

STSDAS Basics

In This Chapter...

Navigating STSDAS / 3-1
Displaying HST Images / 3-4
Analyzing HST Images / 3-8
Displaying HST Spectra / 3-17
Analyzing HST Spectra / 3-21
References / 3-32

The Space Telescope Science Data Analysis System (STSDAS) is the software system for calibrating and analyzing data from the Hubble Space Telescope. The package contains programs—called *tasks*—that perform a wide range of functions supporting the entire data analysis process, from reading tapes, through reduction and analysis, to producing final plots and images. This chapter introduces the basics of STSDAS, showing you how to display your data, leading you through some simple data manipulations, and pointing you towards more sophisticated tasks, some of which are described in the instrument sections of this handbook.

STSDAS is layered on top of the Image Reduction and Analysis Facility (IRAF) software developed at the National Optical Astronomy Observatory (NOAO). Any task in IRAF can be used in STSDAS, and the software is portable across a number of platforms and operating systems. To exploit the power of STSDAS effectively, you need to know the basics of IRAF. If you are not already familiar with IRAF, consult the IRAF Primer in Appendix A before reading further.

3.1 Navigating STSDAS

The tasks in STSDAS are far too numerous and complicated to describe comprehensively in this volume. Instead we will show you where to find the STSDAS tasks appropriate for handling certain jobs. You can refer to online help or the *STSDAS User's Guide* for details on how to use these tasks. Some useful online help commands are:

- `help task` - provides detailed descriptions and examples of each task.
- `help package` - lists the tasks in a given package and their functions.
- `describe task` - provides a detailed description of each task.
- `example task` - provides examples of each task.
- `apropos word` - searches the online help database for tasks relating to the specified word (see Figure A.4 in Appendix A).

3.1.1 STSDAS Structure

STSDAS version 2.0 is structured so that related tasks are logically grouped. In many cases, you can find a task that performs whatever function you need simply by looking in the appropriate package. For example, all of the tasks that are used in the calibration process can be found in the **hst_calib** package and all tasks used for image display and plotting can be found in the **graphics** package. Figure 3.1 shows the STSDAS package structure. Note that IRAF version 2.11 must be installed on your system in order for you to use STSDAS 2.0 and TABLES version 2.0

3.1.2 Packages of General Interest

Images

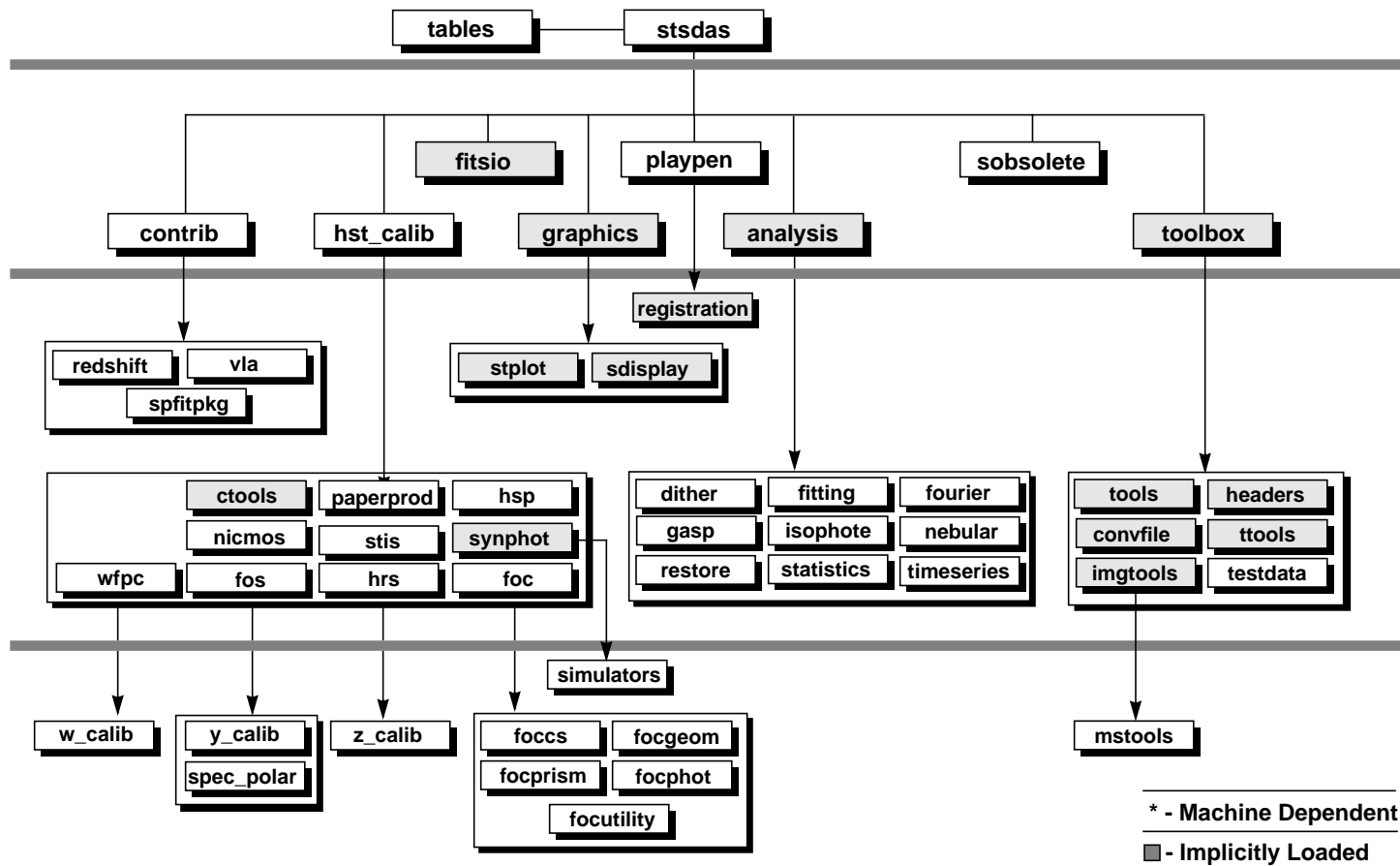
Both IRAF and STSDAS contain a large number of tasks that work with HST images. Some of the packages you should investigate are:

- **images**: This package includes general tasks for copying (**imcopy**), moving (**imrename**), and deleting (**imdelete**) image files. These tasks operate on both the header and data portions of the image. The package also contains a number of general purpose tasks for operations such as rotating and magnifying images.
- **stdas.toolbox.imgtools**: This package contains general tools for working with multigroup GEIS images, including tasks for working with masks, and general purpose tasks for working with the pixel data, such as an interactive pixel editor (**pixedit**).
- **stdas.toolbox.imgtools.mstools**: This package contains tools for working with FITS image extensions, in particular STIS and NICMOS image sets (**imsets**).
- **stdas.analysis**: This package contains general tasks for image analysis.

Tables

Several of the analysis packages in STSDAS, including calibration pipeline tasks, create output files in STSDAS table format, which is a binary row-column format, or in FITS binary table format. (ASCII-format tables are also supported, for input only.) The *STSDAS User's Guide* describes the STSDAS table format in

Figure 3.1: STSDAS Version 2.0 Package Structure



detail. Tasks in the **ttools** package or in the external **tables** package can be used to read, edit, create, and manipulate tables. For example:

- **tread** displays a table, allowing you to move through it with the arrow keys.
- **tprint** displays a table.
- **tcopy** copies tables.
- **tedit** allows you to edit a table.

Many other tasks in **ttools** perform a variety of other functions. See the online help for details.

3.2 Displaying HST Images

This section will be of interest primarily to observers whose datasets contain two-dimensional images, as it explains:

- How to display images in IRAF using the **display** task.
- How to display subsections of images.

Observers viewing WF/PC-1 and WFPC2 data may wish to remove cosmic rays before displaying their data (see page 26-20). The FOC photon-counting hardware does not detect cosmic rays as easily as CCDs, the NICMOS pipeline automatically removes cosmic rays from MULTIACCUM observations, and the STIS pipeline automatically removes cosmic rays from CR-SPLIT association products.

3.2.1 The display Task

The most general IRAF task for displaying image data is the **display** task, the best choice for a first look at HST imaging data. To display an image, you need to:

1. Start an image display server, such as SAOimage, in a separate window from your IRAF session, either from a different xterm window or as a background job before starting IRAF. To start SAOimage, type the following in any xterm or other system window:

```
saimage &
```

2. Load the **images.tv** package from the window where you're running IRAF:

```
cl> images
im> tv
```



Several different display servers, including SAOimage, SAOtng (the next generation of SAOimage), and Ximtool, can be used with IRAF. SAOtng may be retrieved via anonymous FTP from `sao-ftp.harvard.edu` in the directory `~ftp/pub/rd`. Ximtool may be retrieved via anonymous FTP from `iraf.noao.edu` in the directory `~pub/v2103-beta`. Ximtool is particularly handy if you want to blink images.

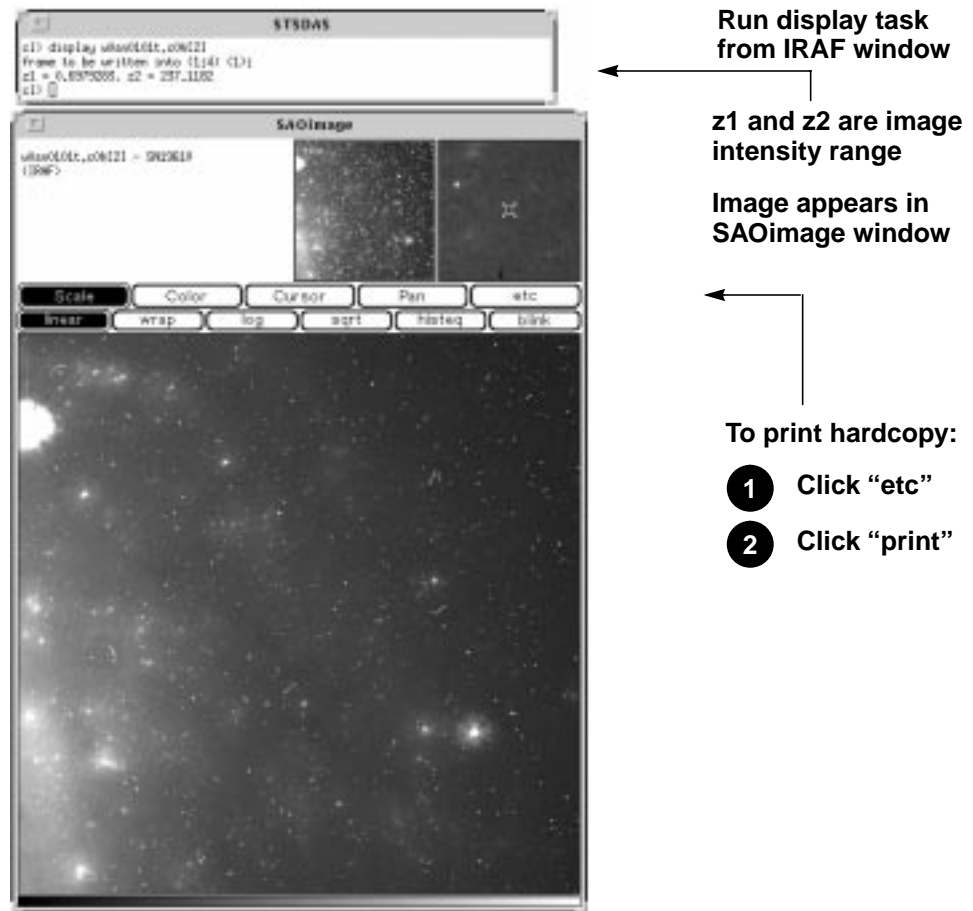
3. Display the image with the IRAF **display** task, using the syntax appropriate for the file format (Chapter 2 explains how to specify GEIS groups and FITS extensions):

```
tv> display fname.c0h[2] 1 (GEIS group 2)
tv> display fname.fits[11] 1 (FITS extension 11)
tv> display fname.fits[sci,3] 1 (FITS extension sci,3)
```

Note that when using **display** or any other task on GEIS images, you do not need to specify a group; the first group is the default. However, when working with FITS files you must specify an extension, unless the FITS file contains only a single image in the primary data unit and has no extensions. Figure 3.2 shows how to display group two of a WF/PC-1 image .



If you want to display all four chips of a WF/PC-1 or WFPC2 image simultaneously, you can create a mosaic with the STSDAS **wmosaic** task in the **hst_calib.wfpc** package. Type `help wmosaic` for details.

Figure 3.2: Displaying an Image

Modifying the Display

There are two ways to adjust how your image is displayed:

- Use the SAOimage command buttons that control zooming, panning, etc.
- Reset the **display** task parameters.

Once an image appears in your SAOimage window, you can use the SAOimage commands displayed near the top of the image window to manipulate or print your image. The *SAOimage Users Guide* describes these commands, although most are fairly intuitive. Just click on the buttons to scale, pan, or print the image, or to perform other commonly-used functions. On-line help is also available at the system level: type `man saoinage` in Unix or `help saoinage` in VMS.

The example in Figure 3.2 shows how you should display an image for a first look. By default, **display** automatically scales the image intensity using a sampling of pixels throughout the image. During your first look, you may want to experiment with the scaling using the `zscale`, `zrange`, `z1` and `z2` parameters. The `zscale` parameter toggles the autoscaling. Setting `zscale-` and `zrange+` tells the task to use minimum and maximum values from the image as the minimum and maximum intensity values. To customize your minimum and

maximum intensity display values, set `zscale-`, `zrange-`, `z1` to the minimum value and `z2` to the maximum value that you want displayed. For example:

```
im> disp w0mw0507v.c0h 1 zrange- zscale- z1=2.78 z2=15.27
```

Notice in Figure 3.2 that when you run **display**, the task shows you the `z1` and `z2` values that it calculates. You can use these starting points in estimating reasonable values for the minimum and maximum intensity display parameters.¹

If you want to display an image with high dynamic range, you may prefer to use logarithmic scaling. However, the log scaling function in SAOimage divides the selected intensity range into 200 linearly spaced levels before taking the log. The resulting intensity levels are rendered in a linear rather than logarithmic sense. You can often obtain better results if you create a separate logarithmic image to display. One way to create a logarithmic image is with the **imcalc** task:

```
im> imcalc x2ce0502t.c1h x2ce0502t.hhh "log10(im1+1.0)"
```

If the peak pixel in your original image contained 2000 counts, for example, you would then display the logarithmic image with `z1=0` and `z2=3.3`.

3.2.2 Working with Image Sections

Sometimes you may want to display only a portion of an image, using the syntax for specifying image sections discussed in Chapter 2. Your specified pixel range should give the starting point and ending point, with a colon separating the two. List the horizontal (*x* axis) range first, followed by the vertical (*y* axis) range. For example, to specify a pixel range from 101 to 200 in the *x* direction and all pixels in the *y* direction from group three of a GEIS format image, you would use a command such as:

```
tv> display image.hhh[3][101:200,*] 1
```

To specify the same pixel range in the second SCI extension of a NICMOS FITS image, you would use a command such as:

```
tv> display image.fits[sci,2][101:200,*] 1
```

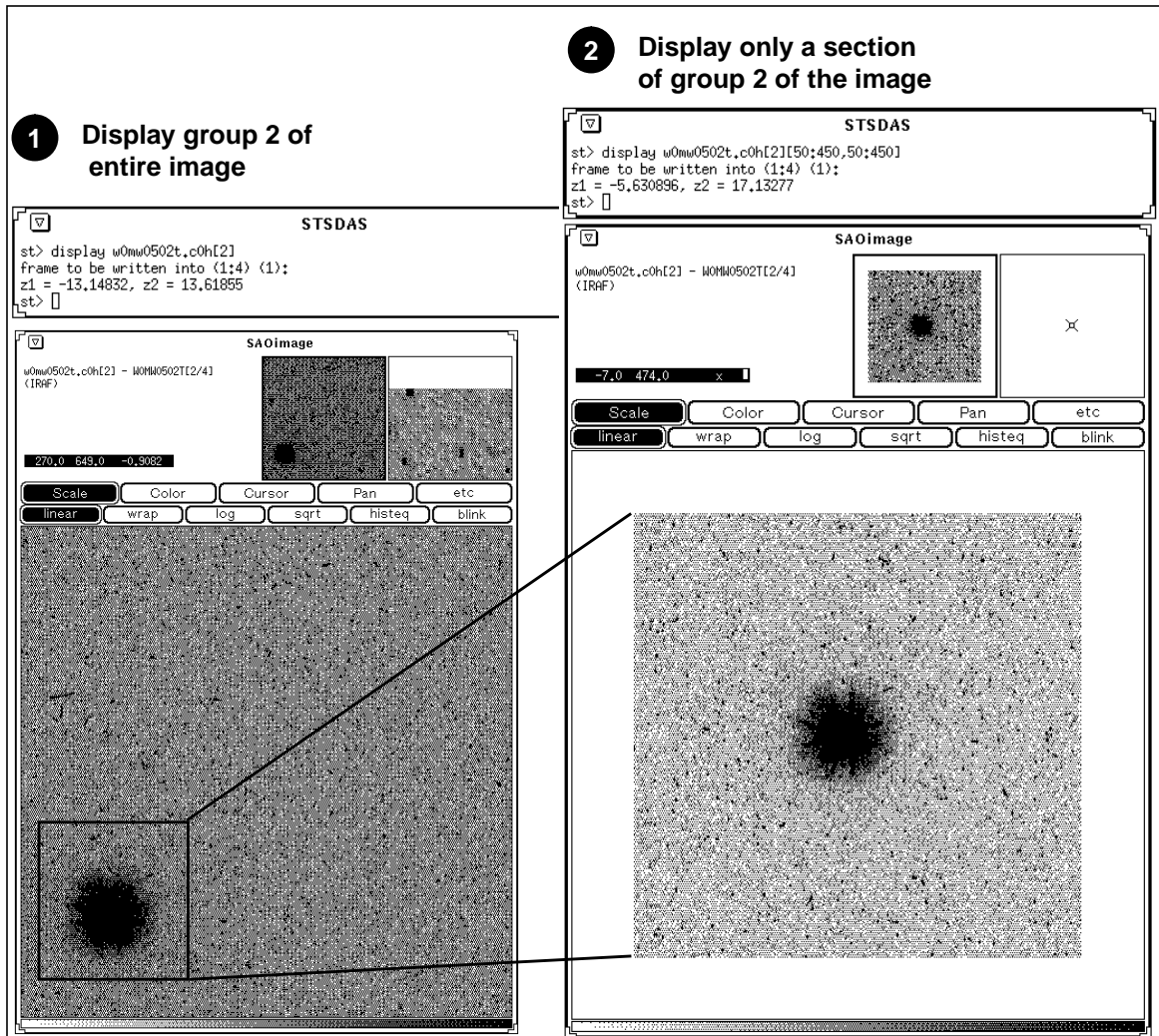


If you specify both a group and an image section of a GEIS file, the group number must come first. When displaying sections of STIS and NICMOS FITS images, you must specify the extension, and the extension designation must come first.

Figure 3.3 shows examples of displaying an image and an image section.

¹. Type `help display` within IRAF to obtain more information about these parameters.

Figure 3.3: Displaying Sections and Groups of an Image



3.3 Analyzing HST Images

This section describes methods for using STSDAS and IRAF to work with two-dimensional image data from HST. Subjects include:

- Relating your image to sky coordinates.
- Examining and manipulating your image.
- Working with STIS and NICMOS imsets.
- Converting counts to fluxes.

3.3.1 Basic Astrometry

This section describes how to determine the orientation of an HST image and the RA and Dec of any pixel or source within it, including:

- Tasks that supply positional information about HST images.
- Methods for improving your absolute astrometric accuracy.

Positional Information

The header of every calibrated HST two-dimensional image contains a linear astrometric plate solution, written in terms of the standard FITS astrometry header keywords: CRPIX1, CRPIX2, CRVAL1, CRVAL2, and the CD matrix—CD1_1, CD1_2, CD2_1, and CD2_2. IRAF/STSDAS tasks can use this information to convert between pixel coordinates and RA and Dec. Two simple tasks that draw on these keywords to relate your image to sky coordinates are:

- **disconlab**: Displays your image with a superimposed RA and Dec grid. Simply open an SAOimage window and type, for example:

```
sd> disconlab n3tc01a5r_cal.fits[1]
```

- **xy2rd**: Translates x and y pixel coordinates to RA and Dec. (The task **rd2xy** inverts this operation.) SAOimage displays the current x,y pixel location of the cursor in the upper-left corner of the window. To find the RA and Dec of the current pixel, you supply these coordinates to **xy2rd** by typing

```
sd> xy2rd n3tc01a5r_cal.fits[1] x y
```

Table 3.1 lists some additional tasks that draw on the standard astrometry keywords.

Observers should be aware that these tasks do not correct for geometric distortion. Only FOC images currently undergo geometric correction during standard pipeline processing (the .c0h/.c0d and .c1h/.c1d FOC images have been geometrically corrected); STIS images will be geometrically corrected in the pipeline once suitable calibration files are in hand. If you need precise relative astrometry, you should use an instrument-specific task that accounts for image distortion, such as the **metric** task for WF/PC-1 and WFPC2 images, described on page 28-18.

Table 3.1: Additional IRAF and STSDAS Astrometry Tasks

Task	Purpose
compass	Plot north and east arrows on an image.
north	Display the orientation of an image based on keywords.
rimcursor	Determine RA and Dec of a pixel in an image.
wcscoords	Use WCS ^a to convert between IRAF coordinate systems.
wcslab	Produce sky projection grids for images.

a. World Coordinate System (WCS). Type “help specwcs” at the IRAF prompt for details.



Do not use tasks like **rimcursor** or **xy2rd** directly on WF/PC-1 or WFPC2 images if you require accurate relative positions. Calibrated WF/PC-1 and WFPC2 images retain a residual distortion which will affect the accuracy of relative positions. Both **wmosaic** and **metric**, found in the **stdas.hst_calib.wfpc** package, correct for this distortion.

Improving Astrometric Accuracy

Differential astrometry (measuring a position of one object relative to another in an image) is easy and relatively accurate for HST images, while absolute astrometry is more difficult, owing to uncertainties in the locations of the instrument apertures relative to the Optical Telescope Assembly (OTA or V1) axis and the inherent uncertainty in the Guide Star positions. However, if you can determine an accurate position for any single star in your HST image, then your absolute astrometric accuracy will be limited only by the accuracy with which you know that star's location and the image orientation.

If there is a star on your image suitable for astrometry, you may wish to extract an image of the sky around this star from the Digitized Sky Survey and measure the position of that star using, for example, the GASP software (described in the *STSDAS User's Guide*). These tools can provide an absolute positional accuracy of approximately $0''.7$. Contact the Help Desk for assistance (send E-mail to help@stsci.edu).

3.3.2 Examining and Manipulating Image Data

This section describes **implot** and **imexamine**, two basic IRAF tools for studying the characteristics of an image, and Table 3.3 lists some useful IRAF/STSDAS tasks for manipulating image data.

implot

The IRAF **implot** task (in the **plot** package) allows you to examine an image interactively by plotting data along a given *line* (*x* axis) or *column* (*y* axis). When you run the task, a large number of commands are available in addition to the usual cursor mode commands common to most IRAF plotting tasks. A complete listing of commands is found in the on-line help, but the most commonly used are listed in Table 3.2. Figure 3.4 shows an example of how to use the **implot** task.

Table 3.2: Basic implot Commands








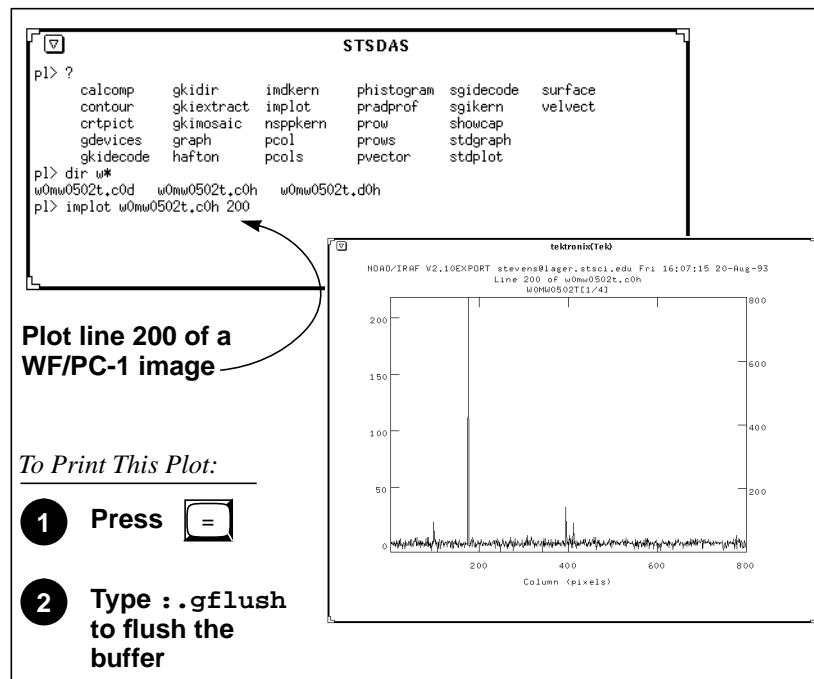
Keystroke	Command
	Display on-line help.
	Plot a line.
	Plot a column.
	Quit implot.
	Move down.
	Move up.
	Display coordinates and pixel values.

Figure 3.4: Plotting Image Data with implot

imexamine

The IRAF **imexamine** task (in the **images.tv** package) is a powerful task that integrates image display with various types of plotting capabilities. Commands can be passed to the task using the image display cursor and the graphics cursor. A complete description of the task and its usage are provided in the online help, available from within the IRAF environment by typing `help imexamine`.

Table 3.3: Image Manipulation Tasks

Task	Package	Purpose
boxcar	images.imfilter	Boxcar smooth a list of images
gcombine	stsdas.toolbox.imgtools	Combine images using various algorithms and rejection schemes
gcopy	stsdas.toolbox.imgtools	Copy GEIS multigroup images
geomap	images.immatch	Compute a coordinate transformation
geotran	images.immatch	Resample an image based on geomap output
grlist	stsdas.graphics.stplot	List of file names of all groups of a GEIS image (to make @lists)
gstatistics	stsdas.toolbox.imgtools	Compute image statistics ^a
imcalc	stsdas.toolbox.imgtools	Perform general arithmetic on GEIS images ^a
imedit	images.tv	Fill in regions of an image by interpolation
imexamine	images.tv	Examine images using display, plots, and text (see page 3-11)
implot	plot	Plot lines and columns of images (see page 3-10)
magnify	images.imgeom	Magnify an image
msarith	stsdas.toolbox.mstools	Performs basic arithmetic on STIS and NICMOS imsets
mscombine	stsdas.toolbox.mstools	Extension of gcombine for STIS and NICMOS imsets
msstatistics	stsdas.toolbox.mstools	Extension of gstatistics for STIS and NICMOS imsets
newcont	stsdas.graphics.stplot	Draw contours of two-dimensional data
pixcoord	stsdas.hst_calib.wfpc	Compute pixel coordinates of stars in a GEIS image
plcreate	xray.ximages	Create a pixel list from a region file (e.g., from SAOimage)
rotate	images.imgeom	Rotate an image
saodump	stsdas.graphics.sdisplay	Make image and colormap files from SAOimage display
siaper	stsdas.graphics.stplot	Plot science instrument apertures of HST

a. Will process all groups of a multigroup GEIS file.

3.3.3 Working with STIS and NICMOS Imsets

STIS and NICMOS data files contain groups of images, called imsets, associated with each individual exposure. A STIS imset comprises SCI, ERR, and DQ images, which hold science, error, and data quality information. A NICMOS imset, in addition to its SCI, ERR, and DQ images, also contains TIME and SAMP images recording the integration time and number of samples corresponding to each pixel of the SCI image. See the STIS and NICMOS Data Structures chapters for more details on imsets.

Here we describe several new STSDAS tasks, located in the `stsdas.toolbox.imgtools.mstools` package, that have been designed to help you work with imsets as units and to deconstruct and rebuild them.

msarith

This tool is an extension of the IRAF task `imarith` to include error and data quality propagation. The `msarith` task supports the four basic arithmetic operations (+, -, *, /) and can operate on individual or multiple imsets. The input operands can be either files or numerical constants; the latter can appear with an associated error, which will propagate into the error array(s) of the output file. Table 3.4 below shows how this task operates on the SCI, ERR, and DQ images in a STIS or NICMOS imset, as well as the additional TIME and SAMP images belonging to NICMOS imsets:

Table 3.4: Task `msarith` Operations

Operation	Operand2	SCI	ERR	DQ	TIME	SAMP
ADD	file	op1+op2	$\sqrt{\sigma_1^2 + \sigma_2^2}$	OR	T1+T2	S1+S2
SUB	file	op1-op2	$\sqrt{\sigma_1^2 + \sigma_2^2}$	OR	T1	S1
MULT	file	op1*op2	$(op1 \times op2) \sqrt{(\sigma_1/op1)^2 + (\sigma_2/op2)^2}$	OR	T1	S1
DIV	file	op1/op2	$(op1/op2) \sqrt{(\sigma_1/op1)^2 + (\sigma_2/op2)^2}$	OR	T1	S1
ADD	constant	op1+op2	$\sqrt{\sigma_1^2 + \sigma_2^2}$
SUB	constant	op1-op2	$\sqrt{\sigma_1^2 + \sigma_2^2}$
MULT	constant	op1*op2	$(op1 \times op2) \sqrt{(\sigma_1/op1)^2 + (\sigma_2/op2)^2}$...	T1*op2	...
DIV	constant	op1/op2	$(op1/op2) \sqrt{(\sigma_1/op1)^2 + (\sigma_2/op2)^2}$...	T1*op2	...

In Table 3.4 the first operand (op1) is always a file, and the second operand (op2) can be either a constant or a file. The ERR arrays of the input files (σ_1 and σ_2) are added in quadrature. If the constant is given with an error (σ_2), the latter is added in quadrature to the input ERR array. Note that in Table 3.4 the pixels in the SCI images are in counts, but `msarith` can also operate on count rates.

mscombine

This task allows you to run the STSDAS task `gcombine` on STIS and NICMOS data files. It divides each imset into its basic components (SCI, ERR, and DQ, plus SAMP and TIME for NICMOS) to make them digestible for `gcombine`. The SCI extensions become the inputs proper to the underlying `gcombine` task, and the ERR extensions become the error maps. The DQ extensions are first combined with a user-specified Boolean mask allowing selective pixel masking and then fed into the data quality maps. If scaling by exposure time is requested, the exposure times of each imset are read from the header keyword PIXVALUE in the TIME extensions.

Once `gcombine` finishes, `mscombine` reassembles the individual output images into imsets and outputs them as a STIS or NICMOS data file. The output images and error maps from `gcombine` form the SCI and ERR extensions of the

output imset. The DQ extension will be a combination of the masking operations and the rejection algorithms executed by **gcombine**. For NICMOS, the TIME extension will be the sum of the TIME values from the input files minus the rejected values, divided on a pixel-by-pixel basis by the number of valid pixels in the output image. The final TIME array will be consistent with the output SCI image (average or median of the science data). The SAMP extension for NICMOS is built from all the input SAMP values, minus the values discarded by masking or rejection.

msstatistics

This tool is an extension of **gstatistics** in the STSDAS package, which is in turn an extension of **imstatistics**. The main novelty is the inclusion of the error and data quality information included with STIS and NICMOS images in computing statistical quantities. In addition to the standard statistical quantities (min, max, sum, mean, standard deviation, median, mode, skewness, kurtosis), two additional quantities have been added to take advantage of the error information: the weighted mean and the weighted variance of the pixel distribution. If x_i is the value at the i -th pixel, with associated error σ_i , the weighted mean and variance used in the task are:

$$\langle x \rangle_w = \frac{\sum_i \frac{x_i}{\sigma_i \times \sigma_i}}{\sum_i \frac{1}{\sigma_i \times \sigma_i}}$$

and:

$$\langle \sigma \rangle_w^2 = \frac{1}{\sum_i \frac{1}{\sigma_i \times \sigma_i}}$$

The data quality information carried by the STIS or NICMOS file is used to reject pixels in the statistical computation. Users can supply additional masks to reject objects or regions from the science arrays.

mssplit and msjoin

The **mssplit** task extracts user-specified imsets from a STIS or NICMOS data file and copies them into separate files. Each output file contains a single imset along with the primary header of the original file. You might find this task useful for reducing the size of a STIS or NICMOS file containing many imsets or for performing analysis on a specific imset. The **msjoin** task inverts the operation of **mssplit**: it assembles separate imsets into a single data file.

There are additional tasks in this package for deleting and sorting imsets, as well as tasks for addressing a specific image class within an imset.

3.3.4 Photometry

Included in this section are:

- A list of IRAF/STSDAS tasks useful for determining source counts.
- Instructions on how to use header keyword information to convert HST counts to fluxes or magnitudes.
- A brief description of **synphot**, the STSDAS synthetic photometry package.

IRAF and STSDAS Photometry Tasks

The following are some useful IRAF/STSDAS packages and tasks for performing photometry on HST images:

- **apphot**: aperture photometry package.
- **daophot**: stellar photometry package useful for crowded fields.
- **isophote**: package for fitting elliptical isophotes.
- **imexamine**: performs simple photometry measurements.
- **imstat**: computes image pixel statistics.
- **imcnts**: sums counts over a specified region, subtracting background.
- **plcreate**: creates pixel masks.

Consult the online help for more details on these tasks and packages. The document “Photometry using IRAF” by Lisa A. Wells, provides a general guide to performing photometry with IRAF; it is available through the IRAF web page:

<http://iraf.noao.edu/>



The **apphot** package allows you to measure fluxes within a series of concentric apertures. This technique can be used to determine the flux in the wings of the PSF, which is useful if you wish to estimate the flux of a saturated star by scaling the flux in the wings of the PSF to an unsaturated PSF.

Converting Counts to Flux or Magnitude

All calibrated HST images record signal in units of counts or Data Numbers (DN)²—NICMOS data is DN s⁻¹. The pipeline calibration tasks do not alter the units of the pixels in the image. Instead they calculate and write the inverse sensitivity conversion factor (PHOTFLAM) and the ST magnitude scale zero point (PHOTZPT) into header keywords in the calibrated data. WF/PC-1 and WFPC2 observers should note that the four chips are calibrated individually, so these photometry keywords belong to the group parameters for each chip.

2. Except for 2-D rectified STIS images, which are in units of I_λ (see Chapter 23).

For all instruments other than NICMOS, PHOTFLAM is defined to be the *mean* flux density F_λ in units of $\text{erg cm}^{-2} \text{s}^{-1} \text{\AA}^{-1}$ that produces 1 count per second in the HST observing mode (PHOTMODE) used for the observation. If the F_λ spectrum of your source is significantly sloped across the bandpass or contains prominent features, such as strong emission lines, you may wish to recalculate the inverse sensitivity using **synphot**, described below. WF/PC-1 observers should note that the PHOTFLAM value calculated during pipeline processing does not include a correction for temporal variations in throughput owing to contamination buildup. Likewise, FOC observers should note that PHOTFLAM values determined by the pipeline before May 18, 1994 do not account for sensitivity differences in formats other than 512 x 512 (see “Format-Dependent Sensitivity” on page 7-10).

To convert from counts or DN to flux in units of $\text{erg cm}^{-2} \text{s}^{-1} \text{\AA}^{-1}$, multiply the total number of counts by the value of the PHOTFLAM header keyword and divide by the value of the EXPTIME keyword (exposure time). You can use the STSDAS task **imcalc** to convert an entire image from counts to flux units. For example, to create a flux-calibrated output image `outimg.fits` from an input image `inimg.fits[1]` with header keywords PHOTFLAM = 2.5E-18 and EXPTIME = 1000.0, you could type:

```
st> imcalc inimg.fits[1] outimg.fits "im1*2.5E-18/1000.0"
```

Calibrated NICMOS data are in units of DN s^{-1} , so the PHOTFLAM values in their headers are in units of $\text{erg cm}^{-2} \text{\AA}^{-1}$. You can simply multiply these images by the value of PHOTFLAM to obtain fluxes in units of $\text{erg cm}^{-2} \text{s}^{-1} \text{\AA}^{-1}$. NICMOS headers also contain the keyword PHOTFNU in units of Jy s. Multiplying your image by the PHOTFNU value will therefore yield fluxes in Janskys.



If your HST image contains a source whose flux you know from ground based measurements, you may choose to determine the final photometry of your HST image from the counts observed for this source.

To convert a measured flux F , in units of $\text{erg cm}^{-2} \text{s}^{-1} \text{\AA}^{-1}$, to an ST magnitude, plug it into the following equation:

$$m = -2.5 \times \log_{10}(F) + \text{PHOTZPT}$$

where the value of the PHOTZPT keyword is the zero point of the ST magnitude scale. The zero point of the ST magnitude system has always been and probably always will be equal to -21.10 , a value chosen so that Vega has an ST magnitude of zero for the Johnson V passband (see Koornneef et al., 1986; Horne, 1988; and the *Synphot Users Guide*).

synphot

The STSDAS synthetic photometry package, called **synphot**, can simulate HST observations of astronomical targets with known spectra. It contains throughput curves for the components of all HST instruments, such as mirrors, filters, gratings, apertures, and detectors, and can generate passband shapes for

any combination of these elements. It can also generate synthetic spectra of many different types, including stellar, blackbody, power-law and H II region spectra, and can convolve these spectra with the throughputs of HST's instruments. You can therefore use it to compare results in many different bands, to cross-calibrate one instrument with another, or to relate your observations to theoretical models.

One useful application of **synphot** is to recalculate the value of PHOTFLAM for a given observation using the latest calibration files. For example, to recalculate PHOTFLAM for an FOC observation, you could use the **calcphot** task in **synphot** as follows:

```
sy> calcphot foc,f/96,x96z1rg,f501n 'unit(1,flam)' counts
```

The first argument to **calcphot** gives the instrument and its configuration, in this case the FOC *f/96* camera in full zoomed format with the F501 filter. (See the **obsmode** task in **synphot** and the *Synphot User's Guide* for help with these observation-mode keywords.) The second tells the task to model a flat F_λ spectrum having unit flux, and the third tells the task to produce output in units of counts per second. After you run **calcphot**, its `result` parameter will contain the count rate expected from the FOC, given this configuration and spectrum. The PHOTFLAM keyword, defined to be the flux required to produce one count per second, simply equals the reciprocal of this value, which you can print to the screen by typing `=1./calcphot.result` at the IRAF prompt.

Please see the *Synphot User's Guide* for more details on this multipurpose package, and see Appendix A for information on getting the **synphot** dataset, which is not included with STSDAS.

Information about retrieving the **synphot** dataset can be found in Appendix A.

3.4 Displaying HST Spectra

This section shows how to plot your HST spectra for a quick first look and how to generate hardcopies of your plots. Because the STIS data format differs from that of FOS and GHRS, we will discuss STIS data separately.

3.4.1 FOS and GHRS Spectra

Before you work with FOS and GHRS data within STSDAS, you will want to convert the FITS files you received from the Archive into GEIS format (see “Converting FITS to GEIS” on page 2-11 for instructions). After conversion, the `.c1h` file will hold the calibrated flux values for each pixel, the `.c0h` file will hold the corresponding wavelengths, and the `.c2h` file will hold the propagated statistical errors.

Each group of an FOS or GHRS GEIS file contains the results of a separate subintegration. FOS readouts taken in ACCUM mode are cumulative, so the last group contains the results of the entire integration. In contrast, GHRS readouts and FOS readouts in RAPID mode are independent. If you want to see the results

of an entire GHRS FP-SPLIT integration, you will need to align and coadd the spectra in the groups of the GHRS file as described in Volume 2 of this handbook. You can also combine all the groups in an FOS or GHRS data file, without wavelength alignment, using the **rcombine** task in the **hst_calib.ctools** package. See online help for details.

The STSDAS task **sgraph** (in the **graphics.stplot** package) can plot the contents of a single GEIS group. For example, if you want to see group 19 of the calibrated FOS spectrum with rootname `y3b10104t` you can type

```
st> sgraph y3b10104t.c1h[19]
```

Given an input flux image (`.c1h`), the task **fwplot** (in the **hst_calib.ctools** package) will look for the corresponding wavelength (`.c0h`) file and plot flux versus wavelength. If requested, it will also look for the error (`.c2h`) file and plot the error bars. To see a plot of the same spectrum as above, but with a wavelength scale and error bars, type

```
st> fwplot y3b10104t.c1h[19] plterr+
```

If you ever need to plot the contents of multiple groups offset from one another on the same graph, you can use the **grspec** task in the **graphics.stplot** package. For example, to plot groups 1, 10, and 19 of a given flux file, you can type

```
st> grspec y3b10104t.c1h 1,10,19
```

Note that **grspec** expects group numbers to be listed as a separate parameter, rather than enclosed in the standard square brackets.

3.4.2 STIS Spectra

STIS data files retrieved from the Archive can contain spectra in two different forms: as long-slit spectral images in FITS IMAGE extensions or as extracted echelle spectra in FITS BINTABLE extensions. Currently only echelle spectra emerge from the pipeline in tabular form, while long-slit spectra emerge as images.

You can use **sgraph** to plot STIS long-slit spectra by specifying the image section that contains the spectrum. For example, to plot the entire x range of the calibrated two-dimensional spectrum in the first extension of the file `o43ba1bnm_x2d.fits`, averaging rows 100 through 1000, you would type

```
st> sgraph o43ba1bnm_x2d.fits[1][*,100:1000]
```

Displaying the long-slit spectral image using the **display** task (see page 3-4) will allow you to see the range of your spectrum in x and y pixel space, so you can choose a suitable image section for plotting.

To plot STIS spectra in BINTABLE extensions, you first need to understand how STIS spectra are stored as binary arrays in FITS table cells. Chapter 2 (page 2-9) discusses this format and describes the *selectors* syntax used to specify these data arrays. Each row of a STIS echelle table contains a separate spectral order, and each column contains data of a certain type, such as WAVELENGTH data or FLUX data. To specify a particular array, you must first type the file name, then the extension containing the BINTABLE, then the column selector, then the

row selector. For example, to select the WAVELENGTH array corresponding to spectral order 80 of the echelle spectrum in extension 4 of `stis.fits`, you would specify the file as:

```
stis.fits[4][c:WAVELENGTH][r:sporder=80]
```

The **sgraph** task and the **igi** plotting package to be discussed below both understand the *selectors* syntax. In particular, if you wanted to plot the flux versus wavelength in STIS echelle order 80, you could type:

```
st> sgraph "stis.fits[4][r:sporder=80] WAVELENGTH FLUX"
```

Remember to include the quotation marks. Otherwise, **sgraph** will complain about too many positional arguments. Note also that **sgraph** understands only row selector syntax; columns are chosen by name.

The STIS-specific **echplot** task is particularly useful for browsing STIS echelle spectra. It can plot single spectral orders, overplot multiple orders on a single plot, or plot up to four orders in separate panels on the same page. For example, to overplot the orders contained in rows two through four and row six on a single page:

```
cl> echplot "stis_x1d.fits[1][r:row=(2:4,6)]" output.igi \
>>> plot_style=m
```


Note that the `plot_style` parameter governs how the spectral orders are plotted. The `plot_style` values `s`, `m`, and `p` plot one order per page, several orders on a single plot, and one order per panel, respectively. The default brightness unit is calibrated FLUX, although you can specify other quantities (e.g., NET counts) using the `flux_col` parameter. See the online help for details.

3.4.3 Producing Hardcopy

This section shows how to generate hardcopies of plots directly and describes **igi**, the Interactive Graphics Interpreter available in STSDAS.

Direct Hardcopies

To print a quick copy of the displayed plot:

1. Type `=gcur` in the command window (where your CL prompt is located).
2. Move the cursor to any location in the graphics window.
3. Press  to write the plot to the graphics buffer.
4. Type `q` to exit graphics mode.
5. At the `cl` prompt, type `gflush`.



Plots will be printed on the printer defined by the IRAF environment variable `stdplot`. Type `show stdplot` to see the current default printer; use `set stdplot = printer_name` to set the default printer.

The PostScript kernel **psikern** allows you to create PostScript files of your IRAF/STSDAS plots. For example, setting the `device` parameter in a plotting task equal to `psi_port` or `psi_land` invokes **psikern** and directs your plot to either a portrait-mode or a landscape mode PostScript file. For example:

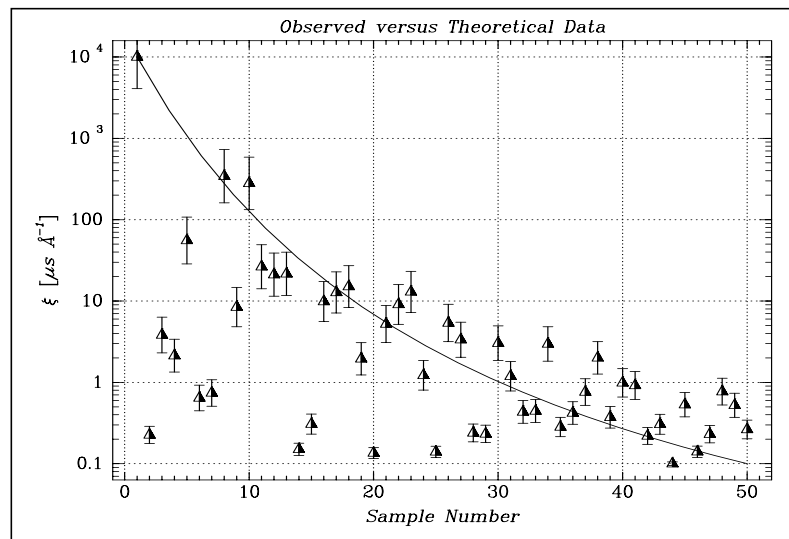
```
st> fwplot y3b10104t.c1h[19] device=psi_land
st> gflush
/tmp/pskxxxx
```

The above commands would write a plot of flux vs. wavelength in landscape-mode into a temporary PostScript file, named `/tmp/pskxxxx` by a UNIX system. See the online help for more about **psikern**, including plotting in color and incorporating PostScript fonts into your plots.

igi

As your plotting needs grow more sophisticated—and especially as you try preparing presentations or publication-quality plots—you should investigate the Interactive Graphics Interpreter, or **igi**. This task, in the STSDAS **stplot** package, can be used with images as well as two- and three-dimensional tables and can draw axes, error bars, labels, and a variety of other features on plots. Different line weights, font styles, and feature shapes are available, enabling you to create complex plots. Figure 3.5 shows a sample plot created in **igi**, however, because **igi** is a complete graphics environment in itself, it is well beyond the scope of this document. You can learn more about **igi** in the *IGI Reference Manual*, available through the STSDAS Web pages

Figure 3.5: Sample igi Plot.



3.5 Analyzing HST Spectra

This section describes some IRAF/STSDAS tasks that can be used for analyzing and manipulating spectral data. Certain of these tasks operate directly on HST data files created by the pipeline. However, a number of the most useful IRAF tasks, such as **splot**, require special preparations of data other than STIS two-dimensional spectra. Before discussing these tasks we will first show how to recast your data into forms that are more generally accessible.

3.5.1 Preparing FOS and GHRS Data

The FOS and GHRS data reduction pipelines store fluxes and wavelengths in separate files. In GEIS format, the `.c1h` file contains the flux information and the `.c0h` file contains the wavelength information. Because IRAF tasks generally require both the flux and wavelength information to reside in the same file, you will probably want to create a new file that combines these quantities.

Several options for combining flux and wavelength information are available:

- **resample**: This simple task resamples your flux data onto a linear wavelength scale, creating a new flux file containing the starting wavelength of the new grid in the `CRVAL1` keyword and the wavelength increment per pixel in the `CD1_1` keyword. Encoding the wavelength information into these standard FITS header keywords makes this format quite portable, but the resampling process loses some of the original flux information. In addition, the error (`.c2h`) and data quality (`.cqh`) files cannot be similarly resampled, limiting the usefulness of this technique.
- **mkmultispec**: This task writes wavelength information into the header of a flux file while preserving all the original information. It is therefore a better choice than **resample** for most applications, and we describe it in more detail below.
- **imtab**: An alternative to writing wavelength information into the header is to use the **imtab** task to create a table recording the wavelength, flux, and if desired, the error data corresponding to each pixel. Many STSDAS tasks, such as those in the STSDAS **fitting** package, can access data in tabular form, so we describe this approach in more detail as well.

mkmultispec

The most convenient method of combining wavelength and flux information, and one that has no effect on the flux data at all, is to use the **mkmultispec** task. This task places wavelength information into the headers of your flux files according to the IRAF multispec-format World Coordinate System (WCS). The multispec coordinate system is intended to be used with spectra having nonlinear dispersions or with images containing multiple spectra, and the format is recognized by many tasks in IRAF V2.10 or later. For a detailed discussion of the multispec WCS, type `help specwcs` at the IRAF prompt.

The **mkmultispec** task can put wavelength information into the flux header files in two different ways. The first involves reading the wavelength data from the .c0h file, fitting the wavelength array with a polynomial function, and then storing the derived function coefficients in the flux header file (.c1h) in multispec format. Legendre, Chebyshev, or cubic spline (spline3) fitting functions of fourth order or larger produce essentially identical results, all having rms residuals less than 10^{-4} Å, much smaller than the uncertainty of the original wavelength information. Because these fits are so accurate, it is usually unnecessary to run the task in interactive mode to examine them.



If there are discontinuities in the wavelengths, which could arise due to the splicing of different gratings, you should run **mkmultispec** in interactive mode to verify the fits.



Because **mkmultispec** can fit only simple types of polynomial functions to wavelength data, this method will *not* work well with FOS prism data, because of the different functional form of the prism-mode dispersion solution. For prism spectra, use the header table mode of **mkmultispec** (see below) or create an STSDAS table using **imtab**.

The other method by which **mkmultispec** can incorporate wavelength information into a flux file is simply to read the wavelength data from the .c0h file and place the entire data array directly into the header of the flux (.c1h) file. This method simply dumps the wavelength value associated with each pixel in the spectrum into the flux header and is selected by setting the parameter `function=table`. To minimize header size, set the parameter `format` to a suitable value. For example, using `format=%8.7g` will retain the original seven digits of precision of the wavelength values, while not consuming too much space in the flux header file.



Be aware that there is a physical limit to the number of header lines that can be used to store the wavelength array (approximately 1000 lines). This limit cannot be overridden. Under ordinary circumstances this limitation is not an issue. However, if many spectral orders have been spliced together, it may not be possible to store the actual wavelength array in the header, and a fit must be done instead.

imtab

Another way to combine wavelengths with fluxes is to create an STSDAS table from your spectrum. The **imtab** task in the STSDAS **ttools** package reads a GEIS format spectral image and writes the list of data values to a column of an STSDAS table, creating a new output table if necessary. The following example shows how

to create a flux, wavelength, and error table from group eight of a GEIS-format FOS dataset:

```
cl> imtab y0cy0108t.c0h[8] y0cy0108t.tab wavelength
cl> imtab y0cy0108t.c1h[8] y0cy0108t.tab flux
cl> imtab y0cy0108t.c2h[8] y0cy0108t.tab error
```

The last word on each command line labels the three columns “wavelength”, “flux”, and “error”.

Constructing tables is a necessary skill if you plan to use certain tasks—such as those in the STSDAS **fitting** package—that do not currently recognize the multispec format WCS header information. Tabulating your spectra is also the best option if you want to join two or more spectra taken with different gratings into a single spectrum covering the complete wavelength range. Because the data are stored as individual wavelength-flux pairs, you do not need to resample, and therefore degrade, the individual spectra to a common, linear dispersion scale before joining them. Instead, you could create separate tables for the spectra from different gratings, and then combine the two tables using, for example, the **tmerge** task:

```
cl> tmerge n5548_h13.tab,n5548_h19.tab n5548.tab append
```

Note that you will first have to edit out any regions of overlapping wavelength from one or the other of the input tables so that the output table will be monotonically increasing (or decreasing) in wavelength.

3.5.2 Preparing STIS Spectra for Analysis

Calibrated STIS spectra emerge from the pipeline either as two-dimensional images (`_x2d` files) or as one-dimensional spectra in tabular form (`_x1d` files.) You can analyze calibrated two-dimensional STIS spectra in IRAF as you would any other long-slit spectral image, because their headers already contain the necessary wavelength information. Tabulated STIS spectra can be analyzed directly using STSDAS tasks that understand the *selectors* syntax described on page 2-9. However, to use IRAF tasks, such as **splot**, that rely on the multispec WCS or to use STSDAS tasks that do not understand three-dimensional tables, you will have to prepare your data appropriately. This section describes two useful tasks for putting your data in the proper form:

- **tomultispec**: This task is the STIS analog to **mkmultispec**, described above; it extracts STIS spectra from tables and writes them as IRAF spectral images with wavelength information in the header.
- **txtable**: This task extracts specified data arrays from STIS table cells and places them in conventional two-dimensional tables for easier access.
- **tximage**: Extracts specified data arrays from STIS table cells and places them into 1-D images. This task can write single group GEIS files.

tomultispec

The **tomultispec** task in the `stdas.hst_calib.ctools` package extracts one or more spectral orders from a STIS table, fits a polynomial dispersion solution to

each wavelength array, and stores the spectra in an output file in original IRAF format (OIF), using the multispec WCS. This task is layered upon the **mkmultispec** task, which performs a similar operation for FOS and GHRS calibrated spectra (see page 3-21). Most of the parameters for **tomultispec** echo those for **mkmultispec**. As a helpful navigational aid, the STIS spectral order numbers are written to the corresponding *beam* numbers in the multispec image; the aperture numbers are indexed sequentially starting from one. You can choose to fit the dispersion solution interactively, but the default fourth-order Chebyshev polynomial will likely suffice for all STIS spectral orders, except for prism-dispersed spectra. However, you cannot use the interactive option if you are selecting more than one order from the input file.

For example, if you want to write all spectral orders from the STIS file `myfile_x1d.fits` to a multispec file, you can type:

```
cl> tomultispec myfile_x1d.fits new_ms.imh
```

Note that the `.imh` suffix on the output file specifies that the output file is to be an OIF file. This format is similar to GEIS format, in that it consists of two files: a header file (`.imh`) and a binary data file (`.pix`). The output format for **tomultispec** will always be OIF.

If you want to select particular spectral orders, rather than writing all the orders to the multispec file, you will need to use the **selectors** syntax. To select only the spectrum stored in row nine of the input table, the previous example would change to:

```
cl> tomultispec "myfile_x1d.fits[r:row=9]" new_ms.imh
```

Note that the double quote marks around the file name and row selector are necessary to avoid syntax errors. To select a range of rows, say rows nine through eleven, you would type:

```
cl> tomultispec "myfile_x1d.fits[r:row=(9:11)]" new_ms.imh
```

You can also select rows based upon values in some other column. For example, to select all rows whose spectral order lies in the range 270 to 272, type:

```
cl> tomultispec "myfile_x1d.fits[r:sporder=(270:272)]" \
>>> new_ms.imh
```

The calibrated flux is extracted by default. However, other intensity data can be specified by setting the `flux_col` parameter.



Be careful not to restrict the search for matching rows too heavily.



Column selectors cannot be used with **tomultispec**.



Choose the type of fitting function for the **tomultispec** dispersion solution with care. Using the `table` option, which writes the entire wavelength array to the image header for each order, will fail if more than about three orders are selected. This restriction results from a limit to the number of keywords that can be used to store the dispersion relation.

txtable

Tabulated STIS spectra are stored as data arrays within individual cells of FITS binary tables (see Chapter 2, page 2-9). These tables are effectively three-dimensional, with each column holding a particular type of quantity (e.g., wavelengths, fluxes), each row holding a different spectral order, and each cell holding a one-dimensional array of values spanning the wavelength space of the order. The **txtable** in the **tables.ttools** package extracts these data arrays from the cells specified with the selectors syntax and stores them in the columns of conventional two-dimensional binary tables.

For example, suppose the first extension of the FITS file `data.fits` contains a STIS echelle spectrum and you want to extract only the wavelength and flux arrays corresponding to spectral order 68. You could then type:

```
tt> txtable "data.fits[1][c:WAVELENGTH,FLUX][r:sorder=68]" \
>>> out_table
```

This command would write the wavelength and flux arrays to the columns of the output table `out_table`. To specify multiple rows in a tabulated echelle spectrum, you would type:

```
tt> txtable "data.fits[1][c:WAVELENGTH,FLUX][r:row=(10:12)]" \
>>> echl
```

This command would generate three separate output files named `echl_r0010.tab`, `echl_r0011.tab`, and `echl_r0012.tab`.

See the online help for more details on **txtable** and the selectors syntax, and remember to include the double quotation marks.

The similar **tximage** task can be used to generate single-group GEIS files from STIS data, which can then be used as input to tasks such as **resample**.

```
tt> tximage "data.fits[1][c:WAVELENGTH][r:row=4]" wave.hhh
tt> tximage "data.fits[1][c:FLUX][r:row=4]" flux.hhh
```

3.5.3 General Tasks for Spectra

IRAF has many tasks for analyzing both one- and two-dimensional spectral data. Many observers will already be familiar with **noao.onedspec** and **noao.twodspect** packages, and those who are not should consult the online help. Table 3.5 lists some of the more commonly used IRAF/STSDAS spectral analysis tasks, and below we briefly describe **splot**, one of the most versatile and useful. Remember that many of these tasks expect to find WCS wavelength information

in the header, so you should first run **mkmultispec** or **tomultispec** on your data, if necessary.

Table 3.5: Tasks for Working with Spectra

Task	Package	Input Format	Purpose
boxcar	images.imfilter	Image	Boxcar smooth a list of images
bplot	noao.onedspec	Multispec image	Plot spectra non-interactively
continuum	noao.onedspec	Image	Continuum normalize spectra
fitprofs	noao.onedspec	Image	Non-interactive Gaussian profile fitting to features in spectra and image lines
gcopy	stsdas.toolbox.imgtools	GEIS image	Copy multigroup images
grlist	stsdas.graphics.stplot	GEIS image	List file names for all groups in a GEIS image; used to make lists for tasks that do not use group syntax
grplot	stsdas.graphics.stplot	GEIS image	Plot arbitrary lines from 1-D image; overplots multiple GEIS groups; no error or wavelength information is used
grspec	stsdas.graphics.stplot	Multispec GEIS image	Plot arbitrary lines from 1-D image; stack GEIS groups
magnify	images.imgeom	Image	Interpolate spectrum on finer (or coarser) pixel scale
nfit1d	stsdas.analysis.fitting	Image, table	Interactive 1-D non-linear curve fitting (see page 3-29)
ngaussfit	stsdas.analysis.fitting	Image, table	Interactive 1-D multiple Gaussian fitting (see page 3-29)
poffsets	stsdas.hst_calib.ctools	GEIS image	Determine pixel offsets between shifted spectra
rapidlook	stsdas.hst_calib.ctools	GEIS image	Create and display a 2-D image of stacked 1-D images
rcombine	stsdas.hst_calib.ctools	GEIS image	Combine (sum or average) GEIS groups in a 1-D image with option of propagating errors and data quality values
resample	stsdas.hst_calib.ctools	GEIS image	Resample FOS and GHRS data to a linear wavelength scale (see page 3-21)
sarith	noao.onedspec	Multispec image	Spectrum arithmetic
scombine	noao.onedspec	Multispec image	Combine spectra
sfit	noao.onedspec	Multispec image	Fit spectra with polynomial function
sgraph	stsdas.graphics.stplot	Image, table	Plot spectra and image lines; allows overplotting of error bars and access to wavelength array (see page 3-17)
spealign	stsdas.hst_calib.ctools	GEIS image	Align and combine shifted spectra (see poffsets)
specplot	noao.onedspec	Multispec image	Stack and plot multiple spectra
splot	noao.onedspec	Multispec image	Plot and analyze spectra & image lines (see page 3-27)

splot

The **splot** task in the IRAF **noao.onedspec** package is a good general analysis tool that can be used to examine, smooth, fit, and perform simple arithmetic operations on spectra. Because it looks in the header for WCS wavelength information, your file must be suitably prepared. Like all IRAF tasks, **splot** can work on only one group at a time from a multigroup GEIS file. You can specify which GEIS group you want to operate on by using the square bracket notation, for example:

```
cl> splot y0cy0108t.c1h[8]
```

If you don't specify a group in brackets, **splot** will assume you want the first group. In order to use **splot** to analyze your FOS or GHRS spectrum, you will first need to write the wavelength information from your .c0h file to the header of your .c1h files in WCS, using the **mkmultispec** task (see page 3-21).

The **splot** task is complex with *many* available options described in detail in the online help. Table 3.6 summarizes a few of the more useful cursor commands for quick reference. When you are using **splot**, a log file saves results produced by the equivalent width or de-blending functions. To specify a file name for this log file, you can set the `save_file` parameter by typing, for example:

```
cl> splot y0cy0108t.c1h[8] save_file=results.log
```

If you have used **tomultispec** to transform a STIS echelle spectrum into .imh/.pix OIF files with WCS wavelength information (see page 3-23), you can step through the spectral orders stored in image lines using the “)”, “(”, and “#” keys. To start with the first entry in your OIF file, type:

```
cl> splot new_ms.imh 1
```

You can then switch to any order for analysis using the “)” key to increment the line number, the “(” key to decrement, and the “#” key to switch to a specified image line. Note the beam label that gives the spectral order cannot be used for navigation. See the online help for details.

Table 3.6: Useful plot Cursor Commands

Command	Purpose
<i>Manipulating spectra</i>	
f	Arithmetic mode; add and subtract spectra
l	Convert spectrum from f_ν to f_λ (invert transformation with “n”)
n	Convert spectrum from f_λ to f_ν
s	Smooth with a boxcar
u	Define linear wavelength scale using two cursor markings
<i>Fitting spectra</i>	
d	Mark two continuum points & de-blend multiple Gaussian line profiles
e	Measure equivalent width by marking points around target line
h	Measure equivalent width assuming Gaussian profile
k	Mark two continuum points and fit a single Gaussian line profile
m	Compute the mean, RMS, and S/N over marked region
t	Enter interactive curve fit function (usually used for continuum fitting)
<i>Displaying and redrawing spectra</i>	
a	Expand and autoscale data range between cursor positions
b	Set plot base level to zero
c	Clear all windowing and redraw full current spectrum
r	Redraw spectrum with current windowing
w	Window the graph
x	Etch-a-sketch mode; connects two cursor positions
y	Overplot standard star values from calibration file
z	Zoom graph by a factor of two in X direction
\$	Switch between physical pixel coordinates and world coordinates
<i>General file manipulation commands</i>	
?	Display help
g	Get another spectrum
i	Write current spectrum to new or existing image
q	Quit and go on to next input spectrum

3.5.4 STSDAS fitting Package

The STSDAS **fitting** package contains several powerful and flexible tasks, listed in Table 3.7, for fitting and analyzing spectra. The **ngaussfit** and **nfit1d** tasks, in particular, are very good for interactively fitting multiple Gaussians and nonlinear functions, respectively, to spectral data. These tasks do not currently recognize the multispec WCS method of storing wavelength information. They recognize the simple sets of dispersion keywords such as W0, WPC and CRPIX, CRVAL, and CDELTA, but these forms apply only to linear coordinate systems and therefore would require resampling of your data onto a linear wavelength scale before being used. However, these tasks do accept input from STSDAS tables, in which you can store the wavelength and flux data value pairs or wavelength, flux, error value triples (“imtab” on page 3-22).

Table 3.7: Tasks in the STSDAS fitting Package

Task	Purpose
convert	Convert ASCII data base format to STSDAS table format
function	Generate functions as images, tables, or lists
gfit1d	Interactive 1-d linear curve fit to images, tables, or lists
i2gaussfit	Iterative 2-d Gaussian fit to noisy images (script)
nfit1d	Interactive 1-d non-linear curve fit to images, tables, or lists
ngaussfit	Interactive 1-d multiple Gaussian fit to images, tables, or lists
n2gaussfit	2-d Gaussian fit to images
prfit	Print contents of fit tables created by fitting task

When using tasks such as **ngaussfit** and **nfit1d**, you must provide initial guesses for the function coefficients as input to the fitting algorithms. You can either specify these initial guesses via parameter settings in the task’s parameter sets (psets) or enter them interactively. For example, suppose you want to fit several features using the **ngaussfit** task. Using the default parameter settings you can start the task by typing:

```
fi> ngaussfit n4449.hhh linefits.tab
```

This command reads spectral data from the image `n4449.hhh` and stores the results of the line fits in the STSDAS table `linefits.tab`. After you start the task, your spectrum should appear in a plot window and the task will be left in cursor input mode. You can use the standard IRAF cursor mode commands to rewindow the plot, restricting your display to the region around a particular feature or features that you want to fit. You may then want to:

- Define a sample region (using the cursor mode **S** command) over which the fit will be computed so that the task will not try to fit the entire spectrum.

- Define an initial guess for the baseline coefficients by placing the cursor at two baseline locations (one on either side of the feature to be fitted) using the **B** keystroke.
- Use the **R** keystroke to redraw the screen and see the baseline that you've just defined.
- Set the initial guesses for the Gaussian centers and heights by placing the cursor at the peak of each feature and typing **P**.
- Press **F** to compute the fit once you've marked all the features you want to fit.

The results will automatically be displayed. You can use the `:show` command to see the coefficient values.

Note that when the **ngaussfit** task is used in this way (i.e., starting with all default values), the initial guess for the FWHM of the features will be set to a value of one. Furthermore, this coefficient and the coefficients defining the baseline are held fixed by default during the computation of the fit, unless you explicitly tell the task through cursor *colon* commands³ to allow these coefficients to vary. It is sometimes best to leave these coefficients fixed during an initial fit, and then to allow them to vary during a second iteration. This rule of thumb also applies to the setting of the `errors` parameter which controls whether or not the task will estimate error values for the derived coefficients. Because the process of error estimation is very CPU-intensive, it is most efficient to leave the error estimation turned off until you've got a good fit, and then turn the error estimation on for one last iteration.

Figure 3.6 shows the results of fitting the H β (4861Å) and [OIII] (4959 and 5007 Å) emission features in the spectrum of NGC 4449. The resulting coefficients and error estimates (in parentheses) are shown in Figure 3.7.

3. See the online help for details and a complete listing of cursor mode colon commands: type `help cursor`.

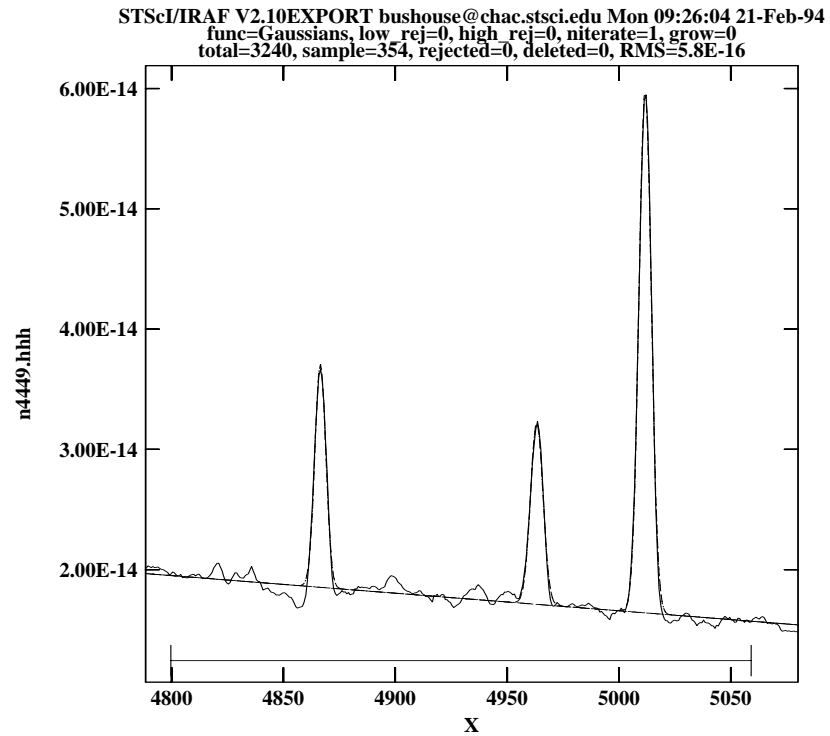
Figure 3.6: Fitting H β and [OIII] Emission Features in NGC 4449

Figure 3.7: Coefficients and Error Estimates

```

function = Gaussians
coeff1 = 8.838438E-14 (0.) - Baseline zeropoint (fix)
coeff2 = -1.435682E-17 (0.) - Baseline slope (fix)
coeff3 = 1.854658E-14 (2.513048E-16) - Feature 1: amplitude (var)
coeff4 = 4866.511 (0.03789007) - Feature 1: center (var)
coeff5 = 5.725897 (0.0905327) - Feature 1: FWHM (var)
coeff6 = 1.516265E-14 (2.740680E-16) - Feature 2: amplitude (var)
coeff7 = 4963.262 (0.06048062) - Feature 2: center (var)
coeff8 = 6.448922 (0.116878) - Feature 2: FWHM (var)
coeff9 = 4.350271E-14 (2.903318E-16) - Feature 3: amplitude (var)
coeff10 = 5011.731 (0.01856957) - Feature 3: center (var)
coeff11 = 6.415922 (0.03769293) - Feature 3: FWHM (var)
rms = 5.837914E-16
grow = 0.
naverage = 1
low_reject = 0.
high_reject = 0.
niterate = 1
sample = 4800.132:5061.308

```

3.5.5 **specfit**

The **specfit** task, in the STSDAS **contrib** package, is another powerful interactive facility for fitting a wide variety of emission-line, absorption-line, and continuum models to a spectrum. This task was written by Gerard Kriss at Johns Hopkins University. Extensive online help is available to guide you through the task,⁴ although because it is a contributed task, little to no support is provided by the STSDAS group.

The input spectrum to **specfit** can be either an IRAF image file or an ASCII file with a simple three-column (wavelength, flux, and error) format. If the input file is an IRAF image, the wavelength scale is set using values of **W0** and **WPC** or **CRVAL1** and **CDELTA1**. Hence, for image input, the spectral data must be on a linear wavelength scale. In order to retain data on a non-linear wavelength scale, it is necessary to provide the input spectrum in an ASCII file, so that you can explicitly specify the wavelength values associated with each data value. The online help explains a few pieces of additional information that must be included as header lines in an input text file.

By selecting a combination of functional forms for various components, you can fit complex spectra with multiple continuum components, blended emission and absorption lines, absorption edges, and extinction. Available functional forms include linear, power-law, broken power-law, blackbody, and optically thin recombination continua, various forms of Gaussian emission and absorption lines, absorption-edge models, Lorentzian line profiles, damped absorption-line profiles, and mean galactic extinction.

3.6 References

3.6.1 Available from STScI

- *STSDAS Users Guide*, version 1.3.3, September 1994.
- *STSDAS Installation and Site Managers Guide*, version 2.0, August 1997.
- *Synphot Users Guide*, August 1997.
- *IGI Reference Manual*, version 3.0, November 1997.

3.6.2 Available from NOAO

- *A Beginners Guide to Using IRAF*, 1994, J. Barnes.
- *User Manual for SAOimage*, 1991, M. Van Hilst.

4. Additional information is available in the *Astronomical Data Analysis Software and Systems III*, ASP Conference Series, Vol. 61, page 437, 1994.

- *Photometry Using IRAF*, 1994, L. Wells.
- *A User's Guide to Stellar CCD Photometry with IRAF*, 1992, P. Massey and L. Davis.

3.6.3 Other References Cited in This Chapter

- Horne, K., 1988, in *New Directions in Spectrophotometry*, A.G.D. Philip, D.S. Hayes, and S.J. Adelman, eds., L. Davis Press, Schenectady NY, p. 145.
- Koorneef, J., R. Bohlin, R. Buser, K. Horne, and D. Turnshek, 1986, in *Highlights of Astronomy*, Vol. 7, J.-P. Swinds, ed., Reidel, Dordrecht, p. 833.
- Kriss, G., 1994, in *Astronomical Data Analysis Software and Systems III*, PASP Conference Series, Vol. 61, p. 437.

