5 UIDL

5.1 WHAT IS UIDL?

Some users may find that their particular analysis task requires more flexibility than UIMAGE provides. Others may have their own IDL routines they wish to use; still others may have an inherent dislike of mouses and menus. For these users, we have UIDL: standard IDL with additional tools useful for COBE analysis. These tools duplicate (and, in some instances, exceed) the functions found in UIMAGE.

The following sections will show you how to:

* get online help for both standard and COBE-written IDL routines
* perform data input and output
* use some of the routines in common analysis situations

> **Note to more experienced users:** You may wish to skim the next section and then jump directly to the command line sequence examples at the end of the chapter.

(We assume that you know some IDL, although we will try to suggest suitable routines to use. If you're an IDL novice, RSI's *IDL Basics*, the *IDL User's Guide*, and the *IDL Reference Guide* are handy references.)

5.2 GETTING INTO UIDL

To access UIDL from the CGIS software, choose the **Go to Command Line (UIDL)** option from the **CGIS Main Menu**. You will see the following instructions:

```
Type EXIT to return to VMS or
Type CGIS to return to the main menu...

UIDL>
```

You are now at the UIDL command line and ready to start your work.

5.3 GETTING OUT OF UIDL

UIDL allows three safe ways to exit, two of which appear as reminders when you enter the UIDL package. Entering `CGIS` at the **UIDL>** prompt returns you to the **CGIS Main Menu** with all of the data acquired during your UIDL (and UIMAGE) session still intact.

The second way to exit is to type `UIMAGE` at the **UIDL>** prompt. This bypasses the **CGIS Main Menu** and places you directly into UIMAGE.

WARNING! Returning to UIMAGE *deletes* any IDL windows you have created. Be sure to save their contents first (using the **TVRD** routine, *e.g.*) before going back to UIMAGE. Any data you have read into or created in UIDL will also be lost if you do not return to UIDL to save it before leaving the CGIS software.

If you have completely finished, have saved everything you want to save, and wish to exit UIDL, UIMAGE, and CGIS, type `EXIT` at the **UIDL>** prompt. You will be returned to a system prompt.

If for some reason you need to stop an IDL process from running, you can often interrupt the offending procedure by typing `^C`.

WARNING! Typing `^Z` or `^Y` will take you completely out of UIDL *and* the CGIS software, losing ALL unsaved data currently in memory.

5.4 GETTING HELP IN UIDL

UIDL contains several different types of help, depending upon whether the tool in question is a standard IDL routine, comes from the IDL User Library, or was written specifically for COBE. If you are unsure whether a routine is associated with IDL or UIDL, begin your quest for help in UHelp.

5.4.1 UHelp

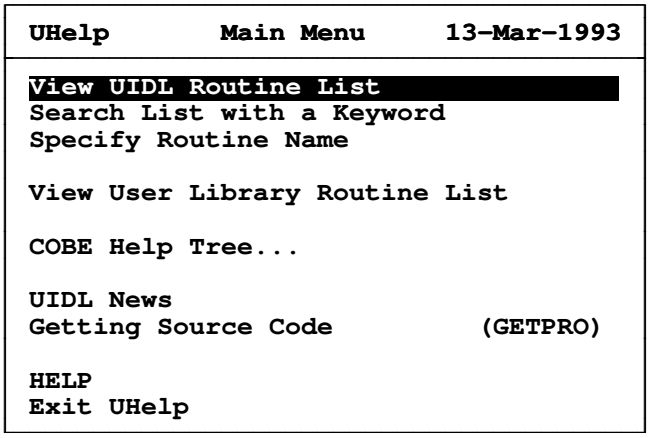
UHelp is a COBE-specific help facility for UIDL. It provides

* instructions for its use
* detailed help listings for each UIDL routine, including calling sequence, warnings, and examples
* a list of the IDL User Library routines
* news about updates to the UIDL software
* a list with brief descriptions of UIDL routines
* instructions for getting copies of the source code for individual UIDL or IDL User Library routines

To invoke UHelp, just type

```
UIDL> UHELP
```

You will see the following menu. Select items using the arrow keys (on a terminal) or by moving the mouse (on a workstation), and invoke your selection by pressing <CR> or <MB1>, respectively:



As in the CGIS Executive and UIMAGE, the ellipsis in a menu option indicates more menus to follow. An ellipsis at the bottom of a menu indicates that there are more choices below.

5.4.1.1 View UIDL Routine List

View UIDL Routine List presents an alphabetized list of UIDL routines along with their one-line descriptions. (You may want to choose this option first to see whether a routine was written by the COBE group.) A version of this list ordered by help categories is provided in Appendix B of this guide. For an up-to-date version of your own, click on the **File Dump** box (workstation users) or press ^F (terminal users).

5.4.1.2 Search List with a Keyword

If you know part of a UIDL routine name, or have a general function in mind, try **Search List with a Keyword**. UHelp prompts for a keyword (text string) and then presents the list of UIDL short descriptions that contain the string. This list is a menu, allowing you to pull up the desired help routine directly. The keyword search may also be invoked directly from the UIDL command line with the command

```
UIDL> UHELP, KEY = 'your_string'
```

For example,

```
UIDL> UHELP, KEY = 'CUBE'
```

displays the following menu:

```
                    KEY SEARCH: Select a Routine

BUILDTEE takes 1-6 cube faces and places them into an unfolded T
DOWNRES "coarsens" the resolution of a skycube or cube face.
GAPFILL interpolates across gaps in an unfolded skycube.
GRABFACE grabs a cube from an unfolded T or sixpack
PIX2DAT extracts a data list from a rasterized skycube by xy coords
PIX2XY creates a raster image (sky cube/face) from a pixel/data list.
REPROJ is a general reprojection facility for skycubes.
SIXPACK packs an unfolded skycube into 3x2 format (no wasted space).
SIXUNPACK "uncompresses" the packed skycube created by SIXPACK.
SMOOTHCUBE convolves an unfolded skycube with an arbitrary kernel.
UPKWHERE unpacks a 'where' vector into cube,level,col,row.

Try another key
Return to MAIN MENU
Exit UHelp
```

5.4.1.3 Specify Routine Name

With **Specify Routine Name**, you are prompted for a specific topic, *i.e.*, the exact routine name. This may also be invoked from the UIDL command line with:

```
UIDL> UHELP, 'routine_name'
```

A sample screen is shown in section 5.4.1.9.

5.4.1.4 View User Library Routine List

The **View User Library Routine List** option gives descriptions of IDL User Library routines, also listed in Appendix F, as well as instructions for displaying detailed help information. See section 5.4.2 for more information about standard IDL help.

5.4.1.5 COBE Help Tree

If you have a particular function in mind, but don't know the routine name, try traversing the **COBE Help Tree...**, where the routine names are displayed by category. (A listing of the help categories is found in Table 5.1.) You will find in using this help tree that some routines have been placed in more than one category.

More information on individual routines may be found in Appendix B, where each routine is listed with a one line description. Check the CGIS **News** (**CGIS Main Menu**) or the **UIDL News** (UHelp) for new developments affecting UIDL routines.

5.4.1.6 UIDL News

Enhancements are made to UIDL, UIMAGE, and CGIS on an on-going basis. To keep abreast of these modifications, choose **UIDL News** from the **UHelp Main Menu**. The news items are dated for easy reference.

5.4.1.7 Getting Source Code (GETPRO)

This option displays the help file for the **GETPRO** routine, which allows you to copy the source code for the specified UIDL or IDL User Library routine into your default directory.

5.4.1.8 HELP

HELP contains an explanation of how to use UHelp and how to get help when viewing text screens from a VT240 terminal. Exiting from the text screen brings up a menu of subtopics describing the **UHelp Main Menu**, the **COBE Help Tree**, and using UHelp from the UIDL command line.

Table 5.1 UIDL HELP TREE CATEGORIES

- COBE Data Input
  - Skymap Data
  - Time Ordered Data
  - Engineer/Hskeep/SpCraft Data
  - PDS/IP Data
- General Data I/O
- Image and Plot*
  - Annotation
  - Printing
  - User-Interactive Programs
- Analysis Tools*
  - Statistics
  - Astronomical Tools
  - Ephemerides
  - DMR Analysis Tools
  - FIRAS Analysis Tools
  - Fitting Tools
- Utilities*
  - Coordinates
  - Time/Ephemerides
  - Skycube
  - String Handling

* Some routines are listed directly under the main category, while others are found under the subcategories.

5.4.1.9 Typical UHelp Text Window

A typical UHelp text window looks like this when displayed on a terminal:

```
                          Routine BUILDTEE

+NAME/ONE LINE DESCRIPTION OF ROUTINE:
    BUILDTEE takes 1-6 cube faces and places them into an unfolded T

DESCRIPTION:
    This function takes up to six square arrays representing skycube
                                    ⋮
                                    ⋮
CALLING SEQUENCE:
    outmap = BUILDTEE (face0, [face1,...,face5], [facenums=facenums], $
                                          [badval=badval], [/sixpack]

ARGUMENTS (I = input, O = output, [] = optional):

   OUTMAP       O   any type     Output unfolded right-T (all six faces)

        Next Scrn=(Space)  Prev Scrn=(B)  Help=(H)  Exit=(Tab)
```

The last line lists the major commands available in UHelp. Arrow keys, <CR>, the space bar, and the letter B move you around in the text. When you reach the top or bottom of the text, the words **TOP!** and **END!**, respectively, appear on the right end of the UHelp status line. The computer also beeps when the beginning or end of the text is reached.

Users of terminals looking for help in moving through UHelp text screens can press H. This brings up the screen shown in Appendix C. To exit a text screen, hit the TAB key. This clears your terminal screen and backs you up one menu level. To return directly to the **UIDL>** prompt while retaining the last screen of help information, type Q.

Workstation users view the help text in a separate window with vertical and horizontal scroll bars. Clicking on the **File Dump** box allows you to capture the text into a file. There is, however, no way to retain the help information on the screen once you return to the UIDL command line. Exit the workstation help window by clicking in the **Done** box.

To send help information to a file instead of your screen, both terminal and workstation users enter

```
UIDL> UHELP, 'routine_name', FILE = 'file_name'
```

for example,

```
UIDL> UHELP, 'REPROJ', FILE = 'REPROJECTION.HELP'
```

An acknowledgement that the file has been written is displayed on your screen.

5.4.2 Standard IDL Help

Although UHelp presents detailed information only for UIDL routines, it does list the User Library routines supplied in standard IDL for fitting, error plotting, *etc*. (A copy of the UHelp User Library Routines listing is found in Appendix F of this guide.) If a routine for which you need help is in the User Library, you can get a comprehensive help listing by typing

```
UIDL> DOC_LIBRARY, 'routine_name'
```

If the routine is neither in the list of UIDL routines nor the IDL User Library, it is probably from standard IDL. In that case, you can get more information by typing

```
UIDL> ?
```

and choosing from the following categories:

* GRAPH_KEYWORDS
* GRAPH_SYSVARS
* PLOT_KEYWORDS
* ROUTINES
* STATLIB
* USERLIB (the User Library routines described above and in Appendix F)
* WIDGETLIB

GRAPH_KEYWORDS, **GRAPH_SYSVARS**, and **PLOT_KEYWORDS** display short descriptions for individual keywords chosen from a list. Choosing one of the other categories causes a list of routines to appear. Select the routine name of interest, and the help information is displayed. This help information contains the name of the routine and its purpose, describes the calling sequence with details about the input and output variables, and in some cases presents examples of usage. Note that although the presentation of IDL help differs greatly between X Windows and non-X Windows terminals, the content is the same.

Exit IDL help by pressing <CR> when prompted for a topic (from a terminal), or clicking on the **Quit** box (on a workstation). Workstation users may retain the help window by clicking in the **Close** box. This shrinks the help window into an icon. It may be reactivated at any time by clicking twice on the icon.

5.5 GETTING DATA INTO AND OUT OF UIDL

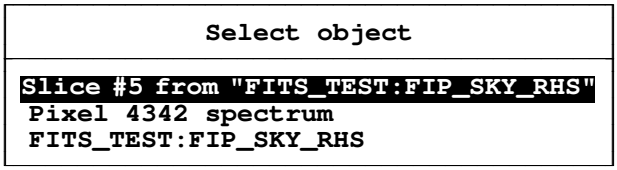
5.5.1 Transporting Data between UIMAGE and UIDL

5.5.1.1 Reading in UIMAGE Objects (UGETDATA)

As also described in chapter 4, data may be moved between UIDL and UIMAGE using the **UGETDATA** and **UPUTDATA** procedures. Data in UIMAGE are accessible in UIDL, but variables must be specified for them. For example, if you have chosen channel 5 of the **SIGNAL** field from face 4 of the FIRAS Initial Product skymap, and you would like to put it into a variable called **FSIGNAL**, type

```
UIDL> UGETDATA, FSIGNAL
```

The **Select Object** menu will appear:



This list will contain all of your UIMAGE objects (graphs, 2-D skymaps, and 3-D spectral cubes). The first object is the one we want to use here. Select it to place this data into the specified variable. Then

```
UIDL>  HELP, FSIGNAL
FSIGNAL     FLOAT     = Array(32, 32)
```

shows that **FSIGNAL** does contain the expected data array size and type. Follow the same simple procedure to read in 3-D objects.

If you select the graph **Pixel 4342 spectrum**, UIMAGE displays the following informational message:

```
A 2-D array is being returned which contains the data that was used
to make the selected graph.  Please index the array by "(*,0)" to
access the X data, and index it by "(*,1)" to access the Y data.
```

X and *Y* refer to the graph axes. For example, if the pixel spectrum were constructed from FIRAS **SIGNAL(5:7)**, **FSIGNAL** would be a 3 × 2 floating point array with frequencies in **FSIGNAL(0:2,0)** and signals in **FSIGNAL(0:2,1)**.

5.5.1.2 Writing Data Sets to UIMAGE (UPUTDATA)

Suppose that you've been working with a FIRAS skycube and have created some modified version of it called **F20BETTER**. You may copy it into UIMAGE by typing

```
UIDL> UPUTDATA, F20BETTER
```

You will then be prompted for some unsupplied quantities such as bad pixel value and coordinate system. You can probably just let the defaults stand for most of these, but *be sure* to give the object some informative title. If you know some of these quantities, you can specify them directly on the UIDL command line. To pass the frequency array **FREQ** along with your data, for example, type

```
UIDL> UPUTDATA, F20BETTER, FREQUENCY = FREQ
```

This is especially useful if you have restored a save set written by UIMAGE that contains the ancillary information in addition to the data. Type `UHELP, 'UPUTDATA'` for a complete list of the ancillary variables.

You cannot put an array of arbitrary size into UIMAGE. Variables must be of the following well-defined sizes:

Resolution	Skycubes	Single Face	Sixpack*
6	128× 96	32× 32	96× 64
7	256× 192	64× 64	192× 128
8	512× 384	128× 128	384× 256
9	1024× 768	256× 256	768× 512

* UIMAGE automatically converts input sixpack arrays to unfolded skycubes.

Three-dimensional arrays of size (*m*, *n*, *p*) will also work as long as *m* and *n* match one of the dimension pairs in the table above. You can repeat this process with as many variables as you like. When you are ready to go back into UIMAGE to work with them, just type

```
UIDL> UIMAGE
```

and UIMAGE will be activated with your newly-created objects included in the menus. All of the objects that were present when you entered UIDL initially will still be there. The color tables will have been reinitialized, however, and any graphics overlays such as coordinate grids will have been lost.

5.5.2 Reading COBE Data Products at the Command Line (DATAIN)

To read the released COBE data products into UIDL without going through UIMAGE, use the **DATAIN** routine. Note that you must know

* the full name of the data set, including directory
* the name of the variable that you want to read (from the FITS extension header -- see Appendix E)
* which elements of the variable arrays you want to read (if you don't want to read them all)
* which face or faces you want (if you don't want a full skymap)

before calling **DATAIN**.

As an example, read in the first 37 channels of the FIRAS right side, fast scan mode PDS data, saving the data itself and the corresponding frequencies:

```
UIDL> DATAIN, 'CGIS_FITS:FIRAS_ALL_SKY_SPECTRA_RHFA.FITS', $
UIDL> DSFIELD = 'REAL_SPE', $
UIDL> SUBSCR = '1:37', DATA = FDATA, FREQUENCY = FFREQ
UIDL> HELP
FDATA           FLOAT     = Array(128, 96, 37)
FFREQ           FLOAT     = Array(37)
```

Similarly,

```
UIDL> DATAIN, 'CGIS_FITS:DIRBE_GALPLANE.FITS', FACE = 0, $
UIDL> DSFIELD = 'PHOTOMET', SUBSCR = '1', DATA = F0B1A
```

reads in the DIRBE photometry for band 1A in face 0.

You can obtain more than just these two variables; get a complete listing by typing `UHELP, 'DATAIN'`.

Note: The remaining subsections of section 5.5 are devoted to reading or writing data in other forms. You may want to skip ahead to the description of analysis tools in section 5.6.

5.5.3 Importing and Exporting COBE IDL Save Sets (CISS)

5.5.3.1 Reading in COBE IDL Save Sets (RESTORE)

You can read in any data that have been previously saved into an IDL save set by using the IDL **RESTORE** command. If the data from a previous example have been saved into a file called **MYDATA.CISS**, the command

```
UIDL>  RESTORE, 'MYDATA.CISS'
```

retrieves the data into their original variables. In addition, if you have created the save set in UIMAGE, the ancillary information such as the bad pixel value accompanies the data. It is also faster to restore data from save sets than to read them from the original data set.

5.5.3.2 Writing IDL Save Sets (SAVE)

You may write data and other variables to IDL save sets for future use, saving access and analysis time later. To save all of your variables, type

```
UIDL> SAVE, FILE = 'MYDATA.SAV', /VAR
```

To save only some of your variables (*e.g.*, the data and some of the ancillary information), use a call like this:

```
UIDL> SAVE, FILE = 'MYDATA.SAV', DATA, BADPIXVAL, SCALE_MAX,$
UIDL> SCALE_MIN
```

5.5.4 Reading in ASCII Files (READCOL)

It is possible to read ASCII data into IDL variables. Given rough coordinates for several regions of the sky, **COORCONV** (discussed below) will let you find the corresponding pixel numbers, but first the data must be entered into UIDL. This can be done using **READCOL**. **READCOL** will read up to 15 columns of data (separated by commas or spaces) with the following formats:

* string (format A)
* byte (B)
* double precision (D)
* floating point (F)
* integer (I)
* longword (L)
* skip a column (X)

You must specify the variable names into which the data will be stored. It is not necessary to read every column, nor to edit the file to eliminate comments; lines whose formats do not match the specified format are ignored.

Assume that a sky coordinate file, called **REGIONS.DAT**, looks like this:

```
REGION      RA (HRS, MIN, SEC)      DEC (DEG, MIN)
LMC          5 30 00                -70 00
SMC          1 45 30                -72 30
SGP          0 55 50                -28 45
NGP         12 50 00                +27 50
```

and that you want to read the data into the IDL variables **NAMES**, **HR**, **MIN**, **SEC**, **DEG**, and **DECMIN**. The appropriate command is

```
UIDL>  READCOL, 'REGIONS.DAT', FORMAT = 'A, F, F, F, F, F', NAMES, $
UIDL>  HR, MIN, SEC, DEG, DECMIN
% READCOL: Skipping Line 1
% READCOL: 4 valid lines read
```

where the A and F specify the format types *string* and *floating point*. The top line doesn't fit this format and is skipped automatically. Each of the IDL variables is created as a 4-element array.

To skip the first column, use

```
UIDL> READCOL, 'REGIONS.DAT', FORMAT = 'X, F, F, F, F, F', HR, MIN, $
UIDL> SEC, DEG, DECMIN
```

5.5.5 Using FITS Files

FITS format is a standard within the astronomical community. Files in this format have a header followed by data. At a minimum, the header contains the following keywords (see section 5.5.5.2 for information on other common FITS header keywords):

SIMPLE = T	(file is in one of defined FITS formats; T = true)
BITPIX = *n*	(specifies data format: *n* = −32 REAL*4; 16 INTEGER*2; 32 INTEGER*4)
NAXIS = *j*	(*j* is number of dimensions in data array)
NAXIS*k* = *m*	($1 \le k \le j$; dimension #*k* has *m* elements)
END	(end of header)

UIDL is capable of reading in data sets in rasterized FITS format using the **READFITS** command and exporting data in FITS format using the **WRITEFITS** command.

5.5.5.1 Reading in FITS Files (READFITS)

You can read rasterized FITS data into UIDL with the routine **READFITS**. In the example below, the data from a FITS file called **MYDATA.FITS** are read into the IDL variable **MYDATA**, which is automatically created as a 128 × 96 floating point array. The header is also saved to an IDL variable, **FITSHEADER**.

```
UIDL>  MYDATA = READFITS( 'MYDATA.FITS', FITSHEADER)
```

The command

```
UIDL>  PRINT, FITSHEADER
```

gives

```
SIMPLE   =    T / Written by IDL  11-Aug-1992 12:33:49.00
 BITPIX  =  -32 /
 NAXIS   =    2 /
 NAXIS1  =  128 /
 NAXIS2  =   96 /
 DATATYPE= 'REAL*4'        / Type of data
 DATE    = '11/08/92'      /
 END
```

for the 8-element string array, one element for each line. Note that the date is given day first, then month.

If the **BZERO** and **BSCALE** keywords appear in the header, the data may have been previously converted to integers by scaling: (*FITS value* = (*real value* - **BZERO**)/**BSCALE**). **READFITS** reverses this scaling by default, resetting **BZERO** to 0. and **BSCALE** to 1. in the process. Use

```
UIDL>  MOREDATA = READFITS( 'filename', /NOSCALE)
```

to retain the integer data values as well as the original values for **BZERO** and **BSCALE**.

5.5.5.2 Writing FITS raster files (WRITEFITS)

Data may be saved into a one plane rasterized FITS file. (Rasterized FITS format is the required input format for any data you plan to analyze using many other software packages, such as IRAF and SAOimage.) To write the data from a 14 × 2 array **HILONLAT** to a file **HIGHPLACES.DAT** in your default directory, enter

```
UIDL> WRITEFITS, 'HIGHPLACES', HILONLAT
```

The routine will supply a simple FITS header with the same keywords shown in the sample header in section 5.5.5.1.

Files may be written to other directories (if you have write privilege to them) by giving the complete file specification:

```
UIDL>  WRITEFITS, 'OLDDISK:[MYOTHERDIR]HIGHPLACES.FITS', HILONLAT
```

If you plan to use your data in another analysis package, or would simply like more information in the FITS header than is provided in the minimal header, you can define your own or modify an existing one with the **SXADDPAR** routine.

The following example shows how to create your own minimal header from scratch:

```
UIDL> SXADDPAR, MYHEADER, 'BITPIX', -32, 'REAL*4'
UIDL> SXADDPAR, MYHEADER, 'NAXIS', 2
UIDL> SXADDPAR, MYHEADER, 'NAXIS1', 14, 'Hot spot positions'
UIDL> SXADDPAR, MYHEADER, 'NAXIS2', 2, 'Longitude, latitude'
UIDL> PRINT, MYHEADER
SIMPLE  =                    T /Sample header
 BITPIX  =                  -32 /REAL*4

 NAXIS   =                    2 /

 NAXIS1  =                   14 /Hot spot positions

 NAXIS2  =                    2 /Longitude, latitude

 END
```

The **END** statement is added for you automatically.

You may include additional entries in your header in the same way, with keywords of your own choosing or some of these standard keywords:

BUNIT	string with data unit; *e.g.*, `'MJy/sr'`.
DATAMAX	maximum value of data
DATAMIN	minimum value of data
BZERO	see section 5.5.5.1
BSCALE	see section 5.5.5.1
BLANK	data value in blank pixels
OBJECT	string with name of object observed
TELESCOP	string with observing telescope
OBSERVER	string with names of observers
DATE-OBS	string with date of observation
DATE	string with file creation date
CTYPE*i*	string with label for axis *i*, $1 \leq i \leq$ **NAXIS**
CRVAL*i*	string with data value in reference pixel for axis *i*, where reference is typically either the first pixel or the central one)
CRPIX*i*	index of the reference pixel for axis *i*, numbering begins with 1
CDELT*i*	spacing of pixels along axis *i*

HINT: To include single quote marks in the text for a string header line, use 2 consecutive single quotes, as in `IT''S`.

The command to include this header in the FITS file is

```
UIDL>  WRITEFITS, 'HIGHPLACES', HILONLAT, MYHEADER
```

5.5.6 Reading in COBE Skymap Format Data (BUILDMAP)

(**NOTE:** The routine discussed in this section does not work outside of the COBECL environment and is not intended for use by guest investigators.)

After some preliminary processing, unreleased COBE data are placed into data archives on the COBECL cluster. Some of these data are in original time order, while some have been further processed and ordered in skymap form. The following examples show how to read these skymaps using the UIDL routine **BUILDMAP**. (More detailed information on using data from the COBE project archives is available in the *Data Access* section of the *CDAC Orientation Handbook*.)

The data set name (including the directory path), variable names (called *fields*), and skycube faces of interest must be known in order to use **BUILDMAP**. Please note that the files in these directories may be offline. You must check that the file and its corresponding index file exist in the specified directory *before* running **BUILDMAP**.

As an example, we'll read in the Galactic Center data from a FIRAS skymap and put the data from just that face into a variable called **FDATA**.

```
UIDL> FDATA = BUILDMAP($
UIDL> FILE = 'FIRCOADD:[FIRAS.ARCHIVES.SKY_93HYBRID]FCF_SKY_
RLSF.ED_9019318_9020711',$
UIDL> FACE = 4, FIELD = 'SPEC_DATA.SPEC(1:140)')
```

FDATA now holds a 32 × 32 × 140 element complex array.

The **$** sign (shown at the ends of the first two lines above) is the IDL continuation character used to break up commands too long to fit on one line. IDL will accept it only in certain places; inserting **$** following a comma between parameters is safe. Also note that the second line wrapped around. This is also acceptable, although it makes editing the line more cumbersome. Had we not specified any particular subscript or range of the signal field array, all frequencies would have been read. Numbering of field subscripts in **BUILDMAP** begins with 1.

BUILDMAP can:

* Read in up to 7 or 8 different fields (see the online help listing), as long as they are the same data type. (This can be a real time-saver.)

* Read in a whole unfolded skycube or a single face.

* Read in the real or imaginary parts of complex data.

* Make an output map in sixpack form, rather than an unfolded skycube, to save space. (Sixpacks are discussed in more detail below.)

To read more than one field of the same data type, in this case, the photometry for two non-adjacent channels of a DIRBE skycube, use a call to **BUILDMAP** like this:

```
UIDL>  BPHOT = BUILDMAP($
UIDL>  FILE = 'DIRBE_PDS1:[DIRBE_EDIT]B_IP_GS.INIT_PROD_931741607', $
UIDL>  FACE = -1, FIELD = 'PHOTOMETRY(1), PHOTOMETRY(7)')
```

(Note the value of `-1` for the **FACE** keyword; this indicates the entire skycube.) The variable **BPHOT** will be a floating point array with dimensions 1024 × 768 × 2. To read adjacent channels, say, 7 through 9, use `FIELD = 'PHOTOMETRY(7:9)'`, resulting in a 1024 × 768 × 3 array.

WARNING: This is an example *only*. IDL memory limitations on your system may not allow you to read in three full DIRBE skymaps.

You may have noticed that the unfolded skycube results in a data array that is half blank. While various celestial features are more easily visualized in terms of the skycube, you can pack the faces more efficiently for storage or when IDL memory limits require it. This packing is referred to as a *sixpack*; the packed cube faces then look like those in Figure 5.1. (We have included a standard unfolded skycube for comparison.)

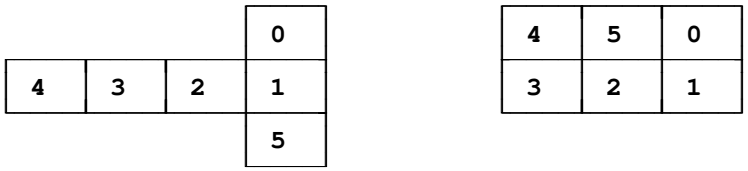


Figure 5.1 Left: An unfolded skycube. Right: An unfolded skycube packed into a sixpack.

Our final example shows how to use **BUILDMAP** to automatically read data into a sixpack, using the DIRBE data set from above:

```
UIDL> BSIX = BUILDMAP($
UIDL> FILE = 'DIRBE_PDS1:[DIRBE_EDIT]B_IP_GS.INIT_PROD_931741607', $
UIDL> FACE = -1, FIELD = 'PHOTOMETRY(2)', /SIXPACK)
```

The resulting floating point array **BSIX** will have dimensions 768 × 512 (compared to 1024 × 768 for an unfolded skycube).

5.6 USING UIDL FOR COMMON ANALYSIS TASKS

This section shows some specific examples using UIDL routines to analyze data from the different COBE instruments. It is not meant to be exhaustive in either the choice of routines or the use of the capabilities of individual routines. Refer to the UHelp files (`UHELP, '`*routine name*`'`) for more specific information.

5.6.1 Packing an Unfolded Skycube to Save Space (SIXPACK)

Several UIDL programs have options that allow the creation of maps directly into sixpack format (see Figure 5.1 above). **SIXPACK** rearranges data in an unfolded skycube into the packed format. Suppose we have a FIRAS spectral cube for elements 5 through 7 of the **REAL_SPE** field in a variable called **FSIGS**, with dimensions 128 × 96 × 3. To pack this into a variable called **FSIGS6P** with dimensions 96 × 64 × 3, enter

```
UIDL> FSIGS6P = SIXPACK( FSIGS )
```

You may then delete **FSIGS**

```
UIDL> DELVAR, FSIGS
```

to clean up.

5.6.2 Unpacking a Packed Skycube (SIXUNPACK)

SIXUNPACK is used to restore a sixpack file to an unfolded skycube or to extract a single slice from a packed spectral cube. (The latter might be especially useful for large DIRBE sets.) Given a variable **BSIG6P**, which holds a DIRBE sixpack of two channels, the call to unpack only the first slice looks like:

```
UIDL> BSIG1 = SIXUNPACK( BSIG6P( *, *, 0) )
```

which returns a floating array of dimension 1024 × 768. (Note that IDL array indices are numbered beginning with 0.)

5.6.3 Building an Unfolded Skycube from Separate Faces (BUILDTEE)

If you are working with smaller data sets, you will probably be working with complete unfolded skycubes. If, on the other hand, you are doing complicated analyses or using a large data set, you may find it more convenient to handle the data on a face-by-face basis. To put the faces back together into an unfolded skycube (sideways *T*) or a sixpack (see Figure 5.1, section 5.5.6), use **BUILDTEE**. The following example takes six separate faces and attaches them together into an unfolded cube called **DMRCUBE**.

```
UIDL> DMRCUBE = BUILDTEE( FACE0, FACE1, FACE2, FACE3, FACE4, FACE5)
```

By specifying which data square matches which face, you can list the faces in any order, or list fewer than six faces. The following example shows an unfolded skycube built from just the four faces containing Galactic Plane data:

```
UIDL> DMRGALPLANE = BUILDTEE( FACE0, FACE2, FACE4, FACE5, $
UIDL> FACENUMS = [0,2,4,5], BADVAL = -1)
```

The **BADVAL** value will be put into any blank spaces in the array. Data points of value zero will *not* be changed.

The faces used for **BUILDTEE** may contain more than one channel or frequency per pixel, as long as the dimension of each is the same. For a FIRAS map, *e.g.*, **FACE0** and **FACE4** might come from **REAL_SPE(6:10)**. You must check that the channels themselves match, as the **BUILDTEE** routine will happily combine a **FACE0** of **REAL_SPE(6:10)** with a **FACE4** of **REAL_SPE(5:9)**.

5.6.4 Making Different Projections (REPROJ)

For some studies or comparisons, the COBE skycube may not be the most helpful way to display data. You can use the **REPROJ** routine to reproject skycube data into another coordinate system.

REPROJ will

* reproject a skycube or a face into another skycube with a different coordinate system overlay or into an Aitoff, Global Sinusoidal, or Mollweide projection
* allow you to specify the minimum and maximum for scaling an image
* give either large or small plot sizes
* allow you to overlay the coordinate grid of your choice (ecliptic, Galactic, equatorial) using either default meridians and parallels or your own defined spacing
* save the reprojected data into an IDL array

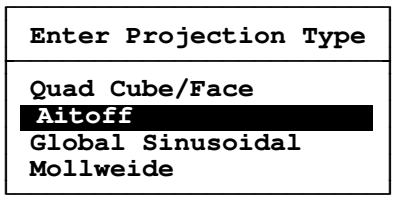
REPROJ ***can only reproject starting from skycubes, sixpacks, or single faces; you cannot, for example, reproject an Aitoff image to skycube format. The initial data are assumed to be in ecliptic coordinates, epoch J2000.***

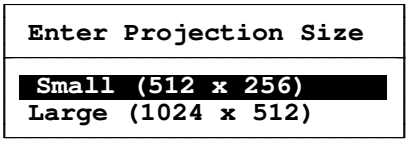
You can display reprojections only on an X Windows terminal. You can still use **REPROJ** on a non-X Windows terminal to form projected data arrays without displaying the result by using the **/NOSHOW** keyword (more on this below). Unless we specify otherwise, the following examples assume that you are working on an X Windows terminal.

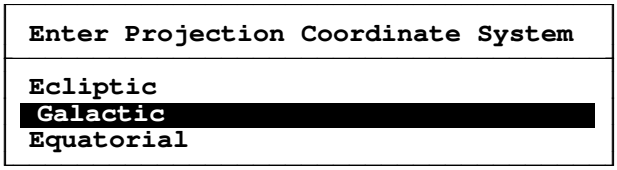
Our data sample for these examples is a single frequency DIRBE skymap that we have placed into the IDL variable **BPHOT1**. Using an X Windows terminal, we'll transform the image into a small Aitoff projection in the Galactic coordinate system. Then we'll put the reprojection into a variable called **BPHOT1P**:

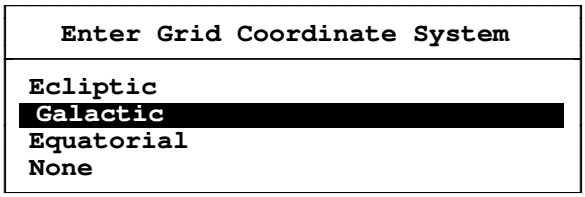
```
UIDL> REPROJ, BPHOT1, BPHOT1P
```

At this point, a series of menus appears, prompting for necessary information that was not entered at the command line. Our choices are highlighted:









The following informational messages may appear on the screen:

```
Restoring AIT_GAL Lookup Table
Building Projection
UIDL>
```

The default reprojection comes with both parallel and meridian overlays *on the display only*; they are not saved in the reprojected raster data, although you may store them yourself using the **TVRD** command. The reprojected raster data, which we've called **BPHOT1P**, may then be printed using **PSIMAGE**.

There are some features of **REPROJ** that must be specified on the command line (*i.e.*, you are not prompted for them). Saving the data into a variable, as we did above, is one. Working with a single face is another. Suppose we just want to look at the Galactic Center region. Having previously stored some face 4 data into the variable **BFACE4**, we can reproject it with

```
UIDL> REPROJ, BFACE4, PROJ = 'A', PSIZE = 'S', COORD = 'G',$
UIDL> GCOORD = 'G', MIN = 1, MAX = 50, FACE = 4
```

where	**PROJ** is the	projection type (**Q**uad Cube, **A**itoff, Global **S**inusoidal, **M**ollweide)
	PSIZE	projection size (**S**mall [512 × 256], **L**arge [1024 × 512])
	COORD	reprojected data coordinate system (**E**cliptic, **G**alactic, e**Q**uatorial)
	GCOORD	grid coordinate system (**E**, **G**, **Q**)
	MIN	image scale minimum
	MAX	image scale maximum
	FACE	face number

Everything done up to now has been put into IDL window 0 by default. Let us keep our last image up and define a new window (#1) with the IDL **WINDOW** command. Say we're working with the Initial Products and would like to set the parallels at 0°, 10°, and 15° to correspond to the data boundaries. For this plot, we'll omit the meridians:

```
UIDL> WINDOW, 1
UIDL> REPROJ, BFACE4, PROJ = 'A', PSIZE = 'S', COORD = 'G',$
UIDL> GCOORD = 'G', MIN = 1, MAX = 50, FACE = 4, WIN = 1, $
UIDL> MERD= -1, PARA = [0, 10, 15]
```

where **MERD** is the meridian (longitude) array (**-1** for none)
PARA parallel (latitude) array (**-1** for none)

Finally, **for non-X Windows users**, the command

```
UIDL> REPROJ, BFACE4, BF4RASTER, /NOSHOW
```

will prompt for the desired coordinate system, *etc.*, as above, and store the reprojected data into **BF4RASTER** without displaying it on your terminal. (Requests for meridians and parallels will be ignored.)

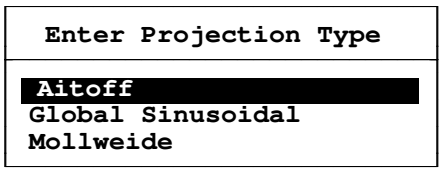
5.6.5 Adding Grids to Displays (PROJGRID)

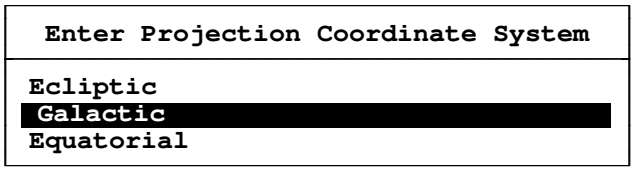
Use **PROJGRID** to put a coordinate grid on currently existing data displays (reprojected or otherwise). Since the coordinate grid overlays do not become part of any recoverable data set, **PROJGRID** can be used *only* on X Window devices. First use the **WSET** command to specify which window contains the data of interest:

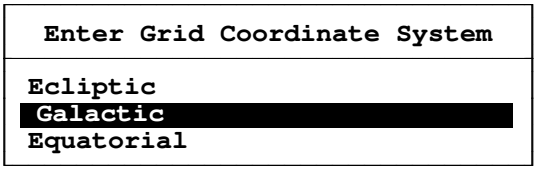
```
UIDL> WSET, window_number
UIDL> PROJGRID, BPHOT1P, W_POS = [32, 32]
```

(Here **BPHOT1P** is the reprojected DIRBE band 1 skymap from our first **REPROJ** example.)

You will be prompted, as in **REPROJ**, for any information not entered on the command line:







(The first two menus appear only if you are putting a grid over a reprojection.) The grid will appear in the window specified and hardcopies can be made as before, if desired.

Note that we specified a window position offset of `[32,32]`. This takes into account the 32-pixel blank border around reprojected displays created by **REPROJ**. The raster variables themselves created by **REPROJ** do not have this border, so no window offset would be needed if you later display one using the **TV** command.

5.6.6 Getting Data from Pixels (PIXINFO) [X]

PIXINFO allows you to determine the value of the data in a given pixel or distinctive region of an image. (This works only on an X Windows terminal.) As points are chosen with the mouse, the display shows the IDL *x* and *y* image coordinates, the longitude/latitude location in a specified coordinate system, the pixel number, and the map value at each position. This sample call to **PIXINFO**,

```
UIDL> COORDS = PIXINFO(MYFACE4, OUTCOORD = 'G', WIN = 0, FACE = 4, $
UIDL> DATVAL = VALUES)
```

stores the galactic coordinates of the selected points into the variable **COORDS**. The corresponding values of the data are put into the **VALUES** variable. The input data, **MYFACE4**, is a DIRBE face 4 map displayed in IDL window 0. Additional information may be requested if you are using reprojected data.

The routine tells you which mouse buttons to use for starting a new line of the value/coordinate display or exiting. The output includes the point on which <MB3> is clicked. You then have a chance to refresh the image and remove the points from it.

The values displayed by **PIXINFO** will also be saved into a journal file if you have opened one. Type `UHELP, 'PIXINFO'` for details on other options.

5.6.7 Converting Coordinates (COORCONV)

Many features of interest in COBE data are located more easily by Galactic, ecliptic, or equatorial coordinates than by skycube pixel number. The pixel coordinate correspondence may be determined with the **COORCONV** routine, which lets you move among pixel number, longitude/latitude, decimal hour/latitude, and coordinate position unit vector formats. You may also use **COORCONV** to determine the FIRAS or DMR (resolution 6) pixel corresponding to a particular DIRBE pixel (resolution 9). **COORCONV** does *not* reproject data, nor does it *change* the resolution of a data set. Use **REPROJ** to do the former and **CONGRID** or **DOWNRES** to do the latter.

For example, suppose you want to find the FIRAS pixels corresponding to the brightest DIRBE pixels in a given data set. For the DIRBE set, we shall use a band 1 skymap from the Initial Product data set, which was previously written into an IDL array called **BPHOT1**.

First identify the interesting pixels:

```
UIDL> DIRBEHIGH = WHERE( BPHOT1 GT 0.2 * MAX(BPHOT1) )
```

DIRBEHIGH, a longword array, contains the pixel numbers of pixels with data values 0.2 or more of the maximum value.

Now find the corresponding FIRAS pixels:

```
UIDL> FIRHIGH = COORCONV( DIRBEHIGH, INFMT = 'P', $
UIDL> OUTFMT = 'P', INCO = 'B', OUTCO = 'F' )
```

where **INFMT** is the input coordinate format (**P**ixel, decimal **H**r/dec, **L**on/lat, **U**nit vector)

OUTFMT output coordinate format (**P**, **H**, **L**, **U**)

INCO input coordinate system or resolution
(for **INFMT** of **'P'**, choices are
F, **D**, or **R6** for FIRAS and DMR (resolution 6)
B or **R9** for DIRBE (resolution 9)
R*n* for resolution *n* ($n \leq 15$))

OUTCO output coordinate system or resolution (same choices as **INCO**)

The array **FIRHIGH** now contains the 12 FIRAS pixel numbers corresponding to the original 12 DIRBE pixels, some of them duplicates, as you might guess from DIRBE's finer spatial resolution.

To turn the DIRBE peaks into Galactic coordinates (longitude/latitude) instead, use

```
UIDL> HILONLAT = COORCONV( DIRBEHIGH, INFMT = 'P', $
UIDL> OUTFMT = 'L', INCO = 'B', OUTCO = 'G' )
```

OUTCO and **INCO** may also have the values **E**, **G**, and **Q** for Ecliptic, Galactic, and eQuatorial (epoch 2000), respectively, as illustrated above. **HILONLAT** is then a 12 × 2 array containing the 12 galactic longitudes and the corresponding latitudes.

COORCONV can also be used to change between ecliptic, Galactic, and equatorial coordinates. See `UHELP, 'COORCONV'` for details.

5.6.8 Making PostScript Graphs and Images (PSIMAGE)

Images and graphics may be stored in PostScript format for eventual printing to a PostScript printer using **PSIMAGE**. **PSIMAGE**

* scales the input image by default from 0 to 255 and makes the largest possible grey scale or color plot that both fits on an 8.5" × 11" page and preserves the *x*:*y* ratio of the original image array

* allows you to
  - change the scaling minimum and maximum,
  - specify the name of the plot file, and
  - shrink the plot size.

For example, suppose we want to plot a DIRBE band 1 skymap, which we have stored in the variable **DIRBE1**. Suppose we also know from trial and error with our screen displays that suitable scaling limits for our purposes are a minimum of 1 and a maximum of 50.

The command

```
UIDL> PSIMAGE, DIRBE1, 1, 50, OUTPUT='DIRBE.PS'
```

puts the scaled grey scale plot into **DIRBE.PS** in our default directory.

If the image contains any overlays (labels, grids, or a bar) that are not in the data array itself, **PSIMAGE** can still be used, but first the window data must be stored into an array:

```
UIDL> DIRBEPLOT = TVRD()
UIDL> PSIMAGE, DIRBEPLOT, SCALE = 0.5
```

The first line reads all of the pixels in the current IDL window into the variable **DIRBEPLOT**. The subsequent call to **PSIMAGE** scales each dimension of the plot by 0.5 before saving it in a file with the default name **IDL.PS**.

To make a color plot using the standard IDL color table #3 (*Red Temperature*), use a call like this:

```
UIDL>  PSIMAGE, DIRBEPLOT, COLOR = 3
```

The resulting **IDL.PS** file may then be sent to a color PostScript printer. See the **PSIMAGE** help file for more information about making color plots.

5.6.9 Fitting to a Multiple Blackbody Model (FITSPEC)

The modeling and fitting tools implemented in UIMAGE are also available to UIDL command line users.

The **FITSPEC** routine is the basis for the UIMAGE multiple blackbody fitting routine described in section 4.4.7.3. Refer to that section for specific descriptions of the parameters.

To use **FITSPEC** from the command line, you must supply

* a spectrum array, which is either the spectrum of a single pixel, an array of spectra, or a spectral cube (with frequency as the third dimension)

* a frequency array, which must be the same size as the frequency dimension of the spectrum

* the spectrum units, if not eplees

* the frequency units, if not icm

The simplest fit is invoked as:

```
UIDL> FIT = FITSPEC(FREQUENCY, SPECTRUM, /MJY, FUNITS = 'GHZ')
```

You will be prompted for initial values of the seven fit parameters. You may ignore the second blackbody component by entering `0` at the **`Second Black Body Temperature`** prompt.

You will then be asked which parameters are to be varied. For best results, you should not try to vary more than two or three parameters at a time.

The output variable, **`FIT`** in this case, holds the value of the fit at each point of **`FREQUENCY`**.

Other capabilities for which additional keywords must be specified include:

* weighting the spectrum
* giving the initial values at the command line for single spectra and arrays
* fixing or varying parameters
* masking out regions from the fitting
* changing the convergence tolerance
* storing the final parameters, errors, and fit residuals
* storing the spectrum with dust fit subtracted and with the cosmic background fit subtracted
* storing a fit failure flag, the χ^2 per degree of freedom, and the number of iterations for the fit

See `UHELP, 'FITSPEC'` for more information.

5.6.10 Fitting to a Bose-Einstein Model (FITMU)

The **`FITMU`** routine is the basis for the UIMAGE Bose-Einstein fitting routine described in section 4.4.7.4. Refer to that section for specific descriptions of the parameters.

To use **FITMU** from the command line, you must supply

* a spectrum array, which is either the spectrum of a single pixel, an array of spectra, or a spectral cube (with frequency as the third dimension)
* a frequency array, which must be the same size as the frequency dimension of the spectrum
* the spectrum units, if not eplees
* the frequency units, if not icm

The simplest fit is invoked as:

```
UIDL> FIT = FITMU(FREQUENCY, SPECTRUM, /MJY, FUNITS = 'GHZ')
```

You will be prompted for initial values of the five fit parameters. You may ignore the residual blackbody component by entering `0` at the **Residual Black Body Temperature** prompt.

You will then be asked which parameters are to be varied. For best results, you should not try to vary more than two or three parameters at a time.

The output variable, **FIT** in this case, holds the value of the fit at each point of **FREQUENCY**.

Other capabilities for which additional keywords must be specified include:

* weighting the spectrum
* giving the initial values at the command line for single spectra and arrays
* fixing or varying parameters
* masking out regions from the fitting
* changing the convergence tolerance
* storing the final parameters, errors, and fit residuals
* storing a fit failure flag, the χ^2 per degree of freedom, and the number of iterations for the fit

See `UHELP, 'FITMU'` for more information.

5.6.11 Subtracting a Dipole and/or Quadrupole (DIPOLE/MULTIPOLE)

The **MULTIPOLE** routine subtracts the dipole and quadrupole terms from a skymap. (A very similar routine called **DIPOLE** subtracts only the dipole from the map. Since the two are extremely alike, only **MULTIPOLE** will be discussed here. See `UHELP,'DIPOLE'` for details on the dipole fitting routine.)

The only required input for **MULTIPOLE** is the skymap data array. Optional input parameters include:

* **GALEXC**: A floating point scalar which specifies the absolute value of the minimum galactic latitude to include in the fit. (*The default is zero; we recommend that you change this to at least 20.*)

* **WEIGHTS**: A weight array

* **BADVAL:** A value indicating bad pixels

MULTIPOLE returns as outputs the mean cosmic microwave background intensity and its error, the dipole amplitude and its error, the galactic latitude and longitude of the pole of the dipole (in degrees), and the error in the dipole direction (in degrees). A simple call to **MULTIPOLE** using the data in an array called **DMRDATA** would look like this:

```
UIDL>  MULTIPOLE, DMRDATA, CMBRINT, SIG_CMBRINT, DIAMP, SIG_DIAMP, $
UIDL>  DILON, DILAT, SIG_DIDIRECTION
```

Optional output parameters include:

* **RESIDMAP** for a 2-D output map of the residuals

* **QPARMS** to return the quadrupole coefficients

* **QSIGMA** to return the error of the quadrupole coefficients

* **QUADMAP** for a 2-D output map of the quadrupole

5.6.12 Modeling Backgrounds (MKBGRMOD)

The routine **MKBGRMOD** is a general 2-D background modeling tool, and forms the basis for the UIMAGE background modeling described in section 4.4.7.5. The corresponding keywords are

CUT	the number of standard deviations to use to identify a source
SRCWIDTH	the size of the pixel patch ($n \times n$) to be removed when a source is found
FIT	the order of the polynomial surface fit
FSIZE	the size of the spatial filter (in pixels)
FILTER	the name of the function to use as a filter
BADVAL	the value of bad pixels (those to be taken out of the fit)
NOPLOT	suppresses plotting
NOCORN	specify when you do not want the corner pixels included in the fit

For combinations of these parameters, see section 4.4.7.5 or type `UHELP, 'MKBGRMOD'`.

The fits work best on relatively small areas; the program cannot handle any area larger than 512 × 512. To do the basic quintic interpolation between pixels specified with the mouse on a 35 × 20 array called **BPATCH**, use the call:

```
UIDL>  BACKGRD = MKBGRMOD(BPATCH)
```

UIDL prints out the parameters it will use and asks whether or not to continue with this fit. Simply hit a <CR> if these parameters are acceptable. The original map is displayed, and in the session window you'll see:

```
Select background points with LEFT MOUSE BUTTON.
Select last point with RIGHT MOUSE BUTTON to exit.
```

As indicated, use <MB1> to mark the points to use in the interpolation and click <MB3> when finished. (<MB3> does mark a point.) The routine then displays the background model with a labeled color bar across the top of the window.

You will find that you can quickly generate many extra windows using **MKBGRMOD**. These may be deleted by determining the window number from `HELP, /DEVICE` and using `WDELETE,` *window_number*. Alternatively, you can just close the window.

5.6.13 Placing Labels on Images

5.6.13.1 MARKMAP

This routine provides three ways to place labels on a map. You can (1) read in a file of coordinates and corresponding labels, (2) enter coordinates and labels from the keyboard, or (3) use the mouse to point and click on positions while entering the labels from the keyboard.

Since **MARKMAP** is equipped with many prompts, users may simply enter `MARKMAP,` *data_array* at the UIDL prompt and navigate through the routine. Alternatively, all necessary information can be entered at the command line in order to avoid prompting. See `UHELP, 'MARKMAP'` for a complete list of possible input parameters.

To use a file of coordinates and labels to mark a map, the file must be in one of two formats. It must either have seven columns for coordinates in Equatorial Sexigesimal coordinate format, *i.e.*, HR MIN SEC DEG DEC_MIN DEC_SEC LABEL, or it must have three columns for coordinates in longitude latitude format, *i.e.*, LONG LAT LABEL. Labels cannot have spaces in them if they are read in from a file (although labels entered from the keyboard may contain spaces). Acceptable values for label coordinates are latitudes between (-90, +90), decimal longitudes between (-360, +360), and RA hours between (0, 24). If the program finds coordinates that do not match one of these criteria, the line will not be plotted; the others, however, will still be displayed.

Suppose, for example, that the file **ECLCOORD.TXT** contains three ASCII columns with ecliptic longitudes, ecliptic latitudes, and labels. To place them on the image made from the array **FDATA**, which we have displayed in IDL window 1, the call would look like:

```
UIDL>  MARKMAP, FDATA, WIN=1, FILE='ECLCOORD.TXT', FCOORDS='E'
```

where `FCOORDS='E'` indicates that the coordinates in the file are ecliptic. (You will be prompted for a coordinate system if you do not input one.)

To enter the labels and coordinates from the keyboard, the simplest call is:

```
UIDL>  MARKMAP, FDATA, WIN=1, ENTRY='KEYBOARD'
```

You will be prompted alternately for coordinate pairs and labels. All of the labels are plotted after the last one has been entered.

Finally, to use the mouse to click on positions and place labels there, use:

```
UIDL>  MARKMAP, FDATA, WIN=1, ENTRY='CURSOR'
```

At the prompt, enter the first label followed by <CR>, and then click <MB1> on the desired location on the image in window 1. Click <MB3> when marking the last position.

5.6.13.2 WINLABEL

To call up **WINLABEL** from the UIDL command line, type

```
UIDL>  WINLABEL, window_number
```

This pulls up the **WINLABEL** widget. See section 4.4.2.8 for detailed instructions on how to use it.

5.7 EXAMPLES OF UIDL COMMAND SEQUENCES

This section contains three sample sessions. Although each features data from a different experiment, the commands presented are not instrument-specific. Through these sessions, we hope to give a sense of how to put some common UIDL commands together to do simple analysis tasks. More detailed instructions on the use of the routines can be found in UHelp. All of the examples assume that you are using a workstation or X Windows terminal.

5.7.1 Example One

This example compares halves of the Galactic Plane as seen by DIRBE band 1. We shall

(1) bring data into UIDL from UIMAGE

(2) display the data

(3) smooth the data and reproject it into a large (1024 × 512) Mollweide projection in galactic coordinates (no displayed grid)

(4) flip the data left to right about the Galactic Center

(5) overlay contours of both the original and the flipped reprojections

(6) make a hardcopy of the composite

We assume that you have used UIMAGE to acquire an unfolded skycube for the DIRBE band 1 photometry field just prior to beginning the example.

Bringing data into UIDL from UIMAGE

```
UIDL>  UGETDATA, B1
```

Store the UIMAGE data in the variable B1.

Select the DIRBE band 1 object from the menu.

```
UIDL>  HELP, B1
```

Determine the array dimensions.

```
B1       FLOAT     = Array(1024, 768)
```

Displaying the data

```
UIDL>  WINDOW, XSIZE = 1024, YSIZE = 768
```

Set up an IDL window.

```
UIDL>  LOADCT, 14
% LOADCT: Loading table STEPS
```

Set up the color table (default is black and white).

```
UIDL>  TV, B1
```

Display the data.

Smoothing and reprojecting the data

```
UIDL>  B1SMOOTH = SMOOTH( B1, 11)
```

Smooth the data with an 11 point boxcar average.

```
UIDL>  TV, B1SMOOTH
```

Display the smoothed image. This replaces the previous display.

```
UIDL>  BMAX = MAX( B1SMOOTH)
```

Find the maximum for display scaling.

```
UIDL>  REPROJ, B1SMOOTH, B1SM_REP, PROJ = 'M', $
UIDL>  PSIZE = 'L', COORD = 'G', MERD = -1, PARA = -1,$
UIDL>  MIN = 1, MAX = BMAX * 0.5
Restoring BIG_MWD_GAL Lookup Table
Building Projection
```

Reproject the smoothed map into a large Mollweide projection. This will replace the smoothed image in window 0. `B1SM_REP`, *holds the reprojected array. The reprojection is in Galactic coordinates, and the drawing of parallels and meridians has been suppressed.*

Flipping the data about the Galactic Center

```
UIDL>  B1SR_FLIP = FLTARR( 1024, 512)
```

Define the array to hold the flipped data.

```
UIDL>  FOR I = 0, 511 DO FOR J = 0, 1023 DO $
UIDL>  B1SR_FLIP( 1023-J, I) = B1SM_REP( J, I)
```

Swap left and right sides of the data array.

Overlaying contours on the original image

```
UIDL>  WINDOW, 1, XSIZE = 1024, YSIZE = 512
```

Define a new window.

```
UIDL>  !X.RANGE = [0, 1023]
UIDL>  !Y.RANGE = [0, 511]
```

Specify the exact ranges for the contour plot axes.

```
UIDL>  !X.STYLE = 1
UIDL>  !Y.STYLE = 1
```

Plot only the exact range.

```
UIDL>  !X.TICKS = 1
UIDL>  !Y.TICKS = 1
```

Do not plot the tick marks around the outside of the contour plot.

```
UIDL>  !P.POSITION = [0, 0, 1, 1]
```

Set the plot window position by giving the normalized coordinates of the lower left and upper right corners. Here, use the entire plot window.

```
UIDL>  CLEVS = [3, 5, 10, 15]
```

Define the contour levels.

```
UIDL>  CONTOUR, B1SM_REP, LEVELS = CLEVS, $
UIDL>  C_COLOR = [220]
```

Overlay the contours on the smoothed, reprojected image, specifying the contour color.

```
UIDL>  CONTOUR, B1SR_FLIP, LEVELS = CLEVS, $
UIDL>  C_COLOR = [255], /NOERASE
```

Overlay the contours for the flipped data in a different color.

```
UIDL>  WINLABEL, 1
```

Write a title on the plot.

Click in the box labeled **Enter text then <cr>** *of the WINLABEL widget.*

Type the title: `Galactic plane asymmetry search`*, then press* <CR>.

Click on the slider bar controlling the Font Size, and drag it to the right until the font size value is 24.

Click in the **OK** *box.*

Click near the center top of the image window. The label will be displayed there, centered around the point at which you clicked.

Click in the **Done** *box of the WINLABEL widget.*

Making a hardcopy of the composite image

```
UIDL>  PICTURE = TVRD()
```

Store the complete image into a variable.

```
UIDL>  PSIMAGE, PICTURE, COLOR = 14, $
UIDL>  OUTPUT = '[MYSCRATCH]:PICTURE.PS'
```

Set up the plot with the same color table we've been using. Send it to a file on your scratch area.

```
UIDL>  $PRINT/QUE=my_color_printer/NOTIFY PICTURE.PS
```

Spawn out of IDL to send the file to your system's color PostScript printer.

5.7.2 Example Two

This example reprojects a DMR data set. In this command sequence, we shall

(1) read in a DMR map from an IDL save set that has the data in sixpack skycube format with ecliptic coordinates

(2) reproject it into a small (512 × 256) Mollweide projection in Galactic coordinates

(3) store rows of the projection into IDL variables

(4) plot one slice of constant latitude of the reprojection, then superimpose two other latitude plots and get hardcopies

(5) save the projected image to a raster FITS file in the current directory

This example assumes that the name of our IDL save set is **DMR31A.CISS**, and that these data have been saved into sixpack form. (This is the default for files written by UIMAGE.) We will put the projected data into a variable **DMRGM**.

Reading in the save set

```
UIDL>  RESTORE, 'DMR31A.CISS'
```

Read the save set data back into UIDL.

```
UIDL>  HELP
DATA        FLOAT = Array(96, 64)
```

Examine the restored data variable name, type, and size. If this were a save set written by UIMAGE, the UIMAGE ancillary variables would also appear.

Reprojecting the data

```
UIDL>  REPROJ, DATA, DMRGM, PROJ = 'M', COORD = 'G', $
UIDL>  GCOORD = 'G'
```

Reproject the image from a galactic coordinate sixpack to a galactic Mollweide projection.

A menu prompt appears with projection size choices (choose **SMALL***).*

```
Restoring MWD_GAL Lookup Table
Adjusting Lookup Tables to Sixpack Format
Building Projection
```

```
UIDL>  HELP, DMRGM
DMRGM       FLOAT = Array(512, 256)
```

Get the array size of the reprojection.

```
UIDL> LOADCT, 15
% Loading table STERN SPECIAL
```
Load in a color table.

Storing the data into IDL variables

```
UIDL>  DMRST0 = DMRGM( *, 128)
```
Make a b=0 strip.

```
UIDL>  DMRSTP10 = DMRGM( *, 135)
```
Make a b=+10 degree strip.

```
UIDL>  DMRSTN10 = DMRGM( *, 121)
```
Make a b=-10 deg strip.

Plotting the strips

```
UIDL>  WINDOW, 1, XSIZE = 512, YSIZE = 200
```
Set up a separate graphics window.

```
UIDL>  !X.STYLE = 1
UIDL>  !X.RANGE = [0, 511]
```
Force the x axis to be exactly the defined length (512).

```
UIDL>  PLOT, DMRST0, LINESTYLE = 0
```
Graph the b=0 strip as a solid line.

```
UIDL>  OPLOT, DMRSTP10, LINESTYLE = 1
```
Graph the b=+10 strip as a dotted line.

```
UIDL>  OPLOT, DMRSTN10, LINESTYLE = 2
```
Graph the b=-10 strip as a dashed line.

```
UIDL>  SET_PLOT, 'PS'
```
Now redo part of the graph into a PostScript file.

```
UIDL>  DEVICE, /LANDSCAPE
```
Use landscape rather than portrait (default) orientation.

```
UIDL>  PLOT, DMRST0, LINESTYLE = 0
UIDL>  OPLOT, DMRSTP10, LINESTYLE = 1
```
Repeat the plot commands.

```
UIDL>  DEVICE, /CLOSE
```
Close the PostScript file.

```
UIDL>  $PRINT/QUE=my_PS_printer IDL.PS
```
Spawn a print job to a black-and-white PostScript printer queue. (Note the $ at the beginning of the line.)

Writing the data to a rasterized FITS file

```
UIDL>  WRITEFITS, 'DMRGM.FITS', DMRGM
```

Use the default minimal header.

5.7.3 Example Three

This example labels regions of a FIRAS map corresponding to a private list of objects. In the following command sequence, we shall

(1) read in **REAL_SPE(5)** from a FIRAS skymap using **DATAIN** and display it

(2) read in a list of object coordinates (right ascension (RA) / declination) and precess them from epoch 1950.0 to epoch 2000.0

(3) write the precessed coordinates to the file **WR2000.CAT**

(4) draw squares around areas in the FIRAS map that correspond to the coordinates of the objects and label them

(5) send the map (including the overlays) to a printer

Our file of object coordinates is called **WR.CAT**, which looks like this:

```
Sixth Catalogue of Wolf-Rayet Stars, excerpt (1950.0 positions)
WR number    RA (hr, min)      Dec (deg, min)
3            01 35.6           +57 54
6            06 52.1           -23 52
7            07 16.2           -13 08
8            07 43.0           -31 47
```

Reading in and displaying the data

```
UIDL> DATAIN, $
UIDL> 'your_PDS_directory:FIRAS_ALL_SKY_SPECTRA_RHFA.FITS',$
UIDL> DSFIELD = 'REAL_SPE', SUBSCR = '5', DATA = FSIG
```

Read in the skymap.

```
UIDL> HELP, FSIG
FSIG        FLOAT = Array(128, 96)
```

Look at the type and size of the data array.

```
UIDL> WINDOW, 0, XS = 512, YS = 384
```
Set up a 512 × 384 pixel IDL window. This is a factor of 4 enlargement.

```
UIDL> MINF = MIN( FSIG( WHERE( FSIG GT 0.)))
```
Determine data minimum and maximum for scaling.

```
UIDL> BIGIMAGE = CONGRID(FSIG, 512, 384)
```
Enlarge the image to 512 × 384.

```
UIDL> TV, BYTSCL( BIGIMAGE, MINF,  MAXF)
```
Display the enlarged image, scaling it between its minimum and maximum values.

```
UIDL> LOADCT, 13
% Loading table RAINBOW
```
Change the color table to something other than black and white.

Reading in the coordinate file and precessing the coordinates

```
UIDL> READCOL, 'WR.CAT', FORMAT = 'A, F, F, F, F',$
UIDL> NUM, RAH, RAM, DECD, DECM

%READCOL: Skipping Line 1
%READCOL: Skipping Line 2
%READCOL: 4 valid lines read
```
Read in the text file.

```
UIDL> NAME = 'WR' + NUM
```
Create the label array for the map markings. '+' is the IDL string concatenation operator.

```
UIDL> DECRA = (RAH + RAM/60.) * 15.
```
Convert the RA to decimal degrees in preparation for use with **PRECESS**.

```
UIDL> DECDEC = DECD / ABS(DECD) * ( ABS(DECD) +$
UIDL> DECM/60.)
```
Convert the declination to decimal degrees.

```
UIDL> PRECESS, DECRA, DECDEC, 1950., 2000.
```
Precess the RA and dec to equinox 2000.0.

Writing out the precessed coordinates

```
UIDL> GET_LUN, UNIT
```
Let IDL select a logical unit number to which to write the output file.

```
UIDL> OPENW, UNIT, 'WR2000.CAT'
```
Open the file `WR2000.CAT` *for writing, and assign the logical unit to it.*

```
UIDL> FOR I = 0, 3 DO $
```
Loop over the number of valid lines read, as determined by **READCOL**.

```
UIDL> PRINTF, UNIT, $
UIDL> FORMAT = '(F8.3, 1X, F7.3, 1X, A3)', DECRA(I),$
UIDL> DECDEC(I), NAME(I)
```
Write the precessed coordinates and the labels to a file.

```
UIDL> CLOSE, UNIT
```
Close up the file.

```
UIDL> FREE_LUN, UNIT
```
Return the unit.

Labeling coordinate locations with squares

```
UIDL> MARKMAP, BIGIMAGE, FILE = 'WR2000.CAT', $
UIDL> FCOORDS = 'QD', WIN = 0, PSYM = 6
% READCOL: 4 valid lines read
```
Label the enlarged image at the locations given in `WR2000.CAT`. *The file coordinates are given in decimal equatorial form. The image to be labeled is in window 0. Use squares for the plot symbols.*

Printing the map with overlays

```
UIDL>  GRAPH1 = TVRD()
```
Save the screen display to an array.

```
UIDL>  PSIMAGE, GRAPH1, OUTPUT = 'WR.PS'
```
Format the array for PostScript printing.

```
UIDL>  $ PRINT/QUE=my_PS_printer  WR.PS
```
Spawn a PostScript print job.