wave Independent, Tabulated",
                                    "Spectral", "None",
                                    "Ext Coefs & Scale Factors",
                                    "Adds Cross_sections to Printout",
                                    "Adds Scatr & Absorption Coefs",
                                    "Full Details" };
  static char scale_label[24] = "Surface Temperature (K)";
  static int min = 2200;
  static int max = 3200;
  static char numb[11][4] = { "220", "230", "240", "250", "260", "270",
                              "280", "290", "300", "310", "320" };

/*---------------------------------------------------------------------
 * --- Create a Bulletin Board Widget
 *---------------------------------------------------------------------
 */
  bboard = create_bboard(w, "Model Options");

/*---------------------------------------------------------------------
 * --- Create a RowColumn Widget, "rowcol".
```

```
 *-----------------------------------------------------------------------
 */
  n = 0;
  XtSetArg(wargs[n], XmNorientation, XmVERTICAL); n++;
  rowcol = XtCreateManagedWidget("Model_Options",
           xmRowColumnWidgetClass, bboard, wargs, n);

/*-----------------------------------------------------------------------
 * --- Create two RowColumn Widgets, "rowcol3" and "rowcol4", within
 *     "rowcol".
 *-----------------------------------------------------------------------
 */
  n = 0;
  XtSetArg(wargs[n], XmNorientation, XmHORIZONTAL); n++;
  rowcol3 = XtCreateManagedWidget("Model_Options",
            xmRowColumnWidgetClass, rowcol, wargs, n);

  hv = 0;
  sd = 2;
  create_separator(rowcol, &hv, &sd);

  n = 0;
  XtSetArg(wargs[n], XmNorientation, XmVERTICAL); n++;
  rowcol4 = XtCreateManagedWidget("Model_Options",
            xmRowColumnWidgetClass, rowcol, wargs, n);

/*-----------------------------------------------------------------------
 * --- Create a RowColumn Widget, "rowcol1", within "rowcol3.
 *-----------------------------------------------------------------------
 */
  n = 0;
  XtSetArg(wargs[n], XmNorientation, XmVERTICAL); n++;
  rowcol1 = XtCreateManagedWidget("Model_Options",
            xmRowColumnWidgetClass, rowcol3, wargs, n);

/*-----------------------------------------------------------------------
 * --- Create a finishbutton, attach it to "rowcol1", and realize it.
 *-----------------------------------------------------------------------
 */
  n = 0;
  button = XmCreatePushButton(rowcol1,"Finished with Selections", wargs,
           n);
  XtAddCallback (button, XmNactivateCallback, cancelCB, bboard);
  XtManageChild (button);

/*-----------------------------------------------------------------------
 * --- Create a Aerosol button, attach it to "rowcol1", and realize it.
 *-----------------------------------------------------------------------
 */
  hv = 0;
  sd = 2;
  create_separator(rowcol1, &hv, &sd);

  n = 0;
  button = XmCreatePushButton(rowcol1,"Aerosol Selections", wargs, n);
  XtAddCallback (button, XmNactivateCallback, aerosol_optCB, bboard);
  XtManageChild (button);

/*-----------------------------------------------------------------------
 * --- Create the radioboxes and toggles for this RowColumn Widget
 *-----------------------------------------------------------------------
 */
```

```
  nc = 1;
  k = 0;
  for (i=0; i<2; i++)
  { create_separator(rowcol1, &hv, &sd);
    radio[i] = create_radiobox(rowcol1, &nc, cat_label[i]);

    for (j=0; j<nct[i]; k++, j++)
    { index[k] = 10*i + j;
      toggle[k] = create_togglebutton(radio[i], tog_label[k], &index[k],
                  modelCB);
    }

    if(mdl1 == 0)
    { if(i == 0)
        j = (int) mdl1_model;
      else if(i == 1)
        j = 7 + metrng_indx;
      XmToggleButtonSetState(toggle[j-1], TRUE, FALSE);
    }
  }

/*-----------------------------------------------------------------------
 * --- Create another RowColumn Widget, "rowcol2", within "rowcol3".
 *-----------------------------------------------------------------------
 */
  hv = 1;
  create_separator(rowcol3, &hv, &sd);

  n = 0;
  XtSetArg(wargs[n], XmNorientation, XmVERTICAL); n++;
  rowcol2 = XtCreateManagedWidget("Model_Options_(continued)",
            xmRowColumnWidgetClass, rowcol3, wargs, n);

/*-----------------------------------------------------------------------
 * --- Create the rest of the radioboxes and toggles
 *-----------------------------------------------------------------------
 */
  hv = 0;
  for (i=2; i<5; i++)
  { create_separator(rowcol2, &hv, &sd);
    radio[i] = create_radiobox(rowcol2, &nc, cat_label[i]);

    for (j=0; j<nct[i]; k++, j++)
    { index[k] = 10*i + j;
      toggle[k] = create_togglebutton(radio[i], tog_label[k], &index[k],
                  modelCB);
    }

    if(mdl1 == 0 && i == 2)
    { j = 10 + (int) mdl1_iseasn;
      XmToggleButtonSetState(toggle[j-1], TRUE, FALSE);
    }

    if(mdl2 == 0)
    { if(i == 3)
        j = 14 + (int) mdl2_ialb;
      else if(i == 4)
        j = 16 + (int) mdl2_ip;
      XmToggleButtonSetState(toggle[j-1], TRUE, FALSE);
    }
  }
```

```
/*-----------------------------------------------------------------------
 * --- Create the Scale in "rowcol4".
 *-----------------------------------------------------------------------
 */

  index[k] = 50;
  wid = 540;
  inc = 0;
  dec = 1;
  if(mdl2 == 0)
    val = 10.0*mdl2_tbound;
  else
    val = min;
  swid = 538;

  scale = create_scale(rowcol4, scale_label, &wid, &min, &max, &inc,
          &dec, &val, &swid, &index[k], modelCB);

  for (j=0; j<11; j++)
  { n = 0;
    XtSetArg (wargs[n], XmNwidth, 45); n++;
    label[j] = XmCreateLabel(scale, numb[j], wargs, n);
  }
  XtManageChildren(label, 11);
}     /*   end model_optCB()   */

/***********************************************************************
 *                          VOID AEROSOL_OPTCB
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  aerosol_optCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  model_optCB
 *=====================================================================
 *<Description>
 *    Presents the various Aerosol Options for IAERSL on the MDL1 card
 *=====================================================================
 *<Called routines>
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelCB               - removes the Rowcol Widget
 *    create_radiobox        - creates a Radiobox Widget
 *    create_togglebutton    - creates a Togglebutton Widget
 *    aerosolCB              - Set one of the input parameters of the
 *                             MDL1 card (IAERSL)
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void aerosol_optCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                   - the ID of the widget for which the
 *                             callback is registered
 *       c                   - the data passed to the routine
 *       call_data           - a pointer to the callback structure which
 *                             contains information on why the callback
 *                             occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
```

```
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void aerosol_optCB (Widget w, XtPointer c, XtPointer call_data)
{
  Widget radio, toggle[12], rowcol;
  int j;
  static int index[12], nc;
  static char cat_label[20] = "AFGL Aerosol Models";
  static char tog_label[12][30] = { "Default aerosol", "No aerosols",
              "Rural aerosol                >",
              "Urban aerosol                >",
              "Maritime aerosol             >",
              "Tropospheric aerosol         >",
              "Fog                          >", "Soot-like aerosols",
              "Oceanic component of maritime",
              "Background stratospheric",
              "Volcanic                     >", "Meteoric dust" };

/*---------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *---------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "AFGL_Aerosol_Models", cancelCB);

/*---------------------------------------------------------------------
 * --- Create the radiobox and toggles
 *---------------------------------------------------------------------
 */
  nc = 1;
  radio = create_radiobox(rowcol, &nc, cat_label);

  for (j=0; j<12; j++)
  { index[j] = j;
    toggle[j] = create_togglebutton(radio, tog_label[j], &index[j],
                aerosolCB);
  }

  if(mdl1 == 0)
  { if(mdl1_iaersl < 1)
      j = mdl1_iaersl + 1;
    else if(mdl1_iaersl == 1 || (mdl1_iaersl > 14 && mdl1_iaersl < 23))
      j = 2;
    else if(mdl1_iaersl == 2 || (mdl1_iaersl > 30 && mdl1_iaersl < 39))
      j = 3;
    else if(mdl1_iaersl == 4 || (mdl1_iaersl > 22 && mdl1_iaersl < 31))
      j = 4;
    else if(mdl1_iaersl == 6 || (mdl1_iaersl > 38 && mdl1_iaersl < 47))
      j = 5;
    else if(mdl1_iaersl > 10 && mdl1_iaersl < 15)
      j = 6;
    else if(mdl1_iaersl == 3)
      j = 7;
    else if(mdl1_iaersl == 5)
      j = 8;
    else if(mdl1_iaersl == 7)
      j = 9;
    else if(mdl1_iaersl > 7 && mdl1_iaersl < 10)
      j = 10;
    else if(mdl1_iaersl == 10)
      j = 11;
```

```
    XmToggleButtonSetState(toggle[j], TRUE, FALSE);
  }
}     /*   end aerosol_optCB()   */

/*********************************************************************
 *                          VOID SPECT1MENUCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  spect1menuCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_spectmenu
 *====================================================================
 *<Description>
 *    Creates the "Spectral Range Intervals" options selection radiobox
 *    menu widget.
 *====================================================================
 *<Called routines>
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelCB               - removes the Rowcol Widget
 *    create_separator       - creates a Separator Widget
 *    create_radiobox        - creates a Radiobox Widget
 *    create_togglebutton    - creates a Togglebutton Widget
 *    spect_optCB            - creates the various Spectral Range scales
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void spect1menuCB(Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                   - the ID of the widget for which the
 *                             callback is registered
 *       c                   - the data passed to the routine
 *       call_data           - a pointer to the callback structure which
 *                             contains information on why the callback
 *                             occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void spect1menuCB(Widget w, XtPointer c, XtPointer call_data)
{
  int *indx = (int *)c;
  Widget radio, toggle[5], rowcol;
  int n, j;
  static int index[5], hv, sd, nc;
  static char cat_label[2][34] = { "Wavenumber Interval (per cm)",
                                   "Wavelength Interval (micrometers)"};
  static char tog_label[10][24] = { "Visible:   8000 - 28000",
                                    "Near IR:   3000 - 13000",
                                    "Mid IR:    1200 -  5200",
                                    "Far IR:     500 -  1500",
                                    "2 Color IR: 600 -  3600",
                                    "Visible:    0.3 -  1.3",
                                    "Near IR:    0.7 -  3.2",
                                    "Mid IR:     2.0 -  7.0",
                                    "Far IR:     6.0 - 16.0",
```

```
                                        "2 Color IR: 3.0 - 13.0" };

/*-----------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-----------------------------------------------------------------------
 */
  rowcol = create_rowcol(w, "Spectral_Interval_Options", cancelCB);

/*-----------------------------------------------------------------------
 * --- Create the radioboxes and toggles
 *-----------------------------------------------------------------------
 */
  hv = 0;
  sd = 2;
  create_separator(rowcol, &hv, &sd);

  nc = 1;
  radio = create_radiobox(rowcol, &nc, cat_label[*indx]);

  for (j=0; j<5; j++)
  { n = index[j] = 5*(*indx) + j;
    toggle[j] = create_togglebutton(radio, tog_label[n], &index[j],
                spect_optCB);
  }

  if(wavn_indx >= 0 && wavn_indx < 5)
    XmToggleButtonSetState(toggle[wavn_indx], TRUE, FALSE);
}     /*   end spectlmenuCB()   */

/***********************************************************************
 *                           VOID AEROSOLCB
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  aerosolCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  aerosol_optCB
 *=====================================================================
 *<Description>
 *    Sets the input parameter IAERSL of the MDL1 card
 *=====================================================================
 *<Called routines>
 *    aero0                 - Presents the AFGL Rural, Urban, and
 *                            Tropospheric Aerosol Options.
 *    aero1                 - Presents the AFGL Maritime Aerosol Options
 *    aero2                 - Presents the AFGL Fog Aerosol Options.
 *    aero3                 - Presents the AFGL Volcanic Aerosol Options
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void aerosolCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - pointer to the data passed to the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
```

```
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void aerosolCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *n = (int *)c;
  static int index;

  if(mdl1 < 0)
  { mdl1 = 0;
    mdl1_iaersl = 0.0;
    mdl1_model = 6.0;
    mdl1_ivis = 1.0;
    mdl1_iseasn = 0.0;
    mdl1_ivulcn = 1.0;
  }

  index = -1;

  if((*n) < 2)
    mdl1_iaersl = (*n) - 1;
  else if((*n) < 4)
  { index = (*n) - 2;
    aero0(&index);
  }
  else if((*n) == 4)
    aero1();
  else if((*n) == 5)
  { index = 2;
    aero0(&index);
  }
  else if((*n) == 6)
    aero2();
  else if((*n) == 7)
    mdl1_iaersl = 3;
  else if((*n) == 8)
    mdl1_iaersl = 5;
  else if((*n) == 9)
    mdl1_iaersl = 7;
  else if((*n) == 10)
    aero3();
  else if((*n) == 11)
    mdl1_iaersl = 10;

  new_file = TRUE;
}     /*   end aerosolCB()    */

/***********************************************************************
 *                          VOID AERO0
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  aero0
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  aerosolCB
 *=====================================================================
 *<Description>
 *    Presents the AFGL Rural, Urban, and Tropospheric Aerosol Options.
 *=====================================================================
```

```
 *<Called routines>
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelCB               - removes the Rowcol Widget
 *    create_radiobox        - creates a Radiobox Widget
 *    create_togglebutton    - creates a Togglebutton Widget
 *    aeros0CB               - Sets the AFGL Aerosol Option
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void aero0(int *indx)
 *    Input:
 *       *indx                - index to indicate desired aerosol group
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void aero0(int *indx)
{
  Widget radio, toggle[9], rowcol;
  int j, ind;
  static int index[9], nc;
  static char cat_label[3][33] = { "AFGL Rural Aerosol Models",
                                   "AFGL Urban Aerosol Models",
                                   "AFGL Tropospheric Aerosol Models"};
  static char tog_label[9][8] =   { "General", " 0% RH", "50% RH",
      "70% RH", "80% RH", "90% RH", "95% RH",  "98% RH", "99% RH" };

  ind = *indx;

/*-------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "AFGL_Models", cancelCB);

/*-------------------------------------------------------------------
 * --- Create the radiobox and toggles
 *-------------------------------------------------------------------
 */
  nc = 1;
  radio = create_radiobox(rowcol, &nc, cat_label[ind]);

  for (j=0; j<9; j++)
  { index[j] = 10*ind + j;
    toggle[j] = create_togglebutton(radio, tog_label[j], &index[j],
                aeros0CB);
  }

  if(ind == 0)
  { if(mdl1_iaersl == 1)
      j = 0;
    else if(mdl1_iaersl > 14 && mdl1_iaersl < 23)
      j = mdl1_iaersl - 14;
    else
      j = -1;
  }
```

```
  else if(ind == 1)
  { if(mdl1_iaersl == 2)
      j = 0;
    else if(mdl1_iaersl > 30 && mdl1_iaersl < 39)
      j = mdl1_iaersl - 30;
    else
      j = -1;
  }
  else if(ind == 2)
  { if(mdl1_iaersl == 6)
      j = 0;
    else if(mdl1_iaersl > 38 && mdl1_iaersl < 47)
      j = mdl1_iaersl - 38;
    else
      j = -1;
  }

  if(j >= 0)
    XmToggleButtonSetState(toggle[j], TRUE, FALSE);
}     /*   end aero0()   */

/*********************************************************************
 *                          VOID AEROS0CB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  aeros0CB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  aero0, aero1, aero2, aero3
 *====================================================================
 *<Description>
 *    Sets the AFGL Aerosol Option
 *====================================================================
 *<Called routines>
 *    none
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void aeros0CB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - pointer to the data passed to the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void aeros0CB (Widget w, XtPointer c, XtPointer call_data)
{
  int *n = (int *)c;
  int ind, i;

  ind = (*n) / 10;
```

```
  i = (*n) % 10;

  if(ind == 0)
  { if(i == 0)
      mdl1_iaersl = 1;
    else
      mdl1_iaersl = i + 14;
  }
  else if(ind == 1)
  { if(i == 0)
      mdl1_iaersl = 2;
    else
      mdl1_iaersl = i + 30;
  }
  else if(ind == 2)
  { if(i == 0)
      mdl1_iaersl = 6;
    else
      mdl1_iaersl = i + 38;
  }
  else if(ind == 3)
  { if(i == 0)
      mdl1_iaersl = 4;
    else
      mdl1_iaersl = i + 22;
  }
  else if(ind == 4)
    mdl1_iaersl = i + 11;
  else if(ind == 5)
    mdl1_iaersl = i + 8;
}     /*   end aeros0CB()    */

/*********************************************************************
 *                          VOID AERO1
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  aero1
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  aerosolCB
 *====================================================================
 *<Description>
 *    Presents the AFGL Maritime Aerosol Options.
 *====================================================================
 *<Called routines>
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelCB               - removes the Rowcol Widget
 *    create_radiobox        - creates a Radiobox Widget
 *    create_togglebutton    - creates a Togglebutton Widget
 *    aeros0CB               - Sets the AFGL Aerosol Option
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void aero1(void)
 *    Input:
 *       None
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
```

```
 *=====================================================================
 *<End>
 ********************************************************************
 */
void aero1(void)
{
  Widget radio, toggle[9], rowcol;
  int j;
  static int index[9], nc;
  static char cat_label[29] = "AFGL Maritime Aerosol Models" ;
  static char tog_label[9][12] = {  "75% oceanic", " 0% RH", "50% RH",
      "70% RH", "80% RH", "90% RH", "95% RH",  "98% RH", "99% RH" };

/*-------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "AFGL_Maritime_Models", cancelCB);

/*-------------------------------------------------------------------
 * --- Create the radiobox and toggles
 *-------------------------------------------------------------------
 */
  nc = 1;
  radio = create_radiobox(rowcol, &nc, cat_label);

  for (j=0; j<9; j++)
  { index[j] = j + 30;
    toggle[j] = create_togglebutton(radio, tog_label[j], &index[j],
                aeros0CB);
  }

  if(mdl1_iaersl == 4)
    j = 0;
  else if(mdl1_iaersl > 22 && mdl1_iaersl < 31)
    j = mdl1_iaersl - 22;
  else
    j = -1;

  if(j >= 0)
    XmToggleButtonSetState(toggle[j], TRUE, FALSE);
}     /*   end aero1()   */

/*******************************************************************
 *                          VOID AERO2
 *******************************************************************
 *<Begin>
 *<Identification>          Name:  aero2
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  aerosolCB
 *===================================================================
 *<Description>
 *    Presents the AFGL Fog Aerosol Options.
 *===================================================================
 *<Called routines>
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelCB               - removes the Rowcol Widget
 *    create_radiobox        - creates a Radiobox Widget
 *    create_togglebutton    - creates a Togglebutton Widget
 *    aeros0CB               - Sets the AFGL Aerosol Option
 *===================================================================
```

```
 *<Parameters>
 *    Formal declaration:
 *       void aero2(void)
 *    Input:
 *       None
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 *********************************************************************
 */
void aero2(void)
{
  Widget radio, toggle[4], rowcol;
  int j;
  static int index[4], nc;
  static char cat_label[16] = "AFGL Fog Models" ;
  static char tog_label[4][25] = { "RRA Fog 1: Advection fog",
                                   "RRA Fog 2: Advection fog",
                                   "RRA Fog 3: Radiation fog",
                                   "RRA Fog 4: Radiation fog" };

/*-------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "AFGL_Fog_Models", cancelCB);

/*-------------------------------------------------------------------
 * --- Create the radiobox and toggles
 *-------------------------------------------------------------------
 */
  nc = 1;
  radio = create_radiobox(rowcol, &nc, cat_label);

  for (j=0; j<4; j++)
  { index[j] = j + 40;
    toggle[j] = create_togglebutton(radio, tog_label[j], &index[j],
                aeros0CB);
  }

  if(mdl1_iaersl > 10 && mdl1_iaersl < 15)
    j = mdl1_iaersl - 11;
  else
    j = -1;

  if(j >= 0)
    XmToggleButtonSetState(toggle[j], TRUE, FALSE);
}     /*   end aero2()   */

/*********************************************************************
 *                          VOID AERO3
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  aero3
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  aerosolCB
```

```
 *=====================================================================
 *<Description>
 *    Presents the AFGL Volcanic Aerosol Options.
 *=====================================================================
 *<Called routines>
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelCB               - removes the Rowcol Widget
 *    create_radiobox        - creates a Radiobox Widget
 *    create_togglebutton    - creates a Togglebutton Widget
 *    aeros0CB               - Sets the AFGL Aerosol Option
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void aero3(void)
 *    Input:
 *       None
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void aero3(void)
{
  Widget radio, toggle[2], rowcol;
  int j;
  static int index[2], nc;
  static char cat_label[26] = "AFGL Volcanic Dust Models" ;
  static char tog_label[2][6] = { "Aged", "Fresh" };

/*-------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "AFGL_Volcanic_Models", cancelCB);

/*-------------------------------------------------------------------
 * --- Create the radiobox and toggles
 *-------------------------------------------------------------------
 */
  nc = 1;
  radio = create_radiobox(rowcol, &nc, cat_label);

  for (j=0; j<2; j++)
  { index[j] = j + 50;
    toggle[j] = create_togglebutton(radio, tog_label[j], &index[j],
                aeros0CB);
  }

  if(mdl1_iaersl > 7 && mdl1_iaersl < 10)
    j = mdl1_iaersl - 8;
  else
    j = -1;

  if(j >= 0)
    XmToggleButtonSetState(toggle[j], TRUE, FALSE);
}     /*   end aero3()   */
```

```
/*********************************************************************
 *                          VOID MODELCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  modelCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  model_optCB
 *====================================================================
 *<Description>
 *    Sets some of the input parameters of the MDL1 and MDL2 cards
 *====================================================================
 *<Called routines>
 *    model2                - sets the input scales for the MDL3 card
 *    metrng_optCB          - sets the Meteorological Range Options
 *    albedo_chg            - sets up the menu for selecting the Albedo
 *                            for each Area.
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void modelCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - pointer to the data passed to the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void modelCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *n = (int *)c;
  int i, j, k, value;
  static int mr;

  XmScaleCallbackStruct * call_value =
                          (XmScaleCallbackStruct *) call_data;
  i = (*n)/10;                          /* RadioBox category        */
  j = (*n) % 10;                        /* Toggle button pushed     */
  value = call_value -> value;

  if((i < 3) && (mdl1 < 0))
  { mdl1 = 0;                           /* Set up default values    */
    mdl1_iaersl = 0.0;
    mdl1_model = 6.0;
    mdl1_ivis = 1.0;
    mdl1_iseasn = 0.0;
    mdl1_ivulcn = 1.0;
  }

  if((i >= 3) && (mdl2 < 0))
  { mdl2 = 0;                           /* Set up default values    */
    mdl2_sn = 0.8;
```

```
    mdl2_tbound = 288.2;
    mdl2_ialb = -1.0;
    mdl2_ip = 0.0;
    cur_ialb = mdl2_ialb;
  }

  if(i == 0)                                /* Temperature Profile Model*/
  { mdl1_model = j+1;
    if(mdl1_model == 7)                     /* User Defined Temp Profile*/
      model2();                             /* Get the User Defined Prof*/
    else
      mdl3 = -1;                            /* No MLD3 card needed      */
  }
  else if(i == 1)                           /* Meteorological Range Opt */
  { mr = j+1;                               /* Select appropriate scale */
    metrng_optCB(NULL, &mr, NULL);          /* Get Met Range            */
  }
  else if(i == 2)                           /* Tropospheric Profile Modl*/
    mdl1_iseasn = j;
  else if(i == 3)                           /* Albedo Type Selection    */
  { cur_ialb = mdl2_ialb;
    mdl2_ialb = j-1;                        /* Albedo Type Choice       */
    if(cur_ialb != mdl2_ialb)               /* If changed,              */
      albedo_chg();                         /*   update albedo areas    */
  }
  else if(i == 4)                           /* Aerosol Profile Printout */
    mdl2_ip = j;
  else if(i == 5)                           /* Surface Temperature (K)  */
    mdl2_tbound = 0.1 * (float)value;

  new_file = TRUE;                          /* inputs changed           */
}     /*   end modelCB()   */

/*********************************************************************
 *                          VOID MODEL2
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  model2
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  modelCB
 *====================================================================
 *<Description>
 *    Sets the input scales of the MDL3 card
 *====================================================================
 *<Called routines>
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelCB               - removes the Rowcol Widget
 *    create_separator       - creates a Separator Widget
 *    create_scale           - creates a Scale Widget
 *    model3CB               - gets the input parameters on the MDL3 card
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void model2( void )
 *    Input:
 *       None
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
```

```
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void model2( void)
{
  Widget scale[6], label[11], rowcol;
  Arg wargs[10];
  int n, i, j;
  static int index[6], hv, sd, wid, min, max, inc, dec, val[6], swid;
  static char temp_label[6][24] = { "Temperature (K) at 5 km",
                                    "Temperature (K) at 4 km",
                                    "Temperature (K) at 3 km",
                                    "Temperature (K) at 2 km",
                                    "Temperature (K) at 1 km",
                                    "Temperature (K) at 0 km" };
  static char temp[11][4] = { "220", "230", "240", "250", "260", "270",
                              "280", "290", "300", "310", "320" };

/*---------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *---------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Temperature_Profile", cancelCB);

/*---------------------------------------------------------------------
 * --- Create the Scales
 *---------------------------------------------------------------------
 */
  hv = 0;
  sd = 2;
  for (i=0; i<6; i++)
  { create_separator(rowcol, &hv, &sd);

    index[i] = i;
    wid = 560;
    min = 2200;
    max = 3200;
    inc = 0;
    dec = 1;
    if(mdl3 == 0)
      val[i] = 10.0*mdl3_t[5-i];
    else
      val[i] = min;
    swid = 538;

    scale[i] = create_scale(rowcol, temp_label[i], &wid, &min, &max,
               &inc, &dec, &val[i], &swid, &index[i], model3CB);

    if(i == 0)
    { for (j=0; j<11; j++)
      { n = 0;
        XtSetArg (wargs[n], XmNwidth, 45); n++;
        label[j] = XmCreateLabel(scale[0], temp[j], wargs, n);
      }
      XtManageChildren(label, 11);
    }
  }
}     /*   end model2()   */

/***********************************************************************
```

```
 *                            VOID MODEL3CB
 ***********************************************************************
 *<Begin>
 *<Identification>           Name:  model3CB
 *                           Type:  C void
 *                       Filename:  visual.c
 *                         Parent:  model2
 *=====================================================================
 *<Description>
 *    Gets the input parameters on the MDL3 card
 *=====================================================================
 *<Called routines>
 *    none
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void model3CB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                   - the ID of the widget for which the
 *                             callback is registered
 *       c                   - pointer to the data passed to the routine
 *       call_data           - a pointer to the callback structure which
 *                             contains information on why the callback
 *                             occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void model3CB (Widget w, XtPointer c, XtPointer call_data)
{
  int *n = (int *)c;
  int value;
  XmScaleCallbackStruct * call_value =
                            (XmScaleCallbackStruct *) call_data;

  value = call_value -> value;

  mdl3_t[5 - (*n)] = 0.1 * (float)value;

  mdl3 = 0;
  new_file = TRUE;
}     /*   end model3CB()    */

/***********************************************************************
 *                            VOID SPECT_OPTCB
 ***********************************************************************
 *<Begin>
 *<Identification>           Name:  spect_optCB
 *                           Type:  C void
 *                       Filename:  visual.c
 *                         Parent:  create_spectlmenu
 *=====================================================================
 *<Description>
 *    Selects the various Spectral Range Options for BLIRB
 *=====================================================================
 *<Called routines>
```

```
 *    create_rowcol          - creates a Rowcol Widget
 *    spect1CB               - removes the rowcol widget
 *    create_separator       - creates a Separator Widget
 *    create_scale           - creates a Scale Widget
 *    spectCB                - Gets the input parameters of the WAVN card
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void spect_optCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                   - the ID of the widget for which the
 *                             callback is registered
 *       c                   - the data passed to the routine
 *       call_data           - a pointer to the callback structure which
 *                             contains information on why the callback
 *                             occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void spect_optCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *indx = (int *)c;
  Widget rowcol, label[3][7], scale[3];
  Arg wargs[10];
  int n, i, j, k, ii, ind, v[3];
  static int index[6], hv, sd, wid, inc, dec[3], val[3], swid;
  static char scale_label[6][33] =
                        { "Lowest Wavenumber (per cm)",
                          "Highest Wavenumber (per cm)",
                          "Number of Wavenumber Intervals",
                          "Lowest Wavelength (micrometers)",
                          "Highest Wavelength (micrometers)",
                          "Number of Wavelength Intervals"
                        };
  static char num[10][6][6] = { {" 8000", "12000", "16000", "20000",
                                 "24000", "28000" },
                                {" 3000", " 5000", " 7000", " 9000",
                                 "11000", "13000" },
                                {"1200", "2000", "2800", "3600", "4400",
                                 "5200" },
                                {" 500", " 700", "900", "1100", "1300",
                                 "1500" },
                                {" 600", "1200", "1800", "2400", "3000",
                                 "3600" },

                                {"0.3", "0.5", "0.7", "0.9", "1.1",
                                 "1.3" },
                                {"0.7", "1.2", "1.7", "2.2", "2.7",
                                 "3.2" },
                                {"2.0", "3.0", "4.0", "5.0", "6.0",
                                 "7.0" },
                                {" 6.0", " 8.0", "10.0", "12.0", "14.0",
                                 "16.0" },
                                {" 3.0", " 5.0", " 7.0", " 9.0", "11.0",
                                 "13.0" }
```

```
                                    };
  static char intr[7][3] = { " 0", " 5", "10", "15", "20", "25", "30" };
  static int minw[10] = { 8000, 3000, 1200, 500, 600, 3000, 7000, 20000,
                          60000, 30000 };
  static int maxw[10] = { 28000, 13000, 5200, 1500, 3600, 13000, 32000,
                          70000, 160000, 130000 };
  static char numb[3][6][6];
  static int min[3], max[3];

  ind = *indx;

  for(j=0; j<6; j++)
  { strcpy(numb[0][j], num[ind][j]);
    strcpy(numb[1][j], num[ind][j]);
  }
  min[0] = min[1] = minw[ind];
  max[0] = max[1] = maxw[ind];

  for(j=0; j<7; j++)
    strcpy(numb[2][j], intr[j]);
  min[2] = 0;
  max[2] = 30;

  if(ind < 5)
    ii = 0;
  else
    ii = 3;

  if(wavn == 0)
  { if(ind < 5)
    { v[0] = wavn_v1;
      v[1] = wavn_v2;
      v[2] = wavn_dv;
    }
    else
    { v[0] = 100000000.0 / wavn_v2;
      v[1] = 100000000.0 / wavn_v1;
      v[2] = wavn_dv;
    }
  }
  else
  { v[0] = min[0];
    v[1] = min[1];
    v[2] = min[2];
  }

/*-----------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-----------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Spectral_Options", spect1CB);

/*-----------------------------------------------------------------------
 * --- Create the Scale
 *-----------------------------------------------------------------------
 */
  hv = 0;
  sd = 2;
  for (i=0; i<3; i++)
  { create_separator(rowcol, &hv, &sd);

    index[i] = 10*ind + i;
```

```
    wid = 460;
    inc = 0;
    if(ind >= 5 && i < 2)
      dec[i] = 4;
    else
      dec[i] = 0;
    swid = 438;

    if(wavn == 0)
    { if(v[i] < min[i])
        val[i] = min[i];
      else if(v[i] > max[i])
        val[i] = max[i];
      else
        val[i] = v[i];
    }
    else
      val[i] = min[i];

    scale[i] = create_scale(rowcol, scale_label[ii+i], &wid, &min[i],
               &max[i], &inc, &dec[i], &val[i], &swid, &index[i],
               spectCB);

    if(i < 2)
      k = 6;
    else
      k = 7;

    for (j=0; j<k; j++)
    { n = 0;
      XtSetArg (wargs[n], XmNwidth, 90); n++;
      label[i][j] = XmCreateLabel(scale[i], numb[i][j], wargs, n);
    }
    XtManageChildren(label[i], k);
  }
}     /*   end spect_optCB()   */

/*********************************************************************
 *                            VOID SPECTCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  spectCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  spect_optCB
 *====================================================================
 *<Description>
 *    Gets the input parameters for the WAVN card
 *====================================================================
 *<Called routines>
 *    none
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void spectCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - pointer to the data passed to the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
```

```
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void spectCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *n = (int *)c;
  static float interval;
  int i, j, value;
  XmScaleCallbackStruct * call_value =
                            (XmScaleCallbackStruct *) call_data;

  value = call_value -> value;
  i = (*n) / 10;
  j = (*n) % 10;

  if(wavn < 0)
  { wavn = 0;
    wavn_v1 = 0.0;
    wavn_v2 = 0.0;
    wavn_dv = 0.0;
  }

  if(j == 0)
  { if(i < 5)
      wavn_v1 = value;
    else
      wavn_v2 = 100000000.0 / (float)value;
  }
  else if(j == 1)
  { if(i < 5)
      wavn_v2 = value;
    else
      wavn_v1 = 100000000.0 / (float)value;
  }
  else if(j == 2)
  { if(value == 0)
      value++;
    wavn_dv = value;
  }

  wavn_indx = i % 5;

  interval = (wavn_v2 - wavn_v1) / wavn_dv;
  new_file = TRUE;
}     /*   end spectCB()    */

/********************************************************************
 *                          VOID SPECT1CB
 ********************************************************************
 *<Begin>
 *<Identification>          Name:  spect1CB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  spect_optCB
 *=====================================================================
```

```
 *<Description>
 *    Removes the calling widget after checking the wavenumber inputs.
 *=====================================================================
 *<Called routines>
 *    drawscene              - Plots the 3-D BLIRB grid points, albedo
 *                             areas, aerosol regions, and the output
 *                             flux.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void spect1CB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                   - the ID of the widget for which the
 *                             callback is registered
 *       c                   - the widget to be removed
 *       call_data           - a pointer to the callback structure which
 *                             contains information on why the callback
 *                             occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 *********************************************************************
 */
void spect1CB(Widget w, XtPointer c, XtPointer call_data)
{
  float temp;

  if(wavn_v2 < wavn_v1)
  { temp = wavn_v2;
    wavn_v2 = wavn_v1;
    wavn_v1 = temp;
  }

  XtUnmanageChild((Widget)c);

  drawscene();
}     /* end spect1CB() */

/*********************************************************************
 *                          VOID AREA_OPTCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  area_optCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_areaImenu
 *=====================================================================
 *<Description>
 *    Sets up the scales for selecting the area dimensions
 *=====================================================================
 *<Called routines>
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelaCB              - removes the rowcol widget
 *    create_separator       - creates a Separator Widget
 *    create_scale           - creates a Scale Widget
 *    areaCB                 - Gets the dimensions for an area
 *=====================================================================
```

```
 *<Parameters>
 *    Formal declaration:
 *       void area_optCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                    - the ID of the widget for which the
 *                              callback is registered
 *       c                    - the data passed to the routine
 *       call_data            - a pointer to the callback structure which
 *                              contains information on why the callback
 *                              occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void area_optCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *indx = (int *)c;
  Widget rowcol, label[6][7], scale[2];
  Arg wargs[10];
  int n, i, j, k, ind, maxx;
  static int index[2], hv, sd, wid, inc, dec, val[2], swid;
  static char scale_label[2][27]={ "Length in X Direction (km)",
                                   "Length in Y Direction (km)" };
  static char numb[2][6][6];
  static int min[2] = { 0, 0 };
  static int max[2];

  for (j=0; j<2; j++)
  { maxx = regn_rh[j][0] - regn_rl[j][0] + 0.1;
    max[j] = 10 * maxx;
    for (i=0; i<6; i++)
      sprintf(numb[j][i], "%5.2f",  0.2 * (float)(i * maxx));
  }

  ind = *indx;

  if(mdl2 < 0)
  { mdl2 = 0;
    mdl2_sn = 0.8;
    mdl2_tbound = 288.2;
    mdl2_ialb = -1;
    mdl2_ip = 0;
    cur_ialb = mdl2_ialb;
  }

/*-------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Area_Dimensions", cancelaCB);

/*-------------------------------------------------------------------
 * --- Create the Scales
 *-------------------------------------------------------------------
 */
  hv = 0;
```

```
  sd = 2;

  for (i=0; i<2; i++)
  { create_separator(rowcol, &hv, &sd);

    index[i] = 10*ind + i;
    wid = 560;
    inc = 1;
    dec = 1;
    swid = 538;

    if(i == 0)
    { if(ind <= area)
        val[i] = 10.0 * (area_ahx[ind] - area_alx[ind]);
      else
        val[i] = min[0];
    }
    else if(i == 1)
    { if(ind <= area)
        val[i] = 10.0 * (area_ahy[ind] - area_aly[ind]);
      else
        val[i] = min[1];
    }

    scale[i] = create_scale(rowcol, scale_label[i], &wid, &min[i],
               &max[i], &inc, &dec, &val[i], &swid, &index[i], areaCB);

    for (j=0; j<6; j++)
    { n = 0;
      XtSetArg (wargs[n], XmNwidth, 90); n++;
      label[i][j] = XmCreateLabel(scale[i], numb[i][j], wargs, n);
    }
    XtManageChildren(label[i], 6);
  }
}     /*   end area_optCB()   */

/*********************************************************************
 *                          VOID AREACB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  areaCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  area_optCB
 *====================================================================
 *<Description>
 *    Gets the albedo area dimensions
 *====================================================================
 *<Called routines>
 *    area_fix              - rectifies the Albedo Area location
 *    create_menubar        - creates the menubar for selecting the
 *                            various options
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void areaCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - pointer to the data passed to the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
```

```
 *                                occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void areaCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *n = (int *)c;
  int i, j, value;
  static Boolean pass = FALSE;
  static Boolean mask[2], go;
  XmScaleCallbackStruct * call_value =
                              (XmScaleCallbackStruct *) call_data;

  value = call_value -> value;
  i = (*n)/10;
  j = (*n) % 10;

  if((i > area) && !pass)
  { area_alx[i] = area_ahx[i] = area_aly[i] = area_ahy[i] =
    area_iamtl[i] = 0.0;
    pass = TRUE;
    mask[0] = mask[1] = FALSE;
  }

  if(j == 0)
  { area_ahx[i] = area_alx[i] + 0.1 * value;
    if(area_ahx[i] > regn_rh[0][0])
      area_ahx[i] = regn_rh[0][0];
  }
  else if(j == 1)
  { area_ahy[i] = area_aly[i] + 0.1 * value;
    if(area_ahy[i] > regn_rh[1][0])
      area_ahy[i] = regn_rh[1][0];
  }

  cur_area = i;
  area_fix();

  if(i > area)
    mask[j] = TRUE;
  go = mask[0] && mask[1];

  if(go)
  { area++;
    pass = FALSE;
    mask[0] = mask[1] = FALSE;
    in_changea = TRUE;

    if(mdl2_ialb < 0)
    { albd++;
      area_iamtl[area] = albd_lalb[albd] = area+1;
      albd_falb[albd] = background_albedo;
    }
    else
      area_iamtl[area] = 0.0;
```

```
    XtUnmanageChild (menu);
    menu = create_menubar(form);
  }

  new_file = TRUE;
}     /*   end areaCB()    */

/*********************************************************************
 *                          VOID AREA_FIX
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  area_fix
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  inputCB, areaCB, cancelmeCB, regnCB
 *====================================================================
 *<Description>
 *    Rectifies the location of an Albedo Area.
 *====================================================================
 *<Called routines>
 *    none
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void area_fix(void);
 *    Input:
 *       None
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void area_fix (void)
{
  int j, k, kk;
  float dif, del;

  del = area_ahx[cur_area] - area_alx[cur_area];
  for (dif = 100000.0, kk = -1, k=0; k<=num_grid_pts[0]; k++)
  { if( fabs(area_alx[cur_area] - axis_pts[0][k]) < dif )
    { dif = fabs(area_alx[cur_area] - axis_pts[0][k]);
      kk = k;
    }
  }
  area_alx[cur_area] = axis_pts[0][kk];
  area_ahx[cur_area] = area_alx[cur_area] + del;

  for (dif = 100000.0, kk = -1, k=0; k<=num_grid_pts[0]; k++)
  { if( fabs(area_ahx[cur_area] - axis_pts[0][k]) < dif )
    { dif = fabs(area_ahx[cur_area] - axis_pts[0][k]);
      kk = k;
    }
  }
  area_ahx[cur_area] = axis_pts[0][kk];

  del = area_ahy[cur_area] - area_aly[cur_area];
  for (dif = 100000.0, kk = -1, k=0; k<=num_grid_pts[1]; k++)
```

```
  { if( fabs(area_aly[cur_area] - axis_pts[1][k]) < dif )
    { dif = fabs(area_aly[cur_area] - axis_pts[1][k]);
      kk = k;
    }
  }
  area_aly[cur_area] = axis_pts[1][kk];
  area_ahy[cur_area] = area_aly[cur_area] + del;

  for (dif = 100000.0, kk = -1, k=0; k<=num_grid_pts[1]; k++)
  { if( fabs(area_ahy[cur_area] - axis_pts[1][k]) < dif )
    { dif = fabs(area_ahy[cur_area] - axis_pts[1][k]);
      kk = k;
    }
  }
  area_ahy[cur_area] = axis_pts[1][kk];
}     /*   end area_fix()   */

/*********************************************************************
 *                          VOID MESHXMENUCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  meshxmenuCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_meshmenu
 *====================================================================
 *<Description>
 *    Sets up the scales for selecting the X Mesh dimensions
 *====================================================================
 *<Called routines>
 *    create_bboard         - creates a BulletinBoard Widget
 *    cancelmeCB            - removes the BulletinBoard Widget
 *    create_separator      - creates a Separator Widget
 *    create_scale          - creates a Scale Widget
 *    meshCB                - gets the dimensions for a mesh
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void meshxmenuCB(Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - the data passed to the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void meshxmenuCB(Widget w, XtPointer c, XtPointer call_data)
{
  Widget rowcol, rowcol1, rowcol2, bboard, button, label[2*ISM][6],
         scale[2*ISM];
  Arg wargs[10];
  int n, i, j, k, ind, maxx;
```

```
  static int index[2*ISM], hv, sd, wid, inc, dec, val[2*ISM], swid;
  static char scale_label[2*ISM][38];
  static char numb[2*ISM][6][6];
  static int min[2*ISM], max[2*ISM];
  static char but_label[15];

  sprintf(but_label, "(Total <= %d)", MAXMX);

  ind = 0;

  if(mes[ind] < (ISM - 1))
    k = mes[ind] + 1;
  else
    k = mes[ind];

  for (j=0; j<=k; j++)
  { min[j] = min[j+k+1] = 0;
    maxx = mes_ms[ind][mes[ind]] + 0.1;
    max[j] = 10 * maxx;
    max[j+k+1] = 40;

    for (i=0; i<6; i++)
    { sprintf(numb[j][i], "%5.2f", 0.2 * (float)(i * maxx));
      sprintf(numb[j+k+1][i], "%d", 8*i);
    }

    if(j <= mes[ind])
      sprintf(scale_label[j], "End of X Mesh %d Interval (km)",
              j+1);
    else
      sprintf(scale_label[j], "End of New X Mesh Interval (km)");
    sprintf(scale_label[j+k+1], "Num Subintervals");
  }

/*-----------------------------------------------------------------------
 * --- Create a Bulletin Board Widget
 *-----------------------------------------------------------------------
 */
  bboard = create_bboard(menu, "X Mesh Options");

/*-----------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-----------------------------------------------------------------------
 */
  n = 0;
  XtSetArg(wargs[n], XmNorientation, XmHORIZONTAL); n++;
  rowcol = XtCreateManagedWidget("X_Mesh_Options",
           xmRowColumnWidgetClass, bboard, wargs, n);

/*-----------------------------------------------------------------------
 * --- Create a RowColumn Widget within the other RowColumn Widget
 *-----------------------------------------------------------------------
 */
  n = 0;
  XtSetArg(wargs[n], XmNorientation, XmVERTICAL); n++;
  rowcol1 = XtCreateManagedWidget("X_Mesh_Options",
            xmRowColumnWidgetClass, rowcol, wargs, n);

/*-----------------------------------------------------------------------
 * --- Create a finishbutton, attach it to "rowcol1", and realize it.
 *-----------------------------------------------------------------------
 */
```

```
  n = 0;
  button = XmCreatePushButton(rowcol1,"Finished with Selections", wargs,
           n);
  XtAddCallback (button, XmNactivateCallback, cancelmeCB, bboard);
  XtManageChild (button);

/*-----------------------------------------------------------------------
 * --- Create the Mesh End Point Scales
 *-----------------------------------------------------------------------
 */
  hv = 0;
  sd = 2;

  for (i=0; i<=k; i++)
  { create_separator(rowcol1, &hv, &sd);

    index[i] = 100*ind + i;
    wid = 460;
    dec = 1;
    swid = 438;
    inc = 1;

    if(i <= mes[ind])
      val[i] = 10.0 * mes_ms[ind][i];
    else
      val[i] = min[i];

    scale[i] = create_scale(rowcol1, scale_label[i], &wid, &min[i],
               &max[i], &inc, &dec, &val[i], &swid, &index[i], meshCB);

    for (j=0; j<6; j++)
    { n = 0;
      XtSetArg (wargs[n], XmNwidth, 90); n++;
      label[i][j] = XmCreateLabel(scale[i], numb[i][j], wargs, n);
    }
    XtManageChildren(label[i], 6);
  }

/*-----------------------------------------------------------------------
 * --- Create another RowColumn Widget within the first
 *-----------------------------------------------------------------------
 */
  hv = 1;
  create_separator(rowcol, &hv, &sd);

  n = 0;
  XtSetArg(wargs[n], XmNorientation, XmVERTICAL); n++;
  rowcol2 = XtCreateManagedWidget("X_Mesh_Options_(continued)",
            xmRowColumnWidgetClass, rowcol, wargs, n);

/*-----------------------------------------------------------------------
 * --- Create a info_button, attach it to "rowcol2", and realize it.
 *-----------------------------------------------------------------------
 */
  n = 0;
  button = XmCreatePushButton(rowcol2, but_label, wargs, n);
  XtManageChild (button);

/*-----------------------------------------------------------------------
 * --- Create the Subinterval Count Scales
 *-----------------------------------------------------------------------
 */
```

```
  hv = 0;

  for (i=k+1; i<=2*k+1; i++)
  { create_separator(rowcol2, &hv, &sd);

    index[i] = 100*ind + i + 50-k-1;
    wid = 160;
    dec = 0;
    swid = 138;
    inc = 1;

    if(i-k-1 <= mes[ind])
    { val[i] = mes_mh[ind][i-k-1];
      if(val[i] > max[i])
        val[i] = max[i];
    }
    else
      val[i] = min[i];

    scale[i] = create_scale(rowcol2, scale_label[i], &wid, &min[i],
               &max[i], &inc, &dec, &val[i], &swid, &index[i], meshCB);

    for (j=0; j<2; j++)
    { n = 0;
      XtSetArg (wargs[n], XmNwidth, 90); n++;
      label[i][j] = XmCreateLabel(scale[i], numb[i][5*j], wargs, n);
    }
    XtManageChildren(label[i], 2);
  }
}     /*   end meshxmenuCB   */

/*********************************************************************
 *                          VOID MESHYMENUCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  meshymenuCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_meshmenu
 *====================================================================
 *<Description>
 *    Sets up the scales for selecting the Y Mesh dimensions
 *====================================================================
 *<Called routines>
 *    create_bboard          - creates a BulletinBoard Widget
 *    cancelmeCB             - removes the BulletinBoard Widget
 *    create_separator       - creates a Separator Widget
 *    create_scale           - creates a Scale Widget
 *    meshCB                 - gets the dimensions for a mesh
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void meshymenuCB(Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                   - the ID of the widget for which the
 *                             callback is registered
 *       c                   - the data passed to the routine
 *       call_data           - a pointer to the callback structure which
 *                             contains information on why the callback
 *                             occurred
 *    Output:
 *       None
```

```
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void meshymenuCB(Widget w, XtPointer c, XtPointer call_data)
{
  Widget rowcol, rowcol1, rowcol2, bboard, button, label[2*ISM][6],
         scale[2*ISM];
  Arg wargs[10];
  int n, i, j, k, ind, maxx;
  static int index[2*ISM], hv, sd, wid, inc, dec, val[2*ISM], swid;
  static char scale_label[2*ISM][38];
  static char numb[2*ISM][6][6];
  static int min[2*ISM], max[2*ISM];
  static char but_label[15];

  sprintf(but_label, "(Total <= %d)", MAXMY);

  ind = 1;

  if(mes[ind] < (ISM - 1))
    k = mes[ind] + 1;
  else
    k = mes[ind];

  for (j=0; j<=k; j++)
  { min[j] = min[j+k+1] = 0;
    maxx = mes_ms[ind][mes[ind]] + 0.1;
    max[j] = 10 * maxx;
    max[j+k+1] = 40;

    for (i=0; i<6; i++)
    { sprintf(numb[j][i], "%5.2f", 0.2 * (float)(i * maxx));
      sprintf(numb[j+k+1][i], "%d", 8*i);
    }

    if(j <= mes[ind])
      sprintf(scale_label[j], "End of Y Mesh %d Interval (km)",
              j+1);
    else
      sprintf(scale_label[j], "End of New Y Mesh Interval (km)");
    sprintf(scale_label[j+k+1], "Num Subintervals");
  }

/*---------------------------------------------------------------------
 * --- Create a Bulletin Board Widget
 *---------------------------------------------------------------------
 */
  bboard = create_bboard(menu, "Y Mesh Options");

/*---------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *---------------------------------------------------------------------
 */
  n = 0;
  XtSetArg(wargs[n], XmNorientation, XmHORIZONTAL); n++;
  rowcol = XtCreateManagedWidget("Y_Mesh_Options",
           xmRowColumnWidgetClass, bboard, wargs, n);
```

```
/*-----------------------------------------------------------------------
 * --- Create a RowColumn Widget within the other RowColumn Widget
 *-----------------------------------------------------------------------
 */
  n = 0;
  XtSetArg(wargs[n], XmNorientation, XmVERTICAL); n++;
  rowcol1 = XtCreateManagedWidget("Y_Mesh_Options",
            xmRowColumnWidgetClass, rowcol, wargs, n);

/*-----------------------------------------------------------------------
 * --- Create a finishbutton, attach it to "rowcol1", and realize it.
 *-----------------------------------------------------------------------
 */
  n = 0;
  button = XmCreatePushButton(rowcol1,"Finished with Selections", wargs,
           n);
  XtAddCallback (button, XmNactivateCallback, cancelmeCB, bboard);
  XtManageChild (button);

/*-----------------------------------------------------------------------
 * --- Create the Mesh End Point Scales
 *-----------------------------------------------------------------------
 */
  hv = 0;
  sd = 2;

  for (i=0; i<=k; i++)
  { create_separator(rowcol1, &hv, &sd);

    index[i] = 100*ind + i;
    wid = 460;
    dec = 1;
    swid = 438;
    inc = 1;

    if(i <= mes[ind])
      val[i] = 10.0 * mes_ms[ind][i];
    else
      val[i] = min[i];

    scale[i] = create_scale(rowcol1, scale_label[i], &wid, &min[i],
               &max[i], &inc, &dec, &val[i], &swid, &index[i], meshCB);

    for (j=0; j<6; j++)
    { n = 0;
      XtSetArg (wargs[n], XmNwidth, 90); n++;
      label[i][j] = XmCreateLabel(scale[i], numb[i][j], wargs, n);
    }
    XtManageChildren(label[i], 6);
  }

/*-----------------------------------------------------------------------
 * --- Create another RowColumn Widget within the first
 *-----------------------------------------------------------------------
 */
  hv = 1;
  create_separator(rowcol, &hv, &sd);

  n = 0;
  XtSetArg(wargs[n], XmNorientation, XmVERTICAL); n++;
  rowcol2 = XtCreateManagedWidget("Y_Mesh_Options_(continued)",
            xmRowColumnWidgetClass, rowcol, wargs, n);
```

```
/*-----------------------------------------------------------------------
 * --- Create a info_button, attach it to "rowcol2", and realize it.
 *-----------------------------------------------------------------------
 */
  n = 0;
  button = XmCreatePushButton(rowcol2, but_label, wargs, n);
  XtManageChild (button);

/*-----------------------------------------------------------------------
 * --- Create the Subinterval Count Scales
 *-----------------------------------------------------------------------
 */
  hv = 0;

  for (i=k+1; i<=2*k+1; i++)
  { create_separator(rowcol2, &hv, &sd);

    index[i] = 100*ind + i + 50-k-1;
    wid = 160;
    dec = 0;
    swid = 138;
    inc = 1;

    if(i-k-1 <= mes[ind])
    { val[i] = mes_mh[ind][i-k-1];
      if(val[i] > max[i])
        val[i] = max[i];
    }
    else
      val[i] = min[i];

    scale[i] = create_scale(rowcol2, scale_label[i], &wid, &min[i],
               &max[i], &inc, &dec, &val[i], &swid, &index[i], meshCB);

    for (j=0; j<2; j++)
    { n = 0;
      XtSetArg (wargs[n], XmNwidth, 90); n++;
      label[i][j] = XmCreateLabel(scale[i], numb[i][5*j], wargs, n);
    }
    XtManageChildren(label[i], 2);
  }
}     /*   end meshymenuCB   */

/***********************************************************************
 *                          VOID MESHZMENUCB
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  meshzmenuCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_meshmenu
 *=====================================================================
 *<Description>
 *    Sets up the scales for selecting the Z Mesh dimensions
 *=====================================================================
 *<Called routines>
 *    create_bboard         - creates a BulletinBoard Widget
 *    cancelmeCB            - removes the BulletinBoard Widget
 *    create_separator      - creates a Separator Widget
 *    create_scale          - creates a Scale Widget
 *    meshCB                - gets the dimensions for a mesh
 *=====================================================================
```

```
 *<Parameters>
 *    Formal declaration:
 *       void meshzmenuCB(Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                    - the ID of the widget for which the
 *                              callback is registered
 *       c                    - the data passed to the routine
 *       call_data            - a pointer to the callback structure which
 *                              contains information on why the callback
 *                              occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void meshzmenuCB(Widget w, XtPointer c, XtPointer call_data)
{
  Widget rowcol, rowcol1, rowcol2, bboard, button, label[2*ISM][6],
         scale[2*ISM];
  Arg wargs[10];
  int n, i, j, k, ind, maxx;
  static int index[2*ISM], hv, sd, wid, inc, dec, val[2*ISM], swid;
  static char scale_label[2*ISM][38];
  static char numb[2*ISM][6][6];
  static int min[2*ISM], max[2*ISM];
  static char but_label[15];

  sprintf(but_label, "(Total <= %d)", MAXMZ);

  ind = 2;

  if(mes[ind] < (ISM - 1))
    k = mes[ind] + 1;
  else
    k = mes[ind];

  for (j=0; j<=k; j++)
  { min[j] = min[j+k+1] = 0;
    maxx = mes_ms[ind][mes[ind]] + 0.1;
    max[j] = 10 * maxx;
    max[j+k+1] = 40;

    for (i=0; i<6; i++)
    { sprintf(numb[j][i], "%5.2f", 0.2 * (float)(i * maxx));
      sprintf(numb[j+k+1][i], "%d", 8*i);
    }

    if(j <= mes[ind])
      sprintf(scale_label[j], "End of Z Mesh %d Interval (km)",
              j+1);
    else
      sprintf(scale_label[j], "End of New Z Mesh Interval (km)");
    sprintf(scale_label[j+k+1], "Num Subintervals");
  }

/*-------------------------------------------------------------------
 * --- Create a Bulletin Board Widget
```

```
 *-----------------------------------------------------------------------
 */
  bboard = create_bboard(menu, "Z Mesh Options");

/*-----------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-----------------------------------------------------------------------
 */
  n = 0;
  XtSetArg(wargs[n], XmNorientation, XmHORIZONTAL); n++;
  rowcol = XtCreateManagedWidget("Z_Mesh_Options",
           xmRowColumnWidgetClass, bboard, wargs, n);

/*-----------------------------------------------------------------------
 * --- Create a RowColumn Widget within the other RowColumn Widget
 *-----------------------------------------------------------------------
 */
  n = 0;
  XtSetArg(wargs[n], XmNorientation, XmVERTICAL); n++;
  rowcol1 = XtCreateManagedWidget("Z_Mesh_Options",
            xmRowColumnWidgetClass, rowcol, wargs, n);

/*-----------------------------------------------------------------------
 * --- Create a finishbutton, attach it to "rowcol1", and realize it.
 *-----------------------------------------------------------------------
 */
  n = 0;
  button = XmCreatePushButton(rowcol1,"Finished with Selections", wargs,
           n);
  XtAddCallback (button, XmNactivateCallback, cancelmeCB, bboard);
  XtManageChild (button);

/*-----------------------------------------------------------------------
 * --- Create the Mesh End Point Scales
 *-----------------------------------------------------------------------
 */
  hv = 0;
  sd = 2;

  for (i=0; i<=k; i++)
  { create_separator(rowcol1, &hv, &sd);

    index[i] = 100*ind + i;
    wid = 460;
    dec = 1;
    swid = 438;
    inc = 1;

    if(i <= mes[ind])
      val[i] = 10.0 * mes_ms[ind][i];
    else
      val[i] = min[i];

    scale[i] = create_scale(rowcol1, scale_label[i], &wid, &min[i],
               &max[i], &inc, &dec, &val[i], &swid, &index[i], meshCB);

    for (j=0; j<6; j++)
    { n = 0;
      XtSetArg (wargs[n], XmNwidth, 90); n++;
      label[i][j] = XmCreateLabel(scale[i], numb[i][j], wargs, n);
    }
    XtManageChildren(label[i], 6);
```

```
  }

/*-----------------------------------------------------------------------
 * --- Create another RowColumn Widget within the first
 *-----------------------------------------------------------------------
 */
  hv = 1;
  create_separator(rowcol, &hv, &sd);

  n = 0;
  XtSetArg(wargs[n], XmNorientation, XmVERTICAL); n++;
  rowcol2 = XtCreateManagedWidget("Z_Mesh_Options_(continued)",
            xmRowColumnWidgetClass, rowcol, wargs, n);

/*-----------------------------------------------------------------------
 * --- Create a info_button, attach it to "rowcol2", and realize it.
 *-----------------------------------------------------------------------
 */
  n = 0;
  button = XmCreatePushButton(rowcol2, but_label, wargs, n);
  XtManageChild (button);

/*-----------------------------------------------------------------------
 * --- Create the Subinterval Count Scales
 *-----------------------------------------------------------------------
 */
  hv = 0;

  for (i=k+1; i<=2*k+1; i++)
  { create_separator(rowcol2, &hv, &sd);

    index[i] = 100*ind + i + 50-k-1;
    wid = 160;
    dec = 0;
    swid = 138;
    inc = 1;

    if(i-k-1 <= mes[ind])
    { val[i] = mes_mh[ind][i-k-1];
      if(val[i] > max[i])
        val[i] = max[i];
    }
    else
      val[i] = min[i];

    scale[i] = create_scale(rowcol2, scale_label[i], &wid, &min[i],
               &max[i], &inc, &dec, &val[i], &swid, &index[i], meshCB);

    for (j=0; j<2; j++)
    { n = 0;
      XtSetArg (wargs[n], XmNwidth, 90); n++;
      label[i][j] = XmCreateLabel(scale[i], numb[i][5*j], wargs, n);
    }
    XtManageChildren(label[i], 2);
  }
}     /*   end meshzmenuCB   */

/***********************************************************************
 *                          VOID MESHCB
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  meshCB
```

```
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  meshxmenuCB, meshymenuCB,
 *                                 meshzmenuCB
 *=====================================================================
 *<Description>
 *    Gets the grid mesh dimensions
 *=====================================================================
 *<Called routines>
 *    none
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void meshCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                   - the ID of the widget for which the
 *                             callback is registered
 *       c                   - pointer to the data passed to the routine
 *       call_data           - a pointer to the callback structure which
 *                             contains information on why the callback
 *                             occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void meshCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *n = (int *)c;
  int i, j, k, m, mm, value;
  float sum;
  Boolean flag;
  XmScaleCallbackStruct * call_value =
                            (XmScaleCallbackStruct *) call_data;

  value = call_value -> value;
  i = (*n) / 100;
  j = (*n) % 100;

  if(j < 50)
  { sum = 0.1 * value;

    if(j < mes[i])
    { k = -1;
      if(j == 0)
      { if(sum > 0.0 && sum < mes_ms[i][1])
          k = j;
      }
      else
      { if(sum > mes_ms[i][j-1] && sum < mes_ms[i][j+1])
          k = j;
      }

      if(k >= 0)
      { mes_ms[i][k] = sum;

        if(mes_del_cnt[i] > 0)
```

```
        { for (m=0; m<mes_del_cnt[i]; m++)
          { if(j == mes_del[i][m])
            { mes_del_cnt[i]--;
              if((mes_del_cnt[i] != 0) && (m != mes_del_cnt[i]))
              { for (mm=m; mm<mes_del_cnt[i]; mm++)
                  mes_del[i][mm] = mes_del[i][mm+1];
              }
            }
          }
        }

        new_file = TRUE;
        in_change = TRUE;
      }
      else
      { mes_del[i][mes_del_cnt[i]] = j;
        mes_del_cnt[i]++;
        new_file = TRUE;
        in_change = TRUE;
      }
    }

    else if(j > mes[i])
    { k = -1;
      if(sum > 0.0 && sum < mes_ms[i][0])
        k = 0;
      else
      { if(mes[i] > 0)
        { for (m=1; m<=mes[i]; m++)
          { if(sum > mes_ms[i][m-1] && sum < mes_ms[i][m])
              k = m;
          }
        }
      }

      if(k >= 0)
      { mes_add[i][0] = sum;
        new_file = TRUE;
        in_change = TRUE;
      }
    }
  }

  else
  { j -= 50;
    if(j <= mes[i])
    { if(value < 1)
      { k = -1;

        if(mes_del_cnt[i] > 0)
        { for (m=0; m<mes_del_cnt[i]; m++)
          { if(j == mes_del[i][m])
              k = m;
          }
        }

        if(k < 0 && j != mes[i])
        { mes_del[i][mes_del_cnt[i]] = j;
          mes_del_cnt[i]++;
        }
      }
```

```
      else
      { for (sum=0.0, m=0; m<=mes[i]; m++)
        { if(m != j)
          { sum += mes_mh[i][m];

            if(mes_del_cnt[i] > 0)
            { for (flag=FALSE, k=0; k<mes_del_cnt[i]; k++)
              { if((m == mes_del[i][k]) && !flag)
                { sum -= mes_mh[i][m];
                  flag = TRUE;
                }
              }
            }
          }
        }

        if(value + sum > MAXXYZ)
          value = MAXXYZ - sum;
        mes_mh[i][j] = value;
      }
    }
    else
      mes_add[i][1] = value;

    new_file = TRUE;
    in_change = TRUE;
  }
}     /*   end meshCB()   */

/*********************************************************************
 *                          VOID AREA_ALBCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  area_albCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_area1menu, cancelaCB
 *====================================================================
 *<Description>
 *    Calls the appropriate function for selecting the area albedo
 *====================================================================
 *<Called routines>
 *    albedo1CB             - Sets the scale for the albedo when
 *                            "mdl2_ialb < 0".
 *    albedo2CB             - Sets the menu for the albedo when
 *                            "mdl2_ialb = 0".
 *    albedo3CB             - Sets the menu for the albedo when
 *                            "mdl2_ialb > 0".
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void area_albCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - the data passed to the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *====================================================================
```

```
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void area_albCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *indx = (int *)c;

  if(mdl2 < 0)
  { mdl2 = 0;
    mdl2_sn = 0.8;
    mdl2_tbound = 288.2;
    mdl2_ialb = -1;
    mdl2_ip = 0;
    cur_ialb = mdl2_ialb;
  }

  if(mdl2_ialb < 0)
    albedo1CB(menu, indx, NULL);
  else if(mdl2_ialb == 0)
    albedo2CB(menu, indx, NULL);
  else if(mdl2_ialb > 0)
    albedo3CB(menu, indx, NULL);
}     /*   end area_albCB()   */

/********************************************************************
 *                          VOID ALBEDO1CB
 ********************************************************************
 *<Begin>
 *<Identification>          Name:  albedo1CB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  area_albCB, albedo_chg
 *=====================================================================
 *<Description>
 *    Sets up the scale for the Albedo value when "mdl2_ialb = -1",
 *    Wave Independent - User Defined Albedo.
 *=====================================================================
 *<Called routines>
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelbCB              - removes the rowcol widget
 *    create_separator       - creates a Separator Widget
 *    create_scale           - creates a Scale Widget
 *    albed1CB               - Gets the albedo value from the scale
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void albedo1CB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                   - the ID of the widget for which the
 *                             callback is registered
 *       c                   - the data passed to the routine
 *       call_data           - a pointer to the callback structure which
 *                             contains information on why the callback
 *                             occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
```

```
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void albedo1CB (Widget w, XtPointer c, XtPointer call_data)
{
  int *indx = (int *)c;
  Widget rowcol, label[6], scale;
  Arg wargs[10];
  int n, i;
  static int hv, sd, wid, inc, dec, val, swid;
  static char scale_label[27];
  static char numb[6][4] = { "0.0", "0.2", "0.4", "0.6", "0.8", "1.0" };
  static int min = 0;
  static int max = 1000;

  sprintf(scale_label, "Albedo Value for Area - %d", (*indx)+1);

/*-------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "User_Defined_Albedo", cancelbCB);

/*-------------------------------------------------------------------
 * --- Create the Scale
 *-------------------------------------------------------------------
 */
  hv = 0;
  sd = 2;
  create_separator(rowcol, &hv, &sd);

  wid = 560;
  inc = 0;
  dec = 3;
  swid = 538;

  if(area > albd)
    val = min;
  else
    val = 1000.0 * albd_falb[(*indx)];

  scale = create_scale(rowcol, scale_label, &wid, &min, &max, &inc,
          &dec, &val, &swid, indx, albed1CB);

  for (i=0; i<6; i++)
  { n = 0;
    XtSetArg (wargs[n], XmNwidth, 90); n++;
    label[i] = XmCreateLabel(scale, numb[i], wargs, n);
  }
  XtManageChildren(label, 6);
}     /*   end albedo1CB()   */

/*******************************************************************
 *                          VOID ALBED1CB
 *******************************************************************
 *<Begin>
 *<Identification>          Name:  albed1CB
 *                          Type:  C void
 *                      Filename:  visual.c
```

```
 *                          Parent:  albedo1CB
 *=====================================================================
 *<Description>
 *    Gets the albedo value when "mdl2_ialb = -1".
 *=====================================================================
 *<Called routines>
 *    None
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void albed1CB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                    - the ID of the widget for which the
 *                              callback is registered
 *       c                    - pointer to the data passed to the routine
 *       call_data            - a pointer to the callback structure which
 *                              contains information on why the callback
 *                              occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void albed1CB (Widget w, XtPointer c, XtPointer call_data)
{
  int *n = (int *)c;
  int i, value;
  XmScaleCallbackStruct * call_value =
                              (XmScaleCallbackStruct *) call_data;

  value = call_value -> value;
  i = *n;

  cur_area = i;
  area_iamtl[i] = albd_lalb[i] = i+1;
  albd_falb[i] = 0.001 * (float)value;

  new_file = TRUE;
  in_change = TRUE;
}     /*   end albed1CB()    */

/********************************************************************
 *                          VOID ALBEDO2CB
 ********************************************************************
 *<Begin>
 *<Identification>          Name:  albedo2CB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  area_albCB, albedo_chg
 *=====================================================================
 *<Description>
 *    Sets up menus for the various Albedo values when "mdl2_ialb = 0",
 *    Wave Independent - Tabulated Albedo.
 *=====================================================================
 *<Called routines>
 *    create_rowcol         - creates a Rowcol Widget
 *    cancelbCB             - removes the rowcol widget
```

```
 *    create_radiobox        - creates a Radiobox Widget
 *    create_togglebutton    - creates a Togglebutton Widget
 *    albed2CB               - Gets the albedo value
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void albedo2CB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                   - the ID of the widget for which the
 *                             callback is registered
 *       c                   - the data passed to the routine
 *       call_data           - a pointer to the callback structure which
 *                             contains information on why the callback
 *                             occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void albedo2CB (Widget w, XtPointer c, XtPointer call_data)
{
  int *indx = (int *)c;
  Widget rowcol, radio, toggle[31];
  int n, j, ind;
  static int index[31], nc;
  static int sub[16] = { 1, 2, 3, 4, 5, 6, 7, 16, 17, 18, 19, 20, 21,
                         22, 29, 30 };
  static char cat_label[41];
  static char tog_label[31][17];

  for (j=0; j<31; j++)
  { strcpy (tog_label[j], broad_type[j]);
    for (n=0; n<16; n++)
    { if(j == sub[n])
        strcat (tog_label[j], " >");
    }
  }

  ind = *indx;
  sprintf(cat_label,"Broad-Band Albedo Surfaces for Area - %d",
          (*indx)+1);

/*-------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Broad-Band_Albedo", cancelbCB);

/*-------------------------------------------------------------------
 * --- Create the radiobox and toggles
 *-------------------------------------------------------------------
 */
  nc = 2;
  radio = create_radiobox(rowcol, &nc, cat_label);

  for (j=0; j<31; j++)
  { index[j] = 100 * ind + j;
```

```
      toggle[j] = create_togglebutton(radio, tog_label[j], &index[j],
                  albed2CB);
    }

    if(area >= 0)
    { if(area_iamtl[ind] == 0)
        j = 0;
      else if(area_iamtl[ind] > 0 && area_iamtl[ind] < 3)
        j = 1;
      else if(area_iamtl[ind] > 2 && area_iamtl[ind] < 5)
        j = 2;
      else if(area_iamtl[ind] > 4 && area_iamtl[ind] < 7)
        j = 3;
      else if(area_iamtl[ind] > 6 && area_iamtl[ind] < 9)
        j = 4;
      else if(area_iamtl[ind] > 8 && area_iamtl[ind] < 11)
        j = 5;
      else if(area_iamtl[ind] > 10 && area_iamtl[ind] < 13)
        j = 6;
      else if(area_iamtl[ind] > 12 && area_iamtl[ind] < 15)
        j = 7;
      else if(area_iamtl[ind] < 22)
        j = area_iamtl[ind] - 7.0;
      else if(area_iamtl[ind] > 22 && area_iamtl[ind] < 25)
        j = 16;
      else if(area_iamtl[ind] > 24 && area_iamtl[ind] < 27)
        j = 17;
      else if(area_iamtl[ind] > 26 && area_iamtl[ind] < 29)
        j = 18;
      else if(area_iamtl[ind] > 28 && area_iamtl[ind] < 32)
        j = 19;
      else if(area_iamtl[ind] > 31 && area_iamtl[ind] < 35)
        j = 20;
      else if(area_iamtl[ind] > 34 && area_iamtl[ind] < 38)
        j = 21;
      else if(area_iamtl[ind] > 37 && area_iamtl[ind] < 41)
        j = 22;
      else if(area_iamtl[ind] < 47)
        j = area_iamtl[ind] - 18.0;
      else if(area_iamtl[ind] > 46 && area_iamtl[ind] < 52)
        j = 29;
      else if(area_iamtl[ind] > 51 && area_iamtl[ind] < 56)
        j = 30;
      else
        j = -1;

      if(j >= 0 && j <= 30)
        XmToggleButtonSetState(toggle[j], TRUE, FALSE);
    }
}     /*   end albedo2CB()   */

/*********************************************************************
 *                          VOID ALBED2CB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  albed2CB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  albedo2CB
 *====================================================================
 *<Description>
 *    Gets the albedo value when "mdl2_ialb = 0".
```

```
 *=====================================================================
 *<Called routines>
 *    albe0                    - Presents Soils and Roads choices
 *    albe1                    - Presents Grass and Tree choices
 *    albe2                    - Presents Snow choices
 *    albe3                    - Presents Ice choices
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void albed2CB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                    - the ID of the widget for which the
 *                              callback is registered
 *       c                    - pointer to the data passed to the routine
 *       call_data            - a pointer to the callback structure which
 *                              contains information on why the callback
 *                              occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void albed2CB (Widget w, XtPointer c, XtPointer call_data)
{
  int *n = (int *)c;
  int i, j;
  static int index;

  i = (*n) / 100;
  j = (*n) % 100;

  index = -1;
  cur_area = i;

  if(j == 0)
    area_iamtl[i] = j;
  else if(j < 8)
  { index = 10*j + j - 1;
    albe0(&index);
  }
  else if(j < 16)
    area_iamtl[i] = j + 7;
  else if(j < 19)
  { index = 10*j + j - 9;
    albe0(&index);
  }
  else if(j < 23)
  { index = 10*j + j - 19;
    albe1(&index);
  }
  else if(j < 29)
    area_iamtl[i] = j + 18;
  else if(j == 29)
    albe2();
  else if(j == 30)
    albe3();
```

```
  new_file = TRUE;
  in_change = TRUE;
}     /*   end albed2CB()    */

/*********************************************************************
 *                          VOID ALBE0
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  albe0
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  albed2CB
 *====================================================================
 *<Description>
 *    Presents Soils and Roads choices of the broad-band albedo groups
 *====================================================================
 *<Called routines>
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelbCB              - removes the Rowcol Widget
 *    create_radiobox        - creates a Radiobox Widget
 *    create_togglebutton    - creates a Togglebutton Widget
 *    albe0CB                - Sets the broad-band albedo
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void albe0(int *indx)
 *    Input:
 *       *indx               - index to indicate desired albedo group
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void albe0(int *indx)
{
  Widget radio, toggle[2], rowcol;
  int j, ind, grp;
  static int index[2], nc;
  static char tog_label[2][4] = { "dry", "wet" };

  grp = (*indx) / 10;
  ind = (*indx) % 10;

/*-------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Soils_&_Roads_State", cancelbCB);

/*-------------------------------------------------------------------
 * --- Create the radiobox and toggles
 *-------------------------------------------------------------------
 */
  nc = 1;
  radio = create_radiobox(rowcol, &nc, broad_type[grp]);

  for (j=0; j<2; j++)
```

```
  { index[j] = 10 * grp + j;
    toggle[j] = create_togglebutton(radio, tog_label[j], &index[j],
                albe0CB);
  }

  if(ind == 0)
  { if(area_iamtl[cur_area] > 0 && area_iamtl[cur_area] < 3)
      j = area_iamtl[cur_area] - 1;
  }
  else if(ind == 1)
  { if(area_iamtl[cur_area] > 2 && area_iamtl[cur_area] < 5)
      j = area_iamtl[cur_area] - 3;
  }
  else if(ind == 2)
  { if(area_iamtl[cur_area] > 4 && area_iamtl[cur_area] < 7)
      j = area_iamtl[cur_area] - 5;
  }
  else if(ind == 3)
  { if(area_iamtl[cur_area] > 6 && area_iamtl[cur_area] < 9)
      j = area_iamtl[cur_area] - 7;
  }
  else if(ind == 4)
  { if(area_iamtl[cur_area] > 8 && area_iamtl[cur_area] < 11)
      j = area_iamtl[cur_area] - 9;
  }
  else if(ind == 5)
  { if(area_iamtl[cur_area] > 10 && area_iamtl[cur_area] < 13)
      j = area_iamtl[cur_area] - 11;
  }
  else if(ind == 6)
  { if(area_iamtl[cur_area] > 12 && area_iamtl[cur_area] < 15)
      j = area_iamtl[cur_area] - 13;
  }
  else if(ind == 7)
  { if(area_iamtl[cur_area] > 22 && area_iamtl[cur_area] < 25)
      j = area_iamtl[cur_area] - 23;
  }
  else if(ind == 8)
  { if(area_iamtl[cur_area] > 24 && area_iamtl[cur_area] < 27)
      j = area_iamtl[cur_area] - 25;
  }
  else if(ind == 9)
  { if(area_iamtl[cur_area] > 26 && area_iamtl[cur_area] < 29)
      j = area_iamtl[cur_area] - 27;
  }
  else
    j = -1;

  if(j >= 0 && j <= 1)
    XmToggleButtonSetState(toggle[j], TRUE, FALSE);
}     /*   end albe0()   */

/*********************************************************************
 *                          VOID ALBE0CB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  albe0CB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  albe0, albe1, albe2, albe3
 *====================================================================
 *<Description>
```

```
 *    Sets the broad-band albedo
 *=====================================================================
 *<Called routines>
 *    none
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void albe0CB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                    - the ID of the widget for which the
 *                              callback is registered
 *       c                    - pointer to the data passed to the routine
 *       call_data            - a pointer to the callback structure which
 *                              contains information on why the callback
 *                              occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void albe0CB (Widget w, XtPointer c, XtPointer call_data)
{
  int *n = (int *)c;
  int grp, i;

  grp = (*n) / 10;
  i = (*n) % 10;

  if(grp == 1)
    area_iamtl[cur_area] = i + 1;
  else if(grp == 2)
    area_iamtl[cur_area] = i + 3;
  else if(grp == 3)
    area_iamtl[cur_area] = i + 5;
  else if(grp == 4)
    area_iamtl[cur_area] = i + 7;
  else if(grp == 5)
    area_iamtl[cur_area] = i + 9;
  else if(grp == 6)
    area_iamtl[cur_area] = i + 11;
  else if(grp == 7)
    area_iamtl[cur_area] = i + 13;
  else if(grp == 16)
    area_iamtl[cur_area] = i + 23;
  else if(grp == 17)
    area_iamtl[cur_area] = i + 25;
  else if(grp == 18)
    area_iamtl[cur_area] = i + 27;
  else if(grp == 19)
    area_iamtl[cur_area] = i + 29;
  else if(grp == 20)
    area_iamtl[cur_area] = i + 32;
  else if(grp == 21)
    area_iamtl[cur_area] = i + 35;
  else if(grp == 22)
    area_iamtl[cur_area] = i + 38;
  else if(grp == 29)
```

```
    area_iamtl[cur_area] = i + 47;
  else if(grp == 30)
    area_iamtl[cur_area] = i + 47;

  new_file = TRUE;
  in_change = TRUE;
}     /*   end albe0CB()   */

/*********************************************************************
 *                              VOID ALBE1
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  albe1
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  albed2CB
 *====================================================================
 *<Description>
 *    Presents Grass and Tree choices of the broad-band albedo groups
 *====================================================================
 *<Called routines>
 *    create_rowcol         - creates a Rowcol Widget
 *    cancelbCB             - removes the Rowcol Widget
 *    create_radiobox       - creates a Radiobox Widget
 *    create_togglebutton   - creates a Togglebutton Widget
 *    albe0CB               - Sets the broad-band albedo
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void albe1(int *indx)
 *    Input:
 *       *indx              - index to indicate desired albedo group
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void albe1(int *indx)
{
  Widget radio, toggle[3], rowcol;
  int j, ind, grp;
  static int index[3], nc;
  static char tog_label[3][12] = { "growing", "dormant", "unspecified"};

  grp = (*indx) / 10;
  ind = (*indx) % 10;

/*-------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Grass_&_Tree_State", cancelbCB);

/*-------------------------------------------------------------------
 * --- Create the radiobox and toggles
 *-------------------------------------------------------------------
 */
```

```
  nc = 1;
  radio = create_radiobox(rowcol, &nc, broad_type[grp]);

  for (j=0; j<3; j++)
  { index[j] = 10 * grp + j;
    toggle[j] = create_togglebutton(radio, tog_label[j], &index[j],
                albe0CB);
  }

  if(ind == 0)
  { if(area_iamtl[cur_area] > 28 && area_iamtl[cur_area] < 32)
      j = area_iamtl[cur_area] - 29;
  }
  else if(ind == 1)
  { if(area_iamtl[cur_area] > 31 && area_iamtl[cur_area] < 35)
      j = area_iamtl[cur_area] - 32;
  }
  else if(ind == 2)
  { if(area_iamtl[cur_area] > 34 && area_iamtl[cur_area] < 38)
      j = area_iamtl[cur_area] - 35;
  }
  else if(ind == 3)
  { if(area_iamtl[cur_area] > 37 && area_iamtl[cur_area] < 41)
      j = area_iamtl[cur_area] - 38;
  }
  else
    j = -1;

  if(j >= 0 && j <= 2)
    XmToggleButtonSetState(toggle[j], TRUE, FALSE);
}     /*   end albe1()   */

/*********************************************************************
 *                          VOID ALBE2
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  albe2
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  albed2CB
 *====================================================================
 *<Description>
 *    Presents Snow choices of the broad-band albedo groups
 *====================================================================
 *<Called routines>
 *    create_rowcol         - creates a Rowcol Widget
 *    cancelbCB             - removes the Rowcol Widget
 *    create_radiobox       - creates a Radiobox Widget
 *    create_togglebutton   - creates a Togglebutton Widget
 *    albe0CB               - Sets the broad-band albedo
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void albe2(void)
 *    Input:
 *       None
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
```

```
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void albe2(void)
{
  Widget radio, toggle[5], rowcol;
  int j;
  static int index[5], nc;
  static char tog_label[5][8] = { "fresh", "dense", "moist", "old",
                                  "melting" };

/*---------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *---------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Snow_State", cancelbCB);

/*---------------------------------------------------------------------
 * --- Create the radiobox and toggles
 *---------------------------------------------------------------------
 */
  nc = 1;
  radio = create_radiobox(rowcol, &nc, broad_type[29]);

  for (j=0; j<5; j++)
  { index[j] = j + 290;
    toggle[j] = create_togglebutton(radio, tog_label[j], &index[j],
                albe0CB);
  }

  if(area_iamtl[cur_area] > 46 && area_iamtl[cur_area] < 52)
    j = area_iamtl[cur_area] - 47;
  else
    j = -1;

  if(j >= 0)
    XmToggleButtonSetState(toggle[j], TRUE, FALSE);
}     /*   end albe2()   */

/***********************************************************************
 *                            VOID ALBE3
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  albe3
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  albed2CB
 *=====================================================================
 *<Description>
 *    Presents Ice choices of the broad-band albedo groups
 *=====================================================================
 *<Called routines>
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelbCB              - removes the Rowcol Widget
 *    create_radiobox        - creates a Radiobox Widget
 *    create_togglebutton    - creates a Togglebutton Widget
 *    albe0CB                - Sets the broad-band albedo
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void albe3(void)
```

```
 *    Input:
 *       None
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void albe3(void)
{
  Widget radio, toggle[4], rowcol;
  int j;
  static int index[4], nc;
  static char tog_label[4][13] = { "white", "grey", "snow and ice",
                                   "dark glass" };

/*-------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Ice_State", cancelbCB);

/*-------------------------------------------------------------------
 * --- Create the radiobox and toggles
 *-------------------------------------------------------------------
 */
  nc = 1;
  radio = create_radiobox(rowcol, &nc, broad_type[30]);

  for (j=0; j<4; j++)
  { index[j] = j + 300;
    toggle[j] = create_togglebutton(radio, tog_label[j], &index[j],
                albe0CB);
  }

  if(area_iamtl[cur_area] > 51 && area_iamtl[cur_area] < 56)
    j = area_iamtl[cur_area] - 52;
  else
    j = -1;

  if(j >= 0)
    XmToggleButtonSetState(toggle[j], TRUE, FALSE);
}     /*   end albe3()   */

/********************************************************************
 *                          VOID ALBEDO3CB
 ********************************************************************
 *<Begin>
 *<Identification>          Name:  albedo3CB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  area_albCB, albedo_chg
 *=====================================================================
 *<Description>
 *    Sets up menus for the various Albedo values when "mdl2_ialb > 0",
 *    Spectral Albedos.
 *=====================================================================
 *<Called routines>
```

```
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelbCB              - removes the rowcol widget
 *    create_radiobox        - creates a Radiobox Widget
 *    create_togglebutton    - creates a Togglebutton Widget
 *    albed3CB               - Gets the albedo value when "mdl2_ialb > 0"
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void albedo3CB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                   - the ID of the widget for which the
 *                             callback is registered
 *       c                   - the data passed to the routine
 *       call_data           - a pointer to the callback structure which
 *                             contains information on why the callback
 *                             occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void albedo3CB (Widget w, XtPointer c, XtPointer call_data)
{
  int *indx = (int *)c;
  Widget radio, toggle[7], rowcol;
  int j, ind;
  static int index[7], nc;
  static char cat_label[45];
  static char spect_label[7][20];

  for (j=0; j<7; j++)
    sprintf(spect_label[j], "Spectral Model - %d", j);

  ind = *indx;
  sprintf(cat_label, "Spectral Surface Albedo Models for Area - %d",
          (*indx)+1);

/*-------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Spectral_Albedo", cancelbCB);

/*-------------------------------------------------------------------
 * --- Create the radiobox and toggles
 *-------------------------------------------------------------------
 */
  nc = 1;
  radio = create_radiobox(rowcol, &nc, cat_label);

  for (j=0; j<7; j++)
  { index[j] = 100 * ind + j;
    toggle[j] = create_togglebutton(radio, spect_label[j], &index[j],
                albed3CB);
  }

  if(area >= 0)
```

```
    { if(area_iamtl[ind] >= 0 && area_iamtl[ind] < 7)
        XmToggleButtonSetState(toggle[(int)area_iamtl[ind]], TRUE, FALSE);
    }
  }     /*   end albedo3CB()   */

 /*********************************************************************
  *                          VOID ALBED3CB
  *********************************************************************
  *<Begin>
  *<Identification>          Name:  albed3CB
  *                          Type:  C void
  *                      Filename:  visual.c
  *                        Parent:  albedo3CB
  *====================================================================
  *<Description>
  *    Gets the albedo value when "mdl2_ialb > 0".
  *====================================================================
  *<Called routines>
  *    None
  *====================================================================
  *<Parameters>
  *    Formal declaration:
  *       void albed3CB( Widget w, XtPointer c, XtPointer call_data)
  *    Input:
  *       w                   - the ID of the widget for which the
  *                             callback is registered
  *       c                   - pointer to the data passed to the routine
  *       call_data           - a pointer to the callback structure which
  *                             contains information on why the callback
  *                             occurred
  *    Output:
  *       None
  *====================================================================
  *<History>
  *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
  *              Developed the original source code.
  *====================================================================
  *<End>
  *********************************************************************
  */
 void albed3CB (Widget w, XtPointer c, XtPointer call_data)
 {
   int *n = (int *)c;
   int i, j;

   i = (*n) / 100;
   j = (*n) % 100;

   area_iamtl[i] = j;
   cur_area = i;

   new_file = TRUE;
   in_change = TRUE;
 }     /*   end albed3CB()   */

 /*********************************************************************
  *                          VOID ALBEDO_CHG
  *********************************************************************
  *<Begin>
  *<Identification>          Name:  albedo_chg
  *                          Type:  C void
  *                      Filename:  visual.c
```

```
*                          Parent:  model1CB
*=====================================================================
*<Description>
*    Sets up the menu for selecting the Albedo for each Area.
*=====================================================================
*<Called routines>
*    create_menubar          - creates the pull-down menus, rollover
*                              menus, and menubar to control them
*
*    create_rowcol           - creates a Rowcol Widget
*    cancelbbCB              - removes the rowcol widget
*    create_radiobox         - creates a Radiobox Widget
*    create_togglebutton     - creates a Togglebutton Widget
*    albedo1CB               - Sets the scale for the albedo when
*                              "mdl2_ialb < 0".
*    albedo2CB               - Sets the menu for the albedo when
*                              "mdl2_ialb = 0".
*    albedo3CB               - Sets the menu for the albedo when
*                              "mdl2_ialb > 0".
*=====================================================================
*<Parameters>
*    Formal declaration:
*       void albedo_chg( void )
*    Input:
*       none
*    Output:
*       None
*=====================================================================
*<History>
*    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
*              Developed the original source code.
*=====================================================================
*<End>
**********************************************************************
*/
void albedo_chg (void)
{
  Widget rowcol, radio, toggle[IAM+1];
  int j;
  static int index[IAM+1], nc;
  static char tog_label[IAM+1][10];
  static char cat_label[29] = "Select Albedos for each Area";

  for(j=0; j<=area; j++)
    sprintf(tog_label[j], "Area - %d", j+1);

  cur_area = area + 1;

  if(mdl2 < 0)                              /* Default input values     */
  { mdl2 = 0;
    mdl2_sn = 0.8;
    mdl2_tbound = 288.2;
    mdl2_ialb = -1;
    mdl2_ip = 0;
    cur_ialb = mdl2_ialb;
  }

  if(cur_ialb < 0)                          /* Wave Indep, User Defined */
  { albd = -1;
    for(j=0; j<=area; j++)                  /* Initialize albedos       */
      area_iamtl[j] = 0;
  }
```

```
  if(mdl2_ialb < 0)                          /* Wave Indep, User Defined */
  { albd = area;
    for(j=0; j<=area; j++)                   /* Initialize ALBD cards    */
    { area_iamtl[j] = albd_lalb[j] = j+1;
      albd_falb[j] = background_albedo;      /*   to Background_Albedo   */
    }
  }
  else                                       /* Tabulated or Spectral    */
  { for(j=0; j<=area; j++)                   /*   clear IAMTL values     */
      area_iamtl[j] = 0;
  }

  XtUnmanageChild (menu);
  menu = create_menubar(form);

/*-----------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-----------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Area_Selection", cancelbbCB);

/*-----------------------------------------------------------------------
 * --- Create the radiobox and toggles
 *-----------------------------------------------------------------------
 */
  nc = 1;
  radio = create_radiobox(rowcol, &nc, cat_label);

  for (j=0; j<=area; j++)
  { index[j] = j;

    if(mdl2_ialb < 0)
    { toggle[j] = create_togglebutton(radio, tog_label[j], &index[j],
                  albedo1CB);                /* Wave Indep, User Defined */
    }
    else if(mdl2_ialb == 0)
    { toggle[j] = create_togglebutton(radio, tog_label[j], &index[j],
                  albedo2CB);                /* Wave Indep, Tabulated    */
    }
    else if(mdl2_ialb > 0)
    { toggle[j] = create_togglebutton(radio, tog_label[j], &index[j],
                  albedo3CB);                /* Spectral                 */
    }
  }

  if(cur_area <= area)
    XmToggleButtonSetState(toggle[cur_area], TRUE, FALSE);
}     /*   end albedo_chg()   */

/***********************************************************************
 *                          VOID AREA_LOCCB
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  area_locCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_area1menu, cancelaCB
 *=====================================================================
 *<Description>
 *    Sets a flag to initiate moving the point where the Albedo Area
 *    is located on the ground.
 *=====================================================================
```

```
 *<Called routines>
 *    reset                    - resets the viewing and plot parameters to
 *                               the original values
 *    drawscene                - Plots the 3-D BLIRB grid points, albedo
 *                               areas, aerosol regions, and the output
 *                               flux.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void area_locCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                     - the ID of the widget for which the
 *                               callback is registered
 *       c                     - the input data from the calling routine
 *       call_data             - a pointer to the callback structure which
 *                               contains information on why the callback
 *                               occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void area_locCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *indx = (int *)c;
  int i;

  move_area = TRUE;
  new_file = TRUE;

  cur_area = *indx;

  cur_lab_obsc = label_obsc;
  label_obsc = TRUE;

/*-------------------------------------------------------------------
 * --- Save the current viewing axis flags and choose a +Z axis option.
 *-------------------------------------------------------------------
 */
  for(i=0; i<3; i++)
    temp_axis[i] = view_axis[i];

  view_axis[0] = view_axis[1] = FALSE;
  view_axis[2] = TRUE;

/*-------------------------------------------------------------------
 * --- Save the current "mov" parameters.
 *-------------------------------------------------------------------
 */
  fac = mov->magfactor;
  nndx = mov->ndx;
  nndy = mov->ndy;
  ttdx = mov->tdx;
  ttdy = mov->tdy;

/*-------------------------------------------------------------------
 * --- Reset the viewpoint and redraw the scene in the window.
```

```
 *-----------------------------------------------------------------
 */
  reset();
  drawscene();
}     /*   end area_locCB()   */

/*****************************************************************
 *                          VOID AREA_DELCB
 *****************************************************************
 *<Begin>
 *<Identification>          Name:  area_delCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_area1menu, regnCB
 *===============================================================
 *<Description>
 *    Deletes an Albedo Area
 *===============================================================
 *<Called routines>
 *    set_albedo            - sets the Albedo values
 *    obsc                  - sets up rowcolumn widget with names of
 *                            materials.
 *    create_menubar        - creates the pull-down menus, rollover
 *                            menus, and menubar to control them
 *    drawscene             - Plots the 3-D BLIRB grid points, albedo
 *                            areas, aerosol regions, and the output
 *                            flux.
 *===============================================================
 *<Parameters>
 *    Formal declaration:
 *       void area_delCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - the data passed to the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *===============================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *===============================================================
 *<End>
 *****************************************************************
 */
void area_delCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *indx = (int *)c;
  int i;

  if( (*indx) != 0 )
  { if( (*indx) < area )
    { for (i=(*indx); i<area; i++)
      { area_alx[i] = area_alx[i+1];
        area_ahx[i] = area_ahx[i+1];
        area_aly[i] = area_aly[i+1];
        area_ahy[i] = area_ahy[i+1];

        if(mdl2_ialb < 0)
```

```
        { area_iamtl[i] = area_iamtl[i+1] - 1;
          albd_lalb[i] = albd_lalb[i+1] - 1;
          albd_falb[i] = albd_falb[i+1];
        }
        else
          area_iamtl[i] = area_iamtl[i+1];
      }
      area--;
      albd--;
    }
    else if( (*indx) == area )
    { area_alx[area] = area_ahx[area] = area_aly[area] = area_ahy[area]=
      area_iamtl[area] = albd_lalb[area] = albd_falb[area] = 0.0;
      area--;
      albd--;
    }

    new_file = TRUE;
    set_albedo();
    obsc();

    XtUnmanageChild (menu);
    menu = create_menubar(form);
  }
  drawscene();
}     /*   end area_delCB()   */

/*********************************************************************
 *                          VOID REGN_OPTCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  regn_optCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_region1menu
 *====================================================================
 *<Description>
 *    Sets up the scales for selecting the region dimensions
 *====================================================================
 *<Called routines>
 *    create_rowcol         - creates a Rowcol Widget
 *    cancelrCB             - removes the rowcol widget
 *    create_separator      - creates a Separator Widget
 *    create_scale          - creates a Scale Widget
 *    regnCB                - Gets the dimensions for a region
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void regn_optCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - the data passed to the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
```

```
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void regn_optCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *indx = (int *)c;
  Widget rowcol, label[3][7], scale[3];
  Arg wargs[10];
  int n, i, j, k, ind, maxx;
  float v[3];
  static int index[3], hv, sd, wid, inc, dec, val[3], swid;
  static char scale_label[3][27]={ "Length in X Direction (km)",
                                   "Length in Y Direction (km)",
                                   "Length in Z Direction (km)" };
  static char numb[3][6][6];
  static int min[3] = { 0, 0, 0 };
  static int max[3];

  for (j=0; j<3; j++)
  { maxx = regn_max_dim[j];
    max[j] = 10 * maxx;
    for (i=0; i<6; i++)
      sprintf(numb[j][i], "%5.2f", 0.2 * (float)(i * maxx));
  }

  ind = *indx;

/*---------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *---------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "BLIRB_Regions", cancelrCB);

/*---------------------------------------------------------------------
 * --- Create the Scale
 *---------------------------------------------------------------------
 */
  hv = 0;
  sd = 2;
  for (i=0; i<3; i++)
  { create_separator(rowcol, &hv, &sd);

    index[i] = 10*ind + i;
    wid = 560;
    inc = 1;
    dec = 1;
    swid = 538;

    if(ind <= regn)
      val[i] = 10.0 * (regn_rh[i][ind] - regn_rl[i][ind]);
    else
      val[i] = min[i];

    scale[i] = create_scale(rowcol, scale_label[i], &wid, &min[i],
               &max[i], &inc, &dec, &val[i], &swid, &index[i], regnCB);

    for (j=0; j<6; j++)
    { n = 0;
      XtSetArg (wargs[n], XmNwidth, 90); n++;
      label[i][j] = XmCreateLabel(scale[i], numb[i][j], wargs, n);
    }
```

```
    XtManageChildren(label[i], 6);
  }
}     /*   end regn_optCB()   */

/*********************************************************************
 *                          VOID REGNCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  regnCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  regn_optCB
 *====================================================================
 *<Description>
 *    Gets the BLIRB region dimensions
 *====================================================================
 *<Called routines>
 *    create_message        - draws a message in a dialog message box
 *    area_delCB            - deletes an Albedo Area
 *    regn_delCB            - deletes a BLIRB Region
 *    set_axis_pts          - sets up the grid points for the X, Y, and
 *                            Z axes.
 *    reset                 - resets the viewing and plot parameters to
 *                            the original values
 *    regn_fix              - rectifies the BLIRB Region location
 *    area_fix              - rectifies the Albedo Area location
 *    create_menubar        - creates the menubar for selecting the
 *                            various options
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void regnCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - pointer to the data passed to the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void regnCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *n = (int *)c;
  int i, j, value;
  float dif1, dif2, maxd;
  static int k;
  static char axis[3][2] = { "X", "Y", "Z" };
  static Boolean pass = FALSE;
  static Boolean mask[3], go;
  static char *msg[] = {
              "                                                    ",
              "                                                    ",
              "\n",
```

```
            "You may wish to use the [Grid Mesh Selection]\n",
            "suboption under the [Modify] option on the menubar\n",
            "to modify the Number of Subintervals to correct the\n",
            "problem.\n",
            "" };
XmScaleCallbackStruct * call_value =
                          (XmScaleCallbackStruct *) call_data;

value = call_value -> value;
i = (*n)/10;
j = (*n) % 10;

if((i > regn) && !pass)                    /* Initialize new region    */
{ regn_izmtl[i] = i+1;
  for (k=0; k<3; k++)
    regn_rl[k][i] = regn_rh[k][i] = 0.0;
  for (k=0; k<MXMTR; k++)
  { mtrl_lmtl[i][k] = 3.0;
    mtrl_wmtl[i][k] = 0.0;
  }
  pass = TRUE;
  mask[0] = mask[1] = mask[2] = FALSE;
}

if(i > regn)
  mask[j] = TRUE;                          /* Dim "j" of new regn input*/
go = mask[0] && mask[1] && mask[2];

regn_rh[j][i] = regn_rl[j][i] + 0.1 * value;

if(i == 0)                                 /* Primary BLIRB Region     */
{ k = regn_rh[j][0] + 0.5;

  if(mes[j] == 0)                          /* Get resulting mesh size  */
  { dif1 = mes_ms[j][mes[j]] / mes_mh[j][mes[j]];
    dif2 = (float)k / mes_mh[j][mes[j]];
  }
  else
  { dif1 = (mes_ms[j][mes[j]]-mes_ms[j][mes[j]-1])/mes_mh[j][mes[j]];
    dif2 = ((float)k - mes_ms[j][mes[j]-1]) / mes_mh[j][mes[j]];
  }

  if( k != (int)mes_ms[j][mes[j]] )
  { sprintf(msg[0], "%s Grid Mesh-%d Endpoint has been moved and\n",
            axis[j], mes[j]+1);
    sprintf(msg[1], "the Grid Spacing changed from %4.1f to %4.1f.\n",
            dif1, dif2);
    create_message( menu, msg, XmDIALOG_WARNING);
  }
  mes_ms[j][mes[j]] = k;

  maxd = regn_rh[0][0];                    /* Get the Max dimension    */
  if(maxd < regn_rh[1][0])
    maxd = regn_rh[1][0];
  if(maxd < regn_rh[2][0])
    maxd = regn_rh[2][0];
  if(maxd > 5.0)
    org_magfactor = 5.0 / maxd;            /* Set display scale factor */
  else
    org_magfactor = 1.0;

  set_axis_pts();                          /* Get the new axis grid pts*/
```

```
    if(regn > 0)                              /* For all but primary regn */
    { for (k=1; k<=regn; k++)
      { cur_regn = k;
        regn_fix();                           /* Adjust regn edges to grid*/

        if(regn_rh[j][k] > regn_rh[j][0]) /* Truncate regn if required*/
          regn_rh[j][k] = regn_rh[j][0];
        if(regn_rl[j][k] >= regn_rh[j][0])
          regn_delCB(NULL, &k, NULL);
      }
    }

    if(j == 0)                                /* For X direction          */
    { area_ahx[0] = regn_rh[0][0];            /* Update primary alb area  */
      if(area > 0)                            /* For other albedo areas   */
      { for(k=1; k<=area; k++)
        { cur_area = k;
          area_fix();                         /* Adjust their edges       */

          if(area_ahx[k] > regn_rh[0][0]) /* Truncate area if required*/
            area_ahx[k] = regn_rh[0][0];
          if(area_alx[k] >= regn_rh[0][0])
            area_delCB(NULL, &k, NULL);
        }
      }
    }
    else if(j == 1)                           /* For Y direction          */
    { area_ahy[0] = regn_rh[1][0];            /* Update primary alb area  */
      if(area > 0)                            /* For other albedo areas   */
      { for(k=1; k<=area; k++)
        { cur_area = k;
          area_fix();                         /* Adjust their edges       */

          if(area_ahy[k] > regn_rh[1][0]) /* Truncate area if required*/
            area_ahy[k] = regn_rh[1][0];
          if(area_aly[k] >= regn_rh[0][0])
            area_delCB(NULL, &k, NULL);
        }
      }
    }
    reset();                                  /* Update the display       */
  }
  else if(i <= regn)                          /* Not primary BLIRB regn   */
  { if(regn_rh[j][i] > regn_rh[j][0])         /* Truncate regn if required*/
      regn_rh[j][i] = regn_rh[j][0];
    cur_regn = i;
    regn_fix();                               /* Adjust edges of region   */
  }

  if(go)                                      /* If all 3 dimensions given*/
  { regn++;                                   /* Inc region count         */
    mtrl++;                                   /* Inc MTRL card count      */
    pass = FALSE;
    mask[0] = mask[1] = mask[2] = FALSE;
    in_change = TRUE;                         /* Indicate regn cnt change */

    cur_regn = regn;
    regn_fix();                               /* Adjust new region edges  */

    XtUnmanageChild (menu);
    menu = create_menubar(form);
  }
```

```
  new_file = TRUE;                          /* Input change              */
}     /*   end regnCB()    */

/*********************************************************************
 *                          VOID REGN_FIX
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  regn_fix
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  inputCB, regnCB, cancelmeCB
 *====================================================================
 *<Description>
 *    Rectifies the location of a BLIRB Region.
 *====================================================================
 *<Called routines>
 *    none
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void regn_fix(void);
 *    Input:
 *       None
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void regn_fix (void)
{
  int j, k, kk;
  float dif, del;

  for (j=0; j<3; j++)
  { del = regn_rh[j][cur_regn] - regn_rl[j][cur_regn];

    for (dif = 100000.0, kk = -1, k=0; k<=num_grid_pts[j]; k++)
    { if( fabs(regn_rl[j][cur_regn] - axis_pts[j][k]) < dif )
      { dif = fabs(regn_rl[j][cur_regn] - axis_pts[j][k]);
        kk = k;
      }
    }
    regn_rl[j][cur_regn] = axis_pts[j][kk];
    regn_rh[j][cur_regn] = regn_rl[j][cur_regn] + del;

    for (dif = 100000.0, kk = -1, k=0; k<=num_grid_pts[j]; k++)
    { if( fabs(regn_rh[j][cur_regn] - axis_pts[j][k]) < dif )
      { dif = fabs(regn_rh[j][cur_regn] - axis_pts[j][k]);
        kk = k;
      }
    }
    regn_rh[j][cur_regn] = axis_pts[j][kk];
  }
}     /*   end regn_fix()    */

/*********************************************************************
 *                          VOID FLAR_LOCCB
```

```
*********************************************************************
*<Begin>
*<Identification>          Name:  flar_locCB
*                          Type:  C void
*                      Filename:  visual.c
*                        Parent:  create_flar1menu, cancelfCB
*====================================================================
*<Description>
*    Sets flags to initiate moving the point where the Flare is
*    located.
*====================================================================
*<Called routines>
*    reset                 - resets the viewing and plot parameters to
*                            the original values
*    drawscene             - Plots the 3-D BLIRB grid points, albedo
*                            areas, aerosol regions, and the output
*                            flux.
*====================================================================
*<Parameters>
*    Formal declaration:
*       void flar_locCB( Widget w, XtPointer c, XtPointer call_data)
*    Input:
*       w                  - the ID of the widget for which the
*                            callback is registered
*       c                  - the input data from the calling routine
*       call_data          - a pointer to the callback structure which
*                            contains information on why the callback
*                            occurred
*    Output:
*       None
*====================================================================
*<History>
*    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
*              Developed the original source code.
*====================================================================
*<End>
*********************************************************************
*/
void flar_locCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *indx = (int *)c;
  int i;

  move_flarh = TRUE;
  move_flarv = TRUE;
  new_file = TRUE;

  cur_flar = *indx;

  cur_lab_obsc = label_obsc;
  label_obsc = TRUE;

  cur_minor_grid = minor_grid;
  minor_grid = TRUE;

/*-------------------------------------------------------------------
 * --- Save the current viewing axis flags and choose a +Z axis option.
 *-------------------------------------------------------------------
 */
  for(i=0; i<3; i++)
    temp_axis[i] = view_axis[i];
```

```
  view_axis[0] = view_axis[1] = FALSE;
  view_axis[2] = TRUE;

/*-----------------------------------------------------------------------
 * --- Save the current "mov" parameters.
 *-----------------------------------------------------------------------
 */
  fac = mov->magfactor;
  nndx = mov->ndx;
  nndy = mov->ndy;
  ttdx = mov->tdx;
  ttdy = mov->tdy;

/*-----------------------------------------------------------------------
 * --- Reset the viewpoint and redraw the scene in the window.
 *-----------------------------------------------------------------------
 */
  reset();
  drawscene();
}     /*   end flar_locCB()   */

/***********************************************************************
 *                           VOID FLAR_DELCB
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  flar_delCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_flar1menu
 *=====================================================================
 *<Description>
 *    Deletes a BLIRB Flare
 *=====================================================================
 *<Called routines>
 *    create_menubar        - creates the pull-down menus, rollover
 *                            menus, and menubar to control them
 *    drawscene             - Plots the 3-D BLIRB grid points, albedo
 *                            areas, aerosol regions, and the output
 *                            flux.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void flar_delCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - the data passed to the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void flar_delCB (Widget w, XtPointer c, XtPointer call_data)
{
```

```
  int *indx = (int *)c;
  int i, j;

  if( (*indx) < flar )
  { for (i=(*indx); i<flar; i++)
    { flar_idflr[i] = flar_idflr[i+1] - 1;
      flar_itflr[i] = flar_itflr[i+1];
      flar_xflar[i] = flar_xflar[i+1];
      flar_yflar[i] = flar_yflar[i+1];
      flar_zflar[i] = flar_zflar[i+1];
      flar_qflar[i] = flar_qflar[i+1];
      flar_tflar[i] = flar_tflar[i+1];
      for (j=0; j<4; j++)
      { flar_frrfup[i][j] = flar_frrfup[i+1][j];
        flar_frrfdn[i][j] = flar_frrfdn[i+1][j];
      }
    }
    flar--;
  }
  else if( (*indx) == flar )
  { flar_idflr[flar] = 0.0;
    flar_itflr[flar] = 0.0;
    flar_xflar[flar] = 0.0;
    flar_yflar[flar] = 0.0;
    flar_zflar[flar] = 0.0;
    flar_qflar[flar] = 0.0;
    flar_tflar[flar] = 0.0;
    for (j=0; j<4; j++)
    { flar_frrfup[flar][j] = 0.0;
      flar_frrfdn[flar][j] = 0.0;
    }
    flar--;
  }

  new_file = TRUE;

  XtUnmanageChild (menu);
  menu = create_menubar(form);

  drawscene();
}     /*   end flar_delCB()   */

/*************************************************************************
 *                          VOID SRCH_LOCCB
 *************************************************************************
 *<Begin>
 *<Identification>          Name:  srch_locCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_srchmenu, cancelslCB
 *=========================================================================
 *<Description>
 *    Sets flags to initiate moving the point where the Slite is
 *    located.
 *=========================================================================
 *<Called routines>
 *    reset                 - resets the viewing and plot parameters to
 *                            the original values
 *    drawscene             - Plots the 3-D BLIRB grid points, albedo
 *                            areas, aerosol regions, and the output
 *                            flux.
 *=========================================================================
```

```
 *<Parameters>
 *    Formal declaration:
 *       void srch_locCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                    - the ID of the widget for which the
 *                              callback is registered
 *       c                    - the input data from the calling routine
 *       call_data            - a pointer to the callback structure which
 *                              contains information on why the callback
 *                              occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 **********************************************************************
 */
void srch_locCB (Widget w, XtPointer c, XtPointer call_data)
{
  int i;

  move_srchh = TRUE;
  move_srchv = TRUE;
  new_file = TRUE;

  cur_lab_obsc = label_obsc;
  label_obsc = TRUE;

  cur_minor_grid = minor_grid;
  minor_grid = TRUE;

/*-------------------------------------------------------------------
 * --- Save the current viewing axis flags and choose a +Z axis option.
 *-------------------------------------------------------------------
 */
  for(i=0; i<3; i++)
    temp_axis[i] = view_axis[i];

  view_axis[0] = view_axis[1] = FALSE;
  view_axis[2] = TRUE;

/*-------------------------------------------------------------------
 * --- Save the current "mov" parameters.
 *-------------------------------------------------------------------
 */
  fac = mov->magfactor;
  nndx = mov->ndx;
  nndy = mov->ndy;
  ttdx = mov->tdx;
  ttdy = mov->tdy;

/*-------------------------------------------------------------------
 * --- Reset the viewpoint and redraw the scene in the window.
 *-------------------------------------------------------------------
 */
  reset();
  drawscene();
}     /*   end srch_locCB()   */
```

```
/*********************************************************************
 *                          VOID SRCH_DELCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  srch_delCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_srchmenu
 *====================================================================
 *<Description>
 *    Deletes a BLIRB SearchLight
 *====================================================================
 *<Called routines>
 *    create_menubar        - creates the pull-down menus, rollover
 *                            menus, and menubar to control them
 *    drawscene             - Plots the 3-D BLIRB grid points, albedo
 *                            areas, aerosol regions, and the output
 *                            flux.
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void srch_delCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - the data passed to the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void srch_delCB (Widget w, XtPointer c, XtPointer call_data)
{
  int i, j;

  srch_xsrch = srch_ysrch = srch_zsrch = srch_thsrch = srch_azsrch =
  srch_psrch = srch_tmsrch = srch_sdiam = 0.0;
  srch = -1;

  new_file = TRUE;

  XtUnmanageChild (menu);
  menu = create_menubar(form);

  drawscene();
}     /*   end srch_delCB()   */

/*********************************************************************
 *                          VOID REGN_LOCCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  regn_locCB
 *                          Type:  C void
 *                      Filename:  visual.c
```

```
 *                        Parent:  create_region1menu, cancelrCB
 *=====================================================================
 *<Description>
 *    Sets flags to initiate moving the point where the Region is
 *    located.
 *=====================================================================
 *<Called routines>
 *    reset                  - resets the viewing and plot parameters to
 *                             the original values
 *    drawscene              - Plots the 3-D BLIRB grid points, albedo
 *                             areas, aerosol regions, and the output
 *                             flux.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void regn_locCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                   - the ID of the widget for which the
 *                             callback is registered
 *       c                   - the input data from the calling routine
 *       call_data           - a pointer to the callback structure which
 *                             contains information on why the callback
 *                             occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void regn_locCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *indx = (int *)c;
  int i;

  move_regnh = TRUE;
  move_regnv = TRUE;
  new_file = TRUE;

  cur_regn = *indx;

  cur_lab_obsc = label_obsc;
  label_obsc = TRUE;

  cur_minor_grid = minor_grid;
  minor_grid = TRUE;

/*-------------------------------------------------------------------
 * --- Save the current viewing axis flags and choose a +Z axis option.
 *-------------------------------------------------------------------
 */
  for(i=0; i<3; i++)
    temp_axis[i] = view_axis[i];

  view_axis[0] = view_axis[1] = FALSE;
  view_axis[2] = TRUE;

/*-------------------------------------------------------------------
 * --- Save the current "mov" parameters.
```

```
 *-----------------------------------------------------------------------
 */
  fac = mov->magfactor;
  nndx = mov->ndx;
  nndy = mov->ndy;
  ttdx = mov->tdx;
  ttdy = mov->tdy;

/*-----------------------------------------------------------------------
 * --- Reset the viewpoint and redraw the scene in the window.
 *-----------------------------------------------------------------------
 */
  reset();
  drawscene();
}     /*   end regn_locCB()    */

/***********************************************************************
 *                           VOID REGN_DELCB
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  regn_delCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_region1menu, regnCB
 *=====================================================================
 *<Description>
 *    Deletes a BLIRB Region
 *=====================================================================
 *<Called routines>
 *    set_aerosol           - sets the Aerosol Material values
 *    obsc                  - sets up rowcolumn widget with names of
 *                            materials.
 *    create_menubar        - creates the pull-down menus, rollover
 *                            menus, and menubar to control them
 *    drawscene             - Plots the 3-D BLIRB grid points, albedo
 *                            areas, aerosol regions, and the output
 *                            flux.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void regn_delCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - the data passed to the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void regn_delCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *indx = (int *)c;
  int i, j;
```

```
  if( (*indx) != 0 )
  { if( (*indx) < regn )
    { for (i=(*indx); i<regn; i++)
      { regn_izmtl[i] = regn_izmtl[i+1] - 1;
        for (j=0; j<3; j++)
        { regn_rl[j][i] = regn_rl[j][i+1];
          regn_rh[j][i] = regn_rh[j][i+1];
        }
        for (j=0; j<MXMTR; j++)
        { mtrl_lmtl[i][j] = mtrl_lmtl[i+1][j];
          mtrl_wmtl[i][j] = mtrl_wmtl[i+1][j];
        }
      }
      regn--;
      mtrl--;
    }
    else if( (*indx) == regn )
    { regn_izmtl[regn] = 0.0;
      for (j=0; j<3; j++)
        regn_rl[j][regn] = regn_rh[j][regn] = 0.0;
      for (j=0; j<MXMTR; j++)
      { mtrl_lmtl[regn][j] = 3.0;
        mtrl_wmtl[regn][j] = 0.0;
      }
      regn--;
      mtrl--;
    }

    new_file = TRUE;
    set_aerosol();
    obsc();

    XtUnmanageChild (menu);
    menu = create_menubar(form);
  }
  drawscene();
}     /*   end regn_delCB()   */

/*********************************************************************
 *                          VOID REGN_MTLCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  regn_mtlCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_region1menu, regn_mtl1
 *====================================================================
 *<Description>
 *    Selects the various Material Options for a BLIRB Region
 *====================================================================
 *<Called routines>
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelmCB              - removes the rowcol widget
 *    create_radiobox        - creates a Radiobox Widget
 *    create_togglebutton    - creates a Togglebutton Widget
 *    mtrlCB                 - Sets one of the input parameters of the
 *                             MTRL card (LMTL).
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void regn_mtlCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
```

```
 *       w                    - the ID of the widget for which the
 *                              callback is registered
 *       c                    - the data passed to the routine
 *       call_data            - a pointer to the callback structure which
 *                              contains information on why the callback
 *                              occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void regn_mtlCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *indx = (int *)c;
  Widget radio, toggle[16], rowcol;
  int i, j, aerosl;
  int reg, mtl;
  static int index[16], nc;
  static char cat_label[27];
  static char tog_label[16][24] = { "LOWTRAN Default Cloud",
      "No Cloud",                    "Rural Aerosol          >",
      "Maritime Aerosol      >",     "Urban Aerosol          >",
      "Tropospheric Aerosol  >",     "Stratospheric Aerosol   ",
      "Volcanic              >",     "Meteoric Dust",
      "Fog                   >",     "Clouds                 >",
      "Rain                  >",     "Snow",
      "Desert Aerosol        >",     "Dust and Dirt          >",
      "Combat Dust and Smoke >" };

  reg = (*indx) / 10;
  mtl = (*indx) % 10;

  sprintf(cat_label, "Material %d for Region %d", mtl+1, reg+1);

/*-------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Material_Options", cancelmCB);

/*-------------------------------------------------------------------
 * --- Create the radiobox and toggles
 *-------------------------------------------------------------------
 */
  nc = 2;
  radio = create_radiobox(rowcol, &nc, cat_label);

  for (j=0; j<16; j++)
  { index[j] = 100 * (100 * (*indx) + j);
    toggle[j] = create_togglebutton(radio, tog_label[j], &index[j],
                mtrlCB);
  }

  if(mtrl >= 0)
  { i = mtrl_lmtl[reg][mtl];
    if(i == 0)
      j = 14;
```

```
    else if(i == 1)
      j = 10;
    else if(i == 2)
      j = 0;
    else if(i == 3)
      j = 1;
    else if((i >= 4 && i <= 7) || (i >= 60 && i <= 67))
      j = 2;
    else if((i >= 8 && i <= 11) || (i >= 44 && i <= 51))
      j = 3;
    else if((i >= 12 && i <= 15) || (i >= 52 && i <= 59))
      j = 4;
    else if(i >= 20 && i <= 23)
      j = 5;
    else if(i == 24)
      j = 6;
    else if(i >= 25 && i <= 26)
      j = 7;
    else if(i == 29)
      j = 8;
    else if((i >= 27 && i <= 28) || (i >= 68 && i <= 69))
      j = 9;
    else if((i == 1) || (i >= 30 && i <= 36) || (i >= 83 && i <= 84))
      j = 10;
    else if(i >= 70 && i <= 72)
      j = 11;
    else if(i == 73)
      j = 12;
    else if(i >= 37 && i <= 40)
      j = 13;
    else if((i == 0) || (i >= 93 && i <= 94))
      j = 14;
    else if(i >= 95 && i <= 100)
      j = 15;

    XmToggleButtonSetState(toggle[j], TRUE, FALSE);
  }
}     /*   end regn_mtlCB()   */

/*********************************************************************
 *                          VOID MTRLCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  mtrlCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  regn_mtlCB
 *====================================================================
 *<Description>
 *    Sets the input parameter LMTL of the MTRL card.
 *====================================================================
 *<Called routines>
 *    mtrl0                 - Presents Rural and Urban Aerosol Material
 *                            Group Options.
 *    mtrl1                 - Presents Maritime Aerosol Material Group
 *                            Options.
 *    mtrl2                 - Presents Tropospheric Aerosol Material
 *                            Group Options.
 *    mtrl3                 - Presents Volcanic Aerosol Material Group
 *                            Options.
 *    mtrl4                 - Presents Fog Aerosol Material Group
 *                            Options.
```

```
 *    mtrl5                    - Presents Clouds Material Group Options.
 *    mtrl6                    - Presents Rain Material Group Options.
 *    mtrl7                    - Presents Desert Aerosol Material Group
 *                               Options.
 *    mtrl8                    - Presents Dust and Dirt Material Group
 *                               Options.
 *    mtrl9                    - Presents Combat Dust and Smoke Material
 *                               Group Options.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void mtrlCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                     - the ID of the widget for which the
 *                               callback is registered
 *       c                     - pointer to the data passed to the routine
 *       call_data             - a pointer to the callback structure which
 *                               contains information on why the callback
 *                               occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void mtrlCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *n = (int *)c;
  int reg, mtl, indx;
  static int index;

  index = (*n) / 100;
  indx = index % 100;
  mtl = ( index % 1000 ) / 100;
  reg = index / 1000;

  cur_regn = reg;
  cur_mtl = mtl;

  if(indx == 0)
    mtrl_lmtl[reg][mtl] = 2;
  else if(indx == 1)
    mtrl_lmtl[reg][mtl] = 3;
  else if(indx == 2)
    mtrl0(&index);
  else if(indx == 3)
    mtrl1(&index);
  else if(indx == 4)
    mtrl0(&index);
  else if(indx == 5)
    mtrl2(&index);
  else if(indx == 6)
    mtrl_lmtl[reg][mtl] = 24;
  else if(indx == 7)
    mtrl3(&index);
  else if(indx == 8)
    mtrl_lmtl[reg][mtl] = 29;
  else if(indx == 9)
```

```
    mtrl4(&index);
  else if(indx == 10)
    mtrl5(&index);
  else if(indx == 11)
    mtrl6(&index);
  else if(indx == 12)
    mtrl_lmtl[reg][mtl] = 73;
  else if(indx == 13)
    mtrl7(&index);
  else if(indx == 14)
    mtrl8(&index);
  else if(indx == 15)
    mtrl9(&index);

  in_change = TRUE;
  new_file = TRUE;
}     /*   end mtrlCB()    */

/*********************************************************************
 *                          VOID MTRL0
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  mtrl0
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  mtrlCB
 *====================================================================
 *<Description>
 *    Presents Rural and Urban Aerosol Material Group Options.
 *====================================================================
 *<Called routines>
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelCB               - removes the Rowcol Widget
 *    create_radiobox        - creates a Radiobox Widget
 *    create_togglebutton    - creates a Togglebutton Widget
 *    mtrl0CB                - Sets the Material Option
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void mtrl0(int *indx)
 *    Input:
 *       *indx               - index to indicate desired material group
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void mtrl0(int *indx)
{
  Widget radio, toggle[12], rowcol;
  int i, j, ind, id;
  static int index[12], nc;
  static char cat_label[2][22] = { "Rural Aerosol Options",
                                   "Urban Aerosol Options" };
  static char tog_label[12][15] = { "LOWTRAN  0% RH", "LOWTRAN 70% RH",
                  "LOWTRAN 80% RH", "LOWTRAN 99% RH", "EOSAEL  0% RH",
                  "EOSAEL 50% RH", "EOSAEL 70% RH", "EOSAEL 80% RH",
```

```
                    "EOSAEL 90% RH", "EOSAEL 95% RH", "EOSAEL 98% RH",
                    "EOSAEL 99% RH" };

  id = (*indx) % 100;
  if(id == 2)
    ind = 0;
  else
    ind = 1;

/*-----------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-----------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Rural_&_Urban_Aerosols", cancelCB);

/*-----------------------------------------------------------------------
 * --- Create the radiobox and toggles
 *-----------------------------------------------------------------------
 */
  nc = 1;
  radio = create_radiobox(rowcol, &nc, cat_label[ind]);

  for (j=0; j<12; j++)
  { index[j] = 100 * (*indx) + j;
    toggle[j] = create_togglebutton(radio, tog_label[j], &index[j],
                mtrl0CB);
  }

  i = mtrl_lmtl[cur_regn][cur_mtl];
  if(id == 2)
  { if(i >= 4 && i <= 7)
      j = i - 4;
    else if(i >= 60 && i <= 67)
      j = i - 56;
  }
  else if(id == 4)
  { if(i >= 12 && i <= 15)
      j = i - 12;
    else if(i >= 52 && i <= 59)
      j = i - 48;
  }
  else
    j = -1;

  if(j >= 0)
    XmToggleButtonSetState(toggle[j], TRUE, FALSE);
}     /*   end mtrl0()   */

/***********************************************************************
 *                          VOID MTRL1
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  mtrl1
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  mtrlCB
 *=====================================================================
 *<Description>
 *    Presents Maritime Aerosol Material Group Options.
 *=====================================================================
 *<Called routines>
 *    create_rowcol          - creates a Rowcol Widget
```

```
 *    cancelCB                - removes the Rowcol Widget
 *    create_radiobox         - creates a Radiobox Widget
 *    create_togglebutton     - creates a Togglebutton Widget
 *    mtrl0CB                 - Sets the Material Option
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void mtrl1(int *indx)
 *    Input:
 *       *indx                - index to indicate desired material group
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void mtrl1(int *indx)
{
  Widget radio, toggle[12], rowcol;
  int i, j, id;
  static int index[12], nc;
  static char cat_label[25] = "Maritime Aerosol Options";
  static char tog_label[12][15] = { "LOWTRAN  0% RH", "LOWTRAN 70% RH",
                  "LOWTRAN 90% RH", "LOWTRAN 99% RH", "EOSAEL  0% RH",
                  "EOSAEL 50% RH", "EOSAEL 70% RH", "EOSAEL 80% RH",
                  "EOSAEL 90% RH", "EOSAEL 95% RH", "EOSAEL 98% RH",
                  "EOSAEL 99% RH" };

  id = (*indx) % 100;

/*-------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Maritime_Aerosols", cancelCB);

/*-------------------------------------------------------------------
 * --- Create the radiobox and toggles
 *-------------------------------------------------------------------
 */
  nc = 1;
  radio = create_radiobox(rowcol, &nc, cat_label);

  for (j=0; j<12; j++)
  { index[j] = 100 * (*indx) + j;
    toggle[j] = create_togglebutton(radio, tog_label[j], &index[j],
                mtrl0CB);
  }

  i = mtrl_lmtl[cur_regn][cur_mtl];
  if(id == 3)
  { if(i >= 8 && i <= 11)
      j = i - 8;
    else if(i >= 44 && i <= 51)
      j = i - 40;
  }
  else
    j = -1;
```

```
  if(j >= 0)
    XmToggleButtonSetState(toggle[j], TRUE, FALSE);
}     /*   end mtrl1()   */

/*********************************************************************
 *                          VOID MTRL2
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  mtrl2
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  mtrlCB
 *====================================================================
 *<Description>
 *    Presents Tropospheric Aerosol Material Group Options.
 *====================================================================
 *<Called routines>
 *    create_rowcol         - creates a Rowcol Widget
 *    cancelCB              - removes the Rowcol Widget
 *    create_radiobox       - creates a Radiobox Widget
 *    create_togglebutton   - creates a Togglebutton Widget
 *    mtrl0CB               - Sets the Material Option
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void mtrl2(int *indx)
 *    Input:
 *       *indx              - index to indicate desired material group
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void mtrl2(int *indx)
{
  Widget radio, toggle[4], rowcol;
  int i, j, id;
  static int index[4], nc;
  static char cat_label[29] = "Tropospheric Aerosol Options";
  static char tog_label[4][15] = { "LOWTRAN  0% RH", "LOWTRAN 70% RH",
                                   "LOWTRAN 90% RH", "LOWTRAN 99% RH" };

  id = (*indx) % 100;

/*-------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Tropospheric_Aerosols", cancelCB);

/*-------------------------------------------------------------------
 * --- Create the radiobox and toggles
 *-------------------------------------------------------------------
 */
  nc = 1;
  radio = create_radiobox(rowcol, &nc, cat_label);
```

```
  for (j=0; j<4; j++)
  { index[j] = 100 * (*indx) + j;
    toggle[j] = create_togglebutton(radio, tog_label[j], &index[j],
                mtrl0CB);
  }

  i = mtrl_lmtl[cur_regn][cur_mtl];
  if(id == 5)
  { if(i >= 20 && i <= 23)
      j = i - 20;
  }
  else
    j = -1;

  if(j >= 0)
    XmToggleButtonSetState(toggle[j], TRUE, FALSE);
}     /*   end mtrl2()    */

/*********************************************************************
 *                          VOID MTRL3
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  mtrl3
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  mtrlCB
 *====================================================================
 *<Description>
 *    Presents Volcanic Aerosol Material Group Options.
 *====================================================================
 *<Called routines>
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelCB               - removes the Rowcol Widget
 *    create_radiobox        - creates a Radiobox Widget
 *    create_togglebutton    - creates a Togglebutton Widget
 *    mtrl0CB                - Sets the Material Option
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void mtrl3(int *indx)
 *    Input:
 *       *indx               - index to indicate desired material group
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void mtrl3(int *indx)
{
  Widget radio, toggle[2], rowcol;
  int i, j, id;
  static int index[2], nc;
  static char cat_label[25] = "Volcanic Aerosol Options";
  static char tog_label[2][14] = { "LOWTRAN Aged", "LOWTRAN Fresh" };

  id = (*indx) % 100;
```

```
/*-----------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-----------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Volcanic_Aerosols", cancelCB);

/*-----------------------------------------------------------------------
 * --- Create the radiobox and toggles
 *-----------------------------------------------------------------------
 */
  nc = 1;
  radio = create_radiobox(rowcol, &nc, cat_label);

  for (j=0; j<2; j++)
  { index[j] = 100 * (*indx) + j;
    toggle[j] = create_togglebutton(radio, tog_label[j], &index[j],
                mtrl0CB);
  }

  i = mtrl_lmtl[cur_regn][cur_mtl];
  if(id == 7)
  { if(i >= 25 && i <= 26)
      j = i - 25;
  }
  else
    j = -1;

  if(j >= 0)
    XmToggleButtonSetState(toggle[j], TRUE, FALSE);
}     /*   end mtrl3()   */

/***********************************************************************
 *                          VOID MTRL4
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  mtrl4
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  mtrlCB
 *=====================================================================
 *<Description>
 *    Presents Fog Aerosol Material Group Options.
 *=====================================================================
 *<Called routines>
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelCB               - removes the Rowcol Widget
 *    create_radiobox        - creates a Radiobox Widget
 *    create_togglebutton    - creates a Togglebutton Widget
 *    mtrl0CB                - Sets the Material Option
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void mtrl4(int *indx)
 *    Input:
 *       *indx               - index to indicate desired material group
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
```

```
 *<End>
 ***********************************************************************
 */
void mtrl4(int *indx)
{
  Widget radio, toggle[4], rowcol;
  int i, j, id;
  static int index[4], nc;
  static char cat_label[20] = "Fog Aerosol Options";
  static char tog_label[4][26] = { "LOWTRAN Radiative",
                                   "LOWTRAN Advective",
                                   "EOSAEL Heavy Advection",
                                   "EOSAEL Moderate Radiation" };

  id = (*indx) % 100;

/*-----------------------------------------------------------------------
 * --- Creae  a RowColumn Widget
 *-----------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Fog_Aerosols", cancelCB);

/*-----------------------------------------------------------------------
 * --- Create the radiobox and toggles
 *-----------------------------------------------------------------------
 */
  nc = 1;
  radio = create_radiobox(rowcol, &nc, cat_label);

  for (j=0; j<4; j++)
  { index[j] = 100 * (*indx) + j;
    toggle[j] = create_togglebutton(radio, tog_label[j], &index[j],
                mtrl0CB);
  }

  i = mtrl_lmtl[cur_regn][cur_mtl];
  if(id == 9)
  { if(i >= 27 && i <= 28)
      j = i - 27;
    else if(i >= 68 && i <= 69)
      j = i - 66;
  }
  else
    j = -1;

  if(j >= 0)
    XmToggleButtonSetState(toggle[j], TRUE, FALSE);
}     /*   end mtrl4()   */

/***********************************************************************
 *                           VOID MTRL5
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  mtrl5
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  mtrlCB
 *=====================================================================
 *<Description>
 *    Presents Clouds Material Group Options.
 *=====================================================================
 *<Called routines>
```

```
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelCB               - removes the Rowcol Widget
 *    create_radiobox        - creates a Radiobox Widget
 *    create_togglebutton    - creates a Togglebutton Widget
 *    mtrl0CB                - Sets the Material Option
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void mtrl5(int *indx)
 *    Input:
 *       *indx               - index to indicate desired material group
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void mtrl5(int *indx)
{
  Widget radio, toggle[10], rowcol;
  int i, j, id;
  static int index[10], nc;
  static char cat_label[14] = "Cloud Options";
  static char tog_label[10][27] = { "Deirmendjian Model C",
       "LOWTRAN Cumulus",            "LOWTRAN Altostratus",
       "LOWTRAN Stratus",            "LOWTRAN Stratus/Strato",
       "LOWTRAN Nimbostratus",       "LOWTRAN Standard Cirrus",
       "LOWTRAN Subvisual Cirrus",   "EOSAEL Fairweather Cumulus",
       "EOSAEL Cumulus Congestus" };

  id = (*indx) % 100;

/*-------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Cloud_Options", cancelCB);

/*-------------------------------------------------------------------
 * --- Create the radiobox and toggles
 *-------------------------------------------------------------------
 */
  nc = 1;
  radio = create_radiobox(rowcol, &nc, cat_label);

  for (j=0; j<10; j++)
  { index[j] = 100 * (*indx) + j;
    toggle[j] = create_togglebutton(radio, tog_label[j], &index[j],
                mtrl0CB);
  }

  i = mtrl_lmtl[cur_regn][cur_mtl];
  if(id == 10)
  { if(i == 1)
      j = 0;
    else if(i >= 30 && i <= 36)
      j = i - 29;
    else if(i >= 83 && i <= 84)
```

```
      j = i - 75;
    }
    else
      j = -1;

    if(j >= 0)
      XmToggleButtonSetState(toggle[j], TRUE, FALSE);
}     /*   end mtrl5()   */

/*********************************************************************
 *                          VOID MTRL6
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  mtrl6
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  mtrlCB
 *====================================================================
 *<Description>
 *    Presents Rain Material Group Options.
 *====================================================================
 *<Called routines>
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelCB               - removes the Rowcol Widget
 *    create_radiobox        - creates a Radiobox Widget
 *    create_togglebutton    - creates a Togglebutton Widget
 *    mtrl0CB                - Sets the Material Option
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void mtrl6(int *indx)
 *    Input:
 *       *indx               - index to indicate desired material group
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void mtrl6(int *indx)
{
  Widget radio, toggle[3], rowcol;
  int i, j, id;
  static int index[3], nc;
  static char cat_label[13] = "Rain Options";
  static char tog_label[3][20] = { "EOSAEL Drizzle",
                                   "EOSAEL Widespread",
                                   "EOSAEL Thunderstorm" };

  id = (*indx) % 100;

/*-------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Rain_Options", cancelCB);

/*-------------------------------------------------------------------
```

```
 * --- Create the radiobox and toggles
 *-----------------------------------------------------------------------
 */
  nc = 1;
  radio = create_radiobox(rowcol, &nc, cat_label);

  for (j=0; j<3; j++)
  { index[j] = 100 * (*indx) + j;
    toggle[j] = create_togglebutton(radio, tog_label[j], &index[j],
                mtrl0CB);
  }

  i = mtrl_lmtl[cur_regn][cur_mtl];
  if(id == 11)
  { if(i >= 70 && i <= 72)
      j = i - 70;
  }
  else
    j = -1;

  if(j >= 0)
    XmToggleButtonSetState(toggle[j], TRUE, FALSE);
}     /*   end mtrl6()    */

/***********************************************************************
 *                          VOID MTRL7
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  mtrl7
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  mtrlCB
 *=====================================================================
 *<Description>
 *    Presents Desert Aerosol Material Group Options.
 *=====================================================================
 *<Called routines>
 *    create_rowcol         - creates a Rowcol Widget
 *    cancelCB              - removes the Rowcol Widget
 *    create_radiobox       - creates a Radiobox Widget
 *    create_togglebutton   - creates a Togglebutton Widget
 *    mtrl0CB               - Sets the Material Option
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void mtrl7(int *indx)
 *    Input:
 *       *indx              - index to indicate desired material group
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void mtrl7(int *indx)
{
  Widget radio, toggle[4], rowcol;
  int i, j, id;
```

```
  static int index[4], nc;
  static char cat_label[23] = "Desert Aerosol Options";
  static char tog_label[4][20] = { "LOWTRAN Wind  0 mps",
                                   "LOWTRAN Wind 10 mps",
                                   "LOWTRAN Wind 20 mps",
                                   "LOWTRAN Wind 30 mps" };

  id = (*indx) % 100;

/*-----------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-----------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Desert_Aerosols", cancelCB);

/*-----------------------------------------------------------------------
 * --- Create the radiobox and toggles
 *-----------------------------------------------------------------------
 */
  nc = 1;
  radio = create_radiobox(rowcol, &nc, cat_label);

  for (j=0; j<4; j++)
  { index[j] = 100 * (*indx) + j;
    toggle[j] = create_togglebutton(radio, tog_label[j], &index[j],
                mtrl0CB);
  }

  i = mtrl_lmtl[cur_regn][cur_mtl];
  if(id == 13)
  { if(i >= 37 && i <= 40)
      j = i - 37;
  }
  else
    j = -1;

  if(j >= 0)
    XmToggleButtonSetState(toggle[j], TRUE, FALSE);
}     /*   end mtrl7()    */

/***********************************************************************
 *                          VOID MTRL8
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  mtrl8
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  mtrlCB
 *=====================================================================
 *<Description>
 *    Presents Dust and Dirt Material Group Options.
 *=====================================================================
 *<Called routines>
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelCB               - removes the Rowcol Widget
 *    create_radiobox        - creates a Radiobox Widget
 *    create_togglebutton    - creates a Togglebutton Widget
 *    mtrl0CB                - Sets the Material Option
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void mtrl8(int *indx)
```

```
 *    Input:
 *       *indx                - index to indicate desired material group
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void mtrl8(int *indx)
{
  Widget radio, toggle[3], rowcol;
  int i, j, id;
  static int index[3], nc;
  static char cat_label[22] = "Dust and Dirt Options";
  static char tog_label[3][26] = { "Dirt",
                                   "EOSAEL Dust Light Loading",
                                   "EOSAEL Dust Heavy Loading" };

  id = (*indx) % 100;

/*-------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Dust_&_Dirt_Options", cancelCB);

/*-------------------------------------------------------------------
 * --- Create the radiobox and toggles
 *-------------------------------------------------------------------
 */
  nc = 1;
  radio = create_radiobox(rowcol, &nc, cat_label);

  for (j=0; j<3; j++)
  { index[j] = 100 * (*indx) + j;
    toggle[j] = create_togglebutton(radio, tog_label[j], &index[j],
                mtrl0CB);
  }

  i = mtrl_lmtl[cur_regn][cur_mtl];
  if(id == 14)
  { if(i == 0)
      j = 0;
    else if(i >= 93 && i <= 94)
      j = i - 92;
  }
  else
    j = -1;

  if(j >= 0)
    XmToggleButtonSetState(toggle[j], TRUE, FALSE);
}     /*   end mtrl8()   */

/*******************************************************************
 *                          VOID MTRL9
 *******************************************************************
 *<Begin>
 *<Identification>          Name:  mtrl9
```

```
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  mtrlCB
 *=====================================================================
 *<Description>
 *    Presents Combat Dust and Smoke Material Group Options.
 *=====================================================================
 *<Called routines>
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelCB               - removes the Rowcol Widget
 *    create_radiobox        - creates a Radiobox Widget
 *    create_togglebutton    - creates a Togglebutton Widget
 *    mtrl0CB                - Sets the Material Option
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void mtrl9(int *indx)
 *    Input:
 *       *indx               - index to indicate desired material group
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void mtrl9(int *indx)
{
  Widget radio, toggle[6], rowcol;
  int i, j, id;
  static int index[6], nc;
  static char cat_label[30] = "Combat Dust and Smoke Options";
  static char tog_label[6][27] = { "EOSAEL High Explosive Dust",
                                   "EOSAEL WP Smoke 17% RH",
                                   "EOSAEL WP Smoke 50% RH",
                                   "EOSAEL WP Smoke 90% RH",
                                   "EOSAEL Fog Oil",
                                   "EOSAEL HC Smoke 85% RH" };

  id = (*indx) % 100;

/*-------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Combat_Dust_&_Smoke_Options", cancelCB);

/*-------------------------------------------------------------------
 * --- Create the radiobox and toggles
 *-------------------------------------------------------------------
 */
  nc = 1;
  radio = create_radiobox(rowcol, &nc, cat_label);

  for (j=0; j<6; j++)
  { index[j] = 100 * (*indx) + j;
    toggle[j] = create_togglebutton(radio, tog_label[j], &index[j],
                mtrl0CB);
  }
```

```
  i = mtrl_lmtl[cur_regn][cur_mtl];
  if(id == 15)
  { if(i >= 95 && i <= 100)
      j = i - 95;
  }
  else
    j = -1;

  if(j >= 0)
    XmToggleButtonSetState(toggle[j], TRUE, FALSE);
}     /*   end mtrl9()    */

/*********************************************************************
 *                          VOID MTRL0CB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  mtrl0CB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  mtrl0, mtrl1, mtrl2, mtrl3, mtrl4,
 *                                 mtrl5, mtrl6, mtrl7, mtrl8, mtrl9
 *====================================================================
 *<Description>
 *    Sets the Material Option
 *====================================================================
 *<Called routines>
 *    none
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void mtrl0CB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - pointer to the data passed to the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void mtrl0CB (Widget w, XtPointer c, XtPointer call_data)
{
  int *n = (int *)c;
  int grp, i;

  grp = (*n) % 10000;
  grp = grp / 100;
  i = (*n) % 100;

  if(grp == 2)
  { if(i < 4)
      mtrl_lmtl[cur_regn][cur_mtl] = i + 4;
    else
      mtrl_lmtl[cur_regn][cur_mtl] = i + 56;
```

```
    }
    else if(grp == 3)
    { if(i < 4)
        mtrl_lmtl[cur_regn][cur_mtl] = i + 8;
      else
        mtrl_lmtl[cur_regn][cur_mtl] = i + 40;
    }
    else if(grp == 4)
    { if(i < 4)
        mtrl_lmtl[cur_regn][cur_mtl] = i + 12;
      else
        mtrl_lmtl[cur_regn][cur_mtl] = i + 48;
    }
    else if(grp == 5)
      mtrl_lmtl[cur_regn][cur_mtl] = i + 20;
    else if(grp == 7)
      mtrl_lmtl[cur_regn][cur_mtl] = i + 25;
    else if(grp == 9)
    { if(i < 2)
        mtrl_lmtl[cur_regn][cur_mtl] = i + 27;
      else
        mtrl_lmtl[cur_regn][cur_mtl] = i + 66;
    }
    else if(grp == 10)
    { if(i == 0)
        mtrl_lmtl[cur_regn][cur_mtl] = 1;
      else if(i < 8)
        mtrl_lmtl[cur_regn][cur_mtl] = i + 29;
      else
        mtrl_lmtl[cur_regn][cur_mtl] = i + 75;
    }
    else if(grp == 11)
      mtrl_lmtl[cur_regn][cur_mtl] = i + 70;
    else if(grp == 13)
      mtrl_lmtl[cur_regn][cur_mtl] = i + 37;
    else if(grp == 14)
    { if(i == 0)
        mtrl_lmtl[cur_regn][cur_mtl] = 0;
      else
        mtrl_lmtl[cur_regn][cur_mtl] = i + 92;
    }
    else if(grp == 15)
      mtrl_lmtl[cur_regn][cur_mtl] = i + 95;

    in_change = TRUE;
    new_file = TRUE;
  }     /*   end mtrl0CB()   */

/*********************************************************************
 *                          VOID OBSC
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  obsc
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  getdata, cancelobCB, obscCB,
 *                                 cancelbCB, area_delCB, regn_delCB,
 *                                 resizeCB, cancelbbCB
 *=====================================================================
 *<Description>
 *    Sets up a rowcolumn widget containing the names of the materials
 *=====================================================================
```

```
 *<Called routines>
 *    plot_out_def1          - Creates a Widget telling the user what
 *                             flux info he is viewing.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void obsc( int *regn_add, int *area_add, int *regn_del,
 *                  int *area_del )
 *    Input:
 *       regn_add            - the added region number
 *       area_add            - the added area number
 *       regn_del            - the deleted region number
 *       area_del            - the deleted area number
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void obsc ( void )
{
  static Widget bboard1 = (Widget)0;
  static Widget bboard2 = (Widget)0;
  static Widget bboard3 = (Widget)0;
  static Widget list1, list2, rowcol, button;
  Arg wargs[20];
  int n, i, j;
  static int numitems1 = 7*IRM;
  static int numitems2 = 3*IAM;
  static int visitem1 = 7;
  static int visitem2 = 3;
  static XmString materials[7*IRM], albedos[3*IAM];
  static char button_label[40], *fname;
  static char *slash = "/";
  char *ptr;

  if(!new_file)
  { fname = ptr = file_name;
    for (i=0; i<strlen(file_name); i++, ptr++)
    { if(ptr[0] == slash[0])
      { fname = ptr;
        fname++;
      }
    }
  }

  if(bboard1 != (Widget)0)
  { XtUnmanageChild(list1);
    XtDestroyWidget(list1);
    XtUnmanageChild(bboard1);
    bboard1 = (Widget)0;
  }
  if(bboard2 != (Widget)0)
  { XtUnmanageChild(list2);
    XtDestroyWidget(list2);
    XtUnmanageChild(bboard2);
    bboard2 = (Widget)0;
  }
```

```
if(bboard3 != (Widget)0)
{ XtUnmanageChild(rowcol);
  XtDestroyWidget(rowcol);
  XtUnmanageChild(bboard3);
  bboard3 = (Widget)0;
}

if(label_obsc)
{ for (i=0; i<IRM; i++)
  { if(i <= regn)
    { sprintf(button_label, "Region - %d", i+1);
      materials[7*i] = XmStringLtoRCreate( button_label, charset);

      for (j=0; j<MXMTR; j++)
      { sprintf(button_label, "MTL %d: %s", j+1,
                mtl_obsc[(int)mtrl_lmtl[i][j]]);
        materials[7*i + 2*j + 1] =
                XmStringLtoRCreate( button_label, charset);

        if(mtrl_wmtl[i][j] > 0.0)
          sprintf(button_label, "VIS %d: %10.4f", j+1,
                  1.0/mtrl_wmtl[i][j]);
        else
          sprintf(button_label, "VIS %d: %10.4f", j+1, 0.0);
        materials[7*i + 2*j + 2] =
                XmStringLtoRCreate( button_label, charset);
      }
    }
    else
    { for (j=0; j<7; j++)
        materials[7*i + j] = XmStringLtoRCreate( " ", charset);
    }
  }

  for (i=0; i<IAM; i++)
  { if(i <= area)
    { sprintf(button_label, "Area - %d", i+1);
      albedos[3*i] = XmStringLtoRCreate( button_label, charset);

      if(mdl2_ialb < 0)
      { sprintf(button_label, "User-Defined Albedo");
        albedos[3*i + 1] = XmStringLtoRCreate( button_label, charset);
        sprintf(button_label, "Value: %10.4f", albd_falb[i]);
        albedos[3*i + 2] = XmStringLtoRCreate( button_label, charset);
      }
      else if(mdl2_ialb == 0)
      { sprintf(button_label, "Broad-Band Albedo");
        albedos[3*i + 1] = XmStringLtoRCreate( button_label, charset);
        sprintf(button_label, "%s", broad_band[(int)area_iamtl[i]]);
        albedos[3*i + 2] = XmStringLtoRCreate( button_label, charset);
      }
      else
      { sprintf(button_label, "Spectral Albedo");
        albedos[3*i + 1] = XmStringLtoRCreate( button_label, charset);
        sprintf(button_label, "Model %d", (int)area_iamtl[i]);
        albedos[3*i + 2] = XmStringLtoRCreate( button_label, charset);
      }
    }
    else
    { for (j=0; j<3; j++)
        albedos[3*i + j] = XmStringLtoRCreate( " ", charset);
    }
```

```
}

if(mtrl_form == (Widget)0)
{ n = 0;
  XtSetArg(wargs[n], XmNbottomAttachment, XmATTACH_FORM); n++;
  XtSetArg(wargs[n], XmNrightAttachment, XmATTACH_FORM); n++;
  XtSetArg(wargs[n], XmNrightOffset, 250); n++;
  XtSetArg(wargs[n], XmNbottomOffset, 0); n++;
  mtrl_form = XmCreateForm(form, "mtrl_form", wargs, n);
  XtManageChild(mtrl_form);
}

if(albd_form == (Widget)0)
{ n = 0;
  XtSetArg(wargs[n], XmNbottomAttachment, XmATTACH_FORM); n++;
  XtSetArg(wargs[n], XmNleftAttachment, XmATTACH_FORM); n++;
  XtSetArg(wargs[n], XmNleftOffset, 0); n++;
  XtSetArg(wargs[n], XmNbottomOffset, 0); n++;
  albd_form = XmCreateForm(form, "albd_form", wargs, n);
  XtManageChild(albd_form);
}

if(file_form == (Widget)0)
{ n = 0;
  XtSetArg(wargs[n], XmNbottomAttachment, XmATTACH_FORM); n++;
  XtSetArg(wargs[n], XmNleftAttachment, XmATTACH_FORM); n++;
  XtSetArg(wargs[n], XmNleftOffset, 410); n++;
  XtSetArg(wargs[n], XmNbottomOffset, 0); n++;
  file_form = XmCreateForm(form, "file_form", wargs, n);
  XtManageChild(file_form);
}

n = 0;
XtSetArg(wargs[n], XmNdialogStyle, XmDIALOG_MODELESS); n++;
XtSetArg(wargs[n], XmNwidth, 215); n++;
XtSetArg(wargs[n], XmNheight, 168); n++;
bboard1 = XmCreateBulletinBoardDialog(mtrl_form, "Region Materials",
          wargs, n);
XtManageChild(bboard1);

n = 0;
XtSetArg(wargs[n], XmNdialogStyle, XmDIALOG_MODELESS); n++;
XtSetArg(wargs[n], XmNwidth, 215); n++;
XtSetArg(wargs[n], XmNheight, 85); n++;
bboard2 = XmCreateBulletinBoardDialog(albd_form, "Area Aerosols",
          wargs, n);
XtManageChild(bboard2);

n = 0;
XtSetArg(wargs[n], XmNdialogStyle, XmDIALOG_MODELESS); n++;
XtSetArg(wargs[n], XmNwidth, 215); n++;
XtSetArg(wargs[n], XmNheight, 50); n++;
bboard3 = XmCreateBulletinBoardDialog(file_form, "Filename", wargs,
          n);
XtManageChild(bboard3);

n = 0;
XtSetArg(wargs[n], XmNlistSpacing, 0); n++;
XtSetArg(wargs[n], XmNmarginHeight, 0); n++;
XtSetArg(wargs[n], XmNmarginWidth, 0); n++;
XtSetArg(wargs[n], XmNlistSizePolicy, XmVARIABLE); n++;
XtSetArg(wargs[n], XmNitemCount, numitems1); n++;
```

```
    XtSetArg(wargs[n], XmNvisibleItemCount, visitem1); n++;
    XtSetArg(wargs[n], XmNitems, materials); n++;
    XtSetArg(wargs[n], XmNscrollBarPlacement, XmBOTTOM_RIGHT); n++;
    XtSetArg(wargs[n], XmNscrollBarDisplayPolicy, XmSTATIC); n++;
    list1 = XmCreateScrolledList(bboard1, "list1", wargs, n);
    XtManageChild(list1);

    n = 0;
    XtSetArg(wargs[n], XmNlistSpacing, 0); n++;
    XtSetArg(wargs[n], XmNmarginHeight, 0); n++;
    XtSetArg(wargs[n], XmNmarginWidth, 0); n++;
    XtSetArg(wargs[n], XmNlistSizePolicy, XmVARIABLE); n++;
    XtSetArg(wargs[n], XmNitemCount, numitems2); n++;
    XtSetArg(wargs[n], XmNvisibleItemCount, visitem2); n++;
    XtSetArg(wargs[n], XmNitems, albedos); n++;
    XtSetArg(wargs[n], XmNscrollBarPlacement, XmBOTTOM_RIGHT); n++;
    XtSetArg(wargs[n], XmNscrollBarDisplayPolicy, XmSTATIC); n++;
    list2 = XmCreateScrolledList(bboard2, "list2", wargs, n);
    XtManageChild(list2);

    n = 0;
    XtSetArg(wargs[n], XmNorientation, XmVERTICAL); n++;
    rowcol = XtCreateManagedWidget("Filename",
             xmRowColumnWidgetClass, bboard3, wargs, n);

    sprintf(button_label, "%s", fname);
    n = 0;
    button = XmCreateLabel(rowcol, button_label, wargs, n);
    XtManageChild (button);
  }

  plot_out_def1();
}     /*   end obsc()    */

/*********************************************************************
 *                          VOID PLOT_OUT_DEF1
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  plot_out_def1
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  obsc, fluxCB, cross_sectionCB,
 *                                 planeCB, waveCB
 *====================================================================
 *<Description>
 *    Creates a Widget telling the user what flux info he is viewing.
 *====================================================================
 *<Called routines>
 *    none
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void plot_out_def1(void)
 *    Input:
 *       None
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
```

```
 *<End>
 ***********************************************************************
 */
void plot_out_def1(void)
{
  static Widget bboard1 = (Widget)0;
  static Widget list1;
  Arg wargs[20];
  int n, which_flux;
  static XmString fluxes[4];
  static char button_label[40];
  static double ten = 10.0;
  static char *label[] = { "Solar Direct Flux",
                           "Solar Reflected Flux",
                           "Solar Diffuse Flux - 1",
                           "Solar Diffuse Flux - 2",
                           "Solar Diffuse Flux - 3",
                           "Solar Diffuse Flux - 4",
                           "Solar Diffuse Flux - 5",
                           "Solar Diffuse Flux - 6",
                           "Solar Diffuse Flux - 7",
                           "Solar Diffuse Flux - 8"
                         };
  static char *axscale[] = { "on a Log Scale",
                             "multiplied by 100,000",
                             "multiplied by 10,000",
                             "multiplied by 1000",
                             "multiplied by 100",
                             "multiplied by 10",
                             "",
                             "divided by 10",
                             "divided by 100",
                             "divided by 1000",
                             "divided by 10,000",
                             "divided by 100,000",
                             "divided by 1,000,000"
                           };
  static int pos_scale = 6;              /* Log10 (Max Flux Divisor) */
  static int neg_scale = -5;             /* Log10 (Max Flux Multiplr)*/
  int idx;


  if(bboard1 != (Widget)0)
  { XtUnmanageChild(list1);
    XtDestroyWidget(list1);
    XtUnmanageChild(bboard1);
    bboard1 = (Widget)0;
  }

  if(!noflux)
  { for(n=0; n<10; n++)
      if(flux_flag[n])
        which_flux = n;

/*-----------------------------------------------------------------------
 * --- Determine the range of the log of the flux values and the flux
 *     scale label index
 *-----------------------------------------------------------------------
 */
    if(maxi_flux[which_flux][cur_nwave] > 0.0)
    { idx = (int)floor(log10((double)maxi_flux[which_flux][cur_nwave]));
      if(idx < neg_scale)
```

```
        idx = neg_scale;
      if(idx > pos_scale)
        idx = pos_scale;
      log_range = (float) pow( ten, (double) idx);
      idx += 1 - neg_scale;
    }
    else
      log_range = 0.0;

/*-----------------------------------------------------------------------
 * --- Get the flux parameter information for the screen and determine
 *     the minimum and maximum flux scale indices for plotting the flux
 *     axis.
 *-----------------------------------------------------------------------
 */
    if(!flux_zero[which_flux][cur_nwave]) /* Use Log Flux values      */
    { idx = 0;

      flux_index_low = mini1_flux[which_flux][cur_nwave];
      if(flux_index_low < 0 || mini1_flux[which_flux][cur_nwave] < 0.0)
        flux_index_low--;
      else
        flux_index_low = 0;

      flux_index_high = maxi1_flux[which_flux][cur_nwave];
      if(flux_index_high > 0 || maxi1_flux[which_flux][cur_nwave] > 0.0)
        flux_index_high++;
      else
        flux_index_high = 0;
    }
                                          /* Scale Flux values Down   */
    else if((mini_flux[which_flux][cur_nwave] >= 0.0) &&
            (maxi_flux[which_flux][cur_nwave] > 0.0))
    { flux_index_low = 0;
      flux_index_high = (int) (maxi_flux[which_flux][cur_nwave] /
                               log_range) + 1;
    }

    else                                  /* Scale Flux values Up     */
    { idx = 1 - neg_scale;
      flux_index_low = flux_index_high = 0;
    }

    if(label_obsc)
    {
/*-----------------------------------------------------------------------
 * --- Print the Type of Flux
 *-----------------------------------------------------------------------
 */
      sprintf(button_label, "%s", label[which_flux]);
      fluxes[0] = XmStringLtoRCreate( button_label, charset);

/*-----------------------------------------------------------------------
 * --- Print the Flux Plot Scale
 *-----------------------------------------------------------------------
 */
      sprintf(button_label, "%s", axscale[idx]);
      fluxes[1] = XmStringLtoRCreate( button_label, charset);

/*-----------------------------------------------------------------------
 * --- Compose and print the Cross-section orientation and value
 *-----------------------------------------------------------------------
```

```
 */
      if(cross_axis[0])
        sprintf(button_label, "X Cross-section = %6.3f km",
                out_m0[0][cross_value[0]]);
      else if(cross_axis[1])
        sprintf(button_label, "Y Cross-section = %6.3f km",
                out_m0[1][cross_value[1]]);
      else if(cross_axis[2])
        sprintf(button_label, "Z Cross-section = %6.3f km",
                out_m0[2][cross_value[2]]);
      fluxes[2] = XmStringLtoRCreate( button_label, charset);

/*-----------------------------------------------------------------------
 * --- Compose and print the Wavenumber value
 *-----------------------------------------------------------------------
 */
      sprintf(button_label, "Wave No. = %9.3f per cm",
              out_waveno[cur_nwave]);
      fluxes[3] = XmStringLtoRCreate( button_label, charset);

/*-----------------------------------------------------------------------
 * --- Create the "form", "BBoard", and "List" Widgets to hold the info
 *-----------------------------------------------------------------------
 */
      if(flux_form == (Widget)0)
      { n = 0;
        XtSetArg(wargs[n], XmNbottomAttachment, XmATTACH_FORM); n++;
        XtSetArg(wargs[n], XmNrightAttachment, XmATTACH_FORM); n++;
        XtSetArg(wargs[n], XmNrightOffset, 630); n++;
        XtSetArg(wargs[n], XmNbottomOffset, 0); n++;
        flux_form = XmCreateForm(form, "flux_form", wargs, n);
        XtManageChild(flux_form);
      }

      n = 0;
      XtSetArg(wargs[n], XmNdialogStyle, XmDIALOG_MODELESS); n++;
      XtSetArg(wargs[n], XmNwidth, 200); n++;
      XtSetArg(wargs[n], XmNheight, 110); n++;
      bboard1 = XmCreateBulletinBoardDialog(flux_form,
                "Flux Information", wargs, n);
      XtManageChild(bboard1);

      n = 0;
      XtSetArg(wargs[n], XmNlistSpacing, 0); n++;
      XtSetArg(wargs[n], XmNmarginHeight, 0); n++;
      XtSetArg(wargs[n], XmNmarginWidth, 0); n++;
      XtSetArg(wargs[n], XmNlistSizePolicy, XmCONSTANT); n++;
      XtSetArg(wargs[n], XmNitemCount, 4); n++;
      XtSetArg(wargs[n], XmNvisibleItemCount, 4); n++;
      XtSetArg(wargs[n], XmNitems, fluxes); n++;
      list1 = XmCreateList(bboard1, "list1", wargs, n);
      XtManageChild(list1);
    }
  }
}   /*  end plot_out_def1()  */

/***********************************************************************
 *                          VOID MTRL1CB
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  mtrl1CB
 *                          Type:  C void
```

```
 *                        Filename:  visual.c
 *                          Parent:  cancelmCB
 *=====================================================================
 *<Description>
 *    Sets up the scale for the Region Material Density
 *=====================================================================
 *<Called routines>
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelobCB             - removes the Rowcol Widget
 *    create_separator       - creates a Separator Widget
 *    create_scale           - creates a Scale Widget
 *    denmtlCB               - Gets the material density
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void mtrl1CB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                   - the ID of the widget for which the
 *                             callback is registered
 *       c                   - the data passed to the routine
 *       call_data           - a pointer to the callback structure which
 *                             contains information on why the callback
 *                             occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void mtrl1CB (Widget w, XtPointer c, XtPointer call_data)
{
  int *indx = (int *)c;
  Widget rowcol, label[6], scale;
  Arg wargs[10];
  int n, i, reg, mtl;
  static int hv, sd, wid, inc, dec, val, swid;
  static char scale_label[55];
  static char numb[6][4] = { "  0", " 20", " 40", " 60", " 80", "100" };
  static int min = 0;
  static int max = 1000;

  reg = (*indx) / 10;
  mtl = (*indx) % 10;

  sprintf(scale_label,
          "Aerosol Component Vis (km) - Material %d Region %d", mtl+1,
          reg+1);

/*-------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-------------------------------------------------------------------
 */
  rowcol = create_rowcol(w, "Mtrl_Density_Options", cancelobCB);

/*-------------------------------------------------------------------
 * --- Create the Scale
 *-------------------------------------------------------------------
 */
```

```
  hv = 0;
  sd = 2;
  create_separator(rowcol, &hv, &sd);

  wid = 560;
  inc = 0;
  dec = 1;
  swid = 538;
  if(mtrl_wmtl[reg][mtl] > 0.0)
    val = 10.0 / mtrl_wmtl[reg][mtl];
  else
    val = 0;
  scale = create_scale(rowcol, scale_label, &wid, &min, &max, &inc,
          &dec, &val, &swid, indx, denmtlCB);

  for (i=0; i<6; i++)
  { n = 0;
    XtSetArg (wargs[n], XmNwidth, 90); n++;
    label[i] = XmCreateLabel(scale, numb[i], wargs, n);
  }
  XtManageChildren(label, 6);
}     /*   end mtrl1CB()   */

/*********************************************************************
 *                          VOID DENMTLCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  denmtlCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  mtrl1CB
 *====================================================================
 *<Description>
 *    Gets the material density
 *====================================================================
 *<Called routines>
 *    none
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void denmtlCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - pointer to the data passed to the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void denmtlCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *indx = (int *)c;
  int value, i, j;
```

```
  XmScaleCallbackStruct * call_value =
                              (XmScaleCallbackStruct *) call_data;

  i = (*indx) / 10;
  j = (*indx) % 10;
  value = call_value -> value;

  if(value > 0)
    mtrl_wmtl[i][j] = 10.0 / (float)value;
  else
    mtrl_wmtl[i][j] = 0.0;
  new_file = TRUE;
}     /*   end denmtlCB()    */

/*********************************************************************
 *                          VOID REGN_MTL1
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  regn_mtl1
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  cancelrCB
 *====================================================================
 *<Description>
 *    Sets up the menu for selecting the Materials and Densities
 *====================================================================
 *<Called routines>
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelCB               - removes the Rowcol Widget
 *    create_radiobox        - creates a Radiobox Widget
 *    create_togglebutton    - creates a Togglebutton Widget
 *    regn_mtlCB             - sets the region material density
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void regn_mtl1( void )
 *    Input:
 *       none
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void regn_mtl1 (void)
{
  Widget radio, toggle[MXMTR], rowcol;
  int j;
  static int index[3], nc;
  static char tog_label[MXMTR][13];
  static char cat_label[24];

  sprintf(cat_label, "Material for Region %d", cur_regn+1);
  cur_mtl = MXMTR + 1;

  for (j=0; j<MXMTR; j++)
    sprintf(tog_label[j], "Material - %d", j+1);
```

```
/*-----------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-----------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Material_Options", cancelCB);

/*-----------------------------------------------------------------------
 * --- Create the radiobox and toggles
 *-----------------------------------------------------------------------
 */
  nc = 1;
  radio = create_radiobox(rowcol, &nc, cat_label);

  for (j=0; j<MXMTR; j++)
  { index[j] = 10 * cur_regn + j;
    toggle[j] = create_togglebutton(radio, tog_label[j], &index[j],
                regn_mtlCB);
  }

  if(cur_mtl < MXMTR)
    XmToggleButtonSetState(toggle[cur_mtl], TRUE, FALSE);
}     /*   end regn_mtl1()    */

/***********************************************************************
 *                          VOID METRNG_OPTCB
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  metrng_optCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  modelCB
 *=====================================================================
 *<Description>
 *    Selects the various Meteorological Range Options for BLIRB
 *=====================================================================
 *<Called routines>
 *    create_rowcol         - creates a Rowcol Widget
 *    cancelCB              - removes the Rowcol Widget
 *    create_separator      - creates a Separator Widget
 *    create_scale          - creates a Scale Widget
 *    metrngCB              - Gets some of the input parameters of the
 *                            MDL1 and MDL2 cards (IVIS and SN).
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void metrng_optCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - the data passed to the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
```

```
 */
void metrng_optCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *indx = (int *)c;
  Widget rowcol, label[6], scale;
  Arg wargs[10];
  int n, i, k;
  static int hv, sd, inc, wid, val, swid;
  static char scale_label[26]={ "Meteorological Range (km)" };
  static char numb[2][6][3] = { { " 0", " 1", " 2", " 3", " 4", " 5" },
                                { " 0", "10", "20", "30", "40", "50" }
                              };
  static int min = 0;
  static int max = 500;
  static int dec[2] = { 2, 1 };

  k = *indx - 1;

/*-----------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-----------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Met_Range_Options", cancelCB);

/*-----------------------------------------------------------------------
 * --- Create the Scale
 *-----------------------------------------------------------------------
 */
  hv = 0;
  sd = 2;
  create_separator(rowcol, &hv, &sd);

  wid = 560;
  inc = 1;
  swid = 538;
  if((mdl1 == 0) && (mdl2 == 0))
  { if(mdl1_ivis == 1)
      val = 500.0 * mdl2_sn;
    else if(mdl1_ivis == 2)
      val = 230.0 * mdl2_sn;
    else if(mdl1_ivis == 3)
      val = 100.0 * mdl2_sn;
    else if(mdl1_ivis == 4)
      val = 50.0 * mdl2_sn;
    else if(mdl1_ivis == 5)
      val = 20.0 * mdl2_sn;
    if(k == 0)
      val = 10 * val;
  }
  else
    val = min;
  if(val > max)
    val = max;

  scale = create_scale(rowcol, scale_label, &wid, &min, &max, &inc,
          &dec[k], &val, &swid, indx, metrngCB);

  for (i=0; i<6; i++)
  { n = 0;
    XtSetArg (wargs[n], XmNwidth, 90); n++;
    label[i] = XmCreateLabel(scale, numb[k][i], wargs, n);
  }
```

```
  XtManageChildren(label, 6);
}     /*   end metrng_optCB()    */

/*********************************************************************
 *                            VOID METRNGCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  metrngCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  metrng_optCB
 *====================================================================
 *<Description>
 *    Gets the IVIS and SN input parameters for the MDL1 and MDL2 cards
 *====================================================================
 *<Called routines>
 *    none
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void metrngCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - pointer to the data passed to the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void metrngCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *indx = (int *)c;
  float sn, dum;
  int value, ivis;
  XmScaleCallbackStruct * call_value =
                            (XmScaleCallbackStruct *) call_data;

  value = call_value -> value;
  if(*indx == 0)                          /* Met Rng <= 5 km          */
    dum = 0.01 *(float)value;
  else                                    /* Met Rng <= 50 km         */
    dum = 0.1 * (float)value;

  if(dum <= 2.0)
  { ivis = 5;
    sn = dum / 2.0;
  }
  else if(dum <= 5.0)
  { ivis = 4;
    sn = dum / 5.0;
  }
  else if(dum <= 10.0)
  { ivis = 3;
```

```
    sn = dum / 10.0;
  }
  else if(dum <= 23.0)
  { ivis = 2;
    sn = dum / 23.0;
  }
  else
  { ivis = 1;
    sn = dum / 50.0;
  }

  if(mdl1 < 0)
  { mdl1 = 0;                                  /* Default values          */
    mdl1_iaersl = 0.0;
    mdl1_model = 6.0;
    mdl1_iseasn = 0.0;
    mdl1_ivulcn = 1.0;
  }
  mdl1_ivis = ivis;

  if(mdl2 < 0)
  { mdl2 = 0;                                  /* Default values          */
    mdl2_tbound = 288.2;
    mdl2_ialb = -1.0;
    mdl2_ip = 0.0;
    cur_ialb = mdl2_ialb;
  }
  mdl2_sn = sn;

  if(dum <= 5.0)
    metrng_indx = 1;
  else
    metrng_indx = 2;

  new_file = TRUE;                             /* inputs changed          */
}     /*   end metrngCB()    */

/*********************************************************************
 *                          VOID SUN_OPTCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  sun_optCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_modifymenu
 *====================================================================
 *<Description>
 *    Selects the various Sun Input Options for BLIRB
 *====================================================================
 *<Called routines>
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelsCB              - removes the Rowcol Widget
 *    create_separator       - creates a Separator Widget
 *    create_radiobox        - creates a Radiobox Widget
 *    create_togglebutton    - creates a Togglebutton Widget
 *    create_scale           - creates a Scale Widget
 *    suninCB                - Set the input parameters of the SUN card
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void sun_optCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
```

```
 *       w                    - the ID of the widget for which the
 *                              callback is registered
 *       c                    - the data passed to the routine
 *       call_data            - a pointer to the callback structure which
 *                              contains information on why the callback
 *                              occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void sun_optCB (Widget w, XtPointer c, XtPointer call_data)
{
  Widget radio[3], toggle[6], rowcol, label[2][10], scale[2];
  Arg wargs[10];
  int n, i, j, k;
  static int index[8], hv, sd, nc, wid, inc, dec, val[2], swid;
  static char cat_label[3][36] = {"Solar Flux and Sky Radiance at 5 Km",
                                  "Sky Radiance Input",
                                  "Spectral Molecular Transmission" };
  static char tog_label[6][14] = { "Parameterized", "LOWTRAN",
                                   "No", "Yes",
                                   "No", "Yes" };
  static char scale_label[2][26]={ "Solar Zenith Angle (deg)",
                                   "Solar Azimuth Angle (deg)" };
  static int min[2] = { 0, 0 };
  static int max[2] = { 90, 360 };
  static char numb[2][10][4] = {
     {"0", "10", "20", "30", "40", "50", "60", "70", "80", "90"},
     {"0", "40", "80", "120", "160", "200", "240", "280", "320", "360"}
     };

/*-------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-------------------------------------------------------------------
 */
  rowcol = create_rowcol(w, "Sun_Options", cancelsCB);

/*-------------------------------------------------------------------
 * --- Create the radioboxes and toggles
 *-------------------------------------------------------------------
 */
  hv = 0;
  sd = 2;
  nc = 1;
  k = 0;
  for (i=0; i<3; i++)
  { create_separator(rowcol, &hv, &sd);
    radio[i] = create_radiobox(rowcol, &nc, cat_label[i]);

    for (j=0; j<2; k++, j++)
    { index[k] = 10*i + j;
      toggle[k] = create_togglebutton(radio[i], tog_label[k], &index[k],
                  suninCB);
    }

    if(sun == 0)
```

```
    { if(i == 0)
        XmToggleButtonSetState(toggle[(int) sun_ifsun], TRUE, FALSE);
      else if(i == 1)
        XmToggleButtonSetState(toggle[2 + (int) sun_isky], TRUE, FALSE);
      else if(i == 2)
        XmToggleButtonSetState(toggle[4 + (int) sun_iftrn], TRUE,FALSE);
    }
  }

/*-----------------------------------------------------------------------
 * --- Create the Scales
 *-----------------------------------------------------------------------
 */
  for (i=0; i<2; k++, i++)
  { create_separator(rowcol, &hv, &sd);

    index[k] = 10*(i+3);
    wid = 780;
    inc = 1;
    dec = 0;
    swid = 758;

    if(sun == 0)
    { if(i == 0)
        val[i] = sun_thsun;
      else if(i == 1)
        val[i] = sun_phsun;
    }
    else
      val[i] = min[i];

    scale[i] = create_scale(rowcol, scale_label[i], &wid, &min[i],
               &max[i], &inc, &dec, &val[i], &swid, &index[k],
               suninCB);

    for (j=0; j<10; j++)
    { n = 0;
      XtSetArg (wargs[n], XmNwidth, 45); n++;
      label[i][j] = XmCreateLabel(scale[i], numb[i][j], wargs, n);
    }
    XtManageChildren(label[i], 10);
  }
}     /*   end sun_optCB()    */

/**********************************************************************
 *                          VOID SUNINCB
 **********************************************************************
 *<Begin>
 *<Identification>          Name:  suninCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  sun_optCB
 *=====================================================================
 *<Description>
 *    Sets the input parameters of the SUN card
 *=====================================================================
 *<Called routines>
 *    none
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void suninCB( Widget w, XtPointer c, XtPointer call_data)
```

```
 *    Input:
 *       w                    - the ID of the widget for which the
 *                              callback is registered
 *       c                    - pointer to the data passed to the routine
 *       call_data            - a pointer to the callback structure which
 *                              contains information on why the callback
 *                              occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void suninCB(Widget w, XtPointer c, XtPointer call_data)
{
  int *n = (int *)c;
  int i, j, value;
  XmScaleCallbackStruct * call_value =
                            (XmScaleCallbackStruct *) call_data;

  value = call_value -> value;

  i = (*n)/10;                          /* RadioBox category        */
  j = (*n) % 10;                        /* Toggle button pushed     */

  if(sun < 0)
  { sun = 0;
    sun_thsun = 0.0;
    sun_phsun = 0.0;
    sun_ifsun = 1.0;
    sun_isky = 0.0;
    sun_iftrn = 0.0;
  }

  if(i == 0)
    sun_ifsun = j;
  else if(i == 1)
    sun_isky = j;
  else if(i == 2)
    sun_iftrn = j;
  else if(i == 3)
  { sun_thsun = value;
    in_change = TRUE;
  }
  else if(i == 4)
  { sun_phsun = value;
    in_change = TRUE;
  }

  new_file = TRUE;
}     /*   end suninCB()    */

/***********************************************************************
 *                          VOID FLAR_OPTCB
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  flar_optCB
 *                          Type:  C void
```

```
 *                        Filename:  visual.c
 *                          Parent:  create_flar1menu
 *=====================================================================
 *<Description>
 *    Selects the various Flare Input Options for BLIRB
 *=====================================================================
 *<Called routines>
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelfCB              - removes the Rowcol Widget
 *    create_separator       - creates a Separator Widget
 *    create_radiobox        - creates a Radiobox Widget
 *    create_togglebutton    - creates a Togglebutton Widget
 *    create_scale           - creates a Scale Widget
 *    flarinCB               - Set the input parameters of the Flare card
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void flar_optCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                   - the ID of the widget for which the
 *                             callback is registered
 *       c                   - the data passed to the routine
 *       call_data           - a pointer to the callback structure which
 *                             contains information on why the callback
 *                             occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void flar_optCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *indx = (int *)c;
  Widget radio, toggle[3], rowcol, label[2][10], scale[2];
  Arg wargs[10];
  int n, i, j, k, kk, ind;
  static int index[5], hv, sd, nc, wid, inc, dec, val[2], swid;
  static char cat_label[11] = "Flare Type";
  static char tog_label[3][18] = { "Isotropic", "10% Up & 90% Down",
                                   "User Defined" };
  static char scale_label[2][24]={ "Flare Intensity (watts)",
                                   "Flare Temperature (K)" };
  static int min[2] = { 0, 1500 };
  static int max[2] = { 100000, 7500 };
  static char numb[2][7][7] = {
     {"0", "20000", "40000", "60000", "80000", "100000", " "},
     {"1500", "2500", "3500", "4500", "5500", "6500", "7500"}
     };

  ind = *indx;

/*-------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Flare_Options", cancelfCB);
```

```
/*-----------------------------------------------------------------------
 * --- Create the radioboxes and toggles
 *-----------------------------------------------------------------------
 */
  hv = 0;
  sd = 2;
  nc = 1;
  kk = 0;
  create_separator(rowcol, &hv, &sd);
  radio = create_radiobox(rowcol, &nc, cat_label);

  for (j=0; j<3; kk++, j++)
  { index[kk] = 10 * ind + j;
    toggle[j] = create_togglebutton(radio, tog_label[j], &index[kk],
                flarinCB);
  }

  if(flar_itflr[ind] >= 0.0 && flar_itflr[ind] < 3.0)
    XmToggleButtonSetState(toggle[(int) flar_itflr[ind]], TRUE, FALSE);

/*-----------------------------------------------------------------------
 * --- Create the Scales
 *-----------------------------------------------------------------------
 */
  for (i=0; i<2; kk++, i++)
  { create_separator(rowcol, &hv, &sd);

    index[kk] = 10 * ind + (i + 3);
    if(i == 0)
    { wid = 560;
      swid = 538;
      k = 6;
    }
    else
    { wid = 660;
      swid = 638;
      k = 7;
    }
    inc = 0;
    dec = 0;

    if(i == 0)
    { if(flar_qflar[ind] >= min[0] && flar_qflar[ind] <= max[0])
        val[i] = flar_qflar[ind];
      else
        val[i] = min[i];
    }
    else if(i == 1)
    { if(flar_tflar[ind] >= min[1] && flar_tflar[ind] <= max[1])
        val[i] = flar_tflar[ind];
      else
        val[i] = min[i];
    }

    scale[i] = create_scale(rowcol, scale_label[i], &wid, &min[i],
               &max[i], &inc, &dec, &val[i], &swid, &index[kk],
               flarinCB);

    for (j=0; j<k; j++)
    { n = 0;
      XtSetArg (wargs[n], XmNwidth, 90); n++;
      label[i][j] = XmCreateLabel(scale[i], numb[i][j], wargs, n);
```

```
    }
    XtManageChildren(label[i], k);
  }
}     /*   end flar_optCB()   */

/*********************************************************************
 *                          VOID FLARINCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  flarinCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  flar_optCB
 *====================================================================
 *<Description>
 *    Sets the input parameters of the Flare card
 *====================================================================
 *<Called routines>
 *    flar2                 - sets up the scales for FLUP & FLDN values
 *    create_menubar        - creates the pull-down menus, rollover
 *                            menus, and menubar to control them
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void flarinCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - pointer to the data passed to the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void flarinCB(Widget w, XtPointer c, XtPointer call_data)
{
  int *n = (int *)c;
  int i, j, value;
  static Boolean pass = FALSE;
  static Boolean mask[3], go;
  XmScaleCallbackStruct * call_value =
                            (XmScaleCallbackStruct *) call_data;

  value = call_value -> value;

  i = (*n) / 10;                          /* Flare number              */
  j = (*n) % 10;                          /* Toggle button pushed      */

  if((i > flar) && !pass)
  { flar_idflr[i] = i+1;
    flar_itflr[i] = flar_xflar[i] = flar_yflar[i] = flar_zflar[i] =
    flar_qflar[i] = flar_tflar[i] =  0.0;
    pass = TRUE;
    mask[0] = mask[1] = mask[2] = FALSE;
```

```
  }

  if(j < 3)
  { flar_itflr[i] = j;
    if(j == 2)
      flar2(i);
    mask[0] = TRUE;
  }
  else if(j == 3)
  { flar_qflar[i] = value;
    mask[1] = TRUE;
  }
  else if(j == 4)
  { flar_tflar[i] = value;
    mask[2] = TRUE;
  }

  cur_flar = i;
  go = mask[0] && mask[1] && mask[2];

  if(go)
  { flar++;
    pass = FALSE;
    mask[0] = mask[1] = mask[2] = FALSE;
    in_changef = TRUE;

    XtUnmanageChild (menu);
    menu = create_menubar(form);
  }

  new_file = TRUE;
}     /*   end flarinCB()    */

/*********************************************************************
 *                          VOID FLAR2
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  flar2
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  flarinCB
 *====================================================================
 *<Description>
 *    Sets the scales for the FLUP and FLDN values
 *====================================================================
 *<Called routines>
 *    create_rowcol         - creates a Rowcol Widget
 *    cancelCB              - removes the Rowcol Widget
 *    create_separator      - creates a Separator Widget
 *    create_scale          - creates a Scale Widget
 *    flar3CB               - gets the input parameters on the FLUP &
 *                            FLDN cards
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void flar2( int ind)
 *    Input:
 *       n                  - the Flare number
 *    Output:
 *       None
 *====================================================================
 *<History>
```

```
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void flar2( int ind)
{
  Widget scale[8], label[6], rowcol;
  Arg wargs[10];
  int n, i, j;
  static int index[8], hv, sd, wid, min, max, inc, dec, val[8], swid;
  static char frac_label[8][39] = {
                          "Fraction of Energy in Up Direction 1",
                          "Fraction of Energy in Up Direction 2",
                          "Fraction of Energy in Up Direction 3",
                          "Fraction of Energy in Up Direction 4",
                          "Fraction of Energy in Down Direction 1",
                          "Fraction of Energy in Down Direction 2",
                          "Fraction of Energy in Down Direction 3",
                          "Fraction of Energy in Down Direction 4" };
  static char frac[6][4] = { "0.0", "0.2", "0.4", "0.6", "0.8", "1.0" };

/*---------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *---------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Flare_Energy_Fractions", cancelCB);

/*---------------------------------------------------------------------
 * --- Create the Scales
 *---------------------------------------------------------------------
 */
  hv = 0;
  sd = 2;
  for (i=0; i<8; i++)
  { create_separator(rowcol, &hv, &sd);

    index[i] = 10 * ind + i;
    wid = 560;
    min = 0;
    max = 1000;
    inc = 0;
    dec = 3;
    if(i < 4)
    { if(flar_frrfup[ind][i] >= 0.0 && flar_frrfup[ind][i] <= 1.0)
        val[i] = 1000.0*flar_frrfup[ind][i];
      else
        val[i] = min;
    }
    else
    { if(flar_frrfdn[ind][i-4] >= 0.0 && flar_frrfdn[ind][i-4] <= 1.0)
        val[i] = 1000.0*flar_frrfdn[ind][i-4];
      else
        val[i] = min;
    }
    swid = 538;

    scale[i] = create_scale(rowcol, frac_label[i], &wid, &min, &max,
               &inc, &dec, &val[i], &swid, &index[i], flar3CB);

    if(i == 0)
```

```
    { for (j=0; j<6; j++)
      { n = 0;
        XtSetArg (wargs[n], XmNwidth, 45); n++;
        label[j] = XmCreateLabel(scale[0], frac[j], wargs, n);
      }
      XtManageChildren(label, 6);
    }
  }
}     /*   end flar2()   */

/*********************************************************************
 *                          VOID FLAR3CB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  flar3CB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  flar2
 *====================================================================
 *<Description>
 *    Gets the input parameters on the FLUP and FLDN cards
 *====================================================================
 *<Called routines>
 *    none
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void flar3CB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - pointer to the data passed to the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void flar3CB (Widget w, XtPointer c, XtPointer call_data)
{
  int *n = (int *)c;
  int i, j, value;
  XmScaleCallbackStruct * call_value =
                            (XmScaleCallbackStruct *) call_data;

  value = call_value -> value;
  i = (*n) / 10;
  j = (*n) % 10;

  if(j < 4)
    flar_frrfup[i][j] = 0.001 * (float)value;
  else
    flar_frrfdn[i][j-4] = 0.001 * (float)value;

  new_file = TRUE;
```

```
}     /*   end flar3CB()   */

/*********************************************************************
 *                          VOID SRCH_OPTCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  srch_optCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_srchmenu
 *====================================================================
 *<Description>
 *    Selects the various SearchLight Input Options for BLIRB
 *====================================================================
 *<Called routines>
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelslCB             - removes the Rowcol Widget
 *    create_separator       - creates a Separator Widget
 *    create_scale           - creates a Scale Widget
 *    srchinCB               - Set the input parameters of the SRCH card
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void srch_optCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                   - the ID of the widget for which the
 *                             callback is registered
 *       c                   - the data passed to the routine
 *       call_data           - a pointer to the callback structure which
 *                             contains information on why the callback
 *                             occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void srch_optCB (Widget w, XtPointer c, XtPointer call_data)
{
  Widget rowcol, label[5][10], scale[5];
  Arg wargs[10];
  int n, i, j, k, kk;
  static int index[5], hv, sd, wid, inc, dec, val[5], swid;
  static char scale_label[5][31] = { "SearchLight Beam Zenith (deg)",
                                     "SearchLight Beam Azimuth (deg)",
                                     "SearchLight Intensity (watts)",
                                     "SearchLight Temperature (K)",
                                     "SearchLight Diameter (m)" };
  static int min[5] = { 0, 0, 0, 1500, 0 };
  static int max[5] = { 90, 360, 100000, 7500, 2500 };
  static char numb[5][7][7] = {
     { "0", "30", "60", "90", "", "", "" },
     { "0", "90", "180", "270", "360", "", "" },
     { "0", "20000", "40000", "60000", "80000", "100000", "" },
     { "1500", "2500", "3500", "4500", "5500", "6500", "7500" },
     { "0", "5", "10", "15", "20", "25", "" }
     };
```

```
/*-----------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-----------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "SearchLight_Options", cancelslCB);

/*-----------------------------------------------------------------------
 * --- Create the Scales
 *-----------------------------------------------------------------------
 */
  hv = 0;
  sd = 2;
  kk = 0;

  for (i=0; i<5; kk++, i++)
  { create_separator(rowcol, &hv, &sd);

    index[kk] = i;
    if(i == 0)
    { wid = 420;
      swid = 398;
      inc = 1;
      dec = 0;
      k = 4;
    }
    else if(i == 1)
    { wid = 420;
      swid = 398;
      inc = 1;
      dec = 0;
      k = 5;
    }
    else if(i == 2)
    { wid = 560;
      swid = 538;
      inc = 0;
      dec = 0;
      k = 6;
    }
    else if(i == 3)
    { wid = 660;
      swid = 638;
      inc = 0;
      dec = 0;
      k = 7;
    }
    else
    { wid = 560;
      swid = 538;
      inc = 0;
      dec = 2;
      k = 6;
    }

    if(i == 0)
    { if(srch_thsrch >= min[0] && srch_thsrch <= max[0])
        val[i] = srch_thsrch;
      else
        val[i] = min[i];
    }
    else if(i == 1)
    { if(srch_azsrch >= min[1] && srch_azsrch <= max[1])
```

```
        val[i] = srch_azsrch;
      else
        val[i] = min[i];
    }
    else if(i == 2)
    { if(srch_psrch >= min[2] && srch_psrch <= max[2])
        val[i] = srch_psrch;
      else
        val[i] = min[i];
    }
    else if(i == 3)
    { if(srch_tmsrch >= min[3] && srch_tmsrch <= max[3])
        val[i] = srch_tmsrch;
      else
        val[i] = min[i];
    }
    else
    { if(srch_sdiam >= min[4]/100 && srch_sdiam <= max[4]/100)
        val[i] = 100.0 * srch_sdiam;
      else
        val[i] = 100 * min[i];
    }

    scale[i] = create_scale(rowcol, scale_label[i], &wid, &min[i],
               &max[i], &inc, &dec, &val[i], &swid, &index[kk],
               srchinCB);

    for (j=0; j<k; j++)
    { n = 0;
      XtSetArg (wargs[n], XmNwidth, 90); n++;
      label[i][j] = XmCreateLabel(scale[i], numb[i][j], wargs, n);
    }
    XtManageChildren(label[i], k);
  }
}     /*   end srch_optCB()   */

/*********************************************************************
 *                          VOID SRCHINCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  srchinCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  srch_optCB
 *====================================================================
 *<Description>
 *    Sets the input parameters of the SRCH card
 *====================================================================
 *<Called routines>
 *    create_menubar        - creates the pull-down menus, rollover
 *                            menus, and menubar to control them
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void srchinCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - pointer to the data passed to the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
```

```
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void srchinCB(Widget w, XtPointer c, XtPointer call_data)
{
  int *n = (int *)c;
  int j, value;
  static Boolean pass = FALSE;
  static Boolean mask[5], go;
  XmScaleCallbackStruct * call_value =
                              (XmScaleCallbackStruct *) call_data;

  value = call_value -> value;

  j = *n;                                  /* Scale selected          */

  if(!pass)
  { srch_thsrch = srch_azsrch = srch_psrch = srch_tmsrch = srch_sdiam
                = 0.0;
    pass = TRUE;
    mask[0] = mask[1] = mask[2] = mask[3] = mask[4] = FALSE;
  }

  if(j == 0)
  { srch_thsrch = value;
    mask[0] = TRUE;
  }
  else if(j == 1)
  { srch_azsrch = value;
    mask[1] = TRUE;
  }
  else if(j == 2)
  { srch_psrch = value;
    mask[2] = TRUE;
  }
  else if(j == 3)
  { srch_tmsrch = value;
    mask[3] = TRUE;
  }
  else if(j == 4)
  { srch_sdiam = 0.01 * (float)value;
    mask[4] = TRUE;
  }

  go = mask[0] && mask[1] && mask[2] && mask[3] && mask[4];

  if(go)
  { if(srch < 0)
    { srch = 0;
      in_changesl = TRUE;

      XtUnmanageChild (menu);
      menu = create_menubar(form);
    }
  }
```

```
  new_file = TRUE;
}     /*   end srchinCB()   */

/*********************************************************************
 *                          VOID COMP_OPTCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  comp_optCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_modifymenu
 *====================================================================
 *<Description>
 *    Selects the various Computation Options for BLIRB
 *====================================================================
 *<Called routines>
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelCB               - removes the Rowcol Widget
 *    create_separator       - creates a Separator Widget
 *    create_radiobox        - creates a Radiobox Widget
 *    create_togglebutton    - creates a Togglebutton Widget
 *    create_scale           - creates a Scale Widget
 *    compCB                 - Set the input parameters of the DOMD card
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void comp_optCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                   - the ID of the widget for which the
 *                             callback is registered
 *       c                   - the data passed to the routine
 *       call_data           - a pointer to the callback structure which
 *                             contains information on why the callback
 *                             occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void comp_optCB (Widget w, XtPointer c, XtPointer call_data)
{
  Widget radio[2], toggle[ISCT+3], rowcol, label[3][6], scale[3];
  Arg wargs[10];
  int n, i, j, k;
  static int nct[2] = { 1, ISCT };
  static int index[ISCT+6], hv, sd, nc, wid, inc, dec[3], val[3], swid;
  static char cat_label[2][29] = { "Delta Function Adjustment",
                                   "Order of Spherical Harmonics" };
  static char tog_label[2][4] = { "No", "Yes" };
  static char togg_label[10];
  static char scale_label[3][34]={ "Maximum Number of Iterations",
                                   "Convergence Criterion",
                                   "Number of Convergence Fail Points"};
  static int min[3] = { 0, 0, 0 };
  static int max[3] = { 100, 1000, 50 };
  static char numb[3][6][5] = {
              { "   0", "  20", "  40", "  60", "  80", " 100" },
```

```
            { "0.00", "0.02", "0.04", "0.06", "0.08", "0.10" },
            { "   0", "  10", "  20", "  30", "  40", "  50" }
        };

/*-----------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-----------------------------------------------------------------------
 */
  rowcol = create_rowcol(menu, "Computation_Options", cancelCB);

/*-----------------------------------------------------------------------
 * --- Create the radioboxes and toggles
 *-----------------------------------------------------------------------
 */
  hv = 0;
  sd = 2;
  nc = 1;
  k = 0;
  for (i=0; i<2; i++)
  { create_separator(rowcol, &hv, &sd);
    radio[i] = create_radiobox(rowcol, &nc, cat_label[i]);

    for (j=0; j<=nct[i]; k++, j++)
    { index[k] = 10*i + j;

      if(i == 0)
      { toggle[k] = create_togglebutton(radio[i], tog_label[k],
                    &index[k], compCB);
      }
      else
      { sprintf(togg_label, "Order %d\n", j);
        toggle[k] = create_togglebutton(radio[i], togg_label, &index[k],
                    compCB);
      }
    }

    if(domd == 0)
    { if(i == 0)
        XmToggleButtonSetState(toggle[(int) domd_idelta], TRUE, FALSE);
      else if(i == 1)
        XmToggleButtonSetState(toggle[2 + (int) domd_isc], TRUE, FALSE);
    }
  }

/*-----------------------------------------------------------------------
 * --- Create the Scales
 *-----------------------------------------------------------------------
 */
  for (i=0; i<3; k++, i++)
  { create_separator(rowcol, &hv, &sd);

    index[k] = 10*(i+2);
    wid = 560;
    inc = 0;
    swid = 538;
    if(i == 1)
      dec[i] = 4;
    else
      dec[i] = 0;

    if(domd == 0)
    { if(i == 0)
```

```
        val[i] = domd_iitl;
      else if(i == 1)
        val[i] = 10000.0*domd_epsi;
      else if(i == 2)
        val[i] = domd_npts;
    }
    else
      val[i] = min[i];

    scale[i] = create_scale(rowcol, scale_label[i], &wid, &min[i],
               &max[i], &inc, &dec[i], &val[i], &swid, &index[k],
               compCB);

    for (j=0; j<6; j++)
    { n = 0;
      XtSetArg (wargs[n], XmNwidth, 45); n++;
      label[i][j] = XmCreateLabel(scale[i], numb[i][j], wargs, n);
    }
    XtManageChildren(label[i], 6);
  }
}     /*   end comp_optCB()   */

/*********************************************************************
 *                          VOID COMPCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  compCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  comp_optCB
 *====================================================================
 *<Description>
 *    Sets the input parameters of the DOMD card
 *====================================================================
 *<Called routines>
 *    none
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void compCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                  - the ID of the widget for which the
 *                            callback is registered
 *       c                  - pointer to the data passed to the routine
 *       call_data          - a pointer to the callback structure which
 *                            contains information on why the callback
 *                            occurred
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void compCB(Widget w, XtPointer c, XtPointer call_data)
{
  int *n = (int *)c;
  int i, j, value;
  XmScaleCallbackStruct * call_value =
```

```
                                (XmScaleCallbackStruct *) call_data;

  value = call_value -> value;
  if(value == 0)
    value++;

  i = (*n)/10;                                /* RadioBox category        */
  j = (*n) % 10;                              /* Toggle button pushed     */

  if(domd < 0)
  { domd = 0;
    domd_isc = 2.0;
    domd_iitl = 10.0;
    domd_epsi = 0.002;
    domd_idelta = 1.0;
    domd_npts = 5.0;
  }

  if(i == 0)
    domd_idelta = j;
  else if(i == 1)
    domd_isc = j;
  else if(i == 2)
    domd_iitl = value;
  else if(i == 3)
    domd_epsi = 0.0001 * (float)value;
  else if(i == 4)
    domd_npts = value;

  new_file = TRUE;
}     /*   end compCB()   */

/*********************************************************************
 *                          VOID OUTPUT_OPTCB
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  output_optCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  create_modifymenu
 *====================================================================
 *<Description>
 *    Selects the Output File Format Option for BLIRB
 *====================================================================
 *<Called routines>
 *    create_rowcol          - creates a Rowcol Widget
 *    cancelCB               - removes the Rowcol Widget
 *    create_separator       - creates a Separator Widget
 *    create_radiobox        - creates a Radiobox Widget
 *    create_togglebutton    - creates a Togglebutton Widget
 *    outputCB               - Set the input parameter of the ASCI card
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void output_optCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                   - the ID of the widget for which the
 *                             callback is registered
 *       c                   - the data passed to the routine
 *       call_data           - a pointer to the callback structure which
 *                             contains information on why the callback
 *                             occurred
```

```
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void output_optCB (Widget w, XtPointer c, XtPointer call_data)
{
  Widget radio, toggle[4], rowcol;
  int j;
  static int index[4], hv, sd, nc;
  static char cat_label[28] = "Radiant Flux Output Options";
  static char tog_label[4][33] = { "No Output File",
                                   "Formatted Output File",
                                   "Unformatted (binary) Output File",
                                   "Both Formatted and Unformatted" };

/*-------------------------------------------------------------------
 * --- Create a RowColumn Widget
 *-------------------------------------------------------------------
 */
  rowcol = create_rowcol(w, "Output_Options", cancelCB);

/*-------------------------------------------------------------------
 * --- Create the radioboxes and toggles
 *-------------------------------------------------------------------
 */
  hv = 0;
  sd = 2;
  nc = 1;
  create_separator(rowcol, &hv, &sd);
  radio = create_radiobox(rowcol, &nc, cat_label);

  for (j=0; j<4; j++)
  { index[j] = j;
    toggle[j] = create_togglebutton(radio, tog_label[j], &index[j],
                outputCB);
  }

  if(asci == 0)
    XmToggleButtonSetState(toggle[(int) asci_irite], TRUE, FALSE);
}     /*   end output_optCB()   */

/*******************************************************************
 *                          VOID OUTPUTCB
 *******************************************************************
 *<Begin>
 *<Identification>          Name:  outputCB
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  output_optCB
 *=====================================================================
 *<Description>
 *    Sets the input parameter of the ASCI card
 *=====================================================================
 *<Called routines>
 *    none
 *=====================================================================
```

```
 *<Parameters>
 *    Formal declaration:
 *       void outputCB( Widget w, XtPointer c, XtPointer call_data)
 *    Input:
 *       w                    - the ID of the widget for which the
 *                              callback is registered
 *       c                    - pointer to the data passed to the routine
 *       call_data            - a pointer to the callback structure which
 *                              contains information on why the callback
 *                              occurred
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 *********************************************************************
 */
void outputCB (Widget w, XtPointer c, XtPointer call_data)
{
  int *n = (int *)c;

  asci = 0;
  asci_irite = *n;                          /* Toggle button pushed     */
}     /*   end outputCB()   */

/*********************************************************************
 *                          VOID GETDATA
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  getdata
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  main, checkfiletypeCB, newfile
 *=====================================================================
 *<Description>
 *    Controls getting the information from the input or outfile file
 *    and processing it.
 *=====================================================================
 *<Called routines>
 *    blirb_inout          - decides whether a BLIRB input or output
 *                           file was selected.  If neither, an error
 *                           flag is returned.
 *    create_message       - draws a message in a dialog message box
 *    readcards            - reads the BLIRB input cards and checks
 *                           them along with getting the output data
 *    set_albedo           - sets the Albedo values
 *    set_aerosol          - sets the Aerosol Material values
 *    set_mtrl_color       - sets the RGB colors for the material types
 *    set_axis_pts         - sets up the grid points for the X, Y, and
 *                           Z axes.
 *    order_mtrl           - arranges the MTRL and REGN cards 1 to 1
 *    order_albd           - arranges the ALBD and AREA cards 1 to 1
 *    reset                - resets the viewing and plot parameters to
 *                           the original values
 *    obsc                 - sets up rowcolumn widget with names of
 *                           materials.
 *    create_menubar       - creates the pull-down menus, rollover
 *                           menus, and menubar to control them
 *=====================================================================
```

```
 *<Parameters>
 *    Formal declaration:
 *       void getdata( void)
 *    Input:
 *       None
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void getdata(void)
{
  float visby, maxd;
  static char *error_mesg[] = {
              "Command Line Filename is neither Input nor Output.\n",
              "Please try another Filename.\n",
              "" };

/*---------------------------------------------------------------------
 * --- Determine whether the datafile is input/output/neither.
 *---------------------------------------------------------------------
 */
  if(!new_file && !def_file)
    blirb_inout();

  if(badfile)
/*---------------------------------------------------------------------
 * --- If file is neither input nor output, display an error message.
 *---------------------------------------------------------------------
 */
    create_message( menu, error_mesg, XmDIALOG_ERROR);
  else
/*---------------------------------------------------------------------
 * --- If file is either input or output, read the BLIRB input cards.
 *---------------------------------------------------------------------
 */
  { if(!new_file && !def_file)
      readcards();

    if(!badfile)
/*---------------------------------------------------------------------
 * --- If file contains no input errors, set the albedo values, aerosol
 *     values, material colors, and axis points.
 *---------------------------------------------------------------------
 */
    { if(!area_order && mdl2_ialb < 0)
        order_albd();                    /* Arranges the ALBD cards  */
      if(!regn_order)
        order_mtrl();                    /* Arranges the MTRL cards  */

      set_albedo();                      /* Set the Albedo values    */
      set_aerosol();                     /* Set Aerosol Mtrl values  */

      if(!mtl_color_set)
        set_mtrl_color();                /* Set the colors of Mtrls  */

      maxd = regn_rh[0][0];
```

```
      if(maxd < regn_rh[1][0])
        maxd = regn_rh[1][0];
      if(maxd < regn_rh[2][0])
        maxd = regn_rh[2][0];
      if(maxd > 5.0)
        org_magfactor = 5.0 / maxd;
      else
        org_magfactor = 1.0;

      set_axis_pts();                    /* Set the axes grid points */

      if(mdl1_ivis == 1)
        visby = 50.0;
      else if(mdl1_ivis == 2)
        visby = 23.0;
      else if(mdl1_ivis == 3)
        visby = 10.0;
      else if(mdl1_ivis == 4)
        visby = 5.0;
      else if(mdl1_ivis == 5)
        visby = 2.0;

      visby *= mdl2_sn;

      if(visby <= 5.0)
        metrng_indx = 1;
      else
        metrng_indx = 2;

      reset();
      obsc();
    }

/*-----------------------------------------------------------------------
 * --- For a new input or output file delete the current menubar and
 *     create a new one.
 *-----------------------------------------------------------------------
 */
    XtUnmanageChild (menu);
    menu = create_menubar(form);
  }
}
/*     end getdata  */

/***********************************************************************
 *                          VOID READCARDS
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  readcards
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  getdata
 *=====================================================================
 *<Description>
 *    Reads the BLIRB input cards and checks them along with getting
 *    the output data.
 *=====================================================================
 *<Called routines>
 *    create_message        - draws a message in a dialog message box
 *    checkinputs           - checks for the presence of required input
 *                            cards
 *    readoutput            - reads the output file information
```

```
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void readcards( void)
 *    Input:
 *       None
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void readcards(void)
{
  char cardlabel[5], card[RLEN], c[5];
  int i, j;
  float dum;
  FILE *fp;
  static char *error_msg[] = {
                "                                                    ",
                "Excess cards ignored.\n",
                "" };
  static char *error_msgx[] = {
                "                                                    ",
                "Card ignored.\n",
                "" };
  static char *error_mesg[] = {
                "Selected Filename not found.\n",
                "Please try another Filename.\n",
                "" };

/*-------------------------------------------------------------------
 * --- Open the requested input file or output file.
 *-------------------------------------------------------------------
 */
  if((fp = fopen(file_name, "r")) != NULL)
  {
/*-------------------------------------------------------------------
 * --- Initialize the Input Card Type Counters and the Grid Point
 *     Counters.
 *-------------------------------------------------------------------
 */
    ilcl = mdl1 = mdl2 = mdl3 = area = regn = mesx = mesy = mesz =
    albd = mtrl = clds = domd = sun = wavn = asci = recl = wavl = vis =
    flar = flen = flup = fldn = srch = sren = blirb = done = -1;

    num_grid_pts[0] = num_grid_pts[1] = num_grid_pts[2] =
    num_grid_main_pts[0] = num_grid_main_pts[1] =
    num_grid_main_pts[2] = -1;

    do                                    /* From VIEW.RJOB Subroutine*/
/*-------------------------------------------------------------------
 * --- Read a BLIRB input card from the file.
 *-------------------------------------------------------------------
 */
    { fgets(card, RLEN, fp);

/*-------------------------------------------------------------------
```

```
 * --- Get the first 5 characters (card identifier) from the card.
 *-----------------------------------------------------------------------
 */
      sscanf(card, "%s", cardlabel);

/*-----------------------------------------------------------------------
 * --- Process the rest of the card depending upon the identifier.
 *-----------------------------------------------------------------------
 */
      if(strncmp(cardlabel,"WAVL",4) == 0)
      { wavl = 0;                          /* WAVL card                 */
        sscanf(card, "%s%10e", c, &wavl_wavl);
      }

      else if(strncmp(cardlabel,"VIS",3) == 0)
      { vis = 0;                           /* VIS card                  */
        sscanf(card, "%s%10e", c, &vis_vis);
      }

      else if(strncmp(cardlabel,"BLIRB",5) == 0)
      { blirb = 0;                         /* BLIRB card                */
        sscanf(card, "%s", c);
      }

      else if(strncmp(cardlabel,"MDL1",4) == 0)
      { mdl1 = 0;                          /* MDL1 card                 */
        sscanf(card, "%s%10e%10e%10e%10e%10e", c, &mdl1_iaersl,
             &mdl1_model, &mdl1_ivis, &mdl1_iseasn, &mdl1_ivulcn);
      }

      else if(strncmp(cardlabel,"MDL2",4) == 0)
      { mdl2 = 0;                          /* MDL2 card                 */
        sscanf(card, "%s%10e%10e%10e%10e", c, &mdl2_sn, &mdl2_tbound,
             &mdl2_ialb, &mdl2_ip);
        if(!blirb_in && (int)mdl1_model != 7)
          fgets(card, RLEN, fp);
        cur_ialb = mdl2_ialb;
      }

      else if(strncmp(cardlabel,"MDL3",4) == 0)
      { mdl3 = 0;                          /* MDL3 card                 */
        sscanf(card, "%s%10e%10e%10e%10e%10e%10e", c, &mdl3_t[0],
             &mdl3_t[1], &mdl3_t[2], &mdl3_t[3], &mdl3_t[4],
             &mdl3_t[5]);
        if(!blirb_in)
          fgets(card, RLEN, fp);
      }

      else if(strncmp(cardlabel,"AREA",4) == 0)
      { area++;                            /* AREA card                 */
        if(area < IAM)
          sscanf(card, "%s%10e%10e%10e%10e%10e", c, &area_alx[area],
               &area_ahx[area], &area_aly[area], &area_ahy[area],
               &area_iamtl[area]);
        else
        { printf(" Too many AREA cards.\n");
          sprintf(error_msg[0],"Too many AREA cards.\n");
          create_message( menu, error_msg, XmDIALOG_ERROR);
        }
      }

      else if(strncmp(cardlabel,"REGN",4) == 0)
```

```
{ regn++;                                   /* REGN card                    */
  if(regn < IRM)
  { sscanf(card, "%s%10e%10e%10e%10e%10e%10e%10e", c,
          &regn_rlx[regn], &regn_rhx[regn], &regn_rly[regn],
          &regn_rhy[regn], &regn_rlz[regn], &regn_rhz[regn],
          &regn_izmtl[regn]);
    regn_rl[0][regn] = regn_rlx[regn];
    regn_rh[0][regn] = regn_rhx[regn];
    regn_rl[1][regn] = regn_rly[regn];
    regn_rh[1][regn] = regn_rhy[regn];
    regn_rl[2][regn] = regn_rlz[regn];
    regn_rh[2][regn] = regn_rhz[regn];
  }
  else
  { printf(" Too many REGN cards.\n");
    printf(" IRM curently set to %d.\n", IRM);
    sprintf(error_msg[0],"Too many REGN cards.\n");
    create_message( menu, error_msg, XmDIALOG_ERROR);
  }
}

else if(strncmp(cardlabel,"MESX",4) == 0)
{ mesx++;                                   /* MESX card                    */
  if(mesx < ISM)
  { sscanf(card, "%s%10e%10e", c, &mesx_mhx[mesx],
          &mesx_xms[mesx]);
    mes_mh[0][mesx] = mesx_mhx[mesx];
    mes_ms[0][mesx] = mesx_xms[mesx];
    mes[0] = mesx;
  }
  else
  { printf(" Too many MESX cards.\n");
    sprintf(error_msg[0],"Too many MESX cards.\n");
    create_message( menu, error_msg, XmDIALOG_ERROR);
  }
}

else if(strncmp(cardlabel,"MESY",4) == 0)
{ mesy++;                                   /* MESY card                    */
  if(mesy < ISM)
  { sscanf(card, "%s%10e%10e", c, &mesy_mhy[mesy],
          &mesy_yms[mesy]);
    mes_mh[1][mesy] = mesy_mhy[mesy];
    mes_ms[1][mesy] = mesy_yms[mesy];
    mes[1] = mesy;
  }
  else
  { printf(" Too many MESY cards.\n");
    sprintf(error_msg[0],"Too many MESY cards.\n");
    create_message( menu, error_msg, XmDIALOG_ERROR);
  }
}

else if(strncmp(cardlabel,"MESZ",4) == 0)
{ mesz++;                                   /* MESZ card                    */
  if(mesz < ISM)
  { sscanf(card, "%s%10e%10e", c, &mesz_mhz[mesz],
          &mesz_zms[mesz]);
    mes_mh[2][mesz] = mesz_mhz[mesz];
    mes_ms[2][mesz] = mesz_zms[mesz];
    mes[2] = mesz;
  }
```

```
    else
    { printf(" Too many MESZ cards.\n");
      sprintf(error_msg[0],"Too many MESZ cards.\n");
      create_message( menu, error_msg, XmDIALOG_ERROR);
    }
  }

  else if(strncmp(cardlabel,"ALBD",4) == 0)
  { albd++;                                /* ALBD card                    */
    if(albd < IDM)
    { if(blirb_in)
        sscanf(card, "%s%10e%10e", c, &albd_lalb[albd],
             &albd_falb[albd]);
      else
        sscanf(card, "%s%10e%10e%10e", c, &albd_lalb[albd],
             &albd_falb[albd], &dum);
    }
    else
    { printf(" Too many ALBD cards.\n");
      sprintf(error_msg[0],"Too many ALBD cards.\n");
      create_message( menu, error_msg, XmDIALOG_ERROR);
    }
  }

  else if(strncmp(cardlabel,"MTRL",4) == 0)
  { mtrl++;                                /* MTRL card                    */
    if(mtrl < ITM)
    { sscanf(card,"%s%10e%10e%10e%10e%10e%10e", c,
           &mtrl_lmtl[mtrl][0], &mtrl_wmtl[mtrl][0],
           &mtrl_lmtl[mtrl][1], &mtrl_wmtl[mtrl][1],
           &mtrl_lmtl[mtrl][2], &mtrl_wmtl[mtrl][2]);
      for(i=0; i<MXMTR; i++)
      { if(mtrl_lmtl[mtrl][i] > 100)
          mtrl_lmtl[mtrl][i] -= 60;
        mtrl_lmtl[mtrl][i] += 3;
      }
    }
    else
    { printf(" Too many MTRL cards.\n");
      sprintf(error_msg[0],"Too many MTRL cards.\n");
      create_message( menu, error_msg, XmDIALOG_ERROR);
    }
  }

  else if(strncmp(cardlabel,"CLDS",4) == 0)
  { clds = 0;                              /* CLDS card                    */
    sscanf(card, "%s%10e%10e%10e", c, &clds_icld, &clds_ibnd,
         &clds_wind);
  }

  else if(strncmp(cardlabel,"DOMD",4) == 0)
  { domd = 0;
    sscanf(card, "%s%10e%10e%10e%10e%10e", c, &domd_isc, &domd_iitl,
         &domd_epsi, &domd_idelta, &domd_npts);
  }

  else if(strncmp(cardlabel,"SUN",3) == 0)
  { sun = 0;                               /* SUN card                     */
    sscanf(card, "%s%10e%10e%10e%10e%10e", c, &sun_thsun,
         &sun_phsun, &sun_ifsun, &sun_isky, &sun_iftrn);
  }
```

```
else if(strncmp(cardlabel,"FLAR",4) == 0)
{ flar++;                                /* FLAR card                    */
  if(flar < NEST)
    sscanf(card, "%s%10e%10e%10e%10e%10e", c, &flar_idflr[flar],
         &flar_itflr[flar], &flar_xflar[flar], &flar_yflar[flar],
         &flar_zflar[flar]);
  else
  { printf(" Too many FLAR cards.\n");
    sprintf(error_msg[0],"Too many FLAR cards.\n");
    create_message( menu, error_msg, XmDIALOG_ERROR);
  }
}

else if(strncmp(cardlabel,"FLEN",4) == 0)
{ flen++;                                /* FLEN card                    */
  if(flen < NEST)
    sscanf(card, "%s%10e%10e%10e", c, &flen_idflr[flen],
         &flen_qflar[flen], &flen_tflar[flen]);
  else
  { printf(" Too many FLEN cards.\n");
    sprintf(error_msg[0],"Too many FLEN cards.\n");
    create_message( menu, error_msg, XmDIALOG_ERROR);
  }
}

else if(strncmp(cardlabel,"FLUP",4) == 0)
{ flup++;                                /* FLUP card                    */
  if(flup < NEST)
    sscanf(card, "%s%10e%10e%10e%10e%10e", c, &flup_idflr[flup],
         &flup_frrfup[flup][0], &flup_frrfup[flup][1],
         &flup_frrfup[flup][2], &flup_frrfup[flup][3]);
  else
  { printf(" Too many FLUP cards.\n");
    sprintf(error_msg[0],"Too many FLUP cards.\n");
    create_message( menu, error_msg, XmDIALOG_ERROR);
  }
}

else if(strncmp(cardlabel,"FLDN",4) == 0)
{ fldn++;                                /* FLDN card                    */
  if(fldn < NEST)
    sscanf(card, "%s%10e%10e%10e%10e%10e", c, &fldn_idflr[fldn],
         &fldn_frrfdn[fldn][0], &fldn_frrfdn[fldn][1],
         &fldn_frrfdn[fldn][2], &fldn_frrfdn[fldn][3]);
  else
  { printf(" Too many FLDN cards.\n");
    sprintf(error_msg[0],"Too many FLDN cards.\n");
    create_message( menu, error_msg, XmDIALOG_ERROR);
  }
}

else if(strncmp(cardlabel,"SRCH",4) == 0)
{ srch = 0;                              /* SRCH card                    */
  sscanf(card, "%s%10e%10e%10e%10e", c, &srch_xsrch,
       &srch_ysrch, &srch_zsrch, &srch_thsrch, &srch_azsrch);
}

else if(strncmp(cardlabel,"SREN",4) == 0)
{ sren = 0;                              /* SREN card                    */
  sscanf(card, "%s%10e%10e%10e", c, &sren_psrch, &sren_tmsrch,
       &sren_sdiam);
}
```

```
      else if(strncmp(cardlabel,"WAVN",4) == 0)
      { wavn = 0;                          /* WAVN card                  */
        sscanf(card, "%s%10e%10e%10e", c, &wavn_v1, &wavn_v2, &wavn_dv);

        wavn_dv = (wavn_v2 - wavn_v1) / wavn_dv;
        if(wavn_v1 >= 8000.0 && wavn_v2 <= 28000)
          wavn_indx = 0;
        else if(wavn_v1 >= 3000.0 && wavn_v2 <= 13000)
          wavn_indx = 1;
        else if(wavn_v1 >= 1200.0 && wavn_v2 <= 5200)
          wavn_indx = 2;
        else if(wavn_v1 >= 500.0 && wavn_v2 <= 1500)
          wavn_indx = 3;
        else if(wavn_v1 >= 600.0 && wavn_v2 <= 3600)
          wavn_indx = 4;
        else
          wavn_indx = -1;
      }

      else if(strncmp(cardlabel,"ASCI",4) == 0)
      { asci = 0;                          /* ASCI card                  */
        sscanf(card, "%s%10e", c, &asci_irite);
      }

      else if(strncmp(cardlabel,"RECL",4) == 0)
      { recl = 0;                          /* RECL card                  */
        sscanf(card, "%s%10e", c, &recl_irpt);
      }

      else if(strncmp(cardlabel,"DONE",4) == 0)
        done = 0;                          /* DONE card                  */

      else if(strncmp(cardlabel,"ILCL",4) == 0)
        ilcl = 0;                          /* ILCL card                  */

      else
      { printf(" Card %s not identified.\n", cardlabel); /* Unknown    */
        sprintf(error_msgx[0],"Card %s not identified.\n", cardlabel);
        create_message( menu, error_msgx, XmDIALOG_ERROR);
      }

    } while (recl != 0 && done != 0);

/*-----------------------------------------------------------------------
 * --- When finished reading the input cards, check for the presence of
 *     the required input cards.
 *-----------------------------------------------------------------------
 */
    checkinputs();

/*-----------------------------------------------------------------------
 * --- If the datafile is an output file, read the remaining output file
 *     data records.
 *-----------------------------------------------------------------------
 */
    if(!blirb_in && (flar < 0) && (srch < 0))
      readoutput(fp);

/*-----------------------------------------------------------------------
 * --- When finished reading all the data, close the file.
 *-----------------------------------------------------------------------
 */
```

```
    fclose(fp);

 /*-----------------------------------------------------------------------
  * --- If this was the first datafile, turn on the Sun display option
  *     and indicate the requested datafile was found.
  *-----------------------------------------------------------------------
  */
    if(!file_found)
      sun_plot = TRUE;

    file_found = TRUE;
   }

 /*-----------------------------------------------------------------------
  * --- If the requested datafile was not found, display an error
  *     message and indicate the requested datafile was not found.
  *-----------------------------------------------------------------------
  */
   else
   { badfile = TRUE;
    create_message( menu, error_mesg, XmDIALOG_ERROR);
   }
}       /* end readcards() */

/***********************************************************************
 *                          VOID READOUTPUT
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  readoutput
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  readcards
 *=====================================================================
 *<Description>
 *    Reads the output file information.
 *=====================================================================
 *<Called routines>
 *    None
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void readoutput( FILE *fp)
 *    Input:
 *       *fp                - the datafile pointer
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void readoutput(FILE *fp)
{
  static float minval = 1.0e-10;
  float dv_dum;
  int i, j, k, m, n;
  float out_m[MAXXYZ+1], vdum, dum;

/*-----------------------------------------------------------------------
```

```
 * --- Determine the number and values of the wavenumbers.
 *-----------------------------------------------------------------------
 */
  dv_dum = (wavn_v2 - wavn_v1) / wavn_dv;
  for (out_nwave=0, vdum=wavn_v1; vdum<wavn_v2; vdum+=dv_dum,
       out_nwave++)
    out_waveno[out_nwave] = vdum;

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine RJOB, get the X, Y, and
 *     Z BLIRB main region grid points and calculate the flux grid
 *     points from them.
 *-----------------------------------------------------------------------
 */
  fscanf(fp, "%d%d%d", &out_imx[0], &out_imx[1], &out_imx[2]);

  if(out_imx[0] > MAXMX)
  { printf(" The X dimension of the flux variable, [out_flux], is %d\n",
            MAXMX);
    printf("     while the current datafile requires %d\n", out_imx[0]);
    printf(" Please increase the value of [MAXMX] in [visual0.h] and");
    printf(" recompile this program.\n\n");
    printf(" Program Aborting!\n");
    exit(0);
  }

  if(out_imx[1] > MAXMY)
  { printf(" The Y dimension of the flux variable, [out_flux], is %d\n",
            MAXMY);
    printf("     while the current datafile requires %d\n", out_imx[1]);
    printf(" Please increase the value of [MAXMY] in [visual0.h] and");
    printf(" recompile this program.\n\n");
    printf(" Program Aborting!\n");
    exit(0);
  }

  if(out_imx[2] > MAXMZ)
  { printf(" The Z dimension of the flux variable, [out_flux], is %d\n",
            MAXMZ);
    printf("     while the current datafile requires %d\n", out_imx[2]);
    printf(" Please increase the value of [MAXMZ] in [visual0.h] and");
    printf(" recompile this program.\n\n");
    printf(" Program Aborting!\n");
    exit(0);
  }

  for (i=0; i<3; i++)
  { if(out_imx[i] > 0)
    { for (j = 0; j<=out_imx[i]; j++)
        fscanf(fp, "%12e", &out_m[j]);     /* Read X,Y,Z grid points    */
      for (j = 0; j<out_imx[i]; j++)
        out_m0[i][j] = 0.5 * (out_m[j] + out_m[j+1]);
    }
  }

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine RJOB, get the surface
 *     albedo indices at each (X,Y) grid point (ISURF).
 *-----------------------------------------------------------------------
 */
  if(out_imx[0] > 0 && out_imx[1] > 0)
    for (i = 0; i<out_imx[1]; i++)
```

```
      for (j = 0; j<out_imx[0]; j++)
        fscanf(fp, "%12e", &dum);

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine RJOB, get the region
 *     material indices at each (X,Y,Z) grid point (IVOLM).
 *-----------------------------------------------------------------------
 */
  if(out_imx[0] > 0 && out_imx[1] > 0 && out_imx[2] > 0)
    for (i = 0; i<out_imx[2]; i++)
      for (j = 0; j<out_imx[1]; j++)
        for (k = 0; k<out_imx[0]; k++)
          fscanf(fp, "%12e", &dum);

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine RFLX, get the LOWTRAN
 *     molecular transmission (TRLW).
 *-----------------------------------------------------------------------
 */
  for (m=0; m<out_nwave; m++)
  { for (j=0; j<=NA; j++)
      fscanf(fp, "%12e", &dum);

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine CLOUDR, get the surface
 *     albedos (SALB).
 *-----------------------------------------------------------------------
 */
    for (j=0; j<=albd; j++)
      fscanf(fp, "%12e", &dum);

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine CLOUDR, get the
 *     extinction coefficients (REXT).
 *-----------------------------------------------------------------------
 */
    for (j=0; j<ITN1; j++)
      fscanf(fp, "%12e", &dum);

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine CLOUDR, get the
 *     scattering coefficients (RSCT).
 *-----------------------------------------------------------------------
 */
    for (j=0; j<ITN1; j++)
      fscanf(fp, "%12e", &dum);

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine CLOUDR, get the
 *     "unknown" coefficients (FDLT).
 *-----------------------------------------------------------------------
 */
    for (j=0; j<ITN1; j++)
      fscanf(fp, "%12e", &dum);

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine CLOUDR, get the phase
 *     function angles (AGL).
 *-----------------------------------------------------------------------
 */
    for (j=0; j<ITN1; j++)
      for (i=0; i<4; i++)
```

```
        fscanf(fp, "%12e", &dum);

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine CLOUDR, get the phase
 *     functions for different materials (PHF).
 *-----------------------------------------------------------------------
 */
    for (j=0; j<ITN1; j++)
      for (i=0; i<4; i++)
        fscanf(fp, "%12e", &dum);

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine CLOUDR, get the
 *     Legendre coefficients (RLEG).
 *-----------------------------------------------------------------------
 */
    for (j=0; j<ITN1; j++)
      for (i=0; i<=(int)domd_isc; i++)
        fscanf(fp, "%12e", &dum);

/*-----------------------------------------------------------------------
 * --- Initialize the "zero flux" flags, minimum and maximum values
 *     of both the flux values and the log flux values.
 *-----------------------------------------------------------------------
 */
    for (j=0; j<10; j++)
    { flux_zero[j][m] = FALSE;
      mini_flux[j][m] = 1.0e+20;
      maxi_flux[j][m] = 0.0;
      mini1_flux[j][m] = 1.0e+20;
      maxi1_flux[j][m] = -1.0e20;
    }

/*-----------------------------------------------------------------------
 * --- Referencing the VIEW program subroutine RFLX, get the direct
 *     solar flux, reflected solar flux, and 8 diffuse flux values at
 *     each (X,Y,Z) flux grid point.  Then, determine the minimums and
 *     maximums of the flux values.
 *-----------------------------------------------------------------------
 */
    for (n=0; n<10; n++)
      for (i=0; i<out_imx[2]; i++)
        for (j=0; j<out_imx[1]; j++)
          for (k=0; k<out_imx[0]; k++)
          { fscanf(fp, "%12e", &out_flux[n][m][k][j][i]);

            if((out_flux[n][m][k][j][i] > -minval) &&
               (out_flux[n][m][k][j][i] <= 0.0))
              flux_zero[n][m] = TRUE;
            if(out_flux[n][m][k][j][i] > 0.0)
            { if(out_flux[n][m][k][j][i] < minval)
                out_flux[n][m][k][j][i] = minval;
            }
            else if(out_flux[n][m][k][j][i] < 0.0)
              out_flux[n][m][k][j][i] = minval;

            if(out_flux[n][m][k][j][i] < mini_flux[n][m])
              mini_flux[n][m] = out_flux[n][m][k][j][i];
            if(out_flux[n][m][k][j][i] > maxi_flux[n][m])
              maxi_flux[n][m] = out_flux[n][m][k][j][i];
          }
```

```
/*-----------------------------------------------------------------
 * --- Convert the flux values to log flux provided there are no zero
 *     or negative values of flux.  Then, determine the minimums and
 *     maximums of the log flux values.
 *-----------------------------------------------------------------
 */
    for (n=0; n<10; n++)
    { if(!flux_zero[n][m])
        for (j=0; j<out_imx[0]; j++)
          for (i=0; i<out_imx[1]; i++)
            for (k=0; k<out_imx[2]; k++)
            { out_flux[n][m][j][i][k] =
                        (float) log10((double) out_flux[n][m][j][i][k]);
              if(out_flux[n][m][j][i][k] < mini1_flux[n][m])
                mini1_flux[n][m] = out_flux[n][m][j][i][k];
              if(out_flux[n][m][j][i][k] > maxi1_flux[n][m])
                maxi1_flux[n][m] = out_flux[n][m][j][i][k];
            }
    }
  }
}      /* end readoutput() */

/*****************************************************************
 *                          VOID CHECKINPUTS
 *****************************************************************
 *<Begin>
 *<Identification>          Name:  checkinputs
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  readcards
 *===============================================================
 *<Description>
 *    Checks for the presence of required input cards.
 *===============================================================
 *<Called routines>
 *    None
 *===============================================================
 *<Parameters>
 *    Formal declaration:
 *       void checkinputs( void)
 *    Input:
 *       None
 *    Output:
 *       None
 *===============================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *===============================================================
 *<End>
 *****************************************************************
 */
void checkinputs(void)
{
  int i, j, k;
  Boolean flag;

  if(mdl1 < 0)
  { printf(" MDL1 input card was not found.\n");
    printf(" Program Aborting!\n");
    exit(0);
  }
```

```
if(mdl2 < 0)
{ printf(" MDL2 input card was not found.\n");
  printf(" Program Aborting!\n");
  exit(0);
}

if((int)mdl1_model == 7 && mdl3 < 0)
{ printf(" MDL3 input card was not found.\n");
  printf(" Program Aborting!\n");
  exit(0);
}

if(area < 0)
{ printf(" AREA input cards were not found.\n");
  printf(" Program Aborting!\n");
  exit(0);
}

if(regn < 0)
{ printf(" REGN input cards were not found.\n");
  printf(" Program Aborting!\n");
  exit(0);
}

if(mes[0] < 0)
{ printf(" MESX input cards were not found.\n");
  printf(" Program Aborting!\n");
  exit(0);
}

if(mes[1] < 0)
{ printf(" MESY input cards were not found.\n");
  printf(" Program Aborting!\n");
  exit(0);
}

if(mes[2] < 0)
{ printf(" MESZ input cards were not found.\n");
  printf(" Program Aborting!\n");
  exit(0);
}

if(albd < 0 && mdl2_ialb == -1.0)
{ printf(" ALBD input cards were not found.\n");
  printf(" Program Aborting!\n");
  exit(0);
}

if(mtrl < 0 && clds_icld == 2.0)
{ printf(" MTRL input cards were not found.\n");
  printf(" Program Aborting!\n");
  exit(0);
}

if(clds < 0)
{ printf(" CLDS input card was not found.\n");
  printf(" Program Aborting!\n");
  exit(0);
}

if(sun < 0)
{ printf(" SUN input card was not found.\n");
```

```
    printf(" Program Aborting!\n");
    exit(0);
  }

  if(flar >= 0)
  { if(flen != flar)
    { printf(" FLAR and FLEN input cards did not match.\n");
      printf(" Program Aborting!\n");
      exit(0);
    }
    else
    { for (i = 0; i <= flar; i++)
      { flag = FALSE;
        for (j = 0; j <= flen; j++)
          if(flar_idflr[i] == flen_idflr[j])
          { flar_qflar[i] = flen_qflar[j];
            flar_tflar[i] = flen_tflar[j];
            flag = TRUE;
          }
        if(!flag)
        { printf(" No FLEN IDFLR matches the FLAR IDFLR (%d).\n",
                  flar_idflr[i]);
          printf(" Program Aborting!\n");
          exit(0);
        }
      }

      for (i = 0; i <= flar; i++)
        if(flar_itflr[i] == 2)
        { if(flup < 0)
          { printf(" FLAR ITFLR = 2 and no FLUP cards are present.\n");
            printf(" Program Aborting!\n");
            exit(0);
          }
          else
          { flag = FALSE;
            for (j = 0; j <= flup; j++)
              if(flar_idflr[i] == flup_idflr[j])
              { for (k = 0; k < 4; k++)
                    flar_frrfup[i][k] = flup_frrfup[j][k];
                flag = TRUE;
              }
            if(!flag)
            { printf(" No FLUP IDFLR matches the FLAR IDFLR (%d).\n",
                      (int)flar_idflr[i]);
              printf(" Program Aborting!\n");
              exit(0);
            }
          }

          if(fldn < 0)
          { printf(" FLAR ITFLR = 2 and no FLDN cards are present.\n");
            printf(" Program Aborting!\n");
            exit(0);
          }
          else
          { flag = FALSE;
            for (j = 0; j <= fldn; j++)
              if(flar_idflr[i] == fldn_idflr[j])
              { for (k = 0; k < 4; k++)
                    flar_frrfdn[i][k] = fldn_frrfdn[j][k];
                flag = TRUE;
```

```
            }
          if(!flag)
          { printf(" No FLDN IDFLR matches the FLAR IDFLR (%d).\n",
                   (int)flar_idflr[i]);
            printf(" Program Aborting!\n");
            exit(0);
          }
        }
      }
    }
  }

  if(srch == 0)
  { if(sren != 0)
    { printf(" SRCH card found and SREN card not found.\n");
      printf(" Program Aborting!\n");
      exit(0);
    }
    else
    { srch_psrch = sren_psrch;
      srch_tmsrch = sren_tmsrch;
      srch_sdiam = sren_sdiam;
    }
  }

  if(wavn < 0)
  { printf(" WAVN input card was not found.\n");
    printf(" Program Aborting!\n");
    exit(0);
  }
}      /* end checkinputs() */

/*********************************************************************
 *                          VOID SET_ALBEDO
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  set_albedo
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  getdata, area_delCB, cancelaCB,
 *                                 cancelbCB, cancelbbCB
 *====================================================================
 *<Description>
 *    Sets the albedo values in the appropriate array.
 *====================================================================
 *<Called routines>
 *    create_message        - draws a message in a dialog message box
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void set_albedo( void)
 *    Input:
 *       None
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
```

```
 */
void set_albedo(void)
{
  static char *msg[] = {
                "                                                          ",
                "                                                          ",
                "                                                          ",
                "" };
  int i,j;

  for (j=0; j <= area; j++)
  { if(area_iamtl[j] >= 0)
    { switch ((int) mdl2_ialb)
      { case -1 :
/*-----------------------------------------------------------------------
 * --- Get the wave independent user-defined albedo values.
 *-----------------------------------------------------------------------
 */
          spect_albedo = FALSE;

          if(area_iamtl[j] == 0.0)
            area_iamt[j] = background_albedo;
          else
          { if(albd >= 0)
            { for (i=0; i <= albd; i++)
                if(area_iamtl[j] == albd_lalb[i])
                { area_iamt[j] = albd_falb[i];

                  if(area_iamt[j] > 1.0)
                  { printf(" IAMTL(%d) = %f\n", j+1, area_iamt[j]);
                    printf(" IAMTL out of range (0.0 - 1.0).\n");
                    printf(" Defaulting to Background Albedo.\n\n");

                    sprintf(msg[0],"IAMTL(%d) = %f\n",j+1,area_iamt[j]);
                    sprintf(msg[1],"IAMTL out of range (0.0 - 1.0).\n");
                    sprintf(msg[2],
                            "Defaulting to Background Albedo.\n");
                    create_message( menu, msg, XmDIALOG_ERROR);
                    area_iamt[j] = background_albedo;
                  }
                }
            }
            else
            { printf(" IALB on the MDL2 card = -1.0\n");
              printf(" But, no ALBD cards were found.\n");
              printf(" Defaulting to Background Albedo.\n\n");

              sprintf(msg[0],"IALB on the MDL2 card = -1.0\n");
              sprintf(msg[1],"But, no ALBD cards were found.\n");
              sprintf(msg[2],"Defaulting to Background Albedo.\n");
              create_message( menu, msg, XmDIALOG_ERROR);
              area_iamt[j] = background_albedo;
            }
          }
          break;

        case 0 :
/*-----------------------------------------------------------------------
 * --- Get the wave independent tabulated broadband albedo values.
 *-----------------------------------------------------------------------
 */
          spect_albedo = FALSE;
```

```
          if(area_iamtl[j] >= 0.0 && area_iamtl[j] <= 55.0)
            area_iamt[j] = broad_albedo[(int) area_iamtl[j]];
          else
          { printf(" IAMTL(%d) is out of range (0.0 - 55.0)\n", j+1);
            printf(" Defaulting to Background Albedo.\n\n");

            sprintf(msg[0],"IAMTL(%d) is out of range (0.0 - 55.0)\n",
                    j+1);
            sprintf(msg[1],"Defaulting to Background Albedo.\n");
            sprintf(msg[2],"");
            create_message( menu, msg, XmDIALOG_ERROR);
            area_iamt[j] = background_albedo;
          }
          break;

        case 1 :
/*-----------------------------------------------------------------------
 * --- Get the spectral albedo values corresponding to 8.0 micrometers.
 *-----------------------------------------------------------------------
 */
          if(area_iamtl[j] >= 0.0 && area_iamtl[j] <= 6.0)
            area_iamt[j] = spectral_albedo[7][(int)area_iamtl[j]];
          else
          { printf(" IAMTL(%d) is out of range (0.0 - 6.0)\n", j+1);
            printf(" Defaulting to Background Albedo.\n\n");

            sprintf(msg[0],"IAMTL(%d) is out of range (0.0 - 6.0)\n",
                    j+1);
            sprintf(msg[1],"Defaulting to Background Albedo.\n");
            sprintf(msg[2],"");
            create_message( menu, msg, XmDIALOG_ERROR);
            area_iamt[j] = background_albedo;
          }
          break;

        default :
          printf(" The value of IALB on card MDL2 = %f\n", mdl2_ialb);
          printf(" Program Aborting!\n");
          exit(0);
      }
    }

/*-----------------------------------------------------------------------
 * --- Check the albedo values to make sure they are between 0 and 1.
 *     If not, default to the background albedo value.
 *-----------------------------------------------------------------------
 */
    if(area_iamt[j] < 0.0 || area_iamt[j] > 1.0)
    { printf(" IAMTL(%d) = %f which is out of range ( >= 0.0).\n",
             j+1, area_iamt[j]);
      printf(" Defaulting to Background Albedo.\n\n");

      sprintf(msg[0],"IAMTL(%d) = %f and is out of range ( >= 0.0).\n",
              j+1, area_iamt[j]);
      sprintf(msg[1],"Defaulting to Background Albedo.\n");
      sprintf(msg[2],"");
      create_message( menu, msg, XmDIALOG_ERROR);
      area_iamt[j] = background_albedo;
    }
  }
}     /* end set_albedo() */
```

```
/*********************************************************************
 *                          VOID SET_AEROSOL
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  set_aerosol
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  getdata, regn_delCB, cancelmCB
 *====================================================================
 *<Description>
 *    Sets the aerosol material values in the appropriate array.
 *====================================================================
 *<Called routines>
 *    create_message        - draws a message in a dialog message box
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void set_aerosol( void)
 *    Input:
 *       None
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void set_aerosol(void)
{
  static char *msg[] = {
                "                                                      ",
                "Defaulting to Background Aerosol.\n",
                "" };
  int i, j;

  for (j=0; j <= regn; j++)
  { if(regn_izmtl[j] < 0.0 || regn_izmtl[j] > mtrl+1)
/*-------------------------------------------------------------------
 * --- For each BLIRB aerosol region, check the material index to make
 *     sure it is within range.  If not, set the material to the
 *     background aerosol material.
 *-------------------------------------------------------------------
 */
    { printf(" IZMTL(%d) = %f which is out of range.\n", j+1,
             regn_izmtl[j]);
      printf(" Defaulting to Background Aerosol.\n\n");
      sprintf(msg[0],"IZMTL(%d) = %f which is out of range.\n", j+1,
             regn_izmtl[j]);
      create_message( menu, msg, XmDIALOG_ERROR);
      regn_izmt[j] = background_aerosol;
    }
/*-------------------------------------------------------------------
 * --- If the index is within range, get the aerosol material type.
 *-------------------------------------------------------------------
 */
    else if(regn_izmtl[j] != 0.0)
      regn_izmt[j] = mtrl_lmtl[(int)regn_izmtl[j] - 1][0];
    else
      regn_izmt[j] = 3;
```

```
  }
}       /* end set_aerosol() */

/*********************************************************************
 *                          VOID SET_MTRL_COLOR
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  set_mtrl_color
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  getdata
 *====================================================================
 *<Description>
 *    Sets the RGB colors for the aerosol material types.
 *====================================================================
 *<Called routines>
 *    None
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void set_mtrl_color( void)
 *    Input:
 *       None
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void set_mtrl_color(void)
{
  int i, j;

/*--------------------------------------------------------------------
 * --- Initialize colors to black.
 *--------------------------------------------------------------------
 */
  for (i=0; i<100; i++)
    for (j=0; j<3; j++)
      mtrl_color[i][j] = 0;

/*--------------------------------------------------------------------
 * --- OTHER Data Base colors:
 *--------------------------------------------------------------------
 */
  for (j=0; j<2; j++)
    mtrl_color[0][j] = 42;                  /* Dirt: Brown              */

  mtrl_color[1][0] = 120;
  for (j=1; j<3; j++)                       /* Deimendjian: Cyan        */
    mtrl_color[1][j] = 255;

  mtrl_color[2][0] = 160;                   /* LOWTRAN Cumulus: Cyan    */
  for (j=1; j<3; j++)
    mtrl_color[2][j] = 255;

/*--------------------------------------------------------------------
 * --- LOWTRAN Data Base colors:
```

```
*-----------------------------------------------------------------
*/
 for (i=4; i<25; i++)                          /* Aerosols: Shades of Blue */
 { mtrl_color[i][1] = 80;
   mtrl_color[i][2] = 111 + 6*i;
 }

 mtrl_color[25][0] = 128;                      /* Aged Volcanic: Gold      */
 mtrl_color[25][1] = 128;

 mtrl_color[26][0] = 188;                      /* Fresh Volcanic: Yellow   */
 mtrl_color[26][1] = 188;

 mtrl_color[27][0] = 255;                      /* Radiative Fog: White     */
 mtrl_color[27][1] = 255;
 mtrl_color[27][2] = 180;

 mtrl_color[28][0] = 255;                      /* Advective Fog: Beige     */
 mtrl_color[28][1] = 255;
 mtrl_color[28][2] = 150;

 mtrl_color[29][0] = 128;                      /* Meteoric Dust: Gray      */
 mtrl_color[29][1] = 128;
 mtrl_color[29][2] = 128;

 for (i=30; i<37; i++)                         /* Clouds: Shades of Cyan   */
   for (j=1; j<3; j++)
     mtrl_color[i][j] = 255;
 mtrl_color[30][0] = 160;                          /* Cumulus              */
 mtrl_color[31][0] = 120;                          /* Altostratus          */
 mtrl_color[32][0] = 80;                           /* Stratus              */
 mtrl_color[33][0] = 40;                           /* Stratus/Strato       */
 mtrl_color[34][0] = 0;                            /* Nimbostratus         */
 mtrl_color[35][0] = 180;                          /* Cirrus               */
 mtrl_color[36][0] = 200;                          /* Subvisual Cirrus     */

 for (i=37; i<41; i++)                         /* Desert: Shades of Brown  */
   for (j=0; j<2; j++)
     mtrl_color[i][j] = (452-10*i);

/*-----------------------------------------------------------------
 * --- EOSAEL Data Base colors:
 *-----------------------------------------------------------------
 */
 for (i=44; i<68; i++)                         /* Aerosols: Shades of Blue */
 { mtrl_color[i][1] = 80;
   mtrl_color[i][2] = 6*i - 147;
 }

 for (i=68; i<70; i++)                         /* Fogs: Shades of Beige    */
 { for (j=0; j<2; j++)
     mtrl_color[i][j] = 255;
   mtrl_color[25][2] = 30*i - 1890;
 }

 for (i=70; i<73; i++)                         /* Rain: Shades of Aqua     */
 { for (j=1; j<3; j++)
     mtrl_color[i][j] = 3040 - 40*i;
   mtrl_color[i][0] = 50;
 }

 for (i=0; i<3; i++)                           /* Snow: White              */
```

```
    mtrl_color[73][i] = 255;

  for (j=83; j<85; j++)
    for (i=1; i<3; i++)
      mtrl_color[j][i] = 255;              /* Clouds: Shades of Cyan    */
  mtrl_color[83][0] = 160;                 /* Fairweather Cumulus       */
  mtrl_color[84][0] = 0;                   /* Cumulus Congestus         */

  for (i=93; i<95; i++)                    /* Dust: Shades of Brown     */
    for (j=0; j<2; j++)
      mtrl_color[i][j] = 2872 - 30*i;

  mtrl_color[95][0] = 100;                 /* HE Dust: Brown            */
  mtrl_color[95][1] = 50;

  for (i=96; i<99; i++)                    /* WP Smokes: Shades Pink    */
  { mtrl_color[i][0] = 255;
    mtrl_color[i][1] = 128;
    mtrl_color[i][2] = 5055 - 50*i;
  }

  for (i=0; i<3; i++)                      /* Fog Oil: Gray             */
    mtrl_color[99][i] = 75;

  for (i=0; i<2; i++)                      /* HC Smoke: Yellow          */
    mtrl_color[100][i] = 255;

  mtl_color_set = TRUE;
}      /* end set_mtrl_color() */

/*********************************************************************
 *                          VOID SET_AXIS_PTS
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  set_axis_pts
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  getdata, regnCB, cancelmeCB
 *====================================================================
 *<Description>
 *    Sets up the grid points for the X, Y, and Z axes.
 *====================================================================
 *<Called routines>
 *    reset                 - resets the "mov" structure to its initial
 *                            values.
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void set_axis_pts( void)
 *    Input:
 *       None
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void set_axis_pts(void)
```

```
{
  int i, j, k;
  float low, del;

  for (k=0; k<3; k++)
/*-----------------------------------------------------------------------
 * --- For each of the 3 axes, determine the lowest region grid point
 *     value.
 *-----------------------------------------------------------------------
 */
  { low = regn_rl[k][0];
    for (i=0; i<= regn; i++)
      if(low > regn_rl[k][i])
        low = regn_rl[k][i];

/*-----------------------------------------------------------------------
 * --- Determine the number of minor grid points and the minor grid
 *     point values.
 *-----------------------------------------------------------------------
 */
    num_grid_pts[k] = 0;
    for (i=0; i <= mes[k]; i++)
    { del = (mes_ms[k][i] - low) / mes_mh[k][i];
      for (axis_pts[k][num_grid_pts[k]] = low, j=0;
           j<(int) mes_mh[k][i]; num_grid_pts[k]++,j++)
        axis_pts[k][num_grid_pts[k]+1] = axis_pts[k][num_grid_pts[k]] +
                                         del;
      low = mes_ms[k][i];
    }

/*-----------------------------------------------------------------------
 * --- Determine the position of the center of the BLIRB main region
 *     and the distance to the plotted Sun along this axis.
 *-----------------------------------------------------------------------
 */
    org_center[k] = 0.5 * org_magfactor * (mes_ms[k][mes[k]] -
                    axis_pts[k][0]) + axis_pts[k][0];
    sun_distance[k] = mes_ms[k][mes[k]] + 1.0;

/*-----------------------------------------------------------------------
 * --- Determine the number of major grid points and the major grid
 *     point values.
 *-----------------------------------------------------------------------
 */
    num_grid_main_pts[k] = 0;
    low = (int)axis_pts[k][0];
    i = mes_ms[k][mes[k]] - axis_pts[k][0];
    for (axis_main_pts[k][num_grid_main_pts[k]] = low, j=0; j<i;
         num_grid_main_pts[k]++, j++)
      axis_main_pts[k][num_grid_main_pts[k]+1] =
                    axis_main_pts[k][num_grid_main_pts[k]] + 1.0;
  }

/*-----------------------------------------------------------------------
 * --- If this is the first datafile to be processed, set the "mov"
 *     structure to its original values.
 *-----------------------------------------------------------------------
 */
  if(nofile)
  { reset();
    nofile = FALSE;
  }
```

```
}       /* end set_axis_pts() */

/*********************************************************************
 *                          VOID ORDER_MTRL
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  order_mtrl
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  getdata
 *   ================================================================
 *<Description>
 *    Arranges the MTRL cards to correspond to the REGN cards 1 to 1.
 *=================================================================
 *<Called routines>
 *    none
 *=================================================================
 *<Parameters>
 *    Formal declaration:
 *       void order_mtrl(void)
 *    Input:
 *       None
 *    Output:
 *       None
 *=================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=================================================================
 *<End>
 *********************************************************************
 */
void order_mtrl(void)
{
  int n, nct, i, j;
  float cnt[MXMTR][IRM], mtl[MXMTR][IRM];
  Boolean found;

  if(!regn_order)
  { if(regn > -1)
    { for (nct = -1, i=0; i<=regn; i++)
      { n = regn_izmtl[i] - 1.0;
        if( n != i )
        { if( n == -1 )
          { nct++;
            regn_izmtl[i] = nct + 1;
            mtl[0][nct] = background_aerosol;
            cnt[0][nct] = 0.0;
            for (j=1; j<MXMTR; j++)
            { mtl[j][nct] = 3.0;
              cnt[j][nct] = 0.0;
            }
          }
          else
          { if(mtrl > -1)
            { found = FALSE;
              if(mtrl >= n)
              { nct++;
                regn_izmtl[i] = nct + 1;
                for (j=0; j<MXMTR; j++)
                { mtl[j][nct] = mtrl_lmtl[n][j];
                  cnt[j][nct] = mtrl_wmtl[n][j];
```

```
                }
                found = TRUE;
              }
              if(!found)
              { printf("regn_izmtl[%d] = %f and no MTRL cards match\n",
                        i, regn_izmtl[i]);
                printf("Program Aborting!\n");
                exit(0);
              }
            }
            else
            { printf("regn_izmtl[%d] = %f and no MTRL cards are");
              printf(" available\n", i, regn_izmtl[i]);
              printf("Program Aborting!\n");
              exit(0);
            }
          }
        }
        else
        { found = FALSE;
          if(mtrl >= n)
          { nct++;
            regn_izmtl[i] = nct + 1;
            for (j=0; j<MXMTR; j++)
            { mtl[j][nct] = mtrl_lmtl[i][j];
              cnt[j][nct] = mtrl_wmtl[i][j];
            }
            found = TRUE;
          }
          if(!found)
          { printf("regn_izmtl[%d] = %f and no MTRL cards match\n",
                    i, regn_izmtl[i]);
            printf("Program Aborting!\n");
            exit(0);
          }
        }
      }

      for (mtrl = -1, i=0; i<=regn; i++)
      { for (j=0; j<MXMTR; j++)
        { mtrl_lmtl[i][j] = mtl[j][i];
          mtrl_wmtl[i][j] = cnt[j][i];
        }
        mtrl++;
      }

      regn_order = TRUE;
    }
  }
}     /*      end order_mtrl  */

/*********************************************************************
 *                          VOID ORDER_ALBD
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  order_albd
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  getdata
 *====================================================================
 *<Description>
 *    Arranges the ALBD cards to correspond to the AREA cards 1 to 1.
```

```
 *=====================================================================
 *<Called routines>
 *    none
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void order_albd(void)
 *    Input:
 *       None
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void order_albd(void)
{
  int nct, i, j;
  float cnt[IAM], mtl[IAM];
  Boolean found;

  if(!area_order && (mdl2_ialb < 0))
  { if(area > -1)
    { for (nct = -1, i=0; i<=area; i++)
      { if( area_iamtl[i] != (i+1) )
        { if( area_iamtl[i] == 0 )
          { nct++;
            cnt[nct] = i+1;
            mtl[nct] = background_albedo;
          }
          else
          { if(albd > -1)
            { found = FALSE;
              for (j=0; j<=albd; j++)
              { if(albd_lalb[j] == area_iamtl[i])
                { nct++;
                  cnt[nct] = i+1;
                  mtl[nct] = albd_falb[j];
                  found = TRUE;
                }
              }
              if(!found)
              { printf("area_iamtl[%d] = %f and no ALBD cards match\n",
                        i, area_iamtl[i]);
                printf("Program Aborting!\n");
                exit(0);
              }
            }
            else
            { printf("area_iamtl[%d] = %f and no ALBD cards are");
              printf(" available\n", i, area_iamtl[i]);
              printf("Program Aborting!\n");
              exit(0);
            }
          }
        }
        else
        { found = FALSE;
```

```
            for (j=0; j<=albd; j++)
            { if(albd_lalb[j] == (i+1))
              { nct++;
                cnt[nct] = i+1;
                mtl[nct] = albd_falb[j];
                found = TRUE;
              }
            }
            if(!found)
            { printf("area_iamtl[%d] = %f and no ALBD cards match\n",
                     i, area_iamtl[i]);
              printf("Program Aborting!\n");
              exit(0);
            }
          }
        }

        for (albd = -1, i=0; i<=area; i++)
        { area_iamtl[i] = albd_lalb[i] = cnt[i];
          albd_falb[i] = mtl[i];
          albd++;
        }
      }
    }
    area_order = TRUE;
}     /*      end order_albd  */

/*****************************************************************
 *                          VOID BLIRB_INOUT
 *****************************************************************
 *<Begin>
 *<Identification>          Name:  blirb_inout
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  getdata, checkfiletypeCB
 *===============================================================
 *<Description>
 *    Decides whether a BLIRB input or output file was selected.  If
 *    neither, an error flag is set.
 *===============================================================
 *<Called routines>
 *    None
 *===============================================================
 *<Parameters>
 *    Formal declaration:
 *       void blirb_inout( void)
 *    Input:
 *       None
 *    Output:
 *       None
 *===============================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *===============================================================
 *<End>
 *****************************************************************
 */
void blirb_inout(void)
{
  int i;
  char *ptr;
```

```
/*  static char *exten = {".Ba"};  */
  static char *exten = {".ia"};

  ptr = file_name;

/*-----------------------------------------------------------------------
 * --- Find the "." in the requested datafile name.
 *-----------------------------------------------------------------------
 */
  for (i = 0; i < strlen(file_name); i++, ptr++)
  { if(ptr[0] == exten[0])
    { ptr++;
      break;
    }
  }

/*-----------------------------------------------------------------------
 * --- If the first character after the "." is "i" the file is assumed
 *     to be a BLIRB input file.
 *-----------------------------------------------------------------------
 */
  if(ptr[0] == exten[1])
  { blirb_in = TRUE;
    badfile = FALSE;

    noflux = TRUE;
    for (i=0; i<10; i++)
      flux_flag[i] = FALSE;
  }
/*-----------------------------------------------------------------------
 * --- If the first character after the "." is "a" the file is assumed
 *     to be a BLIRB output file.
 *-----------------------------------------------------------------------
 */
  else if(ptr[0] == exten[2])
  { blirb_in = FALSE;
    badfile = FALSE;
  }
/*-----------------------------------------------------------------------
 * --- If the first character after the "." is neither an "i" nor a
 *     "a", the file is assumed to be unacceptable as a datafile.
 *-----------------------------------------------------------------------
 */
  else
    badfile = TRUE;
}     /*      end blirb_inout  */

/***********************************************************************
 *                          VOID WRITECARDS
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  writecards
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  savefileCB, getfilenameCB
 *=====================================================================
 *<Description>
 *    Writes the BLIRB input cards to a file.
 *=====================================================================
 *<Called routines>
 *    create_message        - draws a message in a dialog message box
 *    drawscene             - Plots the 3-D BLIRB grid points, albedo
```

```
 *                              areas, aerosol regions, and the output
 *                              flux.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void writecards( void)
 *    Input:
 *       None
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void writecards(void)
{
  char card[RLEN];
  int i, j, k, dum, m, del[40], delr[40], wmtrl, walbd;
  FILE *fp;
  float dv_dum, matter[IRM][MXMTR], weight[IRM][MXMTR], index[IAM],
        value[IAM];
  Boolean flag;
  static char *error_mesg[] = {
                 "",
                 "Error in opening the Savefile.\n",
                 "Input cards not Saved.\n",
                 "" };

/*-------------------------------------------------------------------
 * --- Delete the excess "ALBD" cards
 *-------------------------------------------------------------------
 */
  walbd = albd;
  if(walbd >= 0)
  { for (i=0; i<=walbd; i++)
    { index[i] = albd_lalb[i];
      value[i] = albd_falb[i];
    }
  }

  k = 0;
  if(walbd > 0 && mdl2_ialb < 0)
  { for (i=0; i<walbd; i++)
    { for (j=i+1; j<=walbd; j++)
      { if(value[j] == value[i])
        { area_iamtl[j] = index[j] = area_iamtl[i];
          del[k] = j;
          k++;
        }
      }
    }

    if(k > 1)
    { for (i=0; i<(k-1); i++)
      { flag = FALSE;

        for (j=0; j<(k-1); j++)
        { if(del[j+1] > del[j])
```

```
            { dum = del[j+1];
              del[j+1] = del[j];
              del[j] = dum;
              flag = TRUE;
            }
          }

          if(!flag)
            break;
        }
      }

      dum = 0;
      delr[0] = del[0];
      if(k > 1)
      { for (i=0; i<(k-1); i++)
        { if(del[i+1] != del[i])
          { dum++;
            delr[dum] = del[i+1];
          }
        }

        if(dum > 0)
          k = dum+1;
      }

      if(k > 0)
      { for (i=0; i<k; i++)
        { for (j=0; j<=walbd; j++)
          { if(index[j] > del[i])
            { index[j]--;
              area_iamtl[j]--;
            }
          }
        }

        for (i=0; i<k; i++)
        { if(delr[i] == walbd)
            walbd--;
          else
          { for (j=delr[i]; j<walbd; j++)
            { index[j] = index[j+1];
              value[j] = value[j+1];
            }
            walbd--;
          }
        }
      }
    }

/*-----------------------------------------------------------------------
 * --- Delete the excess "MTRL" cards
 *-----------------------------------------------------------------------
 */
  wmtrl = mtrl;
  if(wmtrl >= 0)
  { for (i=0; i<=wmtrl; i++)
      for (j=0; j<MXMTR; j++)
      { matter[i][j] = mtrl_lmtl[i][j];
        weight[i][j] = mtrl_wmtl[i][j];
      }
  }
```

```
k = 0;
if(wmtrl > 0)
{ for (i=0; i<wmtrl; i++)
  { flag = TRUE;

    for (dum=0; dum<MXMTR; dum++)
    { if( matter[i][dum] != 3.0 || weight[i][dum] != 0.0 )
        flag = FALSE;
    }

    if(flag)
    { regn_izmtl[i] = 0.0;
      delr[k] = i;
      k++;
    }
    else
    { for (j=i+1; j<=wmtrl; j++)
      { flag = TRUE;

        for (dum=0; dum<MXMTR; dum++)
        { if( (matter[j][dum] != matter[i][dum]) ||
              (weight[j][dum] != weight[i][dum]) )
            flag = FALSE;
        }

        if(flag)
        { regn_izmtl[j] = regn_izmtl[i];
          delr[k] = j;
          k++;
        }
      }
    }
  }

  if(k > 1)
  { for (i=0; i<(k-1); i++)
    { flag = FALSE;

      for (j=0; j<(k-1); j++)
      { if(delr[j+1] > delr[j])
        { dum = delr[j+1];
          delr[j+1] = delr[j];
          delr[j] = dum;
          flag = TRUE;
        }
      }

      if(!flag)
        break;
    }
  }

  dum = 0;
  del[0] = delr[0];
  if(k > 1)
  { for (i=0; i<(k-1); i++)
    { if(delr[i+1] != delr[i])
      { dum++;
        del[dum] = delr[i+1];
      }
    }
```

```
      if(dum > 0)
        k = dum+1;
    }

    if(k > 0)
    { for (i=0; i<k; i++)
      { for (j=0; j<=regn; j++)
        { if(regn_izmtl[j] > del[i])
            regn_izmtl[j]--;
        }
      }

      for (i=0; i<k; i++)
      { if(del[i] == wmtrl)
          wmtrl--;
        else
        { for (j=del[i]; j<wmtrl; j++)
          { for (dum=0; dum<MXMTR; dum++)
            { matter[j][dum] = matter[j+1][dum];
              weight[j][dum] = weight[j+1][dum];
            }
          }
          wmtrl--;
        }
      }
    }
  }

/*-----------------------------------------------------------------------
 * --- Open the requested file and write the input cards to it.
 *-----------------------------------------------------------------------
 */
  if((fp = fopen(file_name, "w")) != NULL)
  { if( mdl1 == 0)
    { sprintf(card, "%s%10.4f%10.4f%10.4f%10.4f%10.4f\n", "MDL1      ",
              mdl1_iaersl, mdl1_model, mdl1_ivis, mdl1_iseasn,
              mdl1_ivulcn);
      fputs(card, fp);
    }

    if( mdl2 == 0)
    { sprintf(card, "%s%10.4f%10.4f%10.4f%10.4f\n", "MDL2      ",
              mdl2_sn, mdl2_tbound, mdl2_ialb, mdl2_ip);
      fputs(card, fp);
    }

    if( mdl3 == 0)
    { sprintf(card, "%s%10.4f%10.4f%10.4f%10.4f%10.4f%10.4f\n",
              "MDL3      ", mdl3_t[0], mdl3_t[1], mdl3_t[2], mdl3_t[3],
              mdl3_t[4], mdl3_t[5]);
      fputs(card, fp);
    }

    if( area >= 0)
    { for (j=0; j<=area; j++)
      { sprintf(card, "%s%10.4f%10.4f%10.4f%10.4f%10.4f\n",
                "AREA      ", area_alx[j], area_ahx[j], area_aly[j],
                area_ahy[j], area_iamtl[j]);
        fputs(card, fp);
      }
    }
```

```
if( regn >= 0)
  for (j=0; j<=regn; j++)
  { regn_rlx[j] = regn_rl[0][j];
    regn_rhx[j] = regn_rh[0][j];
    regn_rly[j] = regn_rl[1][j];
    regn_rhy[j] = regn_rh[1][j];
    regn_rlz[j] = regn_rl[2][j];
    regn_rhz[j] = regn_rh[2][j];

    sprintf(card, "%s%10.4f%10.4f%10.4f%10.4f%10.4f%10.4f%10.4f\n",
            "REGN      ", regn_rlx[j], regn_rhx[j], regn_rly[j],
            regn_rhy[j], regn_rlz[j], regn_rhz[j], regn_izmtl[j]);
    fputs(card, fp);
  }

mesx = mes[0];
if( mesx >= 0)
  for (j=0; j<=mesx; j++)
  { mesx_mhx[j] = mes_mh[0][j];
    mesx_xms[j] = mes_ms[0][j];

    sprintf(card, "%s%10.4f%10.4f\n", "MESX      ", mesx_mhx[j],
            mesx_xms[j]);
    fputs(card, fp);
  }

mesy = mes[1];
if( mesy >= 0)
  for (j=0; j<=mesy; j++)
  { mesy_mhy[j] = mes_mh[1][j];
    mesy_yms[j] = mes_ms[1][j];

    sprintf(card, "%s%10.4f%10.4f\n", "MESY      ", mesy_mhy[j],
            mesy_yms[j]);
    fputs(card, fp);
  }

mesz = mes[2];
if( mesz >= 0)
  for (j=0; j<=mesz; j++)
  { mesz_mhz[j] = mes_mh[2][j];
    mesz_zms[j] = mes_ms[2][j];

    sprintf(card, "%s%10.4f%10.4f\n", "MESZ      ", mesz_mhz[j],
            mesz_zms[j]);
    fputs(card, fp);
  }

if( walbd >= 0 && mdl2_ialb < 0)
{ for (j=0; j<=albd; j++)
  { sprintf(card, "%s%10.4f%10.4f\n", "ALBD      ", index[j],
            value[j]);
    fputs(card, fp);
  }
}

if( wmtrl >= 0)
  for (j=0; j<=wmtrl; j++)
  { for(i=0; i<MXMTR; i++)
    { matter[j][i] -= 3;
      if(matter[j][i] > 40)
        matter[j][i] += 60;
```

```
      }

      sprintf(card,"%s%10.4f%10.4f%10.4f%10.4f%10.4f%10.4f\n",
              "MTRL      ", matter[j][0], weight[j][0], matter[j][1],
              weight[j][1], matter[j][2], weight[j][2]);
      fputs(card, fp);
    }

  if( clds == 0)
  { sprintf(card, "%s%10.4f%10.4f%10.4f\n", "CLDS      ", clds_icld,
            clds_ibnd, clds_wind);
    fputs(card, fp);
  }

  if( domd == 0)
  { sprintf(card, "%s%10.4f%10.4f%10.4f%10.4f%10.4f\n", "DOMD      ",
            domd_isc, domd_iitl, domd_epsi, domd_idelta, domd_npts);
    fputs(card, fp);
  }

  if( sun == 0)
  { sprintf(card, "%s%10.4f%10.4f%10.4f%10.4f%10.4f\n", "SUN       ",
            sun_thsun, sun_phsun, sun_ifsun, sun_isky, sun_iftrn);
    fputs(card, fp);
  }

  if( flar >= 0)
  { for (j=0; j<=flar; j++)
    { sprintf(card, "%s%10.4f%10.4f%10.4f%10.4f%10.4f\n",
              "FLAR      ", flar_idflr[j], flar_itflr[j],
              flar_xflar[j], flar_yflar[j], flar_zflar[j]);
      fputs(card, fp);

      sprintf(card, "%s%10.4f%10.1f%10.1f\n",
              "FLEN      ", flar_idflr[j], flar_qflar[j],
              flar_tflar[j]);
      fputs(card, fp);

      if(flar_itflr[j] == 2)
      { sprintf(card, "%s%10.4f%10.4f%10.4f%10.4f%10.4f\n",
                "FLUP      ", flar_idflr[j], flar_frrfup[j][0],
                flar_frrfup[j][1], flar_frrfup[j][2],
                flar_frrfup[j][3]);
        fputs(card, fp);

        sprintf(card, "%s%10.4f%10.4f%10.4f%10.4f%10.4f\n",
                "FLDN      ", flar_idflr[j], flar_frrfdn[j][0],
                flar_frrfdn[j][1], flar_frrfdn[j][2],
                flar_frrfdn[j][3]);
        fputs(card, fp);
      }
    }
  }

  if( srch == 0)
  { sprintf(card, "%s%10.4f%10.4f%10.4f%10.4f%10.4f\n",
            "SRCH      ", srch_xsrch, srch_ysrch, srch_zsrch,
            srch_thsrch, srch_azsrch);
    fputs(card, fp);

    sprintf(card, "%s%10.1f%10.1f%10.4f\n",
            "SREN      ", srch_psrch, srch_tmsrch, srch_sdiam);
```

```
      fputs(card, fp);
    }

    if( wavn == 0)
    { dv_dum = (wavn_v2 - wavn_v1) / wavn_dv;
      sprintf(card, "%s%10.3f%10.3f%10.3f\n", "WAVN      ", wavn_v1,
              wavn_v2, dv_dum);
      fputs(card, fp);
    }

    if( asci == 0)
    { sprintf(card, "%s%10.4f\n", "ASCI      ", asci_irite);
      fputs(card, fp);
    }

    if( recl == 0)
    { sprintf(card, "%s%10.4f\n", "RECL      ", recl_irpt);
      fputs(card, fp);
    }

    sprintf(card, "%s\n", "DONE");
    fputs(card, fp);

/*-----------------------------------------------------------------------
 * --- When finished writing all the data, close the file.
 *-----------------------------------------------------------------------
 */
    fclose(fp);
  }

/*-----------------------------------------------------------------------
 * --- Error occurred when opening the Savefile.
 *-----------------------------------------------------------------------
 */
  else
  { printf(" Error in opening the Savefile.\n");
    printf(" Input cards NOT Saved.\n");
    create_message( menu, error_mesg, XmDIALOG_ERROR);
  }

  drawscene();
}      /* end writecards() */

/***********************************************************************
 *                          XMSTRING XSTR2XMSTR
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  xstr2xmstr
 *                          Type:  C XmString
 *                      Filename:  visual.c
 *                        Parent:  create_message, create_messagef
 *=====================================================================
 *<Description>
 *    Converts arrays of string to an array of compound strings.
 *=====================================================================
 *<Called routines>
 *    None
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       XmString xstr2xmstr( char *chararray[], int n)
 *    Input:
```

```
 *       *chararray[]        - pointer to char array containing the
 *                             message to be displayed
 *       n                   - number of non-NULL characters in string
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Adapted the original SGI graphics course source code.
 *=====================================================================
 *<End>
 *********************************************************************
 */
XmString xstr2xmstr(char *chararray[], int n)
{
  XmString xmstr;
  int i;

/*-------------------------------------------------------------------
 * --- If the array is empty, return an empty XmString string.
 *-------------------------------------------------------------------
 */
  if (n <= 0)
    return (XmStringCreate("", XmSTRING_DEFAULT_CHARSET));

/*-------------------------------------------------------------------
 * --- If not, convert the input string to an XmString string.
 *-------------------------------------------------------------------
 */
  for (xmstr = (XmString) NULL, i = 0; i < n; i++)
  { if(i > 0)
      xmstr = XmStringConcat(xmstr, XmStringSeparatorCreate ());
      xmstr = XmStringConcat(xmstr, XmStringCreate(chararray[i],
                              XmSTRING_DEFAULT_CHARSET));
  }

/*-------------------------------------------------------------------
 * --- Return the XmString string.
 *-------------------------------------------------------------------
 */
  return (xmstr);
}     /* end xstr2xmstr() */

/*********************************************************************
 *                          VOID RESET
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  reset
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  several (main, set_axis_pts, etc.)
 *====================================================================
 *<Description>
 *    Resets the values of the structure "mov" to their initial values.
 *====================================================================
 *<Called routines>
 *    None
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void reset(void)
 *    Input:
```

```
 *       None
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void reset(void)
{
  mov->magfactor = org_magfactor;

  if(view_axis[0])
  { mov->ndx = 10.0 * org_center[1];
    mov->ndy = 10.0 * org_center[2];
  }
  else if(view_axis[1])
  { mov->ndx = 10.0 * org_center[0];
    mov->ndy = 10.0 * org_center[2];
  }
  else if(view_axis[2])
  { mov->ndx = 10.0 * org_center[0];
    mov->ndy = 10.0 * org_center[1];
  }

  mov->tdx = 10.0 * org_offset[0];
  mov->tdy = 10.0 * org_offset[1];
}
/*      end reset()  */

/********************************************************************
 *                          VOID DRAWSCENE
 ********************************************************************
 *<Begin>
 *<Identification>          Name:  drawscene
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  Numerous routines
 *=====================================================================
 *<Description>
 *    Plots the 3-D BLIRB grid points, albedo areas, aerosol regions,
 *    and the output flux.
 *=====================================================================
 *<Called routines>
 *    dist_sun              - determines the "plot-distance" from the
 *                            Earth to the Sun
 *    plot_areas            - plots the BLIRB surface albedo areas
 *    plot_axes             - plots the main BLIRB region axes
 *    plot_sun              - plots the position of the Sun in BLIRB
 *                            space
 *    plot_regions          - plots the 3-D BLIRB aerosol regions
 *    plot_flars            - plots the BLIRB Flares
 *    plot_slite            - plots the BLIRB SearchLights
 *    plot_flux             - plots the requested BLIRB flux field
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void drawscene(void)
 *    Input:
```

```
 *       None
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void drawscene(void)
{
  static long blackcol[] = { 0, 0, 0 };
  static int axis = 1;

  float xpoint, ypoint, zpoint;          /* Eyepoint/Viewpoint       */
  float xref, yref, zref;                /* Reference Point          */

/*-------------------------------------------------------------------
 * --- Determine the viewing window dimensions
 *-------------------------------------------------------------------
 */
  float xu =  0.5 * WINDOW_HEIGHT * (float)xsize/(float)ysize;
  float xl = -xu;
  float yu =  0.5 * WINDOW_HEIGHT;
  float yl = -yu;
  float fac = WINDOW_HEIGHT / (float)ysize;

/*-------------------------------------------------------------------
 * --- Determine the viewpoint or eyepoint
 *-------------------------------------------------------------------
 */
  xpoint = 0.1*(float)mov->ndx;
  ypoint = 0.1*(float)mov->ndy;
  zpoint =  20.0;

/*-------------------------------------------------------------------
 * --- Determine the viewed point or reference point
 *-------------------------------------------------------------------
 */
  if(view_axis[0])
  { xref = org_center[1];
    yref = org_center[2];
    zref = org_center[0];
  }
  else if(view_axis[1])
  { xref = org_center[0];
    yref = org_center[2];
    zref = -org_center[1];
  }
  else if(view_axis[2])
  { xref = org_center[0];
    yref = org_center[1];
    zref = org_center[2];
  }

/*-------------------------------------------------------------------
 * --- Restrict the rotation movement such that the underside of the
 *     albedo areas cannot be seen
 *-------------------------------------------------------------------
 */
```

```
  if(!view_axis[2] && ypoint < yref)
  { ypoint = yref;
    mov->ndy = 10.0 * ypoint;
  }

  ortho( xl, xu, yl, yu, 0.0, 125.0);      /* Use "ortho" projection   */

  pushmatrix();                            /* Save Modelview Matrix    */
/*-----------------------------------------------------------------------
 * --- If the viewing axis has been unchanged since last viewing, then
 *     perform any translation (if necessary).  If the viewing axis has
 *     changed, then save the current viewing axis and set the
 *     translation to (0,0).
 *-----------------------------------------------------------------------
 */
    if(((axis == -1) && view_axis[0]) ||
       ((axis ==  0) && view_axis[1]) ||
       ((axis ==  1) && view_axis[2]))
      translate( -0.01*(float)mov->tdx, -0.01*(float)mov->tdy, 0.0);
    else
    { if(view_axis[0])                     /* Save the currnt view-axis*/
        axis = -1;
      else if(view_axis[1])
        axis =  0;
      else if(view_axis[2])
        axis =  1;

      mov->tdx = 0;                        /* Set translation to (0,0) */
      mov->tdy = 0;
    }

    sun_earth[2] = axis_pts[2][0];

/*-----------------------------------------------------------------------
 * --- Determine the distance from the Sun to the ground
 *-----------------------------------------------------------------------
 */
    dist_sun();

/*-----------------------------------------------------------------------
 * --- Setup the view line-of-sight and scale the plot if necessary.
 *-----------------------------------------------------------------------
 */
    lookat(xpoint, ypoint, zpoint, xref, yref, zref, 0);
    scale(mov->magfactor, mov->magfactor, 1.0);

/*-----------------------------------------------------------------------
 * --- Rotate the plot to get the requested viewing orientation
 *-----------------------------------------------------------------------
 */
    if(!view_axis[2])
      rotate(-900, 'x');
    if(view_axis[0])
      rotate(-900, 'z');

    c3i(blackcol);                         /* set background to black  */
    czclear(0x000000, getgdesc(GD_ZMAX)); /* Clear Screen & Z-buffer  */

/*-----------------------------------------------------------------------
 * --- Plot the components required for the complete BLIRB scene
 *-----------------------------------------------------------------------
 */
```

```
    if(!nofile)
    { pushmatrix();
        zbuffer(TRUE);                      /* Turn on depth checking   */
        plot_areas();                       /* Plot the Albedo Areas    */
        plot_axes();                        /* Plot the Region Axes     */
        if(sun_plot)
          plot_sun();                       /* Plot the Sun position    */
        zbuffer(FALSE);                     /* Turn off depth checking  */
        plot_regions();                     /* Plot the BLIRB Regions   */
        if(flar >= 0)
          plot_flars();                     /* Plot the Flares          */
        if(srch >= 0)
          plot_slite();                     /* Plot the searchLights    */
        if(!noflux)
          plot_flux();                      /* Plot the Output Flux     */
      popmatrix();
    }
  popmatrix();                              /* Restore Modelview Matrix */

  swapbuffers();                            /* Display the current plot */
}   /*  end drawscene()  */

/*********************************************************************
 *                          VOID PLOT_FLUX
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  plot_flux
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  drawscene
 *====================================================================
 *<Description>
 *    Plots the requested BLIRB flux field.
 *====================================================================
 *<Called routines>
 *    plot_flux_axis        - draws the axis and plot scale of the
 *                            requested BLIRB flux
 *    plot_flux_tran        - draws the "3-D surface" in transparent
 *                            color of the requested BLIRB flux
 *    plot_flux_base        - draws the grid lines and the wire frame
 *                            surface for the selected cross-section
 *                            for the requested BLIRB flux.
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void plot_flux(void)
 *    Input:
 *       None
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void plot_flux(void)
{
  int i, n, dum, out_indx[3];
```

```
  for (i=0; i<10; i++)
    if(flux_flag[i])

/*-----------------------------------------------------------------------
 * --- For the requested BLIRB flux field and requested cross-section
 *     orientation, determine the axis index values used in the called
 *     routines.
 *-----------------------------------------------------------------------
 */
    { if(cross_axis[0])
      { out_indx[0] = 0;
        out_indx[1] = 1;
        out_indx[2] = 2;
        n = 0;
      }
      else if(cross_axis[1])
      { out_indx[0] = 1;
        out_indx[1] = 0;
        out_indx[2] = 2;
        n = 1;
      }
      else if(cross_axis[2])
      { out_indx[0] = 2;
        out_indx[1] = 0;
        out_indx[2] = 1;
        n = 2;
      }

/*-----------------------------------------------------------------------
 * --- Plot the flux axis and scale
 *-----------------------------------------------------------------------
 */
      plot_flux_axis(out_indx, cross_value[n]);

/*-----------------------------------------------------------------------
 * --- If transparent colors is requested, plot the flux cross-section
 *     values in 3-D in transparent color.
 *-----------------------------------------------------------------------
 */
      if(transparency)
        plot_flux_tran(out_indx, cross_value[n], i);

/*-----------------------------------------------------------------------
 * --- Plot the flux cross-section values in a 3-D wire mesh
 *-----------------------------------------------------------------------
 */
      plot_flux_base(out_indx, cross_value[n], i);

      dum = out_indx[1];
      out_indx[1] = out_indx[2];
      out_indx[2] = dum;

      plot_flux_base(out_indx, cross_value[n], i);
    }
}   /*  end plot_flux()  */

/***********************************************************************
 *                          VOID PLOT_FLUX_AXIS
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  plot_flux_axis
 *                          Type:  C void
```

```
 *                        Filename:  visual.c
 *                          Parent:  plot_flux
 *=====================================================================
 *<Description>
 *    Draws the axis and plot scale of the requested BLIRB flux.
 *=====================================================================
 *<Called routines>
 *    None
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void plot_flux_axis( int out_indx[3], int cur_imx)
 *    Input:
 *       out_indx[3]         - vector of axis index values
 *       cur_imx             - index for current flux cross-section
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void plot_flux_axis(int out_indx[3], int cur_imx)
{
  static long white[] = { 255, 255, 255};
  int i, j, k, l, m;
  float v0[3], v1[3], v2[3];
  char text[4];

  if(flux_index_low < 0 || flux_index_high > 0)
  { c3i(white);

    k = out_indx[0];
    l = out_indx[1];
    m = out_indx[2];

/*-------------------------------------------------------------------
 * --- Draw the Flux Axis Line
 *-------------------------------------------------------------------
 */
    v0[k] = out_m0[k][cur_imx] + 0.5 * (float) flux_index_low;
    v1[k] = out_m0[k][cur_imx] + 0.5 * (float) flux_index_high;
    v0[l] = v1[l] = out_m0[l][0];
    v0[m] = v1[m] = out_m0[m][0];

    bgnline();
      v3f(v0);
      v3f(v1);
    endline();

/*-------------------------------------------------------------------
 * --- Draw the minor tick marks on the Flux Axis Line
 *-------------------------------------------------------------------
 */
    v1[l] = v0[l] - 0.07071;
    v1[m] = v0[m] - 0.07071;

    for (i=flux_index_low; i<=flux_index_high; i++)
    { v1[k] = v0[k];
```

```
      bgnline();
        v3f(v0);
        v3f(v1);
      endline();

      v0[k] += 0.5;
    }

/*-----------------------------------------------------------------------
 * --- Draw the major tick marks on the Flux Axis Line
 *-----------------------------------------------------------------------
 */
    j = flux_index_low/5;
    flux_index_low = 5 * j;

    v0[k] = out_m0[k][cur_imx] + 0.5 * (float) flux_index_low;
    v1[l] = v0[l] - 0.14142;
    v1[m] = v0[m] - 0.14142;

    v2[l] = v1[l] - 0.2;
    v2[m] = v1[m] - 0.2;

    for (i=flux_index_low; i<=flux_index_high; i += 5)
    { v2[k] = v1[k] = v0[k];

      bgnline();
        v3f(v0);
        v3f(v1);
      endline();

      cmov( v2[0], v2[1], v2[2]);
      sprintf(text, "%d", i);              /* Compose major tick label */
      charstr(text);                       /* Label major tick marks   */

      v0[k] += 2.5;
    }

/*-----------------------------------------------------------------------
 * --- Draw a major tick marks on the end of the positive Flux Axis Line
 *     and label it
 *-----------------------------------------------------------------------
 */
    v2[k] = v1[k] = v0[k] = out_m0[k][cur_imx] +
                            0.5 * (float) flux_index_high;

    bgnline();
      v3f(v0);
      v3f(v1);
    endline();

    cmov( v2[0], v2[1], v2[2]);
    sprintf(text, "%d", flux_index_high);
    charstr(text);
  }
}   /*  end plot_flux_axis()  */

/***********************************************************************
 *                          VOID PLOT_FLUX_TRAN
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  plot_flux_tran
 *                          Type:  C void
```

```
 *                        Filename:  visual.c
 *                          Parent:  plot_flux
 *=====================================================================
 *<Description>
 *    Draws the "3-D surface" in transparent color of the requested
 *    BLIRB flux.
 *=====================================================================
 *<Called routines>
 *    None
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void plot_flux_tran( int out_indx[3], int cur_imx, int flux)
 *    Input:
 *       out_indx[3]         - vector of axis index values
 *       cur_imx             - index for current flux cross-section
 *       flux                - index for the requested type of flux
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void plot_flux_tran( int out_indx[3], int cur_imx, int flux)
{
  static long red4[] = { 255, 0, 0, 0};
  int i, j, k, l, m, ii, jj, kk, iii, jjj, jj1, jj2, kkk;
  float v0[3], v1[3], v2[3], v3[3];

  red4[3] = 2 * trans_index;
  c4i(red4);

  k = out_indx[0];
  l = out_indx[1];
  m = out_indx[2];

/*-------------------------------------------------------------------
 * --- Turn on image blending
 *-------------------------------------------------------------------
 */
  blendfunction(BF_SA, BF_MSA);

/*-------------------------------------------------------------------
 * --- For each rectangle made from the grid points of the
 *     cross-section, determine the abscissa vertices of the rectangle.
 *-------------------------------------------------------------------
 */
  for (i=0; i<out_imx[l]-1; i++)
  { v0[l] = out_m0[l][i];
    v1[l] = out_m0[l][i+1];
    v2[l] = v1[l];
    v3[l] = v0[l];

    for (j=0; j<out_imx[m]-1; j++)
    { v0[m] = out_m0[m][j];
      v1[m] = v0[m];
      v2[m] = out_m0[m][j+1];
      v3[m] = v2[m];
```

```
/*-----------------------------------------------------------------------
 * --- Let the ordinate for each vertex be the requested flux value.
 *-----------------------------------------------------------------------
 */
      switch (k)
      { case 0:
          ii = cur_imx;
          iii = cur_imx;
          jj = i;
          jj1 = i+1;
          jjj = i+1;
          jj2 = i;
          kk = j;
          kkk = j+1;
          break;

        case 1:
          ii = i;
          iii = i+1;
          jj = cur_imx;
          jj1 = cur_imx;
          jjj = cur_imx;
          jj2 = cur_imx;
          kk = j;
          kkk = j+1;
          break;

        case 2:
          ii = i;
          iii = i+1;
          jj = j;
          jj1 = j;
          jjj = j+1;
          jj2 = j+1;
          kk = cur_imx;
          kkk = cur_imx;
          break;
      }

      if(!flux_zero[flux][cur_nwave])
      { v0[k] = 0.5 * out_flux[flux][cur_nwave][ii][jj][kk] +
                out_m0[k][cur_imx];
        v1[k] = 0.5 * out_flux[flux][cur_nwave][iii][jj1][kk] +
                out_m0[k][cur_imx];
        v2[k] = 0.5 * out_flux[flux][cur_nwave][iii][jjj][kkk] +
                out_m0[k][cur_imx];
        v3[k] = 0.5 * out_flux[flux][cur_nwave][ii][jj2][kkk] +
                out_m0[k][cur_imx];
      }
      else if((mini_flux[flux][cur_nwave] >= 0.0) &&
              (maxi_flux[flux][cur_nwave] > 0.0))
      { v0[k] = 0.5 * out_flux[flux][cur_nwave][ii][jj][kk]/log_range +
                out_m0[k][cur_imx];
        v1[k] = 0.5 * out_flux[flux][cur_nwave][iii][jj1][kk]/log_range+
                out_m0[k][cur_imx];
        v2[k] = 0.5 * out_flux[flux][cur_nwave][iii][jjj][kkk]/
                log_range + out_m0[k][cur_imx];
        v3[k] = 0.5 * out_flux[flux][cur_nwave][ii][jj2][kkk]/log_range+
                out_m0[k][cur_imx];
      }
      else
        v0[k] = v1[k] = v2[k] = v3[k] = out_m0[k][cur_imx];
```

```
/*-----------------------------------------------------------------------
 * --- Draw the 3-D Flux polygon from the top
 *-----------------------------------------------------------------------
 */
      bgntmesh();
        v3f(v0);
        v3f(v1);
        v3f(v3);
        v3f(v2);
      endtmesh();

/*-----------------------------------------------------------------------
 * --- Draw the 3-D Flux polygon from the bottom
 *-----------------------------------------------------------------------
 */
      bgntmesh();
        v3f(v0);
        v3f(v3);
        v3f(v1);
        v3f(v2);
      endtmesh();
    }
  }

/*-----------------------------------------------------------------------
 * --- Turn off image blending
 *-----------------------------------------------------------------------
 */
  blendfunction(BF_ONE, BF_ZERO);
}   /*  end plot_flux_tran()  */

/***********************************************************************
 *                           VOID PLOT_FLUX_BASE
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  plot_flux_base
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  plot_flux
 *=====================================================================
 *<Description>
 *    Draws the grid lines and the wire frame surface for the selected
 *    cross-section for the requested BLIRB flux.
 *=====================================================================
 *<Called routines>
 *    None
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void plot_flux_base( int out_indx[3], int cur_imx, int flux)
 *    Input:
 *       out_indx[3]        - vector of axis index values
 *       cur_imx            - index for current flux cross-section
 *       flux               - index for the requested type of flux
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
```

```
 ***********************************************************************
 */
void plot_flux_base( int out_indx[3], int cur_imx, int flux)
{
  static long white[] = { 255, 255, 255};
  static long red[] = { 255, 0, 0};
  static Boolean flag = TRUE;              /* Initialize "flag"        */
  int i, j, k, l, m, ii, jj, kk, iii, jjj, kkk;
  float v0[3], v1[3];

  k = out_indx[0];
  l = out_indx[1];
  m = out_indx[2];

/*-----------------------------------------------------------------------
 * --- For each rectangle made from the grid points of the
 *     cross-section, determine the abscissa vertices of the rectangle.
 *-----------------------------------------------------------------------
 */
  for (i=0; i<out_imx[l]; i++)
  { v0[l] = out_m0[l][i];
    v1[l] = v0[l];

    for (j=0; j<out_imx[m]-1; j++)
    { v0[m] = out_m0[m][j];
      v1[m] = out_m0[m][j+1];

/*-----------------------------------------------------------------------
 * --- Draw the cross-section grid lines
 *-----------------------------------------------------------------------
 */
      v0[k] = v1[k] = out_m0[k][cur_imx];

      c3i(white);

      bgnline();
        v3f(v0);
        v3f(v1);
      endline();

/*-----------------------------------------------------------------------
 * --- Draw the cross-section flux wire mesh 3-D surface
 *-----------------------------------------------------------------------
 */
      switch (k)
      { case 0:
          ii = cur_imx;
          iii = cur_imx;

          if(flag)
          { jj = i;
            jjj = i;
            kk = j;
            kkk = j+1;
          }
          else
          { jj = j;
            jjj = j+1;
            kk = i;
            kkk = i;
          }
```

```
        break;

      case 1:
        jj = cur_imx;
        jjj = cur_imx;

        if(flag)
        { ii = i;
          iii = i;
          kk = j;
          kkk = j+1;
        }
        else
        { ii = j;
          iii = j+1;
          kk = i;
          kkk = i;
        }

        break;

      case 2:
        kk = cur_imx;
        kkk = cur_imx;

        if(flag)
        { ii = i;
          iii = i;
          jj = j;
          jjj = j+1;
        }
        else
        { ii = j;
          iii = j+1;
          jj = i;
          jjj = i;
        }

        break;
    }

    if(!flux_zero[flux][cur_nwave])
    { v0[k] = 0.5 * out_flux[flux][cur_nwave][ii][jj][kk] +
              out_m0[k][cur_imx];
      v1[k] = 0.5 * out_flux[flux][cur_nwave][iii][jjj][kkk] +
              out_m0[k][cur_imx];
    }
    else if((mini_flux[flux][cur_nwave] >= 0.0) &&
            (maxi_flux[flux][cur_nwave] > 0.0))
    { v0[k] = 0.5 * out_flux[flux][cur_nwave][ii][jj][kk] /
              log_range + out_m0[k][cur_imx];
      v1[k] = 0.5 * out_flux[flux][cur_nwave][iii][jjj][kkk] /
              log_range + out_m0[k][cur_imx];
    }
    else
      v0[k] = v1[k] = out_m0[k][cur_imx];

    c3i(red);

    bgnline();
      v3f(v0);
      v3f(v1);
```

```
      endline();
    }
  }

/*-----------------------------------------------------------------------
 * --- Flip the value of the "flag" to indicate whether it is the first
 *     or second pass through the routine.
 *-----------------------------------------------------------------------
 */
  flag = !flag;
}   /*  end plot_flux_base()  */

/***********************************************************************
 *                            VOID DIST_SUN
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  dist_sun
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  drawscene
 *=====================================================================
 *<Description>
 *    Determines the "plot-distance" from the Earth to the Sun.
 *=====================================================================
 *<Called routines>
 *    None
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void dist_sun( void)
 *    Input:
 *       None
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void dist_sun(void)
{
  static float pi_over_180 = 0.01745329;
  float t, p, sp, cp, tt;

  t = pi_over_180 * sun_thsun;
  p = pi_over_180 * sun_phsun;
  tt = ftan(t);

  if(sun_plot == TRUE)
  { if(view_axis[0])
    { sp = fsin(p);
      if((sun_distance[2] - sun_earth[2])*tt*sp <=
         (sun_distance[1] - sun_earth[1]))
        sun_dist = (sun_distance[2] - sun_earth[2]) / fcos(t);
      else
        sun_dist = (sun_distance[1] - sun_earth[1]) / (fsin(t) * sp);
    }

    else if(view_axis[1])
```

```
    { cp = fcos(p);
      if((sun_distance[2] - sun_earth[2])*tt*cp <=
         (sun_distance[0] - sun_earth[0]))
        sun_dist = (sun_distance[2] - sun_earth[2]) / fcos(t);
      else
        sun_dist = (sun_distance[0] - sun_earth[0]) / (fsin(t) * cp);
    }

    else if(view_axis[2])
    { tt = (sun_distance[2] - sun_earth[2]) * tt;
      if(tt < (sun_distance[0] - sun_earth[0]) &&
         tt < (sun_distance[1] - sun_earth[1]))
        sun_dist = (sun_distance[2] - sun_earth[2]) / fcos(t);
      else
      { if((sun_distance[0] - sun_earth[0]) * ftan(p) <=
        (sun_distance[1] - sun_earth[1]))
          sun_dist = (sun_distance[0] - sun_earth[0]) /
                     (fsin(t) * fcos(p));
        else
          sun_dist = (sun_distance[1] - sun_earth[1]) /
                     (fsin(t) * fsin(p));
      }
    }
  }
}     /*    end dist_sun()    */

/*****************************************************************
 *                          VOID PLOT_SUN
 *****************************************************************
 *<Begin>
 *<Identification>          Name:  plot_sun
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  drawscene
 *===============================================================
 *<Description>
 *    Plots the position of the Sun in BLIRB space
 *===============================================================
 *<Called routines>
 *    None
 *===============================================================
 *<Parameters>
 *    Formal declaration:
 *       void plot_sun( void)
 *    Input:
 *       None
 *    Output:
 *       None
 *===============================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *===============================================================
 *<End>
 *****************************************************************
 */
void plot_sun(void)
{
  static float pi_over_180 = 0.01745329;
  static float pi_over_8 = 0.392699;
  static long col[] = { 255, 200, 0};
```

```
  float sinzenith, t, p;
  float v0[3], v1[3], v2[3], v3[3], v4[3];
  int i, j, k;

/*-----------------------------------------------------------------------
 * --- Determine the location of the center of the Sun in the plot
 *-----------------------------------------------------------------------
 */
  t = pi_over_180 * sun_thsun;
  p = pi_over_180 * (90.0 - sun_phsun);
  sinzenith = sun_dist * fsin(t);
  v0[0] = sinzenith * fcos(p) + sun_earth[0];
  v0[1] = sinzenith * fsin(p) + sun_earth[1];
  v0[2] = sun_dist * fcos(t) + sun_earth[2];

  for (i=0; i<3; i++)
    sun_sun[i] = v0[i];

/*-----------------------------------------------------------------------
 * --- Draw the Sun a polygon at a time.  Each polygon will be 22.5 deg
 *     azimuth angle by 22.5 deg zenith angle.
 *-----------------------------------------------------------------------
 */
  c3i(col);

  for (i=0; i<16; i++)
  { v1[0] = v0[0];
    v1[1] = v0[1];
    v1[2] = v0[2] + sun_radius;

    for (j=0; j<3; j++)
      v4[j] = v1[j];

    for (j=0; j<8; j++)
    { sinzenith = sun_radius * fsin(pi_over_8 * (float) (j+1));
      v2[0] = v0[0] + sinzenith * fcos(pi_over_8 * (float) i);
      v2[1] = v0[1] + sinzenith * fsin(pi_over_8 * (float) i);
      v2[2] = v0[2] + sun_radius * fcos(pi_over_8 * (float) (j+1));

      v3[0] = v0[0] + sinzenith * fcos(pi_over_8 * (float) (i+1));
      v3[1] = v0[1] + sinzenith * fsin(pi_over_8 * (float) (i+1));
      v3[2] = v2[2];

      bgntmesh();
        v3f(v1);
        v3f(v2);
        v3f(v4);
        v3f(v3);
      endtmesh();

      for (k=0; k<3; k++)
      { v1[k] = v2[k];
        v4[k] = v3[k];
      }
    }
  }

/*-----------------------------------------------------------------------
 * --- Draw the line from the Sun to the ground.
 *-----------------------------------------------------------------------
 */
  for (i=0; i<3; i++)
```

```
    v1[i] = sun_earth[i];

  bgnline();
    v3f(v0);
    v3f(v1);
  endline();

/*-----------------------------------------------------------------------
 * --- Draw the label for the Sun
 *-----------------------------------------------------------------------
 */
  cmov(v0[0] + 2.0*sun_radius, v0[1] + 2.0*sun_radius, v0[2]);
  charstr("Sun");                          /* Label the Sun */
}     /*  end plot_sun()  */

/***********************************************************************
 *                          VOID PLOT_AREAS
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  plot_areas
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  drawscene
 *=====================================================================
 *<Description>
 *    Plots the BLIRB surface albedo areas.
 *=====================================================================
 *<Called routines>
 *    None
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void plot_areas( void)
 *    Input:
 *       None
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void plot_areas(void)
{
  static long white[] = { 255, 255, 255};
  static long red[] = { 255, 0, 0};
  int i, j;
  long col[3];
  float v0[3], v1[3], v2[3], v3[3];
  float num[3];
  char albedo[10];

  for (i=0; i <= area; i++)
/*-----------------------------------------------------------------------
 * --- Determine the shade of green to be used for each albedo area
 *     and set the color.
 *-----------------------------------------------------------------------
 */
  { if(i != cur_area || !move_area)
```

```
    { j = 25.0 + 230.0 * area_iamt[i];
      col[0] = col[2] = 0;
      col[1] = j;
      c3i(col);
    }
    else
     c3i(red);

/*-----------------------------------------------------------------------
 * --- Determine the vertices of the albedo area.
 *-----------------------------------------------------------------------
 */
    v0[0] = area_alx[i];
    v0[1] = area_aly[i];
    v0[2] = 0.001 * (float) i;

    v1[0] = area_ahx[i];
    v1[1] = v0[1];
    v1[2] = v0[2];

    v2[0] = v1[0];
    v2[1] = area_ahy[i];
    v2[2] = v0[2];

    v3[0] = v0[0];
    v3[1] = v2[1];
    v3[2] = v0[2];

/*-----------------------------------------------------------------------
 * --- Turn on the Z-buffering, shade the albedo area from the top
 *     side, and then turn off the Z-buffering.
 *-----------------------------------------------------------------------
 */
    if(i > 0) zfunction(ZF_ALWAYS);        /* Turn on Z-buffering      */

    bgnpolygon();                          /* Shade the albedo area    */
      v3f(v0);
      v3f(v1);
      v3f(v2);
      v3f(v3);
    endpolygon();

    if(i > 0) zfunction(ZF_LESS);          /* Turn off Z-buffering     */

/*-----------------------------------------------------------------------
 * --- For the whole BLIRB surface area (first area), turn on the
 *     Z-buffering, shade the albedo area from the bottom side, and then
 *     turn off the Z-buffering.
 *-----------------------------------------------------------------------
 */
    if(i == 0) zfunction(ZF_ALWAYS);       /* Turn on Z-buffering      */

    bgnpolygon();                          /* Shade the albedo area    */
      v3f(v0);
      v3f(v3);
      v3f(v2);
      v3f(v1);
    endpolygon();

    if(i == 0) zfunction(ZF_LESS);         /* Turn off Z-buffering     */

/*-----------------------------------------------------------------------
```

```
 * --- If albedo area is to be moved, compose a text message and display
 *     it at the appropriate location.
 *-----------------------------------------------------------------------
 */
    if(i == cur_area && move_area)
    { num[0] = area_alx[i];
      num[1] = area_aly[i] - 0.15;
      num[2] = -0.1;
      cmov( num[0], num[1], num[2]);        /* Move cursor to text locn */

      strcpy(albedo, "Move Area");          /* Compose the text         */

      c3i(red);                             /* Set color to white       */
      zfunction(ZF_ALWAYS);                 /* Turn on Z-buffering      */
      charstr(albedo);                      /* Display the text         */
      zfunction(ZF_LESS);                   /* Turn off Z-buffering     */
    }

/*-----------------------------------------------------------------------
 * --- If albedo area numbers are to be displayed in text form, then
 *     compose the text and display it at the appropriate location.
 *-----------------------------------------------------------------------
 */
    if(label_obsc)
    { num[0] = area_alx[i] + 0.1;
      num[1] = area_aly[i] + 0.1;
      num[2] = 0.05;
      cmov( num[0], num[1], num[2]);        /* Move cursor to text locn */

      sprintf(albedo, "%d", i+1);           /* Compose the text         */

      c3i(white);                           /* Set color to white       */
      zfunction(ZF_ALWAYS);                 /* Turn on Z-buffering      */
      charstr(albedo);                      /* Display the text         */
      zfunction(ZF_LESS);                   /* Turn off Z-buffering     */
    }
  }
}   /*  end plot_areas()  */

/***********************************************************************
 *                          VOID PLOT_REGIONS
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  plot_regions
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  drawscene
 *=====================================================================
 *<Description>
 *    Plots the 3-D BLIRB aerosol regions (volumes).
 *=====================================================================
 *<Called routines>
 *    regn_sides            - draws the 6 sides of the 3-D BLIRB aerosol
 *                            regions in transparent color.
 *    bottom_face           - outlines the 3-D BLIRB aerosol regions
 *                            bottom faces.
 *    top_face              - outlines the 3-D BLIRB aerosol regions
 *                            top faces.
 *    left_face             - outlines the 3-D BLIRB aerosol regions
 *                            left faces.
 *    right_face            - outlines the 3-D BLIRB aerosol regions
 *                            right faces.
```

```
 *    front_face              - outlines the 3-D BLIRB aerosol regions
 *                              front faces.
 *    back_face               - outlines the 3-D BLIRB aerosol regions
 *                              back faces.
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void plot_regions(void)
 *    Input:
 *       None
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 **********************************************************************
 */
void plot_regions(void)
{
  static long col_white[] = { 255, 255, 255};
  static long col_yellow[] = { 255, 255, 0};
  static long col_red[] = { 255, 0, 0};
  int i, k;
  long col[4];
  float regn_vertex[8][3], num[3];
  char text[12];

  for (i=0; i <= regn; i++)
/*-------------------------------------------------------------------
 * --- Determine the vertices of each aerosol region.
 *-------------------------------------------------------------------
 */
  { regn_vertex[0][0] = regn_vertex[3][0] = regn_vertex[4][0] =
                        regn_vertex[7][0] = regn_rl[0][i];

    regn_vertex[1][0] = regn_vertex[2][0] = regn_vertex[5][0] =
                        regn_vertex[6][0] = regn_rh[0][i];

    regn_vertex[0][1] = regn_vertex[1][1] = regn_vertex[4][1] =
                        regn_vertex[5][1] = regn_rl[1][i];

    regn_vertex[2][1] = regn_vertex[3][1] = regn_vertex[7][1] =
                        regn_vertex[6][1] = regn_rh[1][i];

    regn_vertex[0][2] = regn_vertex[1][2] = regn_vertex[2][2] =
                        regn_vertex[3][2] = (regn_rl[2][i]+0.02);

    regn_vertex[4][2] = regn_vertex[5][2] = regn_vertex[6][2] =
                        regn_vertex[7][2] = regn_rh[2][i];

/*-------------------------------------------------------------------
 * --- Determine the color for each aerosol region and set the color.
 *-------------------------------------------------------------------
 */
    for (k=0; k<3; k++)
    { if(i != cur_regn || (!move_regnh && !move_regnv))
        col[k] = mtrl_color[(int) regn_izmt[i]][k];
      else
        col[k] = col_red[k];
```

```
    }
    col[3] = 255;

    c4i(col);

/*-----------------------------------------------------------------------
 * --- If transparent colors are requested, set the transparent colors
 *     and draw the 6 sides of the aerosol region.  Else, set the color
 *     to yellow.
 *-----------------------------------------------------------------------
 */
    if(transparency)
    { col[3] = trans_index;                  /* Set the Transparency     */
      c4i(col);                              /* Set the Aerosol Color    */

      blendfunction(BF_SA, BF_MSA);          /* Image Blending ON        */
      regn_sides(regn_vertex);               /* Draw the region 6 sides  */
      blendfunction(BF_ONE, BF_ZERO);        /* Image Blending OFF       */

      c3i(col_white);                        /* Reset color to white     */
    }
    else if(col[0]+col[1]+col[2] == 0)       /* If color is white        */
      c3i(col_yellow);                       /*   set color to yellow    */

/*-----------------------------------------------------------------------
 * --- Draw the edges of the six faces of the aerosol region.
 *-----------------------------------------------------------------------
 */
    bottom_face(regn_vertex);                /* Outline bottom side      */
    left_face(regn_vertex);                  /* Outline left side        */
    right_face(regn_vertex);                 /* Outline right side       */
    front_face(regn_vertex);                 /* Outline front side       */
    back_face(regn_vertex);                  /* Outline back side        */
    top_face(regn_vertex);                   /* Outline top side         */

/*-----------------------------------------------------------------------
 * --- Determine the location where the region number should be
 *     displayed.
 *-----------------------------------------------------------------------
 */
    if(label_obsc)
    { for (k=0; k<3; k++)
        num[k] = 0.5 * (regn_vertex[0][k] + regn_vertex[6][k]);
      sprintf(text, "%d", i+1);              /* Compose the text         */
      cmov( num[0], num[1], num[2]);         /* Move to text location    */
      charstr(text);                         /* Display the text         */
    }

/*-----------------------------------------------------------------------
 * --- If the region is to be moved, determine the location where the
 *     text message reminder should be displayed.
 *-----------------------------------------------------------------------
 */
    if(i == cur_regn && (move_regnh || move_regnv))
    { c3i(col_red);
      num[0] = regn_vertex[0][0];
      num[1] = regn_vertex[0][1] - 0.15;
      num[2] = regn_vertex[0][2] - 0.15;
      strcpy(text, "Move Region");           /* Compose the text         */
      cmov( num[0], num[1], num[2]);         /* Move to text location    */
      charstr(text);                         /* Display the text         */
    }
```

```
  }
}    /* end plot_regions() */

/*********************************************************************
 *                          VOID REGN_SIDES
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  regn_sides
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  plot_regions
 *====================================================================
 *<Description>
 *    Draws the 6 sides of a 3-D BLIRB aerosol region in transparent
 *    color.
 *====================================================================
 *<Called routines>
 *    None
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void regn_sides( float regn_vertex[8][3])
 *    Input:
 *       regn_vertex        - vertices of the aerosol region
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void regn_sides(float regn_vertex[8][3])
{
/*-------------------------------------------------------------------
 * --- Draw and shade the bottom side.
 *-------------------------------------------------------------------
 */
  bgntmesh();
    v3f(regn_vertex[0]);
    v3f(regn_vertex[3]);
    v3f(regn_vertex[1]);
    v3f(regn_vertex[2]);
  endtmesh();

/*-------------------------------------------------------------------
 * --- Draw and shade the left side.
 *-------------------------------------------------------------------
 */
  bgntmesh();
    v3f(regn_vertex[0]);
    v3f(regn_vertex[4]);
    v3f(regn_vertex[3]);
    v3f(regn_vertex[7]);
  endtmesh();

/*-------------------------------------------------------------------
 * --- Draw and shade the right side.
 *-------------------------------------------------------------------
 */
```

```
  bgntmesh();
    v3f(regn_vertex[1]);
    v3f(regn_vertex[2]);
    v3f(regn_vertex[5]);
    v3f(regn_vertex[6]);
  endtmesh();

/*-----------------------------------------------------------------------
 * --- Draw and shade the front side.
 *-----------------------------------------------------------------------
 */
  bgntmesh();
    v3f(regn_vertex[0]);
    v3f(regn_vertex[1]);
    v3f(regn_vertex[4]);
    v3f(regn_vertex[5]);
  endtmesh();

/*-----------------------------------------------------------------------
 * --- Draw and shade the back side.
 *-----------------------------------------------------------------------
 */
  bgntmesh();
    v3f(regn_vertex[2]);
    v3f(regn_vertex[3]);
    v3f(regn_vertex[6]);
    v3f(regn_vertex[7]);
  endtmesh();

/*-----------------------------------------------------------------------
 * --- Draw and shade the top side.
 *-----------------------------------------------------------------------
 */
  bgntmesh();
    v3f(regn_vertex[4]);
    v3f(regn_vertex[5]);
    v3f(regn_vertex[7]);
    v3f(regn_vertex[6]);
  endtmesh();
}    /* end regn_sides() */

/***********************************************************************
 *                          VOID BOTTOM_FACE
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  bottom_face
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  plot_regions
 *=====================================================================
 *<Description>
 *    Outlines a 3-D BLIRB aerosol region bottom face and indicates
 *    where the line from the Sun to the ground intersects the face.
 *=====================================================================
 *<Called routines>
 *    None
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void bottom_face( float regn_vertex[8][3])
 *    Input:
 *       regn_vertex        - vertices of the aerosol region
```

```
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void bottom_face(float regn_vertex[8][3])
{
  float v0[3], v1[3], v2[3], v3[3], dum, sun[3];

/*-------------------------------------------------------------------
 * --- Outline the region face
 *-------------------------------------------------------------------
 */
  bgnline();
    v3f(regn_vertex[0]);
    v3f(regn_vertex[3]);
    v3f(regn_vertex[2]);
    v3f(regn_vertex[1]);
    v3f(regn_vertex[0]);
  endline();

  if(sun_plot)
/*-------------------------------------------------------------------
 * --- If the Sun is displayed, determine the location where the line
 *     from the Sun to the ground would intersect the face (if it does
 *     intersect the face).
 *-------------------------------------------------------------------
 */
  { sun[2] = regn_vertex[0][2];
    dum = (sun[2] - sun_earth[2]) / (sun_sun[2] - sun_earth[2]);
    sun[0] = dum * (sun_sun[0] - sun_earth[0]) + sun_earth[0];
    sun[1] = dum * (sun_sun[1] - sun_earth[1]) + sun_earth[1];

    if((sun[0] >= regn_vertex[0][0] &&
        sun[0] <= regn_vertex[2][0]) &&
       (sun[1] >= regn_vertex[0][1] &&
        sun[1] <= regn_vertex[2][1]))
/*-------------------------------------------------------------------
 * --- If the line from the Sun to the ground intersects the face,
 *     determine the vertices of a small square polygon on the face
 *     centered about the intersection point to indicate where the
 *     Sun-line intersected the face.
 *-------------------------------------------------------------------
 */
    { v0[0] = sun[0] - 0.02;
      v1[0] = sun[0] + 0.02;
      v2[0] = sun[0] + 0.02;
      v3[0] = sun[0] - 0.02;

      v0[1] = sun[1] - 0.02;
      v1[1] = sun[1] - 0.02;
      v2[1] = sun[1] + 0.02;
      v3[1] = sun[1] + 0.02;

      v0[2] = sun[2];
      v1[2] = sun[2];
      v2[2] = sun[2];
```

```
      v3[2] = sun[2];

/*-----------------------------------------------------------------------
 * --- Draw the polygon on the face.
 *-----------------------------------------------------------------------
 */
      bgnline();
        v3f(v0);
        v3f(v1);
        v3f(v2);
        v3f(v3);
        v3f(v0);
      endline();
    }
  }
}    /* end bottom_face() */

/***********************************************************************
 *                            VOID LEFT_FACE
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  left_face
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  plot_regions
 *=====================================================================
 *<Description>
 *    Outlines a 3-D BLIRB aerosol region left face and indicates
 *    where the line from the Sun to the ground intersects the face.
 *=====================================================================
 *<Called routines>
 *    None
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void left_face( float regn_vertex[8][3])
 *    Input:
 *       regn_vertex        - vertices of the aerosol region
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void left_face(float regn_vertex[8][3])
{
  float v0[3], v1[3], v2[3], v3[3], dum, sun[3];

/*-----------------------------------------------------------------------
 * --- Outline the region face
 *-----------------------------------------------------------------------
 */
  bgnline();
    v3f(regn_vertex[0]);
    v3f(regn_vertex[4]);
    v3f(regn_vertex[7]);
    v3f(regn_vertex[3]);
    v3f(regn_vertex[0]);
```

```
  endline();

  if(sun_plot)
/*-----------------------------------------------------------------------
 * --- If the Sun is displayed, determine the location where the line
 *     from the Sun to the ground would intersect the face (if it does
 *     intersect the face).
 *-----------------------------------------------------------------------
 */
  { sun[0] = regn_vertex[0][0];
    dum = (sun[0] - sun_earth[0]) / (sun_sun[0] - sun_earth[0]);
    sun[1] = dum * (sun_sun[1] - sun_earth[1]) + sun_earth[1];
    sun[2] = dum * (sun_sun[2] - sun_earth[2]) + sun_earth[2];

    if((sun[1] >= regn_vertex[0][1] &&
        sun[1] <= regn_vertex[7][1]) &&
       (sun[2] >= regn_vertex[0][2] &&
        sun[2] <= regn_vertex[7][2]))
/*-----------------------------------------------------------------------
 * --- If the line from the Sun to the ground intersects the face,
 *     determine the vertices of a small square polygon on the face
 *     centered about the intersection point to indicate where the
 *     Sun-line intersected the face.
 *-----------------------------------------------------------------------
 */
    { v0[0] = sun[0];
      v1[0] = sun[0];
      v2[0] = sun[0];
      v3[0] = sun[0];

      v0[1] = sun[1] - 0.02;
      v1[1] = sun[1] - 0.02;
      v2[1] = sun[1] + 0.02;
      v3[1] = sun[1] + 0.02;

      v0[2] = sun[2] - 0.02;
      v1[2] = sun[2] + 0.02;
      v2[2] = sun[2] + 0.02;
      v3[2] = sun[2] - 0.02;

/*-----------------------------------------------------------------------
 * --- Draw the polygon on the face.
 *-----------------------------------------------------------------------
 */
      bgnline();
        v3f(v0);
        v3f(v1);
        v3f(v2);
        v3f(v3);
        v3f(v0);
      endline();
    }
  }
}    /* end left_face() */

/***********************************************************************
 *                          VOID RIGHT_FACE
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  right_face
 *                          Type:  C void
 *                      Filename:  visual.c
```

```
 *                        Parent:  plot_regions
 *=====================================================================
 *<Description>
 *    Outlines a 3-D BLIRB aerosol region right face and indicates
 *    where the line from the Sun to the ground intersects the face.
 *=====================================================================
 *<Called routines>
 *    None
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void right_face( float regn_vertex[8][3])
 *    Input:
 *       regn_vertex        - vertices of the aerosol region
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void right_face(float regn_vertex[8][3])
{
  float v0[3], v1[3], v2[3], v3[3], dum, sun[3];

/*-------------------------------------------------------------------
 * --- Outline the region face
 *-------------------------------------------------------------------
 */
  bgnline();
    v3f(regn_vertex[1]);
    v3f(regn_vertex[2]);
    v3f(regn_vertex[6]);
    v3f(regn_vertex[5]);
    v3f(regn_vertex[1]);
  endline();

  if(sun_plot)
/*-------------------------------------------------------------------
 * --- If the Sun is displayed, determine the location where the line
 *     from the Sun to the ground would intersect the face (if it does
 *     intersect the face).
 *-------------------------------------------------------------------
 */
  { sun[0] = regn_vertex[1][0];
    dum = (sun[0] - sun_earth[0]) / (sun_sun[0] - sun_earth[0]);
    sun[1] = dum * (sun_sun[1] - sun_earth[1]) + sun_earth[1];
    sun[2] = dum * (sun_sun[2] - sun_earth[2]) + sun_earth[2];

    if((sun[1] >= regn_vertex[1][1] &&
        sun[1] <= regn_vertex[6][1]) &&
       (sun[2] >= regn_vertex[1][2] &&
        sun[2] <= regn_vertex[6][2]))
/*-------------------------------------------------------------------
 * --- If the line from the Sun to the ground intersects the face,
 *     determine the vertices of a small square polygon on the face
 *     centered about the intersection point to indicate where the
 *     Sun-line intersected the face.
 *-------------------------------------------------------------------
```

```
 */
    { v0[0] = sun[0];
      v1[0] = sun[0];
      v2[0] = sun[0];
      v3[0] = sun[0];

      v0[1] = sun[1] - 0.02;
      v1[1] = sun[1] - 0.02;
      v2[1] = sun[1] + 0.02;
      v3[1] = sun[1] + 0.02;

      v0[2] = sun[2] - 0.02;
      v1[2] = sun[2] + 0.02;
      v2[2] = sun[2] + 0.02;
      v3[2] = sun[2] - 0.02;

/*-----------------------------------------------------------------------
 * --- Draw the polygon on the face.
 *-----------------------------------------------------------------------
 */
      bgnline();
        v3f(v0);
        v3f(v1);
        v3f(v2);
        v3f(v3);
        v3f(v0);
      endline();
    }
  }
}    /* end right_face() */

/***********************************************************************
 *                          VOID FRONT_FACE
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  front_face
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  plot_regions
 *=====================================================================
 *<Description>
 *    Outlines a 3-D BLIRB aerosol region front face and indicates
 *    where the line from the Sun to the ground intersects the face.
 *=====================================================================
 *<Called routines>
 *    None
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void front_face( float regn_vertex[8][3])
 *    Input:
 *       regn_vertex        - vertices of the aerosol region
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
```

```
void front_face(float regn_vertex[8][3])
{
  float v0[3], v1[3], v2[3], v3[3], dum, sun[3];

/*-----------------------------------------------------------------------
 * --- Outline the region face
 *-----------------------------------------------------------------------
 */
  bgnline();
    v3f(regn_vertex[0]);
    v3f(regn_vertex[1]);
    v3f(regn_vertex[5]);
    v3f(regn_vertex[4]);
    v3f(regn_vertex[0]);
  endline();

  if(sun_plot)
/*-----------------------------------------------------------------------
 * --- If the Sun is displayed, determine the location where the line
 *     from the Sun to the ground would intersect the face (if it does
 *     intersect the face).
 *-----------------------------------------------------------------------
 */
  { sun[1] = regn_vertex[0][1];
    dum = (sun[1] - sun_earth[1]) / (sun_sun[1] - sun_earth[1]);
    sun[0] = dum * (sun_sun[0] - sun_earth[0]) + sun_earth[0];
    sun[2] = dum * (sun_sun[2] - sun_earth[2]) + sun_earth[2];

    if((sun[0] >= regn_vertex[0][0] &&
        sun[0] <= regn_vertex[5][0]) &&
       (sun[2] >= regn_vertex[0][2] &&
        sun[2] <= regn_vertex[5][2]))
/*-----------------------------------------------------------------------
 * --- If the line from the Sun to the ground intersects the face,
 *     determine the vertices of a small square polygon on the face
 *     centered about the intersection point to indicate where the
 *     Sun-line intersected the face.
 *-----------------------------------------------------------------------
 */
    { v0[0] = sun[0] - 0.02;
      v1[0] = sun[0] + 0.02;
      v2[0] = sun[0] + 0.02;
      v3[0] = sun[0] - 0.02;

      v0[1] = sun[1];
      v1[1] = sun[1];
      v2[1] = sun[1];
      v3[1] = sun[1];

      v0[2] = sun[2] - 0.02;
      v1[2] = sun[2] - 0.02;
      v2[2] = sun[2] + 0.02;
      v3[2] = sun[2] + 0.02;

/*-----------------------------------------------------------------------
 * --- Draw the polygon on the face.
 *-----------------------------------------------------------------------
 */
      bgnline();
        v3f(v0);
        v3f(v1);
        v3f(v2);
```

```
        v3f(v3);
        v3f(v0);
      endline();
    }
  }
}    /* end front_face() */

/*********************************************************************
 *                          VOID BACK_FACE
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  back_face
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  plot_regions
 *====================================================================
 *<Description>
 *    Outlines a 3-D BLIRB aerosol region back face and indicates
 *    where the line from the Sun to the ground intersects the face.
 *====================================================================
 *<Called routines>
 *    None
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void back_face( float regn_vertex[8][3])
 *    Input:
 *       regn_vertex        - vertices of the aerosol region
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void back_face(float regn_vertex[8][3])
{
  float v0[3], v1[3], v2[3], v3[3], dum, sun[3];

/*-------------------------------------------------------------------
 * --- Outline the region face
 *-------------------------------------------------------------------
 */
  bgnline();
    v3f(regn_vertex[2]);
    v3f(regn_vertex[3]);
    v3f(regn_vertex[7]);
    v3f(regn_vertex[6]);
    v3f(regn_vertex[2]);
  endline();

  if(sun_plot)
/*-------------------------------------------------------------------
 * --- If the Sun is displayed, determine the location where the line
 *     from the Sun to the ground would intersect the face (if it does
 *     intersect the face).
 *-------------------------------------------------------------------
 */
  { sun[1] = regn_vertex[2][1];
```

```
    dum = (sun[1] - sun_earth[1]) / (sun_sun[1] - sun_earth[1]);
    sun[0] = dum * (sun_sun[0] - sun_earth[0]) + sun_earth[0];
    sun[2] = dum * (sun_sun[2] - sun_earth[2]) + sun_earth[2];

    if((sun[0] >= regn_vertex[7][0] &&
        sun[0] <= regn_vertex[2][0]) &&
       (sun[2] >= regn_vertex[2][2] &&
        sun[2] <= regn_vertex[7][2]))
/*-----------------------------------------------------------------------
 * --- If the line from the Sun to the ground intersects the face,
 *     determine the vertices of a small square polygon on the face
 *     centered about the intersection point to indicate where the
 *     Sun-line intersected the face.
 *-----------------------------------------------------------------------
 */
    { v0[0] = sun[0] - 0.02;
      v1[0] = sun[0] + 0.02;
      v2[0] = sun[0] + 0.02;
      v3[0] = sun[0] - 0.02;

      v0[1] = sun[1];
      v1[1] = sun[1];
      v2[1] = sun[1];
      v3[1] = sun[1];

      v0[2] = sun[2] - 0.02;
      v1[2] = sun[2] - 0.02;
      v2[2] = sun[2] + 0.02;
      v3[2] = sun[2] + 0.02;

/*-----------------------------------------------------------------------
 * --- Draw the polygon on the face.
 *-----------------------------------------------------------------------
 */
      bgnline();
        v3f(v0);
        v3f(v1);
        v3f(v2);
        v3f(v3);
        v3f(v0);
      endline();
    }
  }
}    /* end back_face() */

/***********************************************************************
 *                          VOID TOP_FACE
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  top_face
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  plot_regions
 *=====================================================================
 *<Description>
 *    Outlines a 3-D BLIRB aerosol region top face and indicates
 *    where the line from the Sun to the ground intersects the face.
 *=====================================================================
 *<Called routines>
 *    None
 *=====================================================================
 *<Parameters>
```

```
 *    Formal declaration:
 *       void top_face( float regn_vertex[8][3])
 *    Input:
 *       regn_vertex        - vertices of the aerosol region
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void top_face(float regn_vertex[8][3])
{
  float v0[3], v1[3], v2[3], v3[3], dum, sun[3];

/*---------------------------------------------------------------------
 * --- Outline the region face
 *---------------------------------------------------------------------
 */
  bgnline();
    v3f(regn_vertex[4]);
    v3f(regn_vertex[5]);
    v3f(regn_vertex[6]);
    v3f(regn_vertex[7]);
    v3f(regn_vertex[4]);
  endline();

  if(sun_plot)
/*---------------------------------------------------------------------
 * --- If the Sun is displayed, determine the location where the line
 *     from the Sun to the ground would intersect the face (if it does
 *     intersect the face).
 *---------------------------------------------------------------------
 */
  { sun[2] = regn_vertex[4][2];
    dum = (sun[2] - sun_earth[2]) / (sun_sun[2] - sun_earth[2]);
    sun[0] = dum * (sun_sun[0] - sun_earth[0]) + sun_earth[0];
    sun[1] = dum * (sun_sun[1] - sun_earth[1]) + sun_earth[1];

    if((sun[0] >= regn_vertex[4][0] &&
        sun[0] <= regn_vertex[6][0]) &&
       (sun[1] >= regn_vertex[4][1] &&
        sun[1] <= regn_vertex[6][1]))
/*---------------------------------------------------------------------
 * --- If the line from the Sun to the ground intersects the face,
 *     determine the vertices of a small square polygon on the face
 *     centered about the intersection point to indicate where the
 *     Sun-line intersected the face.
 *---------------------------------------------------------------------
 */
    { v0[0] = sun[0] - 0.02;
      v1[0] = sun[0] + 0.02;
      v2[0] = sun[0] + 0.02;
      v3[0] = sun[0] - 0.02;

      v0[1] = sun[1] - 0.02;
      v1[1] = sun[1] - 0.02;
      v2[1] = sun[1] + 0.02;
      v3[1] = sun[1] + 0.02;
```

```
      v0[2] = sun[2];
      v1[2] = sun[2];
      v2[2] = sun[2];
      v3[2] = sun[2];

/*-----------------------------------------------------------------------
 * --- Draw the polygon on the face.
 *-----------------------------------------------------------------------
 */
      bgnline();
        v3f(v0);
        v3f(v1);
        v3f(v2);
        v3f(v3);
        v3f(v0);
      endline();
    }
  }
}    /* end top_face() */

/***********************************************************************
 *                          VOID PLOT_FLARS
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  plot_flars
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  drawscene
 *=====================================================================
 *<Description>
 *    Plots the BLIRB Flares.
 *=====================================================================
 *<Called routines>
 *    None
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void plot_flars(void)
 *    Input:
 *       None
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void plot_flars(void)
{
  static long col_red[] = { 255, 0, 0};
  int i, k;
  float flar_vertex[8][3], num[3];
  char text[11];

  for (i=0; i <= flar; i++)
/*-----------------------------------------------------------------------
 * --- Determine the vertices of each Flare.
 *-----------------------------------------------------------------------
 */
```

```
  { flar_vertex[0][0] = flar_vertex[3][0] = flar_vertex[4][0] =
                        flar_vertex[7][0] = flar_xflar[i] - 0.05;

    flar_vertex[1][0] = flar_vertex[2][0] = flar_vertex[5][0] =
                        flar_vertex[6][0] = flar_xflar[i] + 0.05;

    flar_vertex[0][1] = flar_vertex[1][1] = flar_vertex[4][1] =
                        flar_vertex[5][1] = flar_yflar[i] - 0.05;

    flar_vertex[2][1] = flar_vertex[3][1] = flar_vertex[7][1] =
                        flar_vertex[6][1] = flar_yflar[i] + 0.05;

    flar_vertex[0][2] = flar_vertex[1][2] = flar_vertex[2][2] =
                        flar_vertex[3][2] = flar_zflar[i] - 0.05;

    flar_vertex[4][2] = flar_vertex[5][2] = flar_vertex[6][2] =
                        flar_vertex[7][2] = flar_zflar[i] + 0.05;

/*-----------------------------------------------------------------------
 * --- Draw the Flare.
 *-----------------------------------------------------------------------
 */
    c3i(col_red);

    bgnline();
      v3f(flar_vertex[0]);
      v3f(flar_vertex[6]);
    endline();

    bgnline();
      v3f(flar_vertex[1]);
      v3f(flar_vertex[7]);
    endline();

    bgnline();
      v3f(flar_vertex[2]);
      v3f(flar_vertex[4]);
    endline();

    bgnline();
      v3f(flar_vertex[3]);
      v3f(flar_vertex[5]);
    endline();

/*-----------------------------------------------------------------------
 * --- Determine the location where the Flare number should be
 *     displayed.
 *-----------------------------------------------------------------------
 */
    if(label_obsc)
    { for (k=0; k<3; k++)
        num[k] = 0.5 * (flar_vertex[0][k] + flar_vertex[6][k]) + 0.10;
      sprintf(text, "%d", i+1);            /* Compose the text          */
      cmov( num[0], num[1], num[2]);       /* Move to text location     */
      charstr(text);                       /* Display the text          */
    }

/*-----------------------------------------------------------------------
 * --- If the Flare is to be moved, determine the location where the
 *     text reminder message should be displayed and display it.
 *-----------------------------------------------------------------------
 */
```

```
    if(i == cur_flar && (move_flarh || move_flarv))
    { num[0] = flar_vertex[0][0];
      num[1] = flar_vertex[0][1] - 0.15;
      num[2] = flar_vertex[0][2] - 0.15;
      strcpy(text, "Move Flare");          /* Compose the text          */
      cmov( num[0], num[1], num[2]);       /* Move to text location     */
      charstr(text);                       /* Display the text          */
    }
  }
}    /* end plot_flars() */

/*********************************************************************
 *                          VOID PLOT_SLITE
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  plot_slite
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  drawscene
 *====================================================================
 *<Description>
 *    Plots the BLIRB SerachLights.
 *====================================================================
 *<Called routines>
 *    None
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void plot_slite(void)
 *    Input:
 *       None
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void plot_slite(void)
{
  static long col_white[] = { 255, 255, 255};
  static long col_red[] = { 255, 0, 0};
  int i, k;
  float flar_vertex[8][3], num[3];
  char text[11];

  for (i=0; i <= srch; i++)
/*-------------------------------------------------------------------
 * --- Determine the vertices of each SearchLight.
 *-------------------------------------------------------------------
 */
  { flar_vertex[0][0] = flar_vertex[3][0] = flar_vertex[4][0] =
                        flar_vertex[7][0] = srch_xsrch - 0.05;

    flar_vertex[1][0] = flar_vertex[2][0] = flar_vertex[5][0] =
                        flar_vertex[6][0] = srch_xsrch + 0.05;

    flar_vertex[0][1] = flar_vertex[1][1] = flar_vertex[4][1] =
                        flar_vertex[5][1] = srch_ysrch - 0.05;
```

```
    flar_vertex[2][1] = flar_vertex[3][1] = flar_vertex[7][1] =
                        flar_vertex[6][1] = srch_ysrch + 0.05;

    flar_vertex[0][2] = flar_vertex[1][2] = flar_vertex[2][2] =
                        flar_vertex[3][2] = srch_zsrch - 0.05;

    flar_vertex[4][2] = flar_vertex[5][2] = flar_vertex[6][2] =
                        flar_vertex[7][2] = srch_zsrch + 0.05;

/*-----------------------------------------------------------------------
 * --- Draw the SearchLight.
 *-----------------------------------------------------------------------
 */
    c3i(col_white);

    bgnline();
      v3f(flar_vertex[0]);
      v3f(flar_vertex[6]);
    endline();

    bgnline();
      v3f(flar_vertex[1]);
      v3f(flar_vertex[7]);
    endline();

    bgnline();
      v3f(flar_vertex[2]);
      v3f(flar_vertex[4]);
    endline();

    bgnline();
      v3f(flar_vertex[3]);
      v3f(flar_vertex[5]);
    endline();

/*-----------------------------------------------------------------------
 * --- Determine the location where the SearchLight "S" should be
 *     displayed.
 *-----------------------------------------------------------------------
 */
    if(label_obsc)
    { for (k=0; k<3; k++)
        num[k] = 0.5 * (flar_vertex[0][k] + flar_vertex[6][k]) + 0.10;
      strcpy(text, "S");                  /* Compose the text          */
      cmov( num[0], num[1], num[2]);      /* Move to text location     */
      charstr(text);                      /* Display the text          */
    }

/*-----------------------------------------------------------------------
 * --- If the SearchLight is to be moved, determine the location where
 *     the text reminder message should be displayed and display it.
 *-----------------------------------------------------------------------
 */
    if(move_srchh || move_srchv)
    { c3i(col_red);
      num[0] = flar_vertex[0][0];
      num[1] = flar_vertex[0][1] - 0.15;
      num[2] = flar_vertex[0][2] - 0.15;
      strcpy(text, "Move Slite");         /* Compose the text          */
      cmov( num[0], num[1], num[2]);      /* Move to text location     */
      charstr(text);                      /* Display the text          */
    }
```

```
  }
}    /* end plot_slite() */

/*********************************************************************
 *                          VOID PLOT_AXES
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  plot_axes
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  drawscene
 *====================================================================
 *<Description>
 *    Draws the main BLIRB region axes.
 *====================================================================
 *<Called routines>
 *    plot_xaxis            - draws the X-axis of the main BLIRB region
 *    plot_yaxis            - draws the Y-axis of the main BLIRB region
 *    plot_zaxis            - draws the Z-axis of the main BLIRB region
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void plot_axes( void)
 *    Input:
 *       None
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void plot_axes(void)
{
  plot_xaxis();                          /* Draw the X axis          */
  plot_yaxis();                          /* Draw the Y axis          */
  plot_zaxis();                          /* Draw the Z axis          */
}   /*  end plot_axes()  */

/*********************************************************************
 *                          VOID PLOT_XAXIS
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  plot_xaxis
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  plot_axes
 *====================================================================
 *<Description>
 *    Draws the X-axis of the main BLIRB region along with drawing
 *    the tick marks, grid lines, and labels.
 *====================================================================
 *<Called routines>
 *    None
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void plot_xaxis( void)
 *    Input:
```

```
 *       None
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ********************************************************************
 */
void plot_xaxis(void)
{
  static long whitecol[] = { 255, 255, 255 };
  static long yellowcol[] = { 255, 255, 0 };

  int j;
  float v0[3], v1[3], v2[3];
  float maxx, offset;
  char text[3];

  maxx = 0;

/*-------------------------------------------------------------------
 * --- Determine the (Y,Z) end points of the grid lines on the ground.
 *-------------------------------------------------------------------
 */
  if(move_regnv && !move_regnh)
  { v0[1] = axis_pts[1][0] - 0.02;
    v1[1] = v0[1];
    v2[1] = axis_pts[1][0] - 0.07071;

    v0[2] = axis_pts[2][num_grid_pts[2]];
    v1[2] = axis_pts[2][0];
    v2[2] = v1[2] - 0.07071;
  }
  else
  { v0[1] = axis_pts[1][num_grid_pts[1]];
    v1[1] = axis_pts[1][0];
    v2[1] = v1[1] - 0.07071;

    v0[2] = axis_pts[2][0] + 0.02;
    v1[2] = v0[2];
    v2[2] = axis_pts[2][0] - 0.07071;
  }

/*-------------------------------------------------------------------
 * --- If minor grid lines are requested, set the color to white,
 *     determine the X position of the grid lines, and draw the lines.
 *-------------------------------------------------------------------
 */
  if(minor_grid)
  { c3i(whitecol);                          /* Set the Axis Color       */

    for (j=0; j<=num_grid_pts[0]; j++)
    { v0[0] = axis_pts[0][j];               /* Determine X position     */
      v1[0] = v0[0];
      v2[0] = v0[0];

      bgnline();                            /* Draw X-Axis + Minor Tic  */
        v3f(v0);
        v3f(v1);
```

```
        v3f(v2);
      endline();

      if(maxx < fabs(v0[0]))                /* Determine greatest X val */
        maxx = fabs(v0[0]);
    }
  }

/*-----------------------------------------------------------------------
 * --- For major grid lines, set the color to yellow, determine the X
 *     position of the grid lines, draw the lines, and label them.
 *-----------------------------------------------------------------------
 */
  v2[1] = v1[1] - 0.14142;
  v2[2] = axis_pts[2][0] - 0.14142;

  c3i(yellowcol);                           /* Set the Axis Color       */

  for (j=0; j<=num_grid_main_pts[0]; j++)
  { v0[0] = axis_main_pts[0][j];            /* Determine X position     */
    v1[0] = v0[0];
    v2[0] = v0[0];

    bgnline();                              /* Draw X-Axis + Major Tic  */
      v3f(v0);
      v3f(v1);
      v3f(v2);
    endline();

    sprintf(text, "%d", (int) axis_main_pts[0][j]); /* Compose Label  */
    cmov( v0[0] - 0.05, v1[1] - 0.4, v2[2] - 0.2);  /* Get in position*/
    charstr(text);                          /* Display the label        */

    if(maxx < fabs(v0[0]))                  /* Determine greatest X val */
      maxx = fabs(v0[0]);
  }

/*-----------------------------------------------------------------------
 * --- Determine the end points of the X-axis line.
 *-----------------------------------------------------------------------
 */
  v0[0] = axis_pts[0][0];
  v0[1] = axis_pts[1][0];

  v1[0] = maxx;
  v1[1] = v0[1];

/*-----------------------------------------------------------------------
 * --- Draw the X-axis line.
 *-----------------------------------------------------------------------
 */
  bgnline();
    v3f(v0);
    v3f(v1);
  endline();

/*-----------------------------------------------------------------------
 * --- Move into position to display the axis label string and display
 *     it.
 *-----------------------------------------------------------------------
 */
  if(view_axis[0])
```

```
    offset = -1.1;
  else
    offset = -0.7;

  cmov( v0[0] + 0.45*(maxx-v0[0]), v0[1] + offset, v2[2] - 0.5);
  charstr("X (km)");
}   /*  end plot_xaxis()  */

/*********************************************************************
 *                          VOID PLOT_YAXIS
 *********************************************************************
 *<Begin>
 *<Identification>          Name:  plot_yaxis
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  plot_axes
 *====================================================================
 *<Description>
 *    Draws the Y-axis of the main BLIRB region along with drawing
 *    the tick marks, grid lines, and labels.
 *====================================================================
 *<Called routines>
 *    None
 *====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void plot_yaxis( void)
 *    Input:
 *       None
 *    Output:
 *       None
 *====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *====================================================================
 *<End>
 *********************************************************************
 */
void plot_yaxis(void)
{
  static long whitecol[] = { 255, 255, 255 };
  static long yellowcol[] = { 255, 255, 0 };

  int j;
  float v0[3], v1[3], v2[3];
  float maxy;
  char text[3];

  maxy = 0;

/*-------------------------------------------------------------------
 * --- Determine the (X,Z) end points of the grid lines on the ground.
 *-------------------------------------------------------------------
 */
  v0[0] = axis_pts[0][num_grid_pts[0]];
  v1[0] = axis_pts[0][0];
  v2[0] = v1[0] - 0.07071;

  v0[2] = axis_pts[2][0] + 0.02;
  v1[2] = v0[2];
  v2[2] = axis_pts[2][0] - 0.07071;
```

```
/*-----------------------------------------------------------------------
 * --- If minor grid lines are requested, set the color to white,
 *     determine the Y position of the grid lines, and draw the lines.
 *-----------------------------------------------------------------------
 */
  if(minor_grid)
  { c3i(whitecol);                          /* Set the Axis Color       */

    for (j=0; j<=num_grid_pts[1]; j++)
    { v0[1] = axis_pts[1][j];               /* Determine Y position     */
      v1[1] = v0[1];
      v2[1] = v0[1];

      bgnline();                            /* Draw Y-Axis + Minor Tic  */
        v3f(v0);
        v3f(v1);
        v3f(v2);
      endline();

      if(maxy < v0[1])                      /* Determine greatest Y val */
        maxy = v0[1];
    }
  }

/*-----------------------------------------------------------------------
 * --- For major grid lines, set the color to yellow, determine the Y
 *     position of the grid lines, draw the lines, and label them.
 *-----------------------------------------------------------------------
 */
  v2[0] = v1[0] - 0.14142;
  v2[2] = axis_pts[2][0] - 0.14142;

  c3i(yellowcol);                           /* Set the Axis Color       */

  for (j=0; j<=num_grid_main_pts[1]; j++)
  { v0[1] = axis_main_pts[1][j];            /* Determine Y position     */
    v1[1] = v0[1];
    v2[1] = v0[1];

    bgnline();                              /* Draw Y-Axis + Major Tic  */
      v3f(v0);
      v3f(v1);
      v3f(v2);
    endline();

    sprintf(text, "%d", (int) axis_main_pts[1][j]); /* Compose label  */
    cmov( v1[0] - 0.27, v0[1] - 0.0625, v2[2] - 0.2); /* Get in positn*/
    charstr(text);                          /* Display the label        */

    if(maxy < v0[1])                        /* Determine greatest Y val */
      maxy = v0[1];
  }

/*-----------------------------------------------------------------------
 * --- Determine the end points of the Y-axis line.
 *-----------------------------------------------------------------------
 */
  v0[0] = axis_pts[0][0];
  v0[1] = axis_pts[1][0];

  v1[0] = v0[0];
  v1[1] = maxy;
```

```
/*-----------------------------------------------------------------------
 * --- Draw the Y-axis line.
 *-----------------------------------------------------------------------
 */
  bgnline();
    v3f(v0);
    v3f(v1);
  endline();

/*-----------------------------------------------------------------------
 * --- Move into position to display the axis label string and display
 *     it.
 *-----------------------------------------------------------------------
 */
  cmov(v0[0] - 1.2, v0[1]+0.5*(maxy-v0[1]) - 0.1, v2[2]-0.5);
  charstr("Y (km)");
}   /*  end plot_yaxis()  */

/***********************************************************************
 *                            VOID PLOT_ZAXIS
 ***********************************************************************
 *<Begin>
 *<Identification>          Name:  plot_zaxis
 *                          Type:  C void
 *                      Filename:  visual.c
 *                        Parent:  plot_axes
 *=====================================================================
 *<Description>
 *    Draws the Z-axis of the main BLIRB region along with drawing
 *    the tick marks, grid lines, and labels.
 *=====================================================================
 *<Called routines>
 *    None
 *=====================================================================
 *<Parameters>
 *    Formal declaration:
 *       void plot_zaxis( void)
 *    Input:
 *       None
 *    Output:
 *       None
 *=====================================================================
 *<History>
 *    09/12/94  AMSRL-BE-S   (505) 678-1570   Elton P. Avara
 *              Developed the original source code.
 *=====================================================================
 *<End>
 ***********************************************************************
 */
void plot_zaxis(void)
{
  static long whitecol[] = { 255, 255, 255 };
  static long yellowcol[] = { 255, 255, 0 };

  int j;
  float v0[3], v1[3], v2[3];
  float maxz, offset;
  char text[3];

  maxz = 0;

/*-----------------------------------------------------------------------
```

```
 * --- Determine the (X,Y) end points of the grid lines on the ground.
 *-----------------------------------------------------------------------
 */
  if(move_regnv && !move_regnh)
  { v0[0] = axis_pts[0][num_grid_pts[0]];
    v1[0] = axis_pts[0][0];
    v2[0] = v1[0] -0.07071;

    v0[1] = axis_pts[1][0] - 0.02;
    v1[1] = v0[1];
    v2[1] = axis_pts[1][0] - 0.07071;
  }
  else
  { v0[0] = axis_pts[0][0] - 0.002;
    v1[0] = axis_pts[0][0] - 0.07071;
    v2[0] = v1[0];

    v0[1] = axis_pts[1][0] - 0.002;
    v1[1] = axis_pts[1][0] - 0.07071;
    v2[1] = v1[1];
  }

/*-----------------------------------------------------------------------
 * --- If minor grid lines are requested, set the color to white,
 *     determine the Z position of the grid lines, and draw the lines.
 *-----------------------------------------------------------------------
 */
  if(minor_grid)
  { c3i(whitecol);                          /* Set the Axis Color       */

    for (j=0; j<=num_grid_pts[2]; j++)
    { v0[2] = axis_pts[2][j];               /* Determine Z position     */
      v1[2] = v0[2];
      v2[2] = v0[2];

      bgnline();                            /* Draw Z-Axis + Minor Tic  */
        v3f(v0);
        v3f(v1);
        v3f(v2);
      endline();

      if(maxz < v0[2])                      /* Determine greatest Z val */
        maxz = v0[2];
    }
  }

/*-----------------------------------------------------------------------
 * --- For major grid lines, set the color to yellow, determine the Z
 *     position of the grid lines, draw the lines, and label them.
 *-----------------------------------------------------------------------
 */
  v1[0] = axis_pts[0][0] - 0.14142;
  v1[1] = axis_pts[1][0] - 0.14142;

  c3i(yellowcol);                           /* Set the Axis Color       */

  for (j=0; j<=num_grid_main_pts[2]; j++)
  { v0[2] = axis_main_pts[2][j];            /* Determine Z position     */
    v1[2] = v0[2];
    v2[2] = v0[2];

    bgnline();                              /* Draw Z-Axis + Major Tic  */
```

```
      v3f(v0);
      v3f(v1);
      v3f(v2);
    endline();

    sprintf(text, "%d", (int) axis_main_pts[2][j]); /* Compose label  */
    cmov( axis_pts[0][0] - 0.3, axis_pts[1][0] - 0.3, v0[2] - 0.05);
    charstr(text);                          /* Display the label        */

    if(maxz < v0[2])                        /* Determine greatest Z val */
      maxz = v0[2];
  }

/*-----------------------------------------------------------------------
 * --- Determine the end points of the Z-axis line.
 *-----------------------------------------------------------------------
 */
  v0[2] = axis_pts[2][0];

  v1[0] = v0[0];
  v1[1] = v0[1];
  v1[2] = maxz;

/*-----------------------------------------------------------------------
 * --- Draw the Z-axis line.
 *-----------------------------------------------------------------------
 */
  bgnline();
    v3f(v0);
    v3f(v1);
  endline();

/*-----------------------------------------------------------------------
 * --- Move into position to display the axis label string and display
 *     it.
 *-----------------------------------------------------------------------
 */
  if(view_axis[0])
    offset = -1.1;
  else
    offset = -0.7;

  cmov( v1[0] - 1.2, v1[1] + offset, v0[2]+0.5*(maxz-v0[2]));
  charstr("Z (km)");
}   /*  end plot_zaxis()  */
```

Distribution

	Copies
ARMY CHEMICAL SCHOOL ATZN CM CC ATTN MR BARNES FT MCCLELLAN AL 36205-5020	1
NASA MARSHAL SPACE FLT CTR ATMOSPHERIC SCIENCES DIV E501 ATTN DR FICHTL HUNTSVILLE AL 35802	1
NASA SPACE FLT CTR ATMOSPHERIC SCIENCES DIV CODE ED 41 1 HUNTSVILLE AL 35812	1
ARMY STRAT DEFNS CMND CSSD SL L ATTN DR LILLY PO BOX 1500 HUNTSVILLE AL 35807-3801	1
ARMY MISSILE CMND AMSMI RD AC AD ATTN DR PETERSON REDSTONE ARSENAL AL 35898-5242	1
ARMY MISSILE CMND AMSMI RD AS SS ATTN MR H F ANDERSON REDSTONE ARSENAL AL 35898-5253	1
ARMY MISSILE CMND AMSMI RD AS SS ATTN MR B WILLIAMS REDSTONE ARSENAL AL 35898-5253	1

ARMY MISSILE CMND 1
AMSMI RD DE SE
ATTN MR GORDON LILL JR
REDSTONE ARSENAL
AL 35898-5245

ARMY MISSILE CMND 1
REDSTONE SCI INFO CTR
AMSMI RD CS R DOC
REDSTONE ARSENAL
AL 35898-5241

ARMY MISSILE CMND 1
AMSMI
REDSTONE ARSENAL
AL 35898-5253

ARMY INTEL CTR 1
AND FT HUACHUCA
ATSI CDC C
FT HUACHUCA AZ 85613-7000

NORTHROP CORPORATION 1
ELECTR SYST DIV
ATTN MRS T BROHAUGH
2301 W 120TH ST BOX 5032
HAWTHORNE CA 90251-5032

NAVAL WEAPONS CTR 1
CODE 3331
ATTN DR SHLANTA
CHINA LAKE CA 93555

PACIFIC MISSILE TEST CTR 1
GEOPHYSICS DIV
ATTN CODE 3250
POINT MUGU CA 93042-5000

LOCKHEED MIS & SPACE CO 1
ATTN KENNETH R HARDY
ORG 91 01 B 255
3251 HANOVER STREET
PALO ALTO CA 94304-1191

NAVAL OCEAN SYST CTR 1
CODE 54
ATTN DR RICHTER
SAN DIEGO CA 92152-5000

METEOROLOGIST IN CHARGE 1
KWAJALEIN MISSILE RANGE
PO BOX 67
APO SAN FRANCISCO
CA 96555

DEPT OF COMMERCE CTR 1
MOUNTAIN ADMINISTRATION
SPPRT CTR LIBRARY R 51
325 S BROADWAY
BOULDER CO 80303

DR HANS J LIEBE 1
NTIA ITS S 3
325 S BROADWAY
BOULDER CO 80303

NCAR LIBRARY SERIALS 1
NATL CTR FOR ATMOS RSCH
PO BOX 3000
BOULDER CO 80307-3000

DEPT OF COMMERCE CTR 1
325 S BROADWAY
BOULDER CO 80303

DAMI POI 1
WASH DC 20310-1067

MIL ASST FOR ENV SCI OFC 1
OF THE UNDERSEC OF DEFNS
FOR RSCH & ENGR R&AT E LS
PENTAGON ROOM 3D129
WASH DC 20301-3080

DEAN RMD 1
ATTN DR GOMEZ
WASH DC 20314

SPACE NAVAL WARFARE SYST CMND PMW 145 1G WASH DC 20362-5100	1
ARMY INFANTRY ATSH CD CS OR ATTN DR E DUTOIT FT BENNING GA 30905-5090	1
AIR WEATHER SERVICE TECH LIBRARY FL4414 3 SCOTT AFB IL 62225-5458	1
USAFETAC DNE ATTN MR GLAUBER SCOTT AFB IL 62225-5008	1
HQ AWS DOO 1 SCOTT AFB IL 62225-5008	1
ARMY SPACE INSTITUTE ATTN ATZI SI 3 FT LEAVENWORTH KS 66027-5300	1
PHILLIPS LABORATORY PL LYP ATTN MR CHISHOLM HANSCOM AFB MA 01731-5000	1
ATMOSPHERIC SCI DIV GEOPHYSICS DIRCTRT PHILLIPS LABORATORY HANSCOM AFB MA 01731-5000	1
PHILLIPS LABORATORY PL LYP 3 HANSCOM AFB MA 01731-5000	1
RAYTHEON COMPANY ATTN DR SONNENSCHEIN 528 BOSTON POST ROAD SUDBURY MA 01776 MAIL STOP 1K9	1

ARMY MATERIEL SYST 1
ANALYSIS ACTIVITY
AMXSY
ATTN MP H COHEN
APG MD 21005-5071

ARMY MATERIEL SYST 1
ANALYSIS ACTIVITY
AMXSY AT
ATTN MR CAMPBELL
APG MD 21005-5071

ARMY MATERIEL SYST 1
ANALYSIS ACTIVITY
AMXSY CR
ATTN MR MARCHET
APG MD 21005-5071

ARL CHEMICAL BIOLOGY 1
NUC EFFECTS DIV
AMSRL SL CO
APG MD 21010-5423

ARMY MATERIEL SYST 1
ANALYSIS ACTIVITY
AMXSY
APG MD 21005-5071

NAVAL RESEARCH LABORATORY 1
CODE 4110
ATTN MR RUHNKE
WASH DC 20375-5000

ARMY MATERIEL SYST 1
ANALYSIS ACTIVITY
AMXSY CS
ATTN MR BRADLEY
APG MD 21005-5071

ARMY RESEARCH LABORATORY 1
AMSRL D
2800 POWDER MILL ROAD
ADELPHI MD 20783-1145

ARMY RESEARCH LABORATORY 1
AMSRL OP SD TP
TECHNICAL PUBLISHING
2800 POWDER MILL ROAD
ADELPHI MD 20783-1145

ARMY RESEARCH LABORATORY 1
AMSRL OP CI SD TL
2800 POWDER MILL ROAD
ADELPHI MD 20783-1145

ARMY RESEARCH LABORATORY 1
AMSRL SS SH
ATTN DR SZTANKAY
2800 POWDER MILL ROAD
ADELPHI MD 20783-1145

ARMY RESEARCH LABORATORY 1
AMSRL
2800 POWDER MILL ROAD
ADELPHI MD 20783-1145

NATIONAL SECURITY AGCY W21 1
ATTN DR LONGBOTHUM
9800 SAVAGE ROAD
FT GEORGE G MEADE
MD 20755-6000

ARMY AVIATION CTR 1
ATZQ D MA
ATTN MR HEATH
FT RUCKER AL 36362

OIC NAVSWC 1
TECH LIBRARY CODE E 232
SILVER SPRINGS
MD 20903-5000

ARMY RSRC OFC 1
ATTN DRXRO GS
PO BOX 12211
RTP NC 27009

DR JERRY DAVIS 1
NCSU
PO BOX 8208
RALEIGH NC 27650-8208

ARMY CCREL CECRL GP ATTN DR DETSCH HANOVER NH 03755-1290	1
ARMY ARDEC SMCAR IMI I BLDG 59 DOVER NJ 07806-5000	1
ARMY SATELLITE COMM AGCY DRCPM SC 3 FT MONMOUTH NJ 07703-5303	1
ARMY COMMUNICATIONS ELECTR CTR FOR EW RSTA AMSEL EW D FT MONMOUTH NJ 07703-5303	1
ARMY COMMUNICATIONS ELECTR CTR FOR EW RSTA AMSEL EW MD FT MONMOUTH NJ 07703-5303	1
ARMY DUGWAY PROVING GRD STEDP MT DA L 3 DUGWAY UT 84022-5000	1
ARMY DUGWAY PROVING GRD STEDP MT M ATTN MR BOWERS DUGWAY UT 84022-5000	1
DEPT OF THE AIR FORCE OL A 2D WEATHER SQUAD MAC HOLLOMAN AFB NM 88330-5000	1
PL WE KIRTLAND AFB NM 87118-6008	1
USAF ROME LAB TECH CORRIDOR W STE 262 RL SUL 26 ELECTR PKWY BLD 106 GRIFFISS AFB NY 13441-4514	1

AFMC DOW WRIGHT PATTERSON AFB OH 0334-5000	1
ARMY FIELD ARTLLRY SCHOOL ATSF TSM TA FT SILL OK 73503-5600	1
NAVAL AIR DEV CTR CODE 5012 ATTN AL SALIK WARMINISTER PA 18974	1
ARMY FOREGN SCI TECH CTR CM 220 7TH STREET NE CHARLOTTESVILLE VA 22901-5396	1
NAVAL SURFACE WEAPONS CTR CODE G63 DAHLGREN VA 22448-5000	1
ARMY OEC CSTE EFS PARK CENTER IV 4501 FORD AVE ALEXANDRIA VA 22302-1458	1
ARMY CORPS OF ENGRS ENGR TOPOGRAPHICS LAB ETL GS LB FT BELVOIR VA 22060	1
TAC DOWP LANGLEY AFB VA 23665-5524	1
ARMY TOPO ENGR CTR CETEC ZC 1 FT BELVOIR VA 22060-5546	1
LOGISTICS CTR ATCL CE FT LEE VA 23801-6000	1

SCI AND TECHNOLOGY 1
101 RESEARCH DRIVE
HAMPTON VA 23666-1340

ARMY NUCLEAR CML AGCY 1
MONA ZB BLDG 2073
SPRINGFIELD VA 22150-3198

ARMY FIELD ARTLLRY SCHOOL 1
ATSF F FD
FT SILL OK 73503-5600

USATRADOC 1
ATCD FA
FT MONROE VA 23651-5170

ARMY TRADOC ANALYSIS CTR 1
ATRC WSS R
WSMR NM 88002-5502

ARMY RESEARCH LABORATORY 1
AMSRL BE M
BATTLEFIELD ENVIR DIR
WSMR NM 88002-5501

ARMY RESEARCH LABORATORY 1
AMSRL BE A
BATTLEFIELD ENVIR DIR
WSMR NM 88002-5501

ARMY RESEARCH LABORATORY 1
AMSRL BE W
BATTLEFIELD ENVIR DIR
WSMR NM 88002-5501

ARMY RESEARCH LABORATORY 1
AMSRL BE
ATTN MR VEAZEY
BATTLEFIELD ENVIR DIR
WSMR NM 88002-5501

DEFNS TECH INFO CTR 1
CENTER DTIC BLS
BLDG 5 CAMERON STATION
ALEXANDRIA
VA 22304-6145

ARMY MISSILE CMND AMSMI REDSTONE ARSENAL AL 35898-5243	1
ARMY DUGWAY PROVING GRD STEDP 3 DUGWAY UT 84022-5000	1
USATRADOC ATCD FA FT MONROE VA 23651-5170	1
ARMY FIELD ARTLRY SCHOOL ATSF FT SILL OK 73503-5600	1
WSMR TECH LIBRARY BR STEWS IM IT WSMR NM 88001	1
Record Copy	10
Total	96