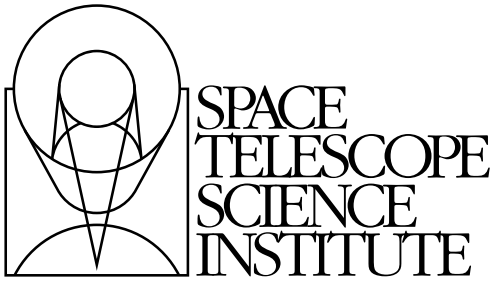
Version 8.0
May 2011

Introduction to the *HST* Data Handbooks



Space Telescope Science Institute
3700 San Martin Drive
Baltimore, Maryland 21218
help@stsci.edu

Operated by the Association of Universities for Research in Astronomy, Inc., for the National Aeronautics and Space Administration

User Support

For prompt answers to any question, please contact the STScI Help Desk.

- Send E-mail to: help@stsci.edu.
- Phone: 410-338-1082.
- Within the USA, you may call toll free: 1-800-544-8125.

World Wide Web

Information and other resources are available on the Web site:

- **URL:** `http://www.stsci.edu`

Revision History

Version	Date	Editors
8.0	May 2011	Ed Smith Technical Editors: Susan Rose & Jim Younger
7.0	January 2009	*HST* Introduction Editors: Brittany Shaw, Jessica Kim Quijano, & Mary Elizabeth Kaiser *HST* Introduction Technical Editors: Susan Rose & Jim Younger
6.0	January 2006	*HST* Introduction Editor: Diane Karakla *HST* Introduction Technical Editors: Susan Rose & Jim Younger.
5.0	July 2004	Diane Karakla and Jennifer Mack, Editors, *HST* Introduction. Susan Rose, Technical Editor, *HST* Introduction.
4.0	January 2002	Bahram Mobasher, Chief Editor, *HST* Data Handbook Michael Corbin, Jin-chung Hsu, Editors, *HST* Introduction
3.1	March 1998	Tony Keyes
3.0, Vol. II	October 1997	Tony Keyes
3.0, Vol. I	October 1997	Mark Voit
2.0	December 1995	Claus Leitherer
1.0	February 1994	Stefi Baum

Contributors

The editors are grateful for the careful reviews and revised material provided by Azalee Boestrom, Howard Bushouse, Stefano Casertano, Warren Hack, Diane Karakla and Matt Lallo.

Citation

In publications, refer to this document as:
Smith, E., et al. 2011, “Introduction to the *HST* Data Handbooks”, Version 8.0, (Baltimore: STScI)

Send comments or corrections to:
Space Telescope Science Institute
3700 San Martin Drive
Baltimore, Maryland 21218
E-mail:help@stsci.edu

Table of Contents

Preface

This *Introduction to the HST Data Handbooks* is intended to provide an introductory level understanding of topics general to the data from all of the Hubble Space Telescope (*HST*) instruments. It introduces methods for: retrieving data produced by observations by the Hubble from *HST* archives; understanding the data structure and where to find important information about each Hubble observation within the data files and "Observation Log" files; and software tools and methods for working with data of different types from different *HST* instruments.

Chapters 2 and 3 of this Introduction provide a stepping stone to understanding the in-depth descriptions of the *HST* Data provided in each instrument's individual Data Handbook. In Chapters 1, 4, 5 and 6 it introduces several important topics not covered in those handbooks. Throughout the document you will find internet links to reference materials that provide more thorough treatments and to others that provide useful information on additional related topics.

With this version, 8.0, the former Instrument Data Handbooks' Introduction: Chapters 1, 2 and 3; and Appendices: A, B and C; have been extracted into Chapters 1-6 respectively of this standalone document instead of having identical copies carried within each separate Instrument's Data Handbook.

Both PDF and HTML versions of the Data Handbooks for *HST*'s instruments:

- Advanced Camera for Surveys (ACS)
- Cosmic Origins Spectrograph (COS)
- Faint Object Camera (FOC)
- Fine Guidance Sensors (FGS)
- Faint Object Spectrograph (FOS)
- Goddard High Resolution Spectrograph (GHRS)
- High Speed Photometer (HSP)
- Near Infrared Camera and Multi-Object Spectrograph (NICMOS)
- Space Telescope Imaging Spectrograph (STIS)
- Wide Field Camera 3 (WFC3)
- Wide Field and Planetary Camera (WFPC)
- Wide Field and Planetary Camera 2 (WFPC2)

may be found at:

`http://www.stsci.edu/hst/HST_overview/documents`

Typographic Conventions

To help you understand the material in this Data Handbook, we will use a few consistent typographic conventions.

Visual Cues

The following typographic cues are used:

- **bold** words identify an **STSDAS**, **IRAF**, or **PyRAF** task or package name.
- `typewriter-like` words identify a file name, system command, or response that is typed or displayed.
- *italic* type indicates a new term, an important point, a mathematical variable, or a task parameter.
- `SMALL CAPS` identifies a header keyword.
- ALL CAPS identifies a table column.

Comments

Occasional side comments point out three types of information, each identified by an icon in the left margin.

Warning: You could corrupt data, produce incorrect results, or create some other kind of severe problem.

Heads Up: Here is something that is often done incorrectly or that is not obvious.

Tip: No problems...just another way to do something or a suggestion that might make your life easier.

Information likely to be updated on a specific Instrument Web site is indicated by this symbol.

CHAPTER 1:

Obtaining *HST* Data

In this chapter...

1.1 Archive Overview

All Hubble Space Telescope (*HST*) data files are stored in the Hubble Data Archive (HDA), which forms part of the Multimission Archive at STScI (MAST)[1]. *HST* Guaranteed Time Observers (GTOs), Guest Observers (GOs), and Archival Researchers can retrieve data in one of the following ways:

- Via ftp/sftp, either directly from the HDA, or by accessing a staging disk set up upon request.
- Via downloaded zip files from the Hubble Legacy Archive (HLA) when the relevant dataset is available. See Section 1.2.2
- Via data storage media (CDs or DVDs) which are shipped to the user.
- Via the use of external hard drives loaned by STScI: Users requesting large amounts of data (>100GB) over a short period of time may request the data be written to hard disk. The disk will be delivered to the user for reading, and then the disk must be returned. Use of this retrieval method must be coordinated through the MAST Help Desk (archive@stsci.edu). The number of external hard drives is limited, so users are strongly encouraged to make arrangements in advance. For reference, a DVD can hold 4.7 GB.

1. MAST currently includes data from *HST*, FUSE, XMM-OM, GALEX, IUE, EUVE, ASTRO (HUT, UIT, WUPPE), ORFEUS (BEFS, IMAPS, TUES), Copernicus, Epoch, Kepler, and ROSAT. Data from the FIRST radio survey, Digitized Sky Survey (DSS) and Sloan Digital Sky Survey (SDSS), the HPOL spectropolarimeter, and the Guide Star Catalog (GSC) are also available.

All datasets retrieved from the HDA, regardless of the method used, will be in FITS (Flexible Image Transport System) format. Further information on *HST* file formats is presented in Chapter 2.

Non-proprietary data in the HDA can be retrieved electronically either by registered HDA users or via anonymous login. Proprietary data may not be retrieved except by a registered HDA user who has the permission of the program's Principal Investigator (PI). Note that *HST* PIs are ***not*** automatically registered. PIs should register before their first observations have been taken. All calibration observations as well as observations made as part of the GO Parallel programs are immediately public. All observations made as part a Treasury Program will either be immediately public or have only a brief proprietary period. The High-Level Science Products (HLSP) section of MAST also contains several sets of publicly available and fully reduced *HST* data such as the Ultra Deep Field and the GEMS survey data. See http://archive.stsci.edu/hlsp/index.html for a complete listing.

The archive recommends to ask for compressed data, which distinctly shortens the retrieval times without any significant information loss.

Advanced Camera for Surveys (ACS), Cosmic Origins Spectrograph (COS), and Wide Field Camera 3 (WFC3) users should note that the preferred option for data retrieval is from the HDA staging disk, via ftp/sftp. Users retrieving large numbers of ACS, COS, or WFC3 files should also consider requesting them on DVDs.

1.1.1 Archive Processing

The HDA contains all observations ever made by *HST* and a catalog that describes the data. Each time a user makes a data request, the HDA delivers data which are either processed with the On-The-Fly-Reprocessing (OTFR) system or simply retrieved from a set of final calibrated and statically stored data, as listed in Table 1.1.

Table 1.1: *HST* Instrument Data Processing

Instrument	HDA Processing
ACS	OTFR
COS	OTFR
FGS	Static
FOC	Static
FOS	Static
GHRS	Static
HSP	Static
NICMOS	Static & OTFR[1]
STIS	Static & OTFR[1]
WFC3	OTFR
WF/PC-1	Static
WFPC2	Static[1]

1. All WFPC2 data, and all STIS and NICMOS data taken before SM4 are stored statically. Post-SM4 STIS and NICMOS data will be processed by OTFR upon retrieval.

The OTFR system reconstructs and calibrates the data at the time of retrieval. This allows users to obtain data calibrated with up-to-date reference files, parameters, and software. OTFR makes use of the OSS/PODPS Unified System (OPUS) system to orchestrate the processing of the data. OPUS retrieves the *HST* telemetry data (POD files) from the Data Archiving and Distribution System (DADS) and creates raw data files in a step called Generic Conversion. Calibration reference files specific to the different modes of *HST* operations are prepared and archived in the Calibration Data Base System (CDBS). During Generic Conversion, CDBS is queried to determine which reference files apply to the specific observation being processed. OPUS then calibrates the data using the specific instrument's calibration software and reference files. After calibration, the files are returned to DADS for distribution to the user, as calibrated and/or raw files.

All STIS data collected before SM4 have been fully reprocessed and calibrated. They are now stored online, along with FOS, GHRS, and FOC data. This online data is available via FTP from archive.stsci.edu. (STIS data taken after SM4 will be processed via OTFR.) All WFPC2 data and NICMOS data taken before SM4 have been reprocessed and are stored statically in DADS.

Data for FOC, FOS, GHRS, HSP, and WF/PC-1 do not pass through OTFR or any pipeline. For FOC, FOS, and GHRS, final calibrated archives have been produced, since no further improvements in the calibration for these instruments are expected.

The user is provided with a copy of the raw and final calibrated data from the static (final) archive once a request is made. For HSP and WF/PC-1, no reprocessing or recalibration has been done nor is any planned. Once raw data from these instruments are retrieved from the HDA, they need to be calibrated locally by the users.

Searches and retrievals are available through the MAST Web site, http://archive.stsci.edu. The MAST Web site allows cross correlation of HDA searches with other MAST mission archives. It offers simple previews of *HST* datasets when available, as well as links to references citing a given dataset using the Astrophysics Data System (ADS). The MAST Web site is discussed in more detail in Section 1.2.

1.1.2 Archive Registration

The simplest way to register for *HST* data is to complete the form on the Web page at: http://archive.stsci.edu/registration. If problems occur, registration requests may also be sent to the HDA Help Desk, at archive@stsci.edu. The PI of each *HST* proposal must request access (i.e., authorization) to their proprietary data for themselves, and for anyone else whom the PI wants to have access to them. PI retrieval permission is not granted automatically, for security reasons. PIs wishing to allow access to their proprietary data should make that request to archive@stsci.edu. When registration is granted, your account will be activated automatically, and you will receive your username and password via e-mail.

1.1.3 Archive Documentation and Help

The MAST Web site provides a wealth of useful information, including an online version of the *HST* Archive Manual available at http://archive.stsci.edu/hst/manual. Investigators expecting to work regularly with *HST* and other datasets supported by MAST should also subscribe to the MAST electronic newsletter by sending an e-mail message to archive_news-request@stsci.edu with the word "*subscribe*" in the body of the message. Questions about the HDA can be directed to archive@stsci.edu, or by phone to (410) 338-4547.

1.2 Obtaining Data via the MAST Web

HDA datasets can be searched for, previewed, and retrieved via the World Wide Web. The starting point for Web-based searches of the HDA is the Multimission Archive at STScI (MAST) Web site:

`http://archive.stsci.edu`[2]

or the Hubble Legacy Archive (described in Section 1.2.2) at:

`http://hla.stsci.edu`

1.2.1 MAST Overview

The MAST home page is shown in Figure 1.1. A powerful feature of MAST is that all of its mission archives, including the HDA, can be searched simultaneously. This is done with the Quick Target Search option shown on the MAST home page. This search will return all datasets for all missions available for a given object or coordinates, according to the search constraints specified by the user (based on the wavelength region of interest), and will provide hypertext links to these datasets. If only *HST* datasets are desired, they can be accessed separately by clicking on the MAST home page from the "Missions" pull-down menu. Searches of the HDA by object class can also be made with the VizieR Catalog Search tool at:

`http://archive.stsci.edu/vizier.php`

The *HST* section of MAST offers tutorials about the HDA, as well as news and a Frequently Asked Questions page. It also provides links to *HST* "Prepared" datasets such as the Ultra Deep Field and the Hubble Deep Field images. Clicking on the "Main Search Form" option of the "Search and Retrieval" menu in the *HST* section brings up the page shown in Figure 1.2. Here the user may query on several search parameters, such as Object Name, Instrument, and Proposal ID. Once these are entered, clicking the "Search" button returns a page listing the datasets found. An example search results page is shown in Figure 1.3. More information about an individual dataset, including a preview (for most datasets), are also available by clicking on the dataset name (Figure 1.4).

Datasets can be selectively marked for retrieval in the search results page. After marking datasets for retrieval, press the button labeled "Submit marked data for retrieval from STDADS". This brings you to the Retrieval Options page, shown in Figure 1.5. Here you may select which files (calibrated, uncalibrated, etc.) you would like retrieved, and where you would like them delivered.

If you are retrieving proprietary data, you will need an archive account. Please refer to Section 1.1.1 for information on how to request one.

2. European archive users should generally use the ESO/ST-ECF Archive at http://archive.eso.org/cms. Canadian users should request public archival data through the CADC Web site at http://cadcwww.dao.nrc.ca. Proprietary data are only available through STScI.

Non-proprietary data may be retrieved with or without an archive account. To retrieve non-proprietary data without an archive account, type “anonymous” in the “Archive Username” field, and your email address in the password field. (The email address is needed so that we can notify you when the retrieval request is finished.)

Options for data delivery include direct ftp and sftp, staging, and CD or DVD. If ftp/sftp delivery is specified, you will need to provide the name of the computer and directory to which the files are to be delivered, as well as your username and password on that computer. (Retrieval requests are encrypted, so there is no danger of your login information being stolen.)

Shortly after submitting the request, you will receive an e-mail message acknowledging its receipt. Another message will be sent after all the requested files have been transferred. The status of the request, including how many files have been transferred and any errors that have occurred, can be checked on a Web page at the address given in the acknowledgment message.

Datasets retrieved to the staging disk using an Archive username and password may be accessed through ftp from archive.stsci.edu using this username and password. (Data that you stage this way will only be visible to you. Therefore, proprietary data as well as non-proprietary data may be safely staged.) Datasets that were retrieved as “anonymous” can be accessed using either an account username and password or through regular anonymous ftp.

Figure 1.1: MAST Home Page

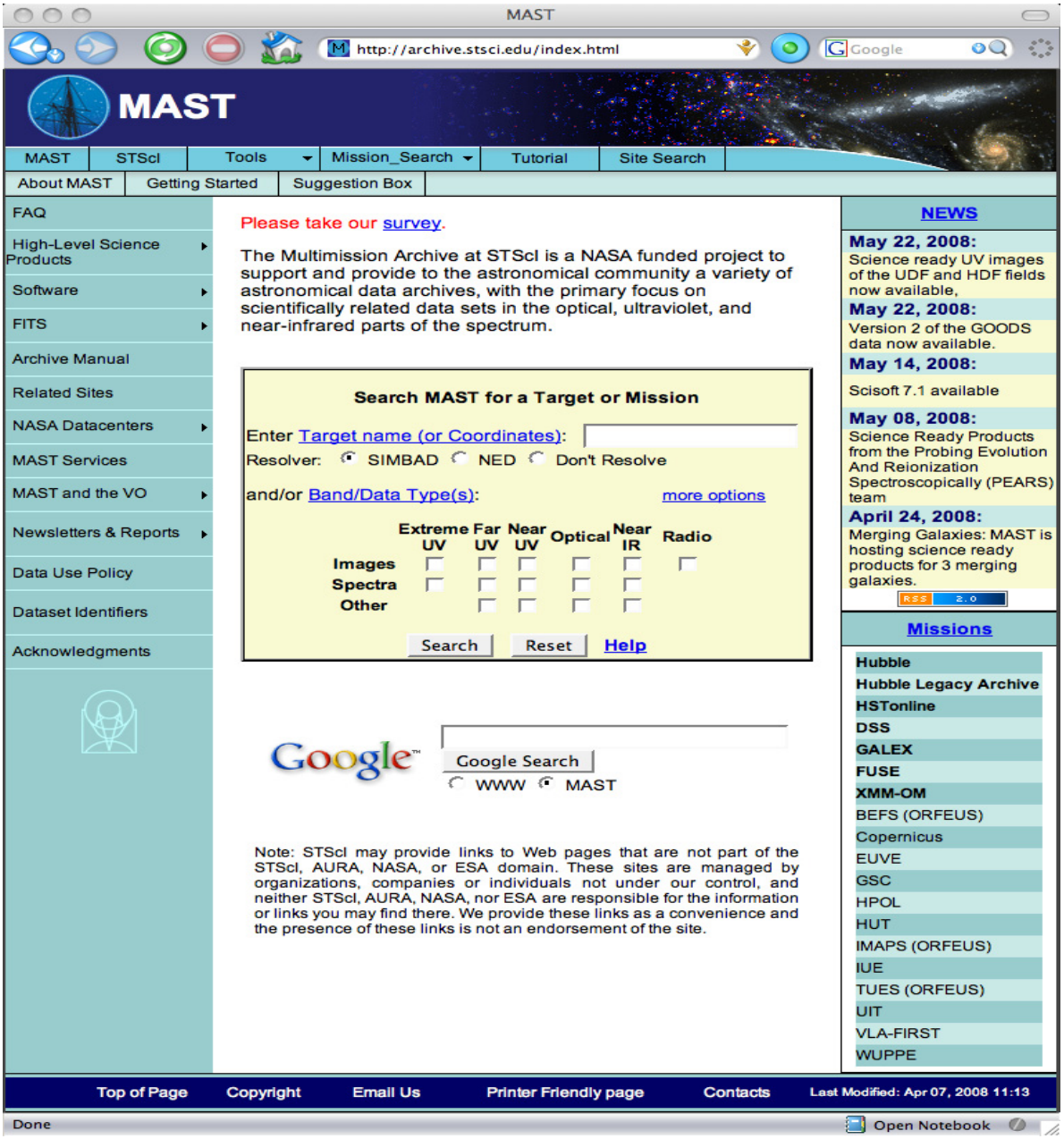


Figure 1.2: *HST* Archive Web Search Form

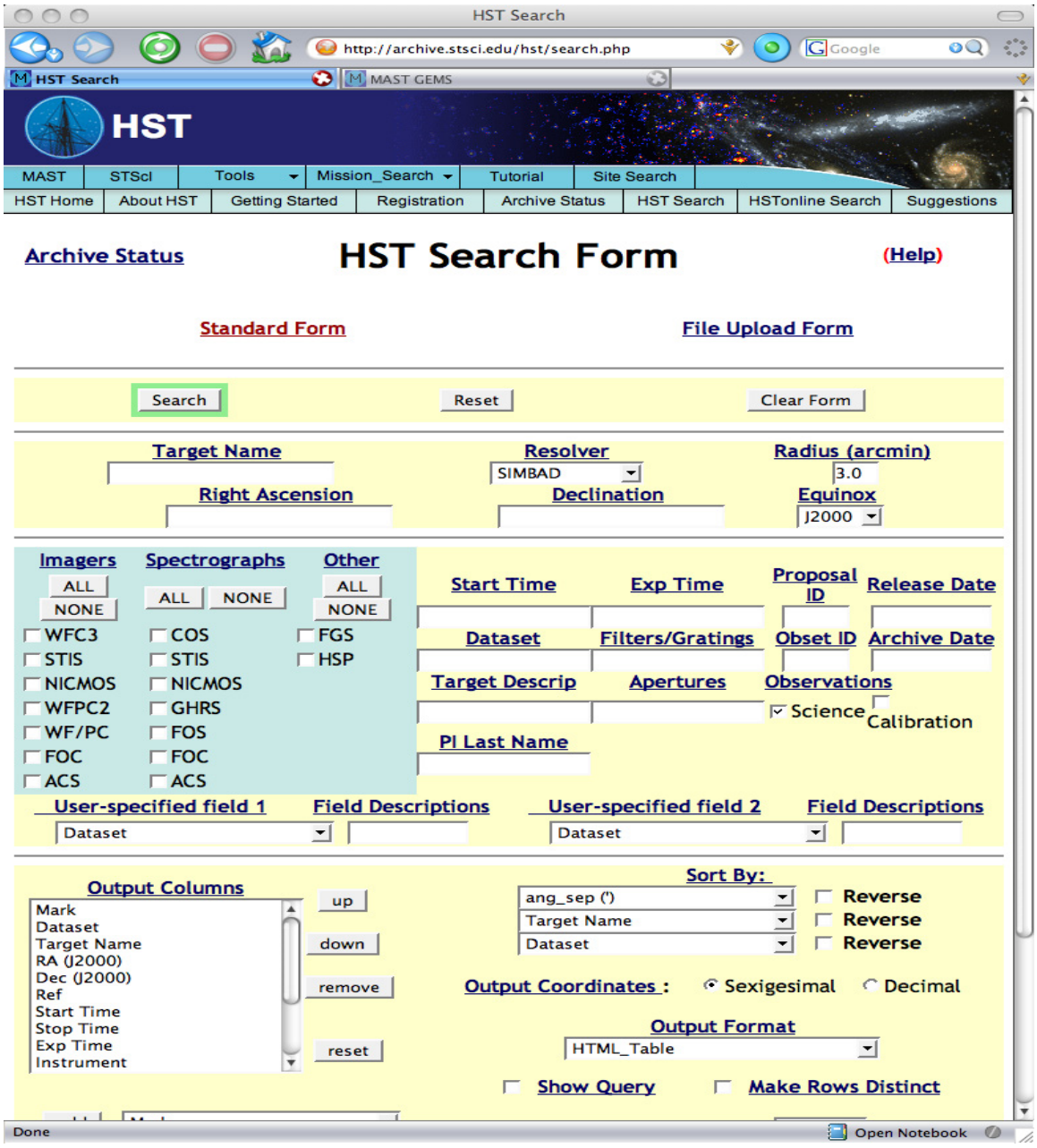
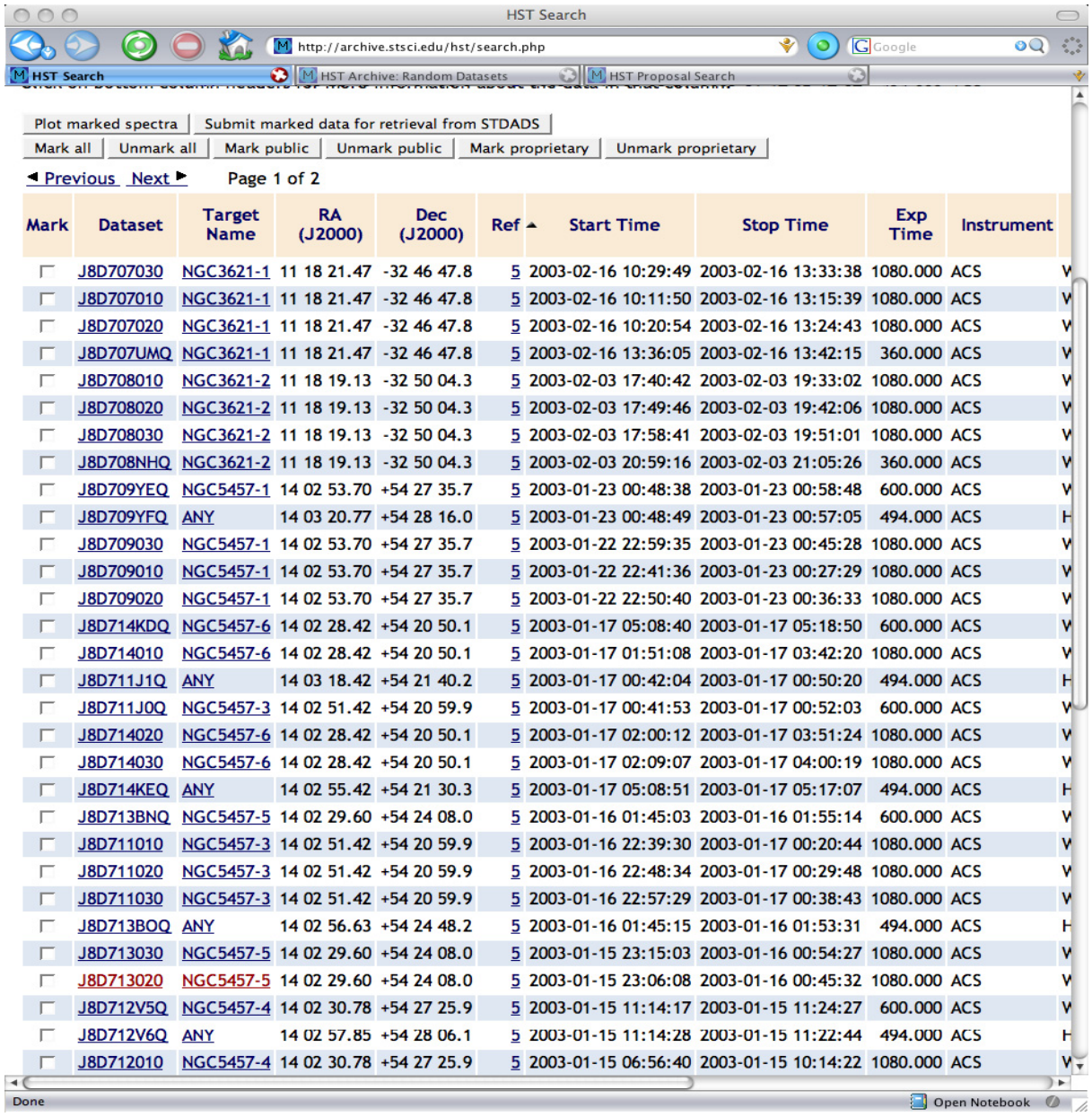


Figure 1.3: *HST* Archive Search Results Page



Mark	Dataset	Target Name	RA (J2000)	Dec (J2000)	Ref	Start Time	Stop Time	Exp Time	Instrument	
☐	J8D707030	NGC3621-1	11 18 21.47	-32 46 47.8	5	2003-02-16 10:29:49	2003-02-16 13:33:38	1080.000	ACS	W
☐	J8D707010	NGC3621-1	11 18 21.47	-32 46 47.8	5	2003-02-16 10:11:50	2003-02-16 13:15:39	1080.000	ACS	W
☐	J8D707020	NGC3621-1	11 18 21.47	-32 46 47.8	5	2003-02-16 10:20:54	2003-02-16 13:24:43	1080.000	ACS	W
☐	J8D707UMQ	NGC3621-1	11 18 21.47	-32 46 47.8	5	2003-02-16 13:36:05	2003-02-16 13:42:15	360.000	ACS	W
☐	J8D708010	NGC3621-2	11 18 19.13	-32 50 04.3	5	2003-02-03 17:40:42	2003-02-03 19:33:02	1080.000	ACS	W
☐	J8D708020	NGC3621-2	11 18 19.13	-32 50 04.3	5	2003-02-03 17:49:46	2003-02-03 19:42:06	1080.000	ACS	W
☐	J8D708030	NGC3621-2	11 18 19.13	-32 50 04.3	5	2003-02-03 17:58:41	2003-02-03 19:51:01	1080.000	ACS	W
☐	J8D708NHQ	NGC3621-2	11 18 19.13	-32 50 04.3	5	2003-02-03 20:59:16	2003-02-03 21:05:26	360.000	ACS	W
☐	J8D709YEQ	NGC5457-1	14 02 53.70	+54 27 35.7	5	2003-01-23 00:48:38	2003-01-23 00:58:48	600.000	ACS	W
☐	J8D709YFQ	ANY	14 03 20.77	+54 28 16.0	5	2003-01-23 00:48:49	2003-01-23 00:57:05	494.000	ACS	H
☐	J8D709030	NGC5457-1	14 02 53.70	+54 27 35.7	5	2003-01-22 22:59:35	2003-01-23 00:45:28	1080.000	ACS	W
☐	J8D709010	NGC5457-1	14 02 53.70	+54 27 35.7	5	2003-01-22 22:41:36	2003-01-23 00:27:29	1080.000	ACS	W
☐	J8D709020	NGC5457-1	14 02 53.70	+54 27 35.7	5	2003-01-22 22:50:40	2003-01-23 00:36:33	1080.000	ACS	W
☐	J8D714KDQ	NGC5457-6	14 02 28.42	+54 20 50.1	5	2003-01-17 05:08:40	2003-01-17 05:18:50	600.000	ACS	W
☐	J8D714010	NGC5457-6	14 02 28.42	+54 20 50.1	5	2003-01-17 01:51:08	2003-01-17 03:42:20	1080.000	ACS	W
☐	J8D711J1Q	ANY	14 03 18.42	+54 21 40.2	5	2003-01-17 00:42:04	2003-01-17 00:50:20	494.000	ACS	H
☐	J8D711J0Q	NGC5457-3	14 02 51.42	+54 20 59.9	5	2003-01-17 00:41:53	2003-01-17 00:52:03	600.000	ACS	W
☐	J8D714020	NGC5457-6	14 02 28.42	+54 20 50.1	5	2003-01-17 02:00:12	2003-01-17 03:51:24	1080.000	ACS	W
☐	J8D714030	NGC5457-6	14 02 28.42	+54 20 50.1	5	2003-01-17 02:09:07	2003-01-17 04:00:19	1080.000	ACS	W
☐	J8D714KEQ	ANY	14 02 55.42	+54 21 30.3	5	2003-01-17 05:08:51	2003-01-17 05:17:07	494.000	ACS	H
☐	J8D713BNQ	NGC5457-5	14 02 29.60	+54 24 08.0	5	2003-01-16 01:45:03	2003-01-16 01:55:14	600.000	ACS	W
☐	J8D711010	NGC5457-3	14 02 51.42	+54 20 59.9	5	2003-01-16 22:39:30	2003-01-17 00:20:44	1080.000	ACS	W
☐	J8D711020	NGC5457-3	14 02 51.42	+54 20 59.9	5	2003-01-16 22:48:34	2003-01-17 00:29:48	1080.000	ACS	W
☐	J8D711030	NGC5457-3	14 02 51.42	+54 20 59.9	5	2003-01-16 22:57:29	2003-01-17 00:38:43	1080.000	ACS	W
☐	J8D713BOQ	ANY	14 02 56.63	+54 24 48.2	5	2003-01-16 01:45:15	2003-01-16 01:53:31	494.000	ACS	H
☐	J8D713030	NGC5457-5	14 02 29.60	+54 24 08.0	5	2003-01-15 23:15:03	2003-01-16 00:54:27	1080.000	ACS	W
☐	J8D713020	NGC5457-5	14 02 29.60	+54 24 08.0	5	2003-01-15 23:06:08	2003-01-16 00:45:32	1080.000	ACS	W
☐	J8D712V5Q	NGC5457-4	14 02 30.78	+54 27 25.9	5	2003-01-15 11:14:17	2003-01-15 11:24:27	600.000	ACS	W
☐	J8D712V6Q	ANY	14 02 57.85	+54 28 06.1	5	2003-01-15 11:14:28	2003-01-15 11:22:44	494.000	ACS	H
☐	J8D712010	NGC5457-4	14 02 30.78	+54 27 25.9	5	2003-01-15 06:56:40	2003-01-15 10:14:22	1080.000	ACS	W

Figure 1.4: *HST* Archive Dataset Preview Page

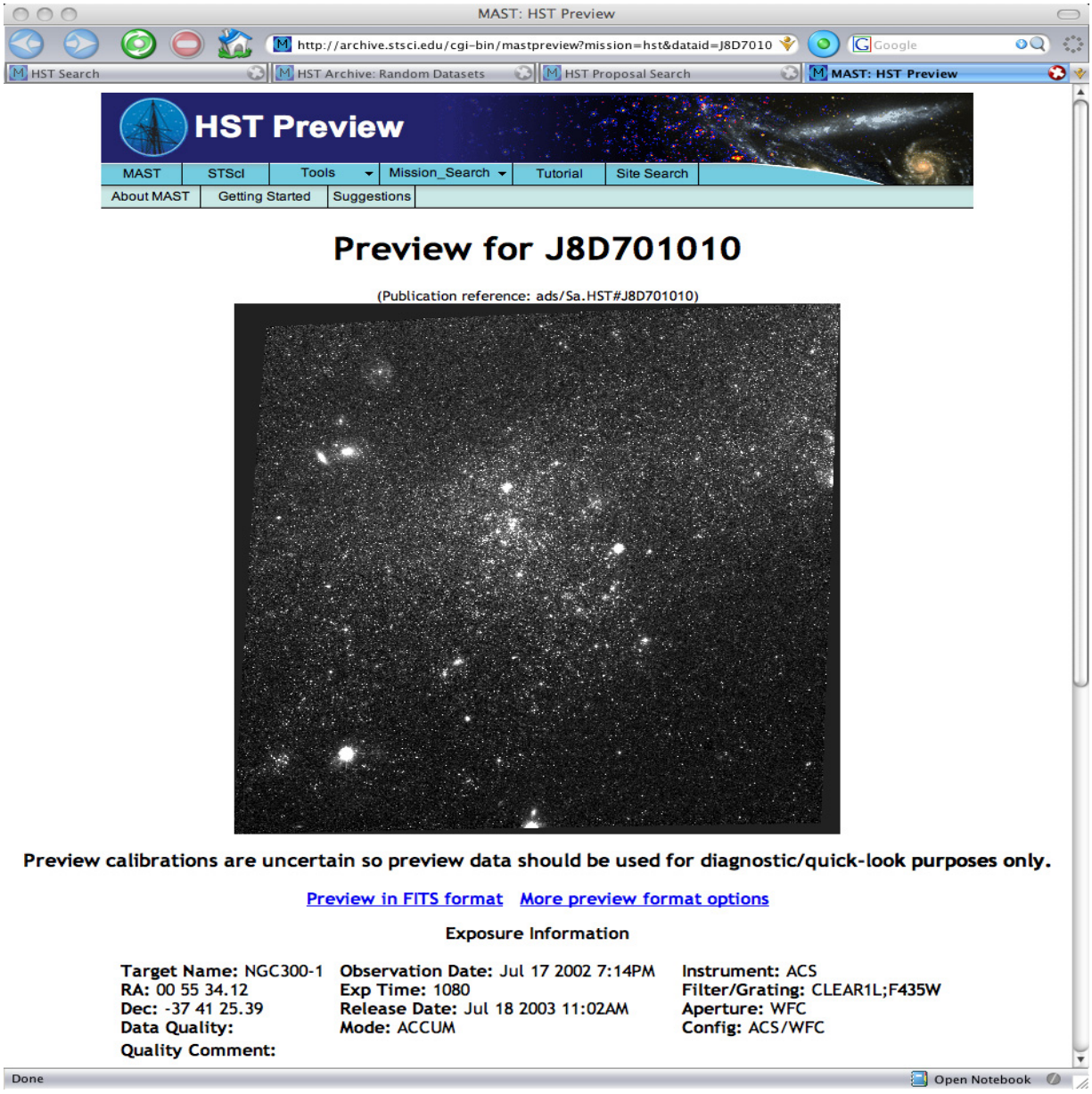
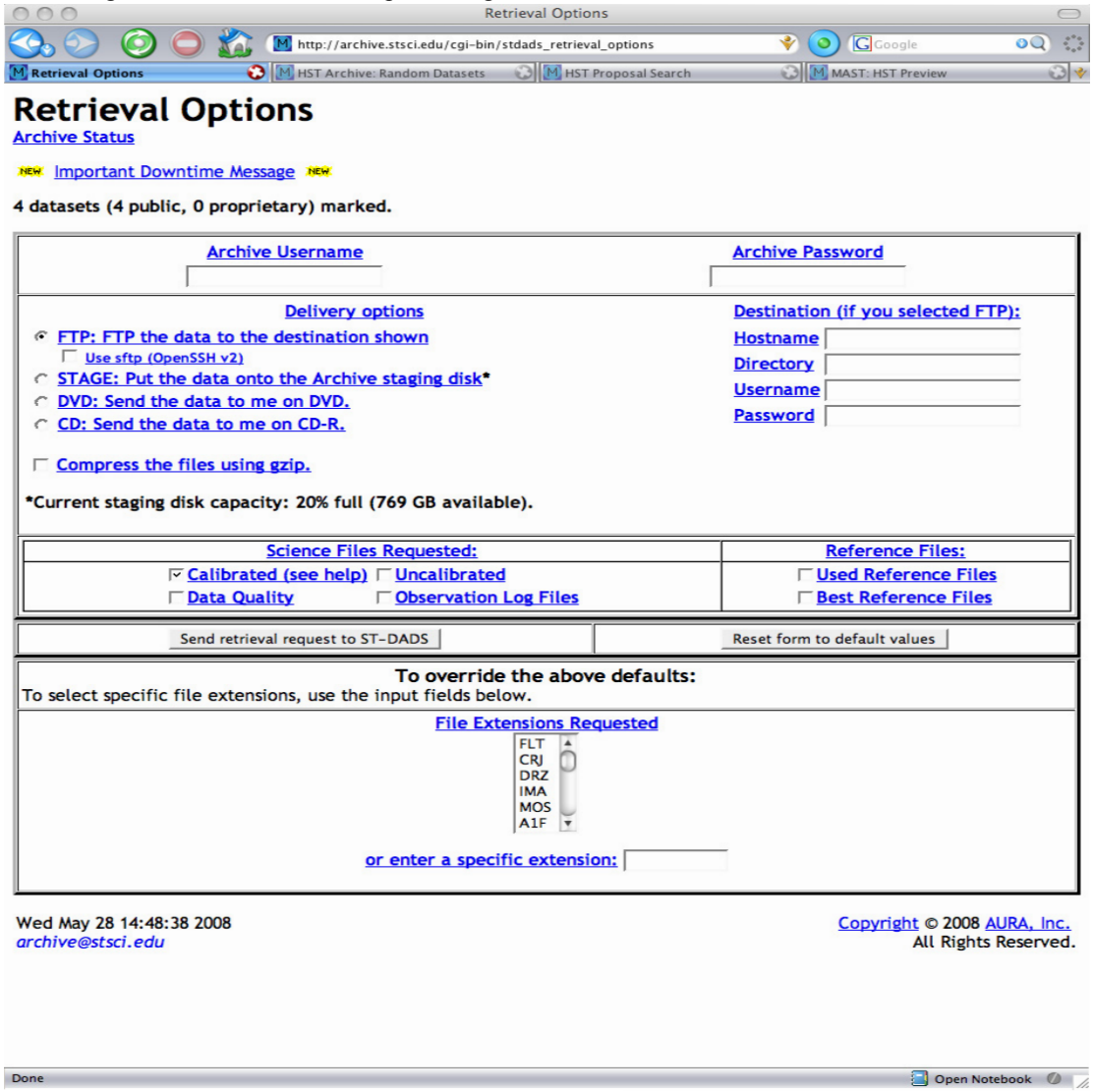


Figure 1.5: *HST* Retrieval Options Page



1.2.2 Hubble Legacy Archive

The Hubble Legacy Archive (HLA) is designed to facilitate access to Hubble Space Telescope data in a form suitable for direct use by a broad variety of science projects. The HLA does this by providing enhanced data products and advanced search and browsing capabilities. The HLA is a joint project of the STScI, the European Coordinating Facility (ST-ECF-Note: ST-ECF ceased operations Jan. 1, 2011; their archives are now served out of ESO/ST-ECF), and the Canadian Astronomy Data Centre (CADC).

At the core of the HLA is a browser-based search and display interface through which users can identify data of interest based on instrument, position on the sky, exposure time, level of processing, availability of catalogs, and other properties of the data. Users can visualize the outline ("footprint") of all data within a desired region of

the sky superimposed on a Digital Sky Survey image, display small previews or cutout views of each dataset, or view the properties of the data offerings in tabular form. Additional selections based on exposure time, date of observation, filter, or other properties are supported. The interface also allows immediate visualization within the browser of the science data at their full resolution, and the data are online and available for immediate download. A shopping cart interface facilitates the selection and retrieval of multiple data sets.

HST datasets are listed in the HLA and are available for visualization within the footprint viewer within approximately one to three weeks after the observations. The advanced HLA data products are added approximately once per year. (A system that will allow much more frequent updates is under development.)

Proprietary data are also listed in the HLA interface, although the datasets cannot be downloaded by the public. Data previews will not be available, but using the footprints viewer the HLA interface may still be useful in order to visualize proprietary observations in the context of other observations of the same sky region.

The offerings of the HLA are rapidly expanding. At the time of this writing (January 2011), the HLA provides enhanced data products for most ACS, NICMOS, and WFPC2 non-proprietary data, including visit-level combined images, and access and immediate display for MAST products for STIS, FOS, and GHRS. Mosaics combining multi-band ACS data from multiple visits are available for nearly 70 pointings, and provide deeper, wider images than can be obtained from individual visits. Extracted spectra are available for most NICMOS and ACS grism data. Source lists for most of the ACS and WFPC2 images have been generated using both DAOPhot and Source Extractor; object positions can be overplotted on the image display, together with sources from external lists such as SDSS, GSC2, 2MASS, GALEX, and FIRST. In addition, over a dozen sets of user-provided high-level science products (HLSP) can be accessed through the same search and display interface; this includes the results of many of the *HST* Treasury and public programs, such as HDF, UDF, COSMOS, GOODS, GEMS, ANGST, and several others. These are a subset of the HLSP described in Section 1.2.3.

Extensive descriptions of the HLA products and their properties are available through the HLA Help Center. The Release Notes summarize the evolution of HLA products over time. Figures 1.2.3, 1.7, and 1.8 show examples of the HLA Web pages. The HLA main page can be accessed at:

`http://hla.stsci.edu.`

Figure 1.6: HLA Example Page With ACS WFC Broad Band and Coadded (Color) Images

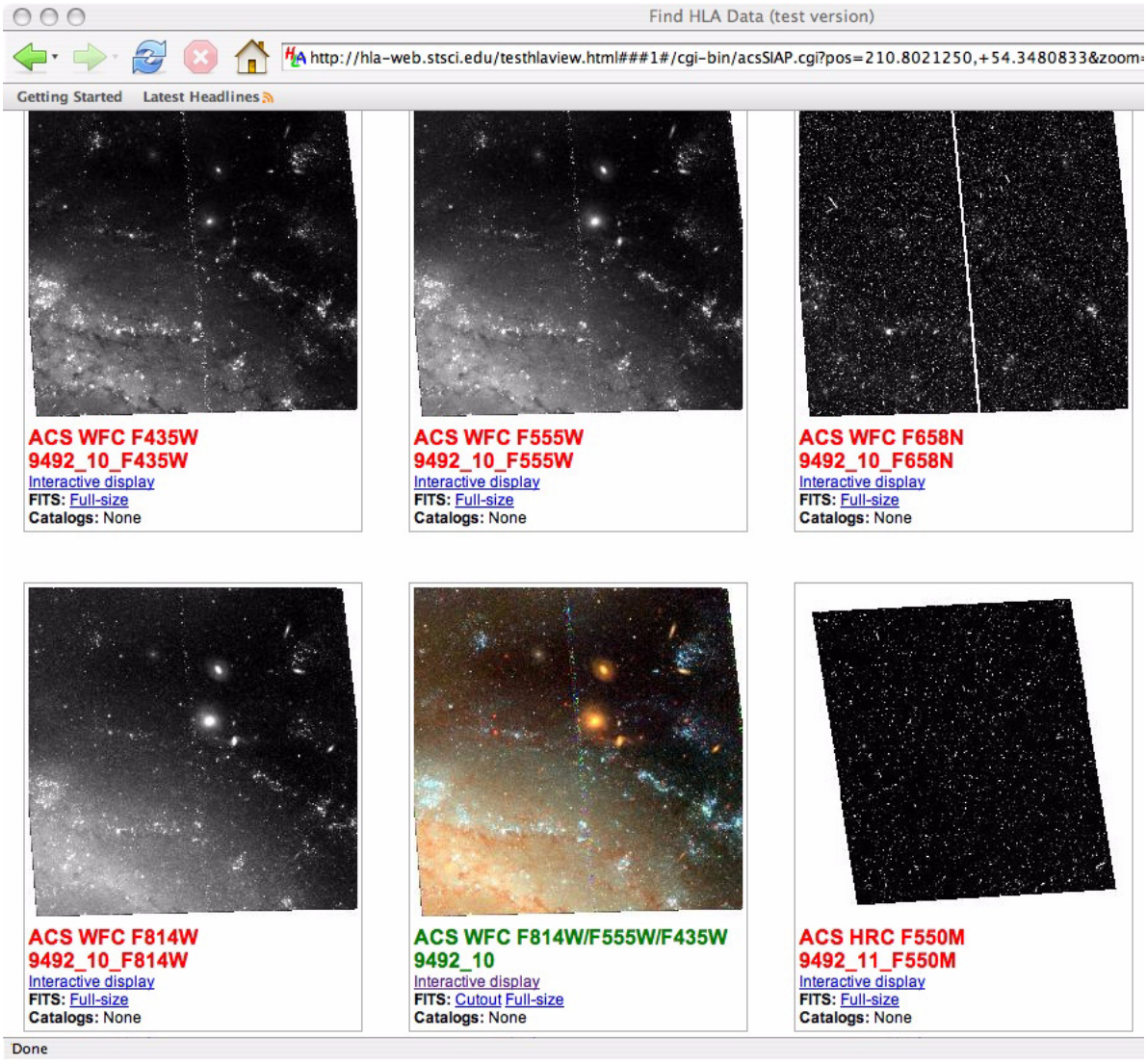


Figure 1.7: STIS Eta Car Spectral Image and Segments of a Wavelength Calibrated Spectrum

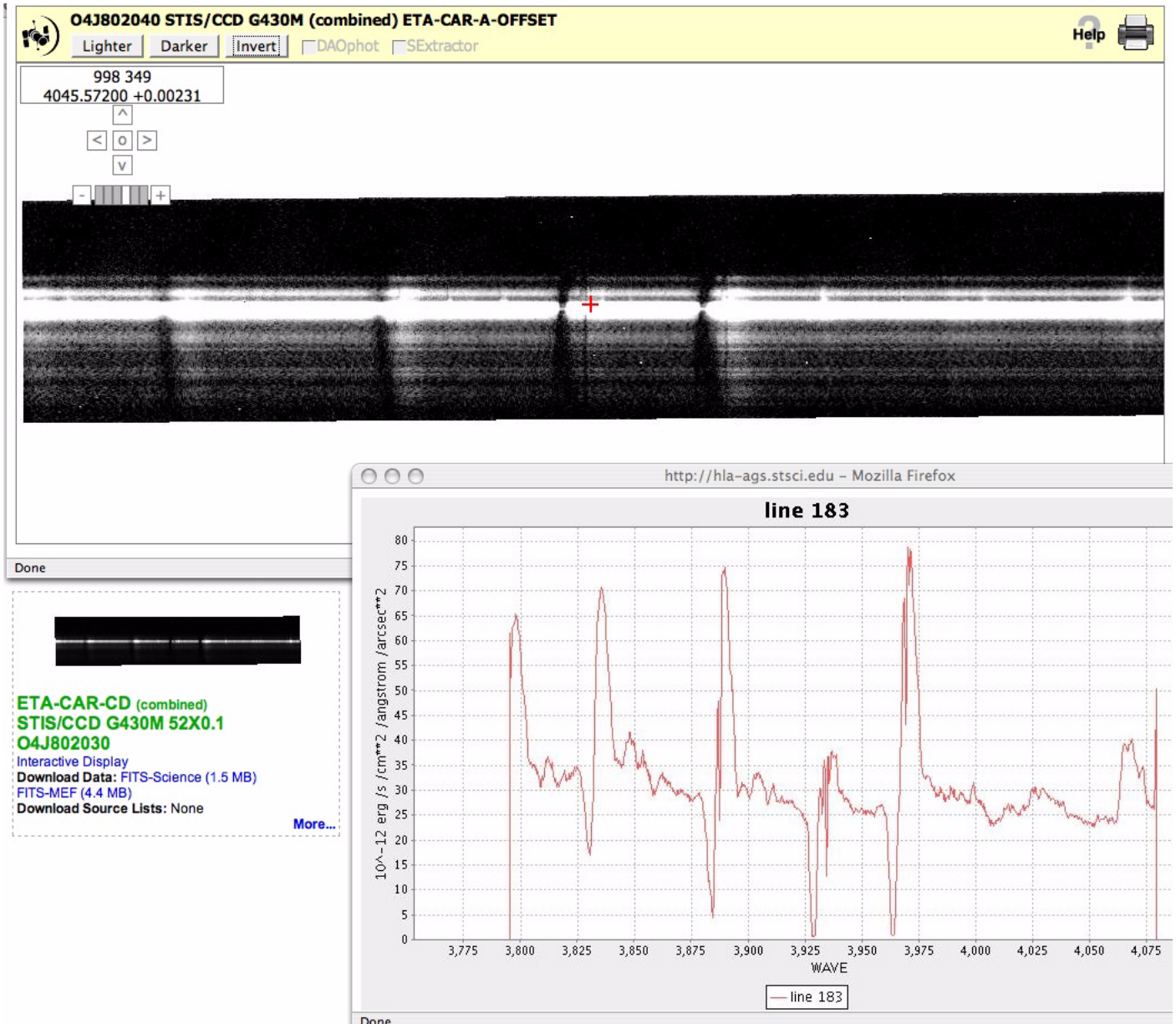
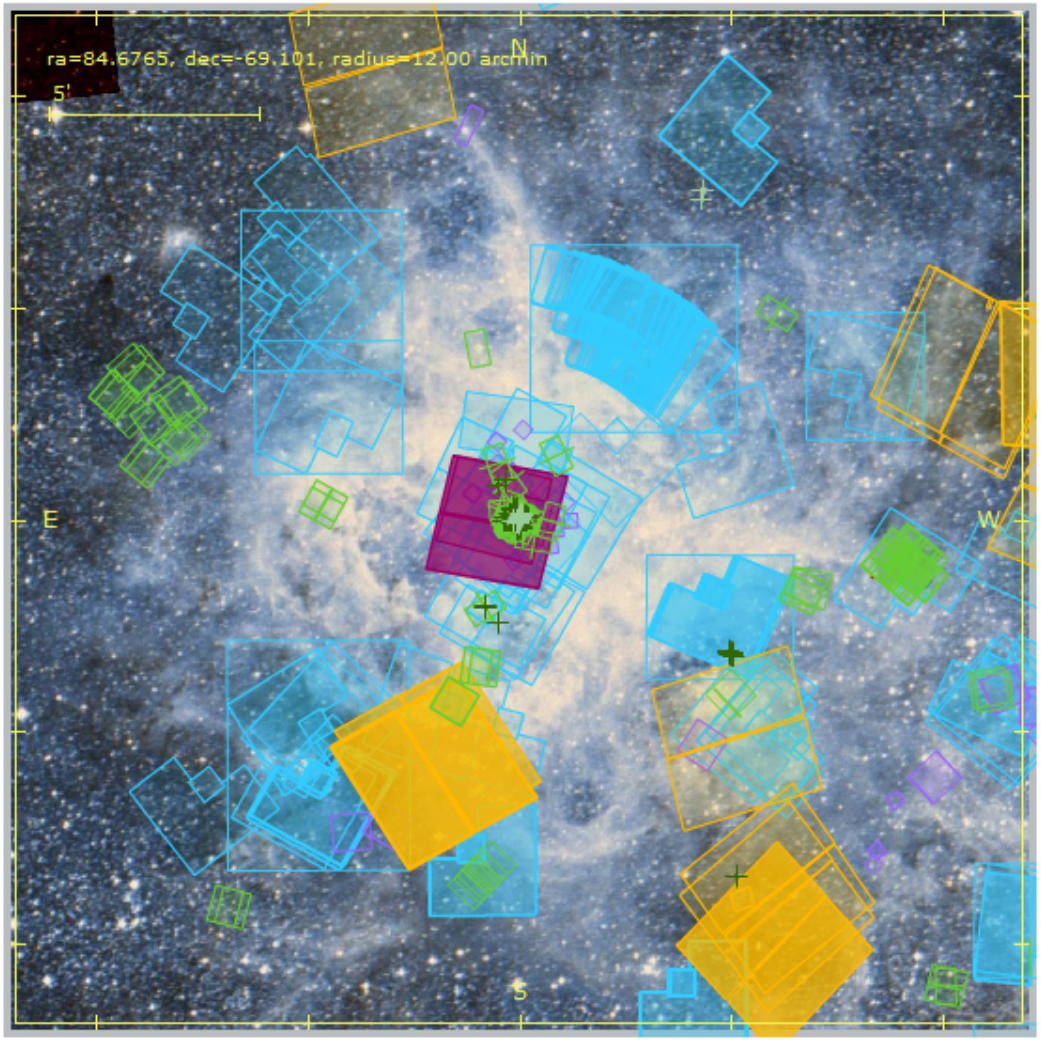


Figure 1.8: HLA Footprint view showing *HST* Instrument. FOVs Overlaid on 30 Doradus.

Inventory | Images | Footprints | Cart, 0 kB | Grism Spectra (ST-ECF) | Help Center

Instruments	#Footprints
☑ ALL	1917
☑ ACS	116
☑ ACSGrism	0
☑ WFPC2	424
☑ WFPC2-PC	325
☑ NICMOS	196
☑ NICGrism	0
☑ WFC3	58
☑ COS	47
☑ STIS	466
☑ FOS	208
☑ GHRS	77



Click to select the interactive footprint graphic: you can then zoom in using the = or + keys, or zoom out using -.

30 dor **RA = 84.676490 Dec = -69.100900 r = 0.200000 [05:38:42.358 -69:06:03.24]**

The recent "Data Release 5" includes new and expanded data products and significant interface enhancements. Products planned for release in DR5 or soon thereafter include combined WFC3 images, prototype COS and STIS products, and a greatly expanded number of multi-visit ACS mosaics. A new footprint display allows panning and zooming and is more fully integrated with the rest of the interface. Other improvements under consideration for future releases include expanding the generation of source lists and mosaics to additional instruments, more spectroscopic products (e.g., coadded multi-visit spectra, multi-wavelength spliced spectra), and a more advanced user interface for spectra (e.g., quick look spectral extractions, line identification lists). Some of the more general goals of the HLA are to make more *HST* data available through the Virtual Observatory (VO), to move toward a sky atlas user-view rather than a collection of datasets, and to develop an "all-*HST*-sky" source list.

The HLA is designed to be complementary to the MAST interface for access to *HST* data. Users needing access to known data sets, or those who plan to run independent data reduction and image combination, will often find it more convenient to use the MAST interface to access and retrieve the data of interest. Those who need access to proprietary data may also prefer to use the MAST interface. On the other hand, users who wish to search for data covering a region of interest and are interested in a graphical display of their relative position, or users who may be interested in quick-look capabilities, science-ready combined data, and/or source listings, will generally find the HLA offerings to be more convenient and more directly accessible.

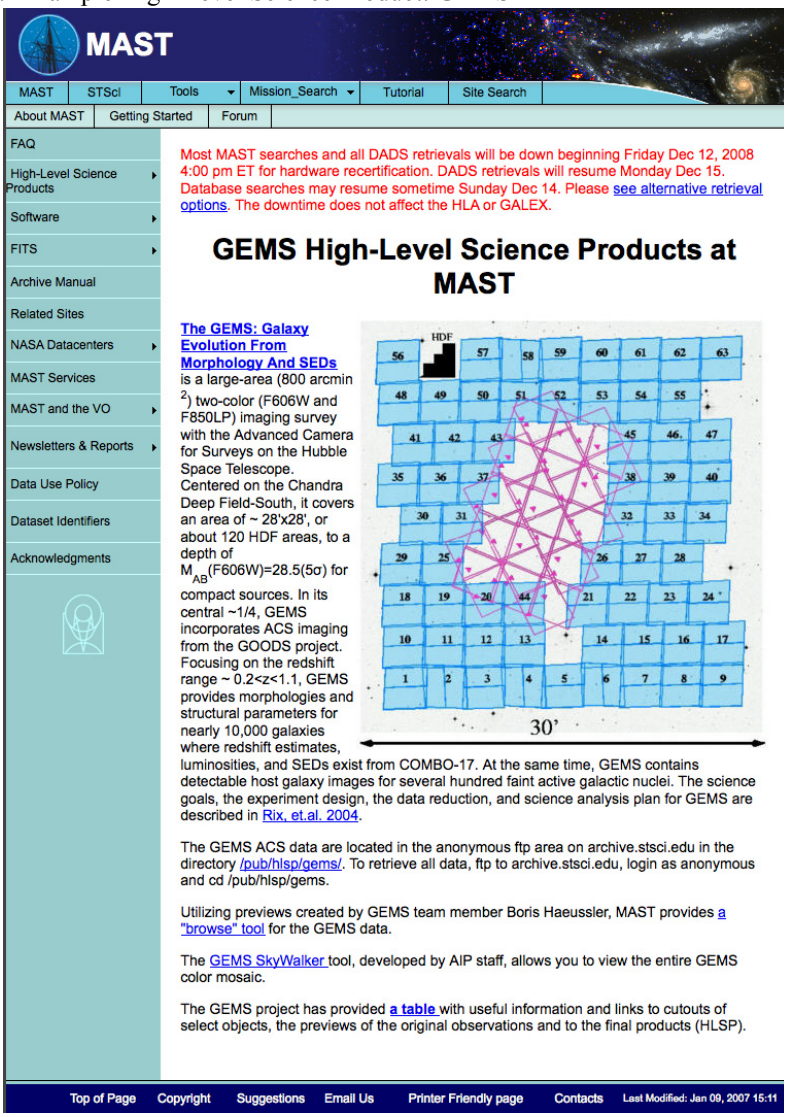
1.2.3 High-Level Science Products

MAST also contains a number of *user-contributed*, High-Level Science Products (HLSP), which are accessible at http://archive.stsci.edu/hlsp/index.html. High-Level Science Products are generally fully processed (reduced, coadded, cosmic-ray cleaned, etc.) images and spectra that are ready for scientific analysis. HLSP also include files such as object catalogs, spectral atlases, and README files describing a given set of data. Most of the data originate from the Treasury, Archival Legacy and Large Programs (TALL) from Cycle 11 onward, but some contributions from smaller *HST* programs and other MAST missions are also included.

A screen shot of the Web page for the ACS product Galaxy Evolution from Morphology and SEDs (GEMS) is shown in Figure 1.9.

Users who are interested in contributing to the HLSP, are referred to the Guidelines for Contributing High-Level Science Products to MAST (http://archive.stsci.edu/hlsp/hlsp_guidelines.html, please make sure to get the latest version). Furthermore, they are asked to contact the archive scientist involved as soon as they start working on the data.

Figure 1.9: Example High-Level Science Product: GEMS



1.3 Obtaining Data with StarView

The Java version of StarView is no longer supported by STScI. However, a Web-based Flash-enabled version of StarView has been created to replace it. This Web-based version provides much of the same functionality as the Java version, including customizing screens. It is available at:

http://starview.stsci.edu/

CHAPTER 2:

HST File Formats

In this chapter...

Chapters 2 and 3 assume that the reader is familiar with basic IRAF/STSDAS syntax. If terms such as `login.cl`, **epar**, **lpar**, and **apropos** are not familiar to the reader, we recommend skipping to the IRAF Primer included in Chapter 4.

2.1 *HST* Data Files

The STScI pipeline automatically processes and calibrates all the data received from *HST* and assembles them into a form suitable for most scientific analyses.

Data from WFPC2, ACS, COS, NICMOS, STIS, and WFC3 are made available to observers as files in Multi-Extension FITS (MEF) format, which is directly readable by most **PyRAF**/**IRAF**/**STSDAS** tasks.

For first generation instruments (FGS, FOC, FOS, GHRS, HSP, and WF/PC-1), the data are in a format known as Generic Edited Information Set (GEIS). Byte ordering for binary data in GEIS files is machine-dependent. Therefore, for purposes of archiving and file transfer, GEIS data are converted to a modified FITS format known as "waivered FITS." It is necessary to convert waivered FITS files back to GEIS format for data processing and analysis using **IRAF**/**STSDAS** tasks.

All WFPC2 data are now available in either waivered FITS or MEF formats. The user may specify either format when retrieving that data from the HDA. WFPC2 data, in either GEIS or MEF formats, can be fully processed with STSDAS tasks.

For older instruments, Heritage Instruments, (FOC, FOS, FGS, GHRS, HSP, WF/PC-1, and WFPC2) data are provided in waivered FITS format, which needs to be converted back to GEIS format before processing. Note that WFPC2 data is also available in MEF format from the HDA. Newer instruments (ACS, COS, NICMOS, STIS, and WFC3) generate and store files in FITS format and should not be converted to GEIS.

Table 2.1: *HST* Instrument File Formats

Instrument	Status	Format from HDA	Format for STSDAS Analysis
ACS	active	MEF	MEF
COS	active	MEF	MEF
FGS	active	waivered FITS	GEIS
FOC	decommissioned SM3B	waivered FITS	GEIS
FOS	decommissioned SM2	waivered FITS	GEIS
GHRS	decommissioned SM2	waivered FITS	GEIS
HSP	decommissioned SM1	waivered FITS	GEIS
NICMOS	active	MEF	MEF
STIS	active	MEF	MEF
WFC3	active	MEF	MEF
WFPC2	decommissioned during SM4	waivered FITS & MEF[1]	GEIS & MEF[1]
WFPC1	decommissioned SM1	waivered FITS	GEIS

1. HDA now also stores WFPC2 data in MEF format.

This chapter describes GEIS and MEF formats, in more detail. ACS, COS, NICMOS, STIS, WFPC2, and WFC3 observers should pay particular attention to the section on MEF files, which shows how to identify and access the contents of these files and covers some important conventions regarding header keywords. Users of the heritage instruments should consult the material on data groups and conversion from waivered FITS to GEIS format found in Section 2.3.1 before proceeding to Chapter 3.

Throughout this document, references to FITS files will always mean HST Multi-Extension FITS format files. References to waivered FITS files will always be explicitly stated.

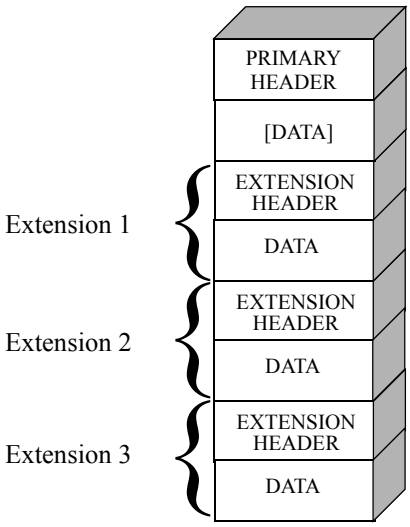
2.2 Multi-Extension FITS File Format

Flexible Image Transport System (FITS)[1] is a standard format for exchanging astronomical data, independent of the hardware platform and software environment.

A file in FITS format consists of a series of Header Data Units (HDUs), each containing two components: an ASCII text header and the binary data. The header contains a series of *keywords* that describe the data in a particular HDU and the data component may immediately follow the header.

For *HST* FITS files, the first HDU, or primary header, contains no data. The primary header may be followed by one or more HDUs called extensions. Extensions may take the form of images, binary tables, or ASCII text tables. The data type for each extension is recorded in the `XTENSION` header keyword. Figure 2.1 schematically illustrates the structure of a FITS file and its extensions.

Figure 2.1: FITS File Structure



Each FITS extension header contains the required keyword `XTENSION`, which specifies the extension type and has one of the following values: IMAGE, BINTABLE, and TABLE, corresponding to an image, binary table, and ASCII table, respectively.

1. A description of FITS format and various supporting documents can be found at the Web site http://fits.gsfc.nasa.gov/fits_home.html

Table 2.2: *HST* Header Keyword Descriptions

HDU Keyword	Description	Values
XTENSION	Data type for extension	• IMAGE • BINTABLE (binary table) • TABLE (ASCII table)
EXTVER	Imset number (see Table 2.3)	Integer
EXTNAME	Extension names that describe the type of data component	• SCI (science image) • ERR (error image) • DQ (data quality image) • SAMP[1] (number of sample) • TIME[1] (exposure time) • EVENTS[2] (photon event list) • GTI[2] (good time interval) • WHT (weight image) • CTX (context image)
PIXVALUE[3]	Allowed for any extension except SCI, and used for an image with uniform value for all pixels	Real number

1. Only found in NICMOS and WFC3 data.
2. Only found in COS and STIS data.
3. When an image has the same value at all pixels (e.g., data quality value), the extension has no data component. Instead, the constant pixel value is stored in the header keyword PIXVALUE.

A set of FITS extension images which are logically related to one another is called an *imset*. For example, the error image and the data quality image are in the same imset as the science image itself. The keyword EXTNAME is used to specify the extension names of different images in the same imset.

2.2.1 Working with Multi-Extension FITS Images in IRAF

The FITS image kernel included in **IRAF** version 2.12.2 and higher treats each extension like a standard **IRAF** image. This subsection provides an overview of Multi-Extension FITS file syntax and keyword options needed to extract specific types of data. The following discussion describes how to specify the image extensions in FITS files that you would like to process with **IRAF**/**STSDAS** tasks and presumes that you are using **IRAF** 2.12.2 or higher. It covers how to:

- List a FITS file's extensions
- Access data in particular FITS extension
- Inherit keywords from the primary header
- Append new extensions to existing FITS files

Retaining the .fits suffix at the end of every FITS file name in your file specifications will ensure that IRAF both reads and writes these images in FITS format.

In order to work with ACS, NICMOS, or STIS data, users will need to upgrade to at least IRAF 2.12.2 and STSDAS 3.2. Processing of WFC3 data will require instead STSDAS 3.8 or higher. COS will require STSDAS 3.9 or later and its pipeline runs only under PyRAF (Section 3.2).

Generating a FITS File Listing

Once you have downloaded any FITS files from the HDA, you may want an inventory of their contents. To generate a listing of a FITS file's extensions, you can use the **catfits** task in the **tables** package. For example, Table 2.3 illustrates the first 11 lines generated by **catfits** from a NICMOS `MULTIACCUM` FITS file containing only images. The output columns from **catfits** contain the following information:

- The first column lists the extension numbers. Note that the primary header extension number is zero.
- The second column lists the extension type whose value is specified in the keyword XTENSION. (In Table 2.3 this column is FITSNAME).
- The third column lists the extension name, given by the keyword EXTNAME. (In Table 2.3 this column is FILENAME).
- The fourth column lists the imset number, given in the EXTVER keyword. (In Table 2.3 this column is EXTVE).

Several **STSDAS** tasks can work with entire imsets (see Section 3.4.3), but most operate on individual images. See the "Data Structure" chapters in Part II of this Data Handbook for more information on the contents of a particular instrument's imsets.

Table 2.3: NICMOS MULTIACCUM Listing from **catfits**

```
tt> catfits n3t501c2r_raw.fits

EXT#  FITSNAME      FILENAME              EXTVE  DIMENS     BITPI OBJECT

0     n3t501c2r_raw n3t501c2r_raw.fits                     16    n3t501c2r_raw.f
1       IMAGE       SCI                   1     256x256    16    n3t501c2r_raw.f
2       IMAGE       ERR                   1                -32
3       IMAGE       DQ                    1                16
4       IMAGE       SAMP                  1                16
5       IMAGE       TIME                  1                -32
6       IMAGE       SCI                   2     256x256    16
7       IMAGE       ERR                   2                -32
8       IMAGE       DQ                    2                16
9       IMAGE       SAMP                  2                16
10      IMAGE       TIME                  2                -32
```

Accessing FITS Images

After you have identified which FITS image extension you wish to process, you can direct an **IRAF**/**STSDAS** task to access that extension using the following syntax:

`fitsfile.fits[`*extension number*`] [`*image section*`]`

or

`fitsfile.fits[`*keyword options*`] [`*image section*`]`

Specifying the extension number is the most basic method of accessing an HDU in a FITS file, but it is not necessarily the most useful syntax. Referring to an extension's EXTNAME and EXTVER in the `[keyword options]` is often more convenient. If a number follows an EXTNAME, **IRAF** interprets the number as an EXTVER. For example, if extension number 6 holds the science image belonging to the imset with EXTVER = 2, as in the **catfits** listing above, it can be specified in two equivalent ways:

```
fitsfile.fits[6]
fitsfile.fits[sci,2]
```

Designating an EXTNAME without an EXTVER refers to the first extension in the file with the specified value of EXTNAME. Thus, `fitsfile.fits[sci]` is the same as `fitsfile.fits[sci,1]`.

The syntax for designating image sections follows the **IRAF** standard. So, in the current example, the specifications

```
fitsfile.fits[6][100:199,100:299]
fitsfile.fits[sci,2][100:199,100:299]
```

both refer to a 100 by 200 pixel subsection of the same science image in `fitsfile.fits`.

Header Keywords and Inheritance

ACS, COS, NICMOS, STIS, and WFC3 data files use an **IRAF** image kernel convention that allows HDU extensions, under certain circumstances, to *inherit* keywords from the primary header. When this inheritance takes place, the primary header keywords are practically indistinguishable from the extension header keywords. This feature circumvents the large scale duplication of keywords that share the same value for all extensions. The primary header keywords effectively become global keywords for all image extensions. Note, the FITS standard does not include keyword inheritance, and while the idea itself is simple, its consequences are often complex and sometimes surprising to users.

In general, keyword inheritance is the default, and **IRAF**/**STSDAS** applications will join the primary and extension headers and treat them as one. For example, using **imheader** as follows on a FITS file will print both primary and extension header keywords to the screen:

```
cl> imheader fitsfile.fits[sci,2] long+ | page
```

Using **imcopy** on such an extension will combine the primary and extension headers in the output HDU, even if the output is going to an extension of another FITS file. Once **IRAF** has performed the act of inheriting the primary header keywords, it will normally turn the inheritance feature off in any output file it creates unless specifically told to do otherwise.

If you need to change the value of one of the global keywords inherited from the primary header, you must edit the primary header itself (i.e., "extension" [0]).

Keyword inheritance is not always desirable. For example, if you use **imcopy** to copy all the extensions of a FITS file to a separate output file, **IRAF** will write primary header keywords redundantly into each extension header. You can suppress global keyword inheritance by using the NOINHERIT keyword option in the file specification. For example:

```
im> imcopy fitsfile.fits[6][noinherit] outfile.fits
im> imcopy fitsfile.fits[sci,2,noinherit] outfile.fits
```

The resulting `outfile.fits` contains no global keywords from `fitsfile.fits`, except for keywords which were present in the extension header. In the first example, where the FITS image uses an absolute extension number, `noinherit` is the only entry needed in the FITS option field. In the second command, the `noinherit` option is bracketed with the `EXTNAME` and `EXTVER` keyword. For a complete explanation of FITS file name specifications, see:

http://iraf.noao.edu/iraf/web/docs/fitsuserguide.html

Appending Image Extensions to FITS Files

IRAF/STSDAS tasks that produce FITS images as output can either create new FITS files or append new image extensions to existing FITS files. You may find the following examples useful if you plan to write scripts to reduce MEF FITS files:

If the specified output file does not yet exist, it is created containing only the specified extension of the original file. For example, to copy the contents of the primary header of `fitsfile.fits` to a new FITS file named `outfile.fits`:

```
cl> imcopy fitsfile.fits[0] outfile.fits
```

Note that **imcopy** will yield an error if an extension is not specified in the command. If the specified output file already exists and you want to append a new extension to it, you must include the APPEND option in the output file specification. The example below appends extension `[sci,2]` of `fitsfile.fits` to the existing file `outfile.fits`, while retaining the original EXTNAME and EXTVER of the extension. The `noinherit` keyword option prevents the copying of the primary header keywords from the input file into the output extension header:

```
cl> imcopy fitsfile.fits[sci,2,noinherit] \
>>> outfile.fits[append]
```

Note that the backslash is added to indicate that the remainder of the command follows on the next line, after the ">>>" prompt.

To change the EXTNAME or EXTVER of the appended extension, you can specify new values for these keywords in the output extension:

```
cl> imcopy fitsfile.fits[sci,2,noinherit] \
>>> outfile.fits[sci,3,append]
```

For obvious reasons, it is generally not advisable for two file extensions in the same FITS file to share the same EXTNAME and EXTVER values. However, if you must append an extension to an output file already containing an extension with the same EXTNAME/EXTVER pair you can do so with the DUPNAME option:

```
cl> imcopy fitsfile.fits[7] \
>>> outfile.fits[append,dupname]
```

If you need to replace an existing extension with a new output extension, you can use the OVERWRITE option as follows. Overwriting can cause a lengthy rewrite of the whole file to insert the new extension, if its size is not the same as the extension it replaces.

```
cl> imcopy fitsfile.fits[sci,2,noinherit] \
>>> outfile.fits[sci,2,overwrite]
```

2.2.2 Working with FITS Table Extensions

FITS tables are used to store certain types of data from ACS, COS, NICMOS, STIS, WFC3, and WFPC2. For these instruments, OPUS produces association tables that list the exposures used to construct association products. In addition, reference data may be stored in FITS tables. This section describes:

- How to access and read FITS table extensions.
- How to specify data arrays in FITS table cells.

This discussion assumes you are using **STSDAS** 3.2 or later. (The **IRAF** FITS kernel deals only with FITS images. The **tables** package in, **STSDAS** handles FITS table extensions.)

Accessing FITS Tables

You can access data in FITS table extensions using the same tasks appropriate for any other **STSDAS** table. The syntax for accessing a specific FITS table is similar to the syntax for accessing FITS images (see Section 2.2.1), with the following exceptions:

- The FITS table interface does not support header keyword inheritance.
- FITS tables must reside in a FITS table extension, in either ASCII form (XTENSION=TABLE) or binary form (XTENSION=BINTABLE).

For example, running **catfits** on the NICMOS association table `n3tc01010_asn.fits` provides the following output:

```
fi> catfits n3tc01010_asn.fits


EXT#  FITSNAME      FILENAME               EXTVE ...
0     n3tc01010_asn N3TC01010_ASN.FITS ...
1       BINTABLE    ASN                  1 ...
```

Extension number 1 holds the association table, which has EXTNAME=ASN and EXTVER=1. You can use the **tprint** task in the **STSDAS tables** package to print the contents of this table, and the following commands are all equivalent:

```
tt> tprint n3tc01010_asn.fits
tt> tprint n3tc01010_asn.fits[1]
tt> tprint n3tc01010_asn.fits[asn,1]
```

STSDAS tables tasks can read both FITS TABLE and BINTABLE extensions, but they can write tabular results only as BINTABLE extensions. Tasks that write to a table in-place (e.g., **tedit**) can modify an existing FITS extension, and tasks that create a new table (e.g., **tcopy**) will create a new extension when writing to an existing FITS file. If the designated output file does not already exist, the task will create a new FITS

file with the output table in the first extension. If the output file already exists, the task will append the new table to the end of the existing file; the APPEND option necessary for appending FITS image extensions is not required. As with FITS images, you can specify the EXTNAME and EXTVER of the output extension explicitly, if you want to assign them values different from those in the input HDU. You can also specify the `OVERWRITE` option if you want the output table to supplant an existing FITS extension. For example, you could type:

```
tt> tcopy n3tc01010_asn.fits out.fits[3][asn,2,overwrite]
```

This command would copy the table in the first extension of `n3tc01010_asn.fits` into the third extension of `out.fits`, while reassigning it the EXTNAME/EXTVER pair `[asn,2]` and overwriting the previous contents of the extension. Note that overwriting is the only time when it is valid to specify an extension, EXTNAME, and an EXTVER in the output specification.

Specifying Arrays in FITS Table Cells

A standard FITS table consists of columns and rows forming a two-dimensional grid of cells; however, each of these cells can contain a data array, effectively creating a table of higher dimensionality. Tables containing extracted STIS and COS spectra take advantage of this feature. Each column of a STIS or COS spectral table holds data values corresponding to a particular physical attribute, such as wavelength, net flux, or background flux. For STIS, each row contains data corresponding to one spectral order, and tables holding echelle spectra can contain many rows. Each cell of such a spectral table can contain a one-dimensional data array corresponding to that cell's physical attribute and spectral order.

In order to analyze tabular spectral data with **STSDAS** tasks other than **sgraph** and **igi** (which have been appropriately modified to handle three-dimensional data tables), you will need to extract the desired arrays from the three-dimensional table. Two **IRAF** tasks, named **tximage** and **txtable**, can be used to extract the table-cell arrays. Complementary tasks, named **tiimage** and **titable**, will insert arrays back into table cells. The task **tscopy** will copy rows, columns, and subsets of tables. To specify the arrays which should be extracted from or inserted into the table cells, you will need to use the *selectors* syntax to specify the desired row and column. The general syntax for selecting a particular cell is:

```
intable.fits[extension number][c:column_selector][r:row_selector]
or
intable.fits[keyword options][c:column_selector][r:row_selector]
```

A *column selector* is a list of column patterns separated by commas. The column pattern is either a column name, a file name containing a list of column names, or a pattern using the **IRAF** pattern matching syntax (type "`help system.match`" for

a description of the **IRAF** pattern matching syntax). To obtain a list of the column names, you can run the **tlcol** task (type "`tlcol infile.fits`").

A row selector parameter can be used to specify a certain row in the table. For example, if you specify:

```
infile.fits[3][c:WAVELENGTH,FLUX][r:SPORDER=(68:70)]
```

IRAF will extract data from the table stored in the third extension of `infile.fits`, specifically from the columns labeled WAVELENGTH and FLUX, and will restrict the extraction to the rows where the spectral order (SPORDER) is within the range 68–70, inclusive. Alternatively, if you specify:

```
infile.fits[sci,2][c:FLUX][r:row=(20:30)]
```

IRAF will obtain data from the table stored in the FITS file extension with an EXTNAME=SCI and EXTVER=2. The data will come from the column FLUX and be restricted to the row numbers 20–30, inclusive. All **STSDAS** and TABLES tasks are now able to use row and column selection. For a complete explanation of the table selector syntax, type "`help selectors`".

2.3 GEIS File Format

GEIS format[2] is the standard format for reducing data from FGS, FOC, FOS, GHRS, HSP, WF/PC-1, and WFPC2. Data from these instruments are distributed by the HDA in waivered FITS files and must be converted to GEIS format. Note that WFPC2 data is now available from the HDA in waivered FITS and MEF format files[2].

All *HST* images in GEIS format consist of two components: a *header file* (with suffix ending in "h"), and a separate *binary data file* (with suffix ending in "d"). Both files must reside in the same directory for processing.

GEIS header files (e.g., `w0lo0105t.c1h`), consist entirely of ASCII text in fixed-length records of 80 bytes. These records contain header keywords that specify the properties of the image itself and the parameters used in executing the observation and processing the data.

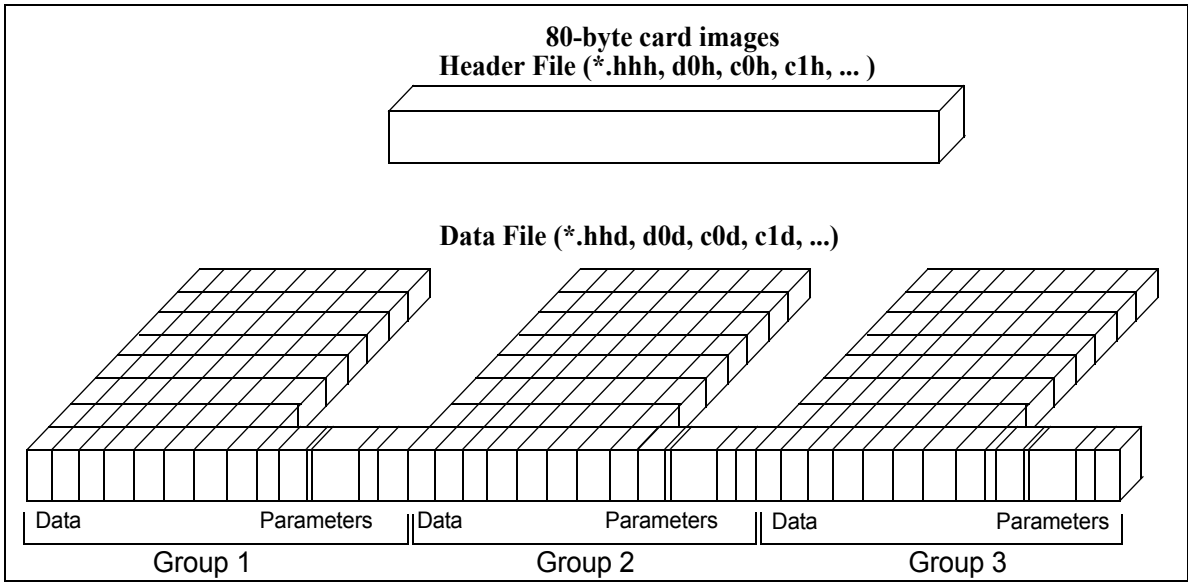
GEIS binary data files, (e.g., `w0lo0105t.c1d`), contain one or more *groups* of binary data. Each group comprises a data array followed by an associated block of binary parameters called the Group Parameter Block (GPB). Each group of a GEIS file has identical array sizes, data types, and group parameters. Figure 2.2 depicts the structure of a GEIS data file graphically.

2. GEIS files are also commonly referred to as **STSDAS** images.

The three-letter identifier (e.g., d0h) that follows the rootname of a GEIS format HST data file (see Chapter 5 for more on HST file names) has often been called an "extension" in the past. However, because of the potential for confusion with FITS extensions, this handbook will refer to these three-letter identifiers as "suffixes."

The binary content of GEIS files is machine dependent. Copying GEIS files directly from one platform to another (e.g., from a Mac to a Sun) may result in unreadable data. The task STRFITS can be used to convert GEIS to waivered FITS format before transferring.

Figure 2.2: GEIS File Structure



2.3.1 Converting waivered FITS to GEIS

The HDA stores and distributes datasets from FGS, FOC, FOS, GHRS, HSP, WF/PC-1, and WFPC2 in waivered FITS format.

We highly recommend that users convert waivered FITS datasets back into their native GEIS format before processing them.

Your data must be in GEIS format for you to use the **STSDAS** software tools developed specifically for analysis of these data. It is important to use the **strfits** task found in **stsdas.fitsio** or in **tables.fitsio** to perform the conversion from waivered FITS format to the GEIS format. A special convention is used to map GEIS format to waivered FITS format. While other FITS readers may be able to read portions of the data correctly, they are unlikely to reconstruct the entire data file properly.

To recreate the original multi-group GEIS file using **strfits**, you must first type:

```
cl> set imtype="hhh"
```

This command tells **IRAF** to write output files in GEIS format. You then need to set the **strfits** parameters `xdimtogf` and `oldirafname` both to "yes". For example, after you have set `imtype = hhh`, you can convert the FITS file `*_hhf.fits` into the GEIS format files `*.hhh` and `*.hhd` by typing:

```
cl> strfits *_hhf.fits "" xdim=yes oldiraf=yes
```

As another example, the waivered FITS WFPC2 dataset `u6n20101m_clf.fits` can be converted using **strfits** to created two GEIS files: `u6n20101m.clh` (a header file) and `u6n20101m.cld` (a data file).

2.3.2 GEIS Data Groups

One of the original advantages of GEIS format was that it could accommodate multiple images within a single file. This feature is useful because a single *HST* observation often produces multiple images or spectra. For example, a single WF/PC-1 or WFPC2 exposure generates four simultaneous images, one for each CCD chip. Likewise, a single FOS or GHRS dataset may comprise many spectra. The data corresponding to each CCD (for WF/PC-1 or WFPC2), or each readout (FOS) or bin (GHRS), are stored sequentially in the groups of a single GEIS binary data file. The header file corresponding to this data file contains the information that applies to the observation as a whole (i.e., to all the groups in the image), and the group-specific keyword information is stored in the group parameter block of each data group in the binary data file.

The *number* of groups produced by a given observation depends upon the instrument configuration, the observing mode, and the observing parameters. Table 2.4 lists the *contents* and the number of groups in the final calibrated image for the most commonly-used modes of each instrument that uses the GEIS data format.

Table 2.4: Groups in Calibrated Images by Instrument and Mode for instruments using GEIS format.

Instrument	Mode	Number of Groups	Description
FGS	All	7	FGS data are not reduced with **IRAF** and **STSDAS**. Therefore, FGS groups have different meaning than for the other instruments.
FOC	All	1	All FOC images have only a single group.
FOS	ACCUM	*n*	Group *n* contains accumulated counts from groups (subintegrations) 1, 2, ... *n*. The last group is the full exposure.
	RAPID	*n*	Each group is an independent subintegration with exposure time given by group parameter EXPOSURE.
HSP	All	1	HSP datasets always have only a single group that represents either digital star data (`.d0h`, `.c0h`), digital sky data (`.d1h`, `.c1h`), analog star data (`.d2h`, `.c2h`), or analog sky data (`.d3h`, `.c3h`).
GHRS	ACCUM	*n*	Each group is an independent subintegration with exposure time given by group parameter EXPOSURE. If FP-SPLIT mode was used, the groups will be shifted in wavelength space. The independent subintegrations should be coadded prior to analysis.
	RAPID	*n*	Each group is a separate subintegration with exposure time given by group parameter EXPOSURE.
WF/PC-1	WF	4	Group *n* represents CCD chip *n*, e.g., group 1 is chip 1 (unless not all chips were used). Group parameter DETECTOR always gives chip used.
	PC	4	Group *n* is chip *n* + 4, e.g., group 1 is chip 5. If not all chips were used, see the DETECTOR parameter which always gives the chip used.
WFPC2	All	4	Planetary chip is group 1, detector 1. Wide Field chips are groups 2–4 for detectors 2–4. If not all chips were used, see the DETECTOR keyword.

2.3.3 Working with GEIS Files

This section briefly explains how to work with information in GEIS header and data files.

GEIS Headers

Header keyword information relevant to each group of a GEIS file resides in two places, the primary header file and the parameter block associated with a group. Because GEIS header files are composed solely of ASCII text, they are easy to view or print using standard Unix text-handling facilities. However, the group parameters are

stored in the binary data file. To access them you need to use an **IRAF** task such as **imheader**, as shown in the section titled "Printing Header Information".

You can use the **IRAF hedit** task to edit the keywords in GEIS headers. While it is possible to edit GEIS header files using standard Unix text editors, you must maintain their standard 80-character line length. The **hedit** task automatically preserves this line length. If you need to add or delete group parameters, you can use the **STSDAS groupmod** task in the **stsdas.hst_calib.ctools** package. The **STSDAS chcalpar** task is useful for updating header keywords containing calibration switches and calibration reference files.

Always edit headers using tasks like hedit, eheader, groupmod, or chcalpar. Editing headers with a standard text editor may corrupt the files by creating incorrect line lengths.

GEIS Data Files

Numerous **IRAF/STSDAS** tasks exist for working with GEIS images. Most of these tasks operate on only one image at a time, so you usually need to specify the GEIS file group to be processed. If you do not specify a group, the task will operate on the first group by default.

Specifying a Group

To specify a particular group in a GEIS file, append the desired group number in square brackets to the file name (e.g., `z2bd010ft.d0h[10]`). For example, to apply the **imarith** task to group 10 of a GEIS image, type the following:

```
cl> imarith indata.hhh[10] + 77.0 outdata.hhh
```

(Always refer to a GEIS file by its header file name, with suffix ending in "h", even though mathematically you are operating on the data portion.)

The command above will add 77.0 to the data in group 10 of the file `indata.hhh`, and will write the output to a new single-group file called `outdata.hhh`. Any operation performed on a single group of a multi-group GEIS file results in an output file containing a single group.

Specifying an Image Section

If you wish to process only part of an image, you can specify the image section after the group specification in the following manner:

```
cl> imarith indata.hhh[2][100:199,200:399] * 32.0 outdata.hhh
```

This command extracts a 100 by 200 pixel subsection of the image in the second group of the file `indata.hhh`, multiplies it by a factor of 32.0, and stores the result in a new output file, `outdata.hhh`, which is a 100 by 200 pixel single group GEIS file.

An image section of one group of a GEIS image may be overwritten or operated upon, leaving the rest of the image intact. For example, the following two commands will first create `outdata.hhh` and then overwrite a section of it:

```
cl> imarith indata.hhh * 0.0 outdata.hhh
```

```
cl> imarith indata.hhh[2][100:199,200:399] * 32.0 \
>>> outdata.hhh[100:199,200:399]
```

Printing Header Information

As for MEF files, the task **imheader** extracts and prints information about a GEIS image. This task prints the image name, dimensions (including the number of groups), pixel type, and title of the image when it is run in default mode. For example:

```
cl> imhead indata.hhh
indata.hhh[1/64][500][real]: INDATA[1/64]
```

The output line indicates that `indata.hhh` is a multi-group GEIS file which contains 64 groups of data, each consisting of an array 500 pixels in length. The data type of the values is real (floating point). Note that since no group designation was provided, the task defaulted to the first group. To reveal more information regarding group 10, you can type:

```
cl> imhead indata.hhh[10] long+ | page
```

This will generate a long listing of both the ASCII header parameters in the `*.hhh` file and the specific group parameters for group 10 of the `*.hhd` file.

Other Group-Related Tasks

Currently, **IRAF** tasks and many **STSDAS** tasks cannot simultaneously process all the groups in an input image and write the results to corresponding groups in an output image in one step. However, there are several **STSDAS** tasks, particularly in the **toolbox.imgtools** and **hst_calib.ctools** packages, written to support group format data. Please refer to the *STSDAS User's Guide* for more details about working with GEIS images.

2.3.4 Waivered FITS Format

File formats for the first and second generation *HST* instruments (FGS, FOC, FOS, HSP, WF/PC-1, GHRS, and WFPC2) were developed before the standardization of MEF format. The waivered FITS format was developed in response to the need for a machine independent storage format for these data and was based on the idea of stacking multi-group GEIS data as a new dimension in a FITS image.

For example, a WFPC2 science data GEIS file with four groups has four 800x800 pixel images in its data file. When this GEIS file is converted to a waivered FITS file (using the **IRAF** task **stwfits**), the resulting FITS file has the dimensions of 800x800x4 (a three-dimensional image!) in its primary HDU. Similarly, an FOS GEIS file may have 40 groups, each group being a one-dimensional image (spectrum) that is 2064 pixels in length. The waivered FITS file equivalent of this FOS GEIS file has one 2D image of the size 2064x40 as its primary HDU.

In the case of a 4-group WFPC2 image, the first extension of the waivered FITS file is a table containing four rows. Each row represents a group. Each column in the table will correspond to a group keyword. Each element in the table contains keyword values for a specific group. This can be viewed using the **tread** command:

```
st> tread u2eo030ft_c0f.fits[1]
```

You can also display the values of specific keywords using a command like **tdump**, which in the example below, writes the values to a file called “params.txt”:

```
st> tdump u2eo030ft_c0f.fits[1] columns="PHOTMODE, CRVAL1,\
CRVAL2, BIASEVEN, BIASODD" datafile=params.txt
```

```
File "params.txt"
WFPC2, 1, A2D15, F487N CAL  204.716  70.286  295.994  295.966
WFPC2, 2, A2D15, F487N CAL  204.667  70.283  318.476  318.966
WFPC2, 3, A2D15, F487N CAL  204.680  70.304  303.804  303.966
WFPC2, 4, A2D15, F487N CAL  204.742  70.300  306.539  306.966
```

The data component of a multi-group GEIS file, when converted to waivered FITS, is stored in the primary HDU of the waivered FITS image as a multi-dimensional image. The task **display** can be used to view one group image at a time. For instance, to view group 2 of a 4-group waivered FITS WFPC2 image, type:

```
st> display u2eo030ft_c0f.fits[0][*,*,2]
```

The **display** task reads the image from the primary HDU, and specifies the image using three-dimensional syntax, where the "*" represents all pixels in *x* and *y*.

If you want to view the central 400x400 section of the WFPC2 image, you can use the following command:

```
t> display u2eo030ft_c0f.fits[0][200:600,200:600,2]
```

It is STRONGLY recommended that all waivered FITS files be converted back to GEIS format, by using the task strfits, before further processing and analysis with IRAF/STSDAS tasks.

CHAPTER 3:

Analyzing *HST* Data

In this chapter...

Chapters 2 and 3 assume that the reader is familiar with basic IRAF/STSDAS syntax. If terms such as `login.cl`, **epar**, **lpar**, and **apropos** are not familiar to the reader, we recommend skipping to the IRAF Primer included in Chapter 4.

3.1 Analysis Options for *HST* Data

HST data can be manipulated with several different software packages. In this section we introduce a few of the software language options.

3.1.1 IRAF/STSDAS

STSDAS is an **IRAF** package developed by STScI for the reduction and analysis of *HST* data. The package contains 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. Sections 3.2 through 3.7 introduce the basics of **STSDAS**, illustrating how to display *HST* data, presenting some simple data manipulations, and pointing you towards more sophisticated tasks.

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**, you need to know

the basics of **IRAF**. If you are not already familiar with **IRAF**/**PyRAF**, consult the **IRAF**/**PyRAF** Primer in Chapter 4 before reading further.

3.1.2 PyRAF

PyRAF is a command language for **IRAF** that is based on Python. It has a number of advantages over the **IRAF** CL. Most importantly, with few exceptions, it allows use of exactly the same syntax as the **IRAF** CL. Some of the advantages that it provides are:

- true command line recall (with arrow key editing)
- command and filename completion
- GUI-based graphics windows, previous plot recall, multiple graphics windows
- a GUI epar parameter editor with help displayed in a separate window
- IDL-like capabilities
- true error handling for scripts (shows which line of a script fails when errors occur)
- can script **IRAF** tasks in Python language
- exception handling capability

Since **PyRAF** is so highly compatible with the **IRAF** CL, virtually all of the examples shown in this handbook will work the same for **PyRAF**. Minor differences include the user prompt and the graphics windows appearance.

More information on **PyRAF** can be found at:

http://www.stsci.edu/resources/software_hardware/pyraf

3.1.3 Python

Python is used for astronomical data reduction applications. It is a freely available, general-purpose, dynamically-typed interactive language that provides modules for scientific programming and is used for astronomical data reduction application. These modules include:

- numpy: IDL-style array manipulation facilities
- PyFITS: read and write FITS files to and from arrays
- matplotlib: plotting and image display package
- numdisplay: display arrays to SAOimage, DS9, and Ximtool
- **PyRAF**: run **IRAF** tasks from Python

Python is a very powerful language that is well suited to writing programs to solve many needs beside scientific analysis. Tools are available to read (but currently not write) GEIS files.

Python can make use of **PyRAF** to allow access to **IRAF** tasks. Tutorials are available which illustrate the use of Python for interactive data analysis in astronomy (in much the same style as is now popular with IDL). The initial focus of these tutorials is the use of interactive tasks for the novice user. The more advanced tutorials focus on teaching the details of Python programming. The tutorials can be downloaded from:

`http://www.scipy.org/Topical_Software`

STScI is developing most of its new calibration and data analysis software in Python. More information on the use of Python to analyze *HST* data can be obtained from:

`http://www.stsci.edu/resources/software_hardware`

3.1.4 Interactive Data Language (IDL)

IDL is an array-based, interactive programming language that provides many numerical analysis and visualization tools. It is typically much easier to develop new analysis and visualization applications and utilities in IDL than in Fortran or C. As a result, it is very popular in the astronomical community with many astronomers using it for their analysis of *HST* data.

It can be obtained from ITT Visual Information Solutions (http://www.ittvis.com/idl/), for a fee. Libraries for reading *HST* data are part of the freely available ASTRON library (http://idlastro.gsfc.nasa.gov) which has links to other IDL astronomy libraries.

3.1.5 Fortran and C

For those who wish to write their own Fortran or C applications, we recommend using the FITSIO library for reading FITS files (http://heasarc.gsfc.nasa.gov/docs/software/fitsio/fitsio.html; note that the C library is called CFITSIO).

This library does not support GEIS format directly so users will need to use the waivered FITS format obtained from the archive and manually extract the needed information.

3.1.6 Java

The most widely used FITS libraries for Java are the Java FITS Utilities (http://heasarc.gsfc.nasa.gov/docs/heasarc/fits/java/v1.0/) and the Java FITS Class Library (http://www.eso.org/~pgrosbol/fits_java/jfits.html). These libraries do not support GEIS format, but can handle waivered FITS format.

3.2 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:

- `apropos` *word* - searches the online help database for tasks relating to the specified word (see Figure 4.4).
- `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.
- `examples` *task* - provides examples of each task.

3.2.1 STSDAS Structure

STSDAS is structured so that related tasks are grouped together as packages. For example, tasks used in the calibration process can be found in the **hst_calib** package, and tasks used for image display and plotting can be found in the **graphics** package. Table 3.1 shows the current **STSDAS** package structure. The current version of **IRAF** (v2.15 as of November 2010) and TABLES (v3.13 as of January 2011) must be installed on your system in order to use the latest version of **STSDAS**. It is always a good idea to make sure you have the most recent versions of the software as they often include important code updates. Newer versions of IRAF, STSDAS, and TABLES are released approximately every 6 months to provide extensively tested updates for support of new instruments or to correct bugs in previous versions. Bug fixes and new versions of this package in the process of being tested can be downloaded as the IRAFX environment from:

`http://stsdas.stsci.edu/irafx/`

These pre-release versions are updated weekly to include new code and bug fixes which have not been extensively tested or verified for accuracy by the Institute. They also require a very different installation procedure and can be installed without root access, but are only provided as binaries for the Mac OSX Leopard and Linux Red Hat Enterprise 4 operating systems.. Please check the following Web site for the latest information:

`http://www.stsci.edu/resources/software_hardware/`

Table 3.1: **STSDAS** Version 3.13 Package Structure

analysis	**Data analysis package.**
dither	Dithered image combination.
fitting	Curve fitting tools.
fourier	Fourier analysis.
gasp	Access the *HST* Guide Star Catalog on CD-ROM.
isophote	Elliptical isophote image analysis.
nebular	Tasks for analyzing nebular emission lines
restore	Deconvolve or filter 1- or 2-dimensional images.
statistics	Statistical analysis software.
slitless	Slitless spectroscopy packages (e.g. aXe)
contrib	**User-contributed software.**
acoadd	Image coaddition using lLucy Method with optional acceleration
gaussfit	Least squares and robust estimation program
plucy	Multiple channel photometric image restoration
slitless	Extract a spectrum from a direct/Grism image pari
redshift	Tasks for determining redshifts and dispersions.
spfitpkg	Fitting spectra with non-linear chi-square minimization.
vla	Spectral image reduction for VLA data.
fitsio	**FITS and GEIS input/output for *HST* data (images and tables).**
graphics	**Graphics and image display package.**
sdisplay	Image display package for SAOImage display device.
stplot	General plotting utilities.
hst_calib	***HST* Science Instrument calibration package.**
acs	Tasks for calibrating ACS data.
ctools	General calibration tools.
foc	Tasks for calibrating FOC data.
focprism	FOC prism package.
fos	Tasks for calibrating FOS data.
spec_polar	Tasks for reducing and analyzing FOS polarimetry.
hrs	Tasks for calibrating HRS data.
nicmos	Tasks for calibrating NICMOS data.
mstools	General-purpose tasks that handle NICMOS imsets
hstcos	Tasks for calibrating COS data.
paperprod	Tasks for generating paper products.
stis	Tasks for calibrating STIS data.
synphot	Synthetic photometry and modelling instrument response.
simulators	Synthetic photometry simulation package.
wfc3	Tasks for calibrating WFC3 data.
wfpc	Tasks for calibrating WF/PC-1 and WFPC2 data.
w_calib	Tasks for deriving the WF/PC-1 instrument calibration.
playpen	**Miscellaneous experimental tasks.**

sobsolete	**Package of tasks that have been retired. (a sampling below)**
foccs	FOC calibration software package.
focgeom	FOC geometry package.
focphot	FOC photometry package.
focutility	Obsolete FOC utility package.
hsp	Tasks for calibrating HSP data.
olddither	Older version (V1.2) of dither.
registration	Compute registration parameters and resample unaligned data files.
testdata	Tools for creating artificial images.
timeseries	Time series photometry data reduction and analysis.
y_calib	Tasks supporting the FOS calibration process.
z_calib	Tasks supporting the HRS calibration process.
toolbox	**General tools package.**
convfile	Reformat images between VAX and Sun.
headers	Tools for modifying image headers.
imgtools	Tools for manipulating & examining images and bad pixel lists.
mstools	Tasks to handle MEF IMSETs.
tools	Generic data handling and utility tools.
ttools	Table manipulation tools.

3.2.2 Packages of General Interest

For 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 **IRAF** package includes general tasks for copying (**imcopy**), moving (**imrename**), deleting (**imdelete**), displaying (**tv**), and examining (**imexamine**) 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 image statistics, rotating and magnifying images, and registering and dewarping images.
- **stsdas.toolbox.imgtools**: This package contains tools for working with 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**), and **gcombine** for coadding GEIS images. Also of note are the tasks **imcalc** for performing image arithmetic, and **rd2xy** and **xy2rd** for converting between RA, Dec and x,y pixel coordinates. Many of these tasks will also work with single group or waivered FITS format files.
- **stsdas.toolbox.imgtools.mstools**: This package contains tools for working with FITS image files, in particular ACS, NICMOS, STIS, and WFC3 image sets (imsets). **Msstatistics**, for example, will print statistics on each image set (imset) in an image file. Similarly, **msarith** can be used for image arithmetic and uses information from the Data Quality (DQ), Error (ERR), and if available, the TIME and SAMP extensions in the calculation.

- **stsdas.analysis.dither.multiDrizzle**: This routine is provided as a "one-touch" interface for performing all the tasks necessary for registering dithered *HST* images, performing cosmic ray rejection, removing geometric distortion and performing the final image combination using "**drizzle**". We are currently developing a replacement for MultiDrizzle which will rely on an extended image header for all the astrometric and distortion information used to geometrically correct and align the images. This new task will be announced as soon as a version has been successfully tested with data from the current imaging instruments on *HST*.
- **stsdas.analysis**: This package contains general tasks for image analysis, such as Fourier analysis (**fourier**), dithering (**dither**), and **fitting**.

For 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 supported, but only for input.) The *STSDAS User's Guide* describes the **STSDAS** table format in 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.

3.3 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

3.3.1 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 DS9, in a separate window from your **IRAF** session, either from a different xterm window or as a back-

ground job before starting **IRAF**. To start DS9, type the following in a Unix window:

```
> ds9 &
```

Several different display servers, including SAOimage, DS9 (the next generation of SAOimage), and Ximtool, can be used with IRAF. DS9 may be retrieved from http://hea-www.harvard.edu/RD/ds9/. Ximtool may be retrieved from http://iraf.noao.edu.

2. Make sure that **IRAF** has been set to expect the size of the image you wish to display. This is controlled with the `stdimage` keyword. As long as this is set to the largest image size you would like to display, the entire image will be available in your viewer. Otherwise, **IRAF** will only display up to the currently set limit. If you would like a general setting that will display images up to 8192 × 8192, set `stdimage` equal to `imt7`, otherwise, set `stdimage` equal to the length of one side of the image. Other types of images may require larger formats, like `imt4096` for ACS and WFC3 images, for example, while `imt1024` is often adequate for WFPC2 and NICMOS images. To set the size of the image buffer to 1024 × 1024, or the size of the image you would like to display, type the following in the **IRAF** window:

```
cl> set stdimage = imt1024
```

3. Load the **images.tv** package from the window where you are running **IRAF**:

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

Note that parameters, like `stdimage`, and frequently used tasks, like **tv,** are generally defined in the `login.cl` file.

4. Display the image in frame 1 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. Figure 3.1 shows how to display chip 1 of an ACS/WFC 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.

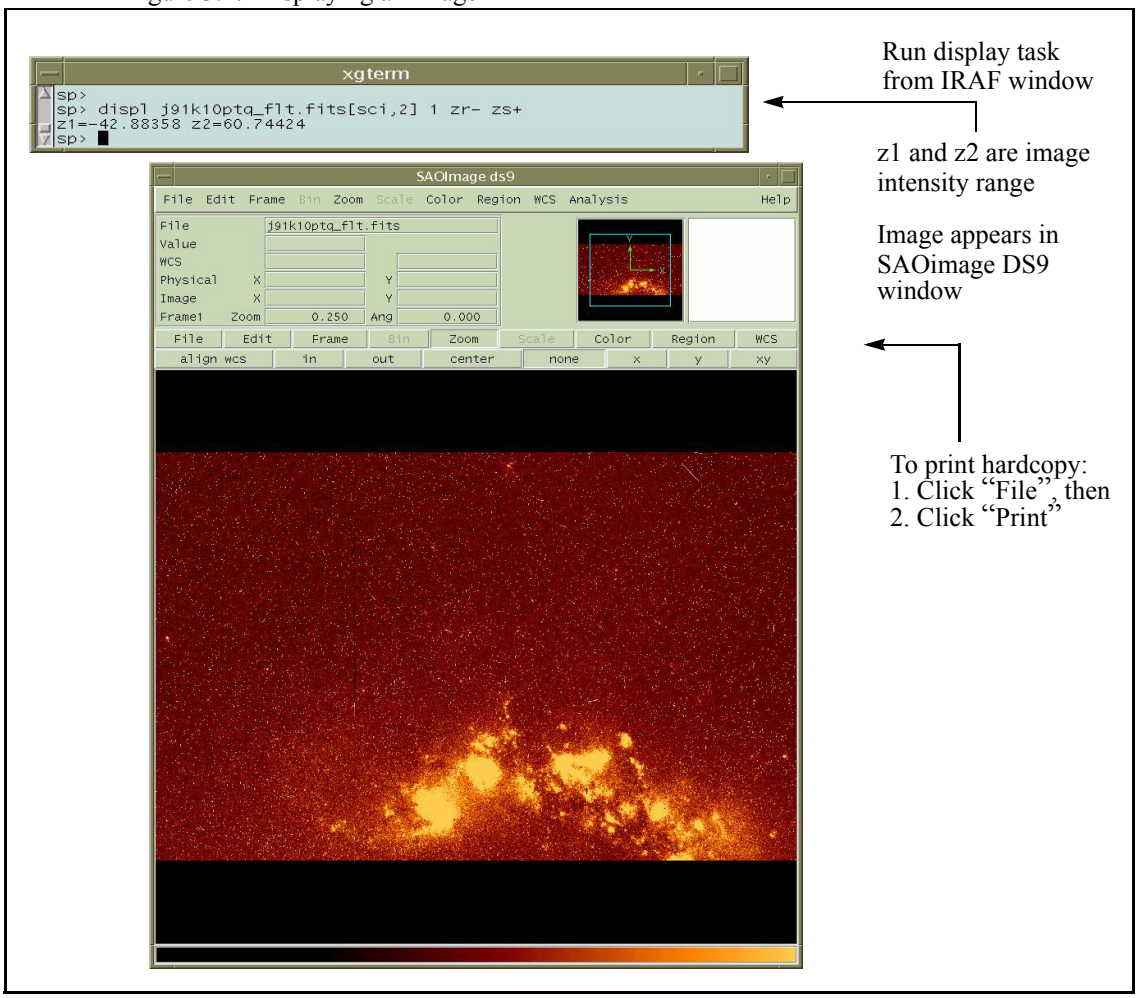
Modifying the Display

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, either on the command line or using the **epar** command.

Once an image appears in your DS9 window, you can use the SAOimage commands displayed near the top of the image window to manipulate or print your image. The *SAOimage User's 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. Online help is also available at the system level: type `man saoimage` in Unix.

Figure 3.1: Displaying an Image



The example in Figure 3.1 shows how you could 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 intensity scaling using the `zscale`, `zrange`, `z1` and `z2` parameters. The `zscale` parameter toggles the auto scaling. Setting `zrange+` (and `zscale-`) tells the task to display the image using the minimum and maximum values in the image. To customize your minimum and maximum intensity display values, set `z1` to the minimum value and `z2` to the maximum value that you want displayed. You must also set `zscale-` and `zrange-` to disable these parameters. To ensure the entire image is displayed, you should set the `fill+` option. For example:

```
im> disp j91k10ptq_flt.fits[sci,2] 1 zrange- zscale- z1=2.78
z2=15.27 fill+
```

Notice in Figure 3.1 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 greater dynamic range, you may prefer to use logarithmic scaling. However, the log scaling function in DS9 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 a 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`. Otherwise, you can simply use:

```
im> display x2ce0502t.c1h ztrans=log
```

3.3.2 Working with Image Sections

To display only a portion of an image, use 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, type:

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

To specify the same pixel range in the second SCI extension of a NICMOS FITS image:

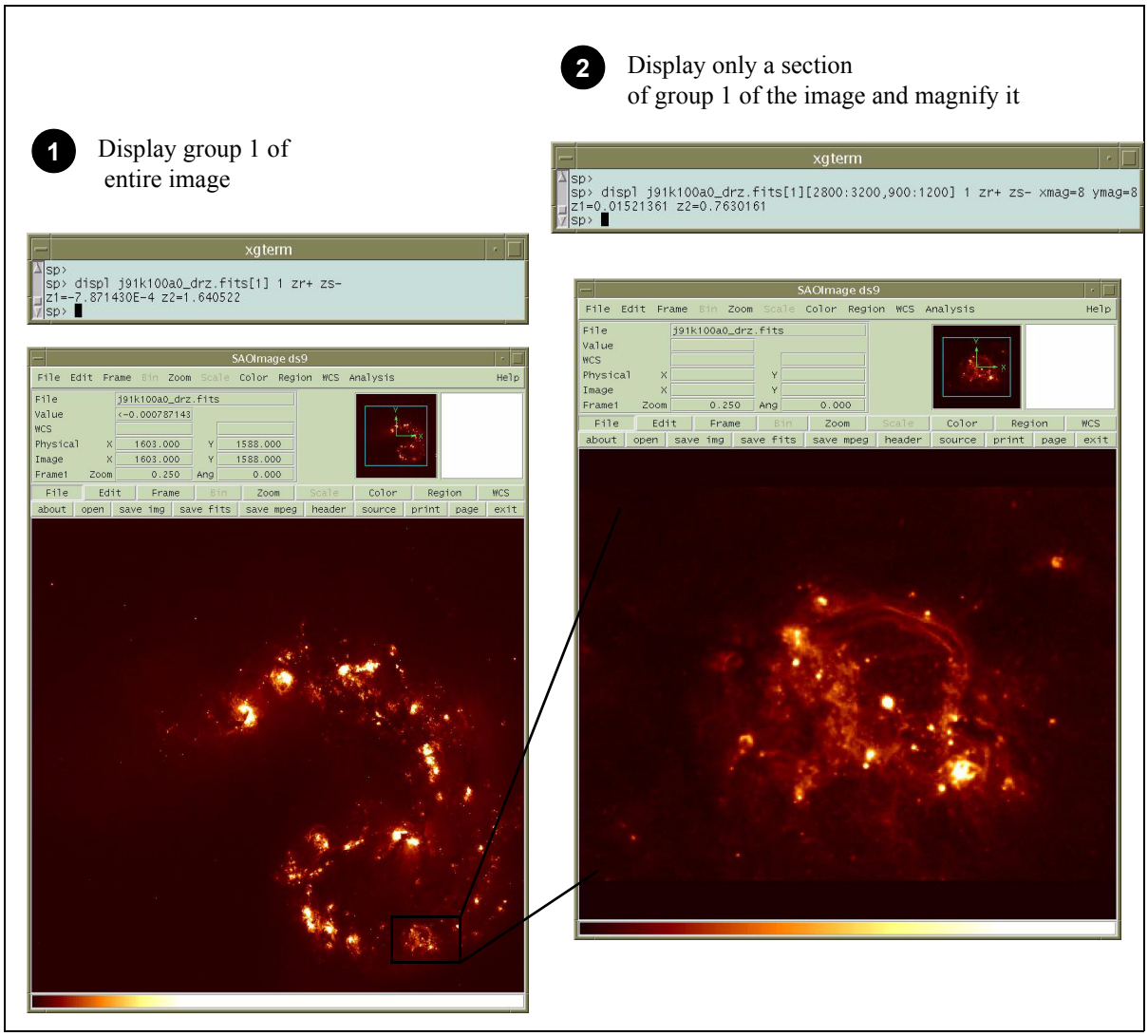
```
tv> display image.fits[sci,2][101:200,*] 1
```

If you specify both a group and an image section of a GEIS file, the group number must come first. When displaying sections of FITS image extensions, you must specify the extension number followed by the image section.

Figure 3.2 shows examples of displaying an image and an image section.

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

Figure 3.2: Displaying Sections and Groups of an Image



3.4 Analyzing *HST* Images

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

- Relating your image to sky coordinates
- Examining and manipulating your image
- Working with Multi-Extension FITS imsets
- Converting counts to fluxes

3.4.1 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: reference pixel values (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 contours and grid of RA and Dec. 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.) DS9 displays the current *x,y* pixel location of the cursor in the upper-left corner of the window. These tasks will only give accurate results when they are run on images which have been corrected for distortion. To find the RA and Dec of the current pixel, you supply these coordinates to **xy2rd** by typing:

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

Note, the `hms` option formats the results in hours, minutes, and seconds.

Observers should be aware that *these tasks do not correct for geometric distortion*. Only ACS, FOC, STIS, WFC3[2], and WFPC2 images currently undergo geometric

correction during standard pipeline processing. Table 3.2 lists some additional tasks that make use of the standard astrometry keywords.

Table 3.2: Additional **IRAF** and **STSDAS** Astrometry Tasks

Task	Package	Purpose
compass	stsdas.graphics.sdisplay	Plot north and east arrows on an image.
imexamine	cl.images.tv	(`rimexamine`) Mutli-purpose tool for examining images - statistics, photometry, and astrometry.
north	stsdas.hst_calib.ctools	Display the orientation of an image based on keywords.
rimcursor	cl.lists	Determine RA and Dec of a pixel in an image.
wcslab	cl.images.tv	Produce sky projection grids for images.

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. Absolute astrometry, on the other hand, 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 guide star positions. Generally, observations obtained during the same visit using the same guide star acquisition are well-registered. Observations separated by one or more guide star acquisitions will typically have small shifts. 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 find its absolute position from the Guide Star Catalog II (GSC2), which is on the IAU recommended International Celestial Reference Frame, and has a typical error of 0.3". Contact the Help Desk if you require further assistance.

2. After installation on *HST* during Servicing Mission 4.

3.4.2 Examining and Manipulating Image Data

This section describes **implot** and **imexamine**, two basic **IRAF** tools for studying the characteristics of an image. Table 3.4 lists many useful **IRAF/STSDAS** tasks for manipulating images. The list is not exhaustive, just a sample of what is available.

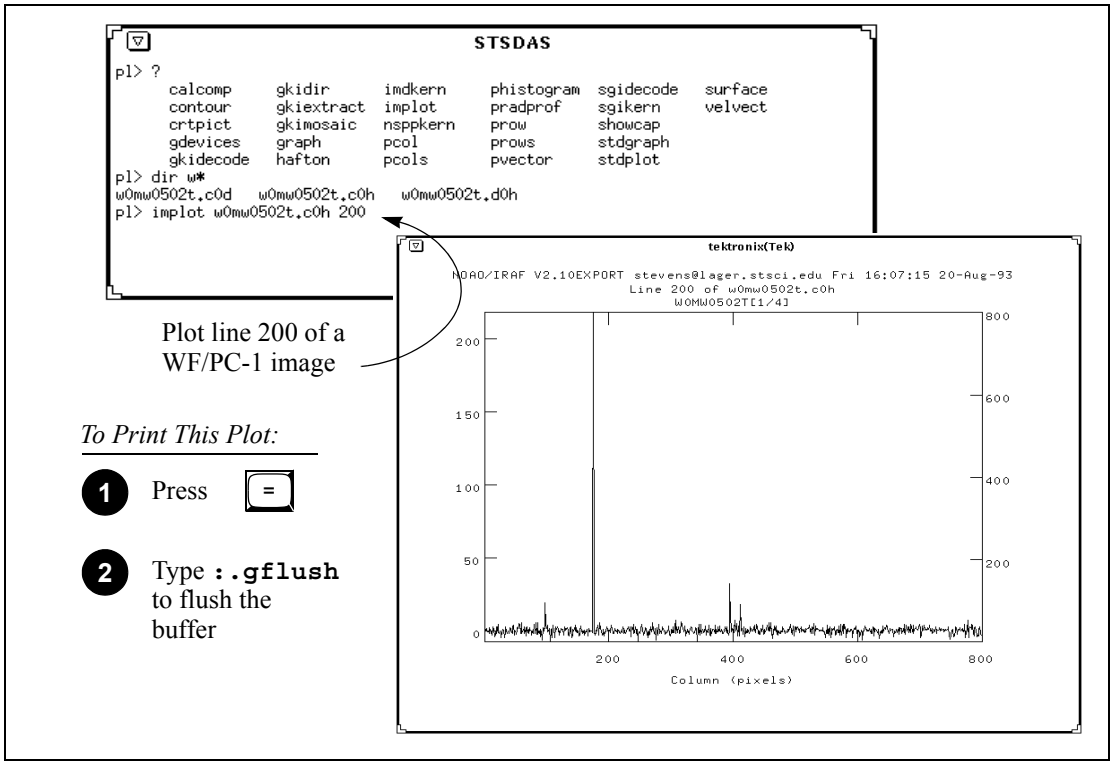
implot

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 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 online help, but those most commonly used are listed in Table 3.3. Figure 3.3 shows an example of how to use the **implot** task to plot a row of data.

Table 3.3: Basic Implot Commands

Keystroke	Command
?	Display online help.
L	Plot a line.
C	Plot a column.
J	Move down.
K	Move up.
Space	Display coordinates and pixel values.
Q	Quit implot.

Figure 3.3: Plotting Image Data with Implot (Step 2 Only Works in **IRAF**)



imexamine

The **imexamine** task (in the **images.tv** package) is a powerful **IRAF** 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`.

3.4.3 Working with FITS Imsets

ACS, COS, NICMOS, STIS, and WFC3 data files contain groups of images, called *imsets*, associated with each individual exposure. See Table 2.2 and the Data Structures chapters in Part II for more details on imsets.

Table 3.4: Image Manipulation Tasks.

Task	Package	Purpose
boxcar	images.imfilter	Boxcar smooth a list of images
blkavg	images.imgeom	Block average or sum an image
fmedian	images.imfilter	Box median filter a list of images
gcombine	stsdas.toolbox.imgtools	Combine GEIS images using various algorithms and rejection schemes[1]
gcopy	stsdas.toolbox.imgtools	Copy multigroup GEIS images[1]
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)[1]
grplot	stsdas.graphics.stplot	Plot lines of a group-format 1D GEIS image (spectrum)[1]
gstatistics	stsdas.toolbox.imgtools	Compute image statistics[1]
histogram	stsdas.graphics.stplot	Plot or list a histogram of a table column, image, or list
igi	stsdas.graphics.stplot	Interactive Graphics Interpreter plotting package
imcalc	stsdas.toolbox.imgtools	Perform general arithmetic on images
imcombine	images.immatch	Combine images using various algorithms
imedit	images.tv	Fill in regions of an image by background estimation or copy and paste
imexamine	images.tv	Examine images using display, plots, and text (see Section imexamine)
implot	plot	Plot lines and columns of images (see Section implot)
magnify	images.imgeom	Magnify an image
msarith	stsdas.toolbox.mstools	Performs basic arithmetic on *HST* FITS imsets[1]
mscombine	stsdas.toolbox.mstools	Extension of **gcombine** for *HST* FITS imsets[1]
msstatistics	stsdas.toolbox.mstools	Extension of **gstatistics** for *HST* FITS imsets[1]
newcont	stsdas.graphics.stplot	Draw contours of two-dimensional data
pixcoord	stsdas.hst_calib.wfpc	Compute pixel coordinates of stars in a GEIS image[1]
plcreate	xray.ximages	Create a pixel list from a region file (e.g., from SAOimage regions) Useful for masking of images.
rotate	images.imgeom	Rotate an image
saodump	stsdas.graphics.sdisplay	Make image and color map files from SAOimage display
sgraph	stsdas.graphics.stplot	Graph image sections or lists
siaper	stsdas.graphics.stplot	Plot science instrument apertures of *HST*
xregister	images.immatch	Register 1D or 2D images using cross-correlation

1. Will process all groups of a multi-group GEIS file.

Here we describe several **STSDAS** tasks, located in the **stsdas.toolbox.imgtools.mstools** package, that have been designed to 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 have associated errors, which will propagate into the error array(s) of the output file.

mscombine

This task runs the **STSDAS** task **gcombine** on ACS, COS, NICMOS, STIS, and WFC3 data files. It separates each imset into its basic components (e.g., SCI, ERR, DQ, SAMP, TIME). The SCI extensions then become the inputs for 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 are then combined 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 (NICMOS and WFC3/IR data only).

Once **gcombine** has finished, **mscombine** then reassembles the individual output images into imsets and outputs them as one data file. The output images and error maps from **gcombine** form the SCI and ERR extensions of the output imset(s). The DQ extension will be a combination of the masking operations and the rejection algorithms executed in **gcombine**. TIME extensions 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 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 feature is the inclusion of the error and data quality information included with *HST* FITS 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 in the imset 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 FITS 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 file containing many imsets or for performing analysis on a specific imset. The **msjoin** task does the opposite of **mssplit**: it assembles separate imsets into a single data file.

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

3.4.4 Photometry

Included in this section are:

- A list of **IRAF**/**STSDAS** tasks useful for determining source flux.
- Instructions on how to use header keyword information to convert *HST* image units 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/docs/photom.html

The apphot package allows you to measure fluxes within a series of concentric apertures. This technique can be used, for example, 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 to Units of Flux or Magnitude

Calibrated HST images obtained from the HDA store signal in various units. Table 3.5 lists a selection of *HST* instrument image units. Refer to the instrument specific Data Handbooks for instruments not included in this list.

Table 3.5: Examples of Calibrated *HST* Image Units

Instrument	Calibrated Data Units	Drizzled Data Units
ACS	e^-	e^-
COS	DN/s	N/A[1]
NICMOS	DN/s	DN/s
STIS	DN	N/A[1]
WFC3/UVIS	e^-/s	e^-/s
WFC3/IR	e^-/s	e^-/s
WFPC2	DN	DN/s

1. No drizzled products from the pipeline.

The pipeline calibration tasks do not alter the units in the images when performing the photometric correction step. Instead they calculate and write the 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. For ACS observers, the PHOTFLAM values for the two WFC chips are defined to be the same.

PHOTFLAM is defined as the *mean* flux density F_λ in units of erg cm^{-2} s^{-1} $Å^{-1}$ that produces 1 count per second in the *HST* observing mode (PHOTMODE) used for the observation. (Note that the word "counts" may refer to DN or electrons, depending on the instrument used.) For example, calibrated ACS images are already corrected for the instrumental gain, and the PHOTFLAM values are computed accordingly. The PHOTFLAM values for WFPC2, on the other hand, are dependent on the gain.

Calibrated images, in units of DN or electrons (e.g., STIS or WFPC2), may be converted to flux in units of erg cm^{-2} s^{-1} $Å^{-1}$ by multiplying the image by the value of the PHOTFLAM header keyword and dividing by the value of the EXPTIME keyword (exposure time). Calibrated images in units of signal rates (e.g., NICMOS data in DN s^{-1} and drizzled ACS data in electrons s^{-1}), may simply be multiplied by the PHOTFLAM value to obtain the flux in units of erg cm^{-2} s^{-1} $Å^{-1}$. NICMOS and WFC3/IR image headers also contain the keyword PHOTFNU in units of Janskys. Multiplying these images by the PHOTFNU value will therefore yield fluxes in Janskys.

The **STSDAS** task **imcalc** may be used to convert an 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, type:

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

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 PHOTFLAM using **synphot**, described below. WF/PC-1 and WFPC2 observers should note that the PHOTFLAM values calculated during pipeline processing do not include a correction for temporal variations in throughput owing to contamination buildup, or Charge Transfer Efficiency (CTE) effects. However, for WFPC2, new header keywords provide parameters that can be used to derive corrections for each chip, in units of magnitude, for contamination (ZP_CORR) and CTE (CTE1E2, CTE_1E3, CTE1E4). 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 512×512. Consult the instrument section (Part II) of the Data Handbook for more information.

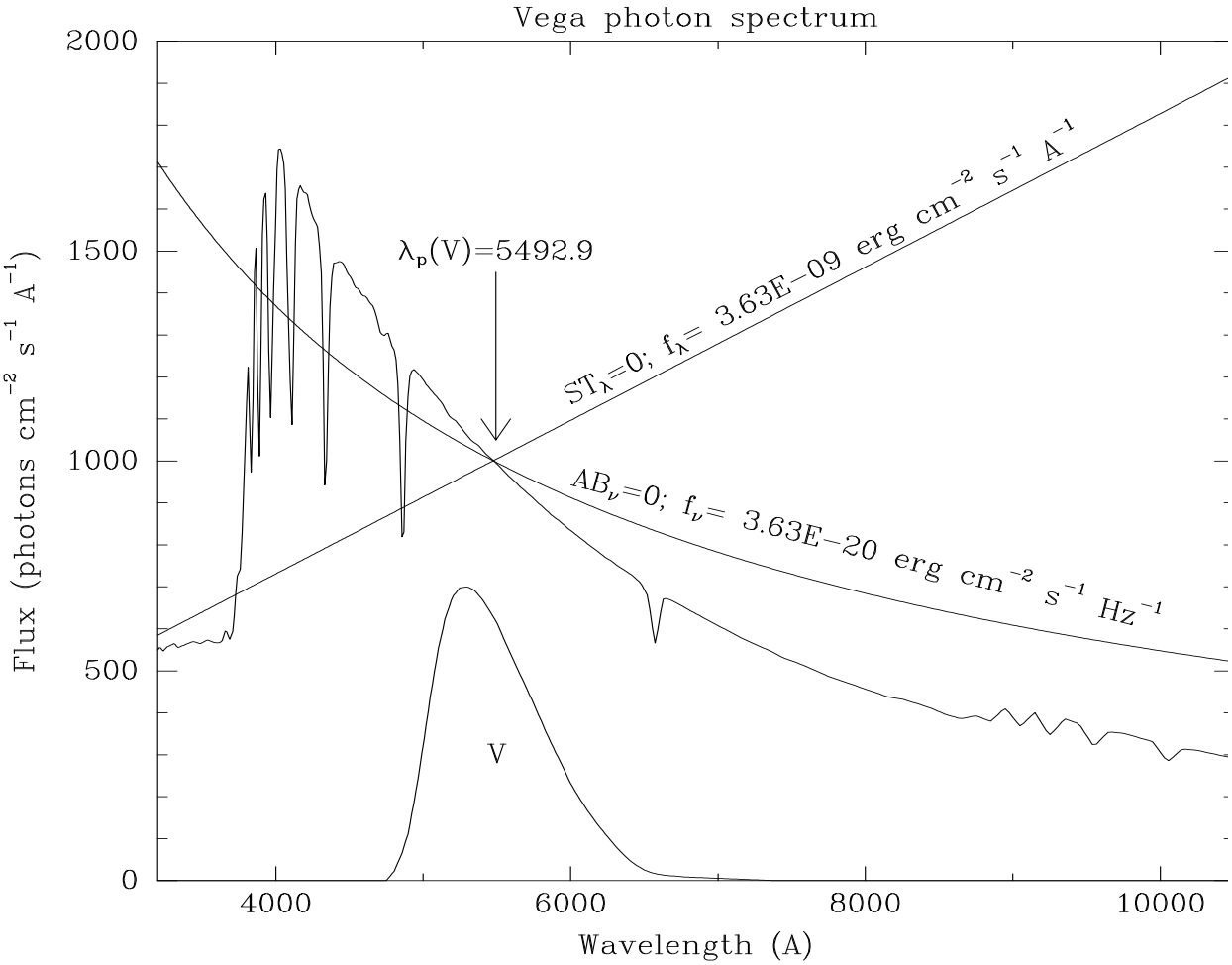
If your HST image contains a source whose flux you know from ground based measurements, you may choose to calibrate the final photometry of your HST image from the counts observed for this source.

To convert a measured flux *F*, in units of erg cm^{-2} s^{-1} $Å^{-1}$, to an ST magnitude, the following equation may be used:

$$m = -2.5 \times \log10\ (F) + \text{PHOTZPT}$$

where the value of the PHOTZPT keyword is the zero point of the ST magnitude (STMAG) scale. The STMAG system is based on constant flux per unit wavelength. The zero point of the STMAG system is equal to –21.10, a value chosen so that Vega has an ST magnitude of zero for the Johnson *V* passband (see Figure 3.4; Koornneef et al., 1986; Horne 1988; and the *Synphot User's Guide*).

Figure 3.4: Standard Photometric Systems Illustrated.



Further zeropoint corrections are necessary for converting from STMAG to other systems like Johnson/Cousins, and depend on the color of your sources. See specific photometry examples in the instrument sections of this Handbook (Part II).

Synphot

The **STSDAS** synthetic photometry package, called **synphot**, can simulate *HST* observations of astronomical targets with known spectra. It makes use of a data set that contains throughput curves of all *HST* optical components, such as mirrors, filters, gratings, apertures, and detectors, and it can generate passband shapes for any combination of these elements. It can also generate synthetic spectra of many different types of sources, including stellar, blackbody, power-law and H II regions, 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 *HST* sensitivity tables. The **bandpar** task may be used to compute the photometric parameters of a passband using the combined

throughputs of the individual *HST* components. For example, to recalculate PHOTFLAM for an ACS observation, type:

```
sy> bandpar acs,wfc1,f555w
```

where the observation mode string is a comma separated list consisting of the instrument and its configuration, in this case the ACS detector with the WFC chip 1 and the `F555W` filter. (See the **obsmode** task in **synphot** and the *Synphot User's Guide* for help with these observation mode keywords.) To see a list of observation mode keywords for the ACS, type:

```
sy> obsmode acs
```

Using the default parameters, the **bandpar** command shown above will print to the screen a table of photometric parameters. The URESP parameter contains the flux (in F_λ) of a source that produces a response of one count per second in this passband and is therefore identical to PHOTFLAM.

Please see the *Synphot User's Guide* for more details on this package. See the *Synphot Data User's Guide* and Section 4.5 for more information on how to use and obtain the **synphot** data set, which is not included with **STSDAS**.

3.4.5 Combining Dithered *HST* Datasets with MultiDrizzle

Many *HST* observations make use of the technique of dithering, or offsetting the telescope to different locations in order to move the target around the detector. This is done for several reasons, including sub-pixel offsets to improve PSF sampling, offsets to move bad pixels around to different locations on the sky, or large shifts comparable to the detector size, to create large mosaics of the target field.

The recommended software to combine dithered *HST* datasets is **MultiDrizzle** (Koekemoer et al. 2002), which is a **PyRAF** script designed to provide fully automated image registration, cosmic ray cleaning, and final image combination using the **drizzle** software (Fruchter & Hook 2002) and **PyDrizzle**. **MultiDrizzle** is currently available within **STSDAS** and has been tested on a representative set of commonly-used ACS, NICMOS, STIS, WFPC2, and WFC3 observing modes.

The only required input to **MultiDrizzle** is a list of calibrated science images. The user may also choose to provide additional input such as bad pixel masks, a delta-shift file, or a reference image. The script performs the following steps:

- Calculate and subtract a background sky value for each exposure.
- Search for additional bad pixels that are strongly negative, which may not have been flagged in the data quality arrays.

- Determine shifts from the WCS information in the image headers. If the user has supplied a delta-shift file, those shifts are applied in addition to the header offsets. The user may also choose to provide the absolute shifts to use for each image, ignoring the header information.
- Drizzle each input image onto a separate output image. All images remain in the same reference plane such that any pixel (*x*,*y*) is the same logical coordinate in each image. During the drizzling process, each image is corrected for any geometric distortion that might be present.
- Combine the separately drizzled images to create a median image, in order to obtain an estimate of the cleaned final image.
- Transform sections of the median image, corresponding to the location of each input image, back to the original distorted input frames and calculate the derivative of each image.
- Compare the original input image against the transformed median and its derivative in order to create a cosmic ray mask for each exposure.
- Combine the cosmic ray masks with any additional user-supplied bad pixel masks and finally drizzle all the original input exposures onto a single output image.

The various steps can each be turned on or off by the user, since there may be cases where not all the steps need to be run, or some of them may have already been run. In addition, parameters controlling the behavior of each step can be adjusted by the user. The default parameter values are set such that the script should produce a scientifically-useful combined, drizzled image in a single operation. However, this may not be the optimal scientific image for a given set of exposures, therefore access is provided to parameters of **drizzle** and other steps for fine-tuning the results.

Please refer to the MultiDrizzle Handbook v3.0 (Fruchter & Sosey et al. 2009), and the online help documentation for MultiDrizzle within **PyRAF**, for further information about the various parameters for the script. The handbook also contains basic examples for each of the current instruments.

In general, the code has been tested on a wide variety of the most commonly used observing modes and should produce useful results with the default parameters. Since the software is actively being improved, it is important to check for updates if you find problems with a particular dataset. Users are encouraged to send e-mail to help@stsci.edu for assistance.

We are currently developing a replacement for MultiDrizzle which will rely on an extended image header for all the astrometric and distortion information used to geometrically correct and align the images. This new task will be announced as soon as a version has been successfully tested with data from the current imaging instruments on *HST*.

3.5 Displaying *HST* Spectra

This section describes how to plot the most common *HST* spectra (COS, FOS, GHRS, and STIS) in **IRAF**/**STSDAS** for a quick first look.

We will not discuss ACS, NICMOS, or WFC3 grism or prism data. The tools for extracting, displaying and analyzing spectra from these instrument modes are discussed in the instrument sections in Part II (see sections on **aXe** for ACS and WFC3, and **NICMOSlook** for NICMOS).

3.5.1 Specview

Specview is a very useful tool for displaying and analyzing spectra from most *HST* instrument configurations in their native archival format, as well as data from a variety of other spectral instruments. It is a Java application for 1D interactive spectral visualization and analysis.

Specview was written at STScI in Java (Busko 1999) and is distributed in standalone application and applet formats. The application version requires that either the Java Development Kit (JDK) or Java Runtime Environment (JRE) be installed in your system and accessible from your path.

Specview is capable of overplotting spectra from different instruments, measuring, modelling, and fitting spectral features, spectral line identification, and it allows somewhat elaborate plot annotation. More information about Specview, together with screen shots, demos, and the software for download are available at:

`http://www.stsci.edu/resources/software_hardware/specview`

3.5.2 COS and STIS Spectra

Both COS and STIS data files retrieved from the HDA can contain spectra in two different forms: as spectral images in FITS IMAGE extensions or as extracted spectra in FITS BINTABLE extensions.

Plotting COS and STIS Imaging Spectra

You can use **sgraph** in the **graphics.stplot** package of **STSDAS** to plot spectral images by specifying the image section that contains the spectrum. For example, to plot the entire *x*-range of the calibrated two-dimensional STIS 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]
```

Similarly, to plot the calibrated two-dimensional COS spectrum in the first extension of the file `l61h54cxr_flt_a.fits`, averaging rows 451 through 480, you would type:

```
st> sgraph l61h54cxr_flt_a.fits[1][*,451:480]
```

Displaying a spectral image using the **display** task (see Section 3.3.1) allows you to see the range of your spectrum in *x*- and *y*-pixel space, so you can choose a suitable image section for plotting.

Plotting COS and STIS Tabular Spectra

To plot COS or STIS spectra in BINTABLE extensions, you first need to understand how the spectra are stored as binary arrays in FITS table cells. Section 2.2.2 discusses this format and describes the *selectors* syntax used to specify these data arrays. To specify a particular array, you must first type the file name, then the extension containing the BINTABLE, followed by the column selector, and finally the row selector.

COS Row Selector Examples

COS tabular spectra contain two or three rows corresponding to either the FUV segments of the NUV stripes. For example, to select the WAVELENGTH array corresponding to segment A of the FUV spectrum in extension 1 of `cos_fuv.fits`, you would specify the file as either:

```
cos_fuv.fits[1][c:WAVELENGTH][r:segment=FUVA]
or
cos_fuv.fits[1][c:WAVELENGTH][r:=1]
```

To select the WAVELENGTH array corresponding to stripe C of the NUV spectrum in extension 1 of `cos_nuv.fits`, you would specify the file as either:

```
cos_nuv.fits[1][c:WAVELENGTH][r:segment=NUVC]
or
cos_nuv.fits[1][c:WAVELENGTH][r:=3]
```

STIS Row Selector Examples

Each row of a STIS tabular spectrum contains a separate spectral order (first-order spectra will have one row, while echelle spectra will have many rows), and each column contains data of a certain type, such as wavelength or flux. To specify a particular array, you must first type the file name, then the extension containing the BINTABLE, followed by the column selector, and finally the row selector. For example, to select the WAVELENGTH array corresponding to spectral order 80 of the

echelle spectrum in extension 4 (EXTNAME=SCI, EXTVER=2) of `stis.fits`, you would specify the file as either:

```
stis.fits[4][c:WAVELENGTH][r:sporder=80]
or
stis.fits[sci,2][c:WAVELENGTH][r:sporder=80]
```

Plotting Tasks

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

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

Similarly, to plot flux vs. wavelength in COS segment FUVA, you could type

```
st> sgraph "cos.fits[1][r:segment=FUVA] WAVELENGTH FLUX"
```

Remember to include the quotation marks, otherwise, **sgraph** will complain about too many 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 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.

3.5.3 FOS and GHRS Spectra

Before working with FOS and GHRS data within **STSDAS**, you will want to convert the FITS files you received from the Archive into GEIS format (see Section 2.3.1 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 sub-integration. 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. 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.

Sgraph 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 `y3bl0104t,` you can type:

```
st> sgraph y3bl0104t.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 y3bl0104t.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 y3bl0104t.c1h 1,10,19
```

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

3.5.4 Producing Hardcopy

This section shows how to generate hard copies of plots directly and describes **igi**, the Interactive Graphics Interpreter available in **STSDAS**. If you are working in the Python/**PyRAF** environment, the plotting library **matplotlib** is available. It uses most of the MATLAB syntax.

Direct Hardcopies

To print a quick copy of the displayed plot from the cl window:

1. Type `=gcur` in the cl command window.
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`.

From a **PyRAF** window, making hard copies is simpler: just select `print` from the menu at the top of the graphics window.

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. This parameter is generally defined in the*** `login.cl` ***file.***

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:

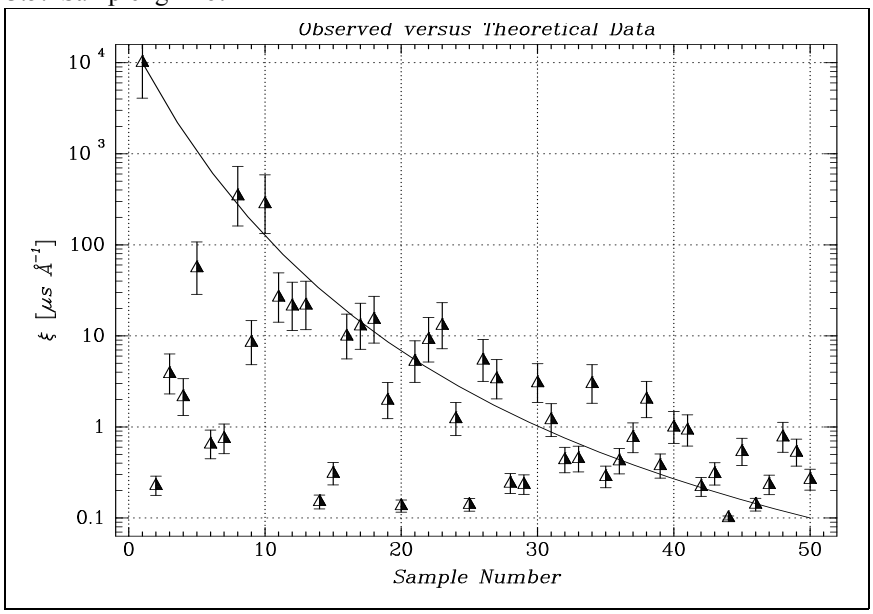
```
st> sgraph o43balbnm_x2d.fits[1][*,100:1000] device=psi_land
st> gflush
/tmp/psk<i>xxx</i>
```

The above commands would write a plot in landscape-mode into a temporary PostScript file, named `/tmp/psk`*xxxx* 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



3.6 Analyzing *HST* Spectra

This section describes some **IRAF**/**STSDAS** tasks that can be used for analyzing and manipulating spectral data. Some 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 specially prepared data (except for STIS two-dimensional spectra). Before discussing these tasks we will first describe how to recast your data into forms that are more generally accessible.

Alternatively, many other useful tools are now available for the analysis of *HST* spectra which operate outside of **IRAF**; among them is **specview**, a Java tool for viewing and analyzing spectra from a variety of instruments. It is briefly described in the previous section, Section 3.5.

3.6.1 Preparing COS and STIS Spectra for Analysis in IRAF or PyRAF and STSDAS

Calibrated spectra emerge from the pipeline either as two-dimensional spectral images (`x2d` STIS files) or as one-dimensional spectra stored in tabular form (`x1d` files for COS and STIS). You can analyze calibrated two-dimensional spectra in **IRAF** as you would analyze any other spectral image, because their headers already contain the necessary wavelength information.

Tabulated COS and STIS spectra can be analyzed directly using **STSDAS** tasks that understand the *selectors* syntax described in Section 2.2.2. However, in order to use **IRAF** tasks that rely on the multispec format WCS, such as **splot**, or other **STSDAS** tasks that do not understand three-dimensional tables, you will have to

prepare your data appropriately. This section describes some useful tasks for putting your data in the proper form:

- **tomultispec**: This task is the analog to **mkmultispec** and is useful for working with three-dimensional tables. It extracts COS and 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 three-dimensional table cells and places them in conventional two-dimensional tables for easier access.
- **tximage**: This task extracts specified data arrays from a 3-D table cells and places them into 1-D images. This task can write single group GEIS files.

tomultispec

The **tomultispec** task in the **stsdas.hst_calib.ctools** package extracts one row (or several) from a COS or 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 "mkmultispec" in Section 3.6.2). 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 rows from the file `myfile_x1d.fits` to a multispec file, type

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

The output file format of **tomultispec** will be OIF regardless of the specified extension. This format is similar to GEIS format, (see Section 4.3.6). OIF format files have a header component (suffix `.imh`) and a binary data component (suffix `.pix`).

If you want to select particular rows, rather than writing all the rows 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, 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
```

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

Column selectors cannot be used with `tomultispec`***. Tomultispec extracts the calibrated FLUX by default. However, other intensity data (e.g., NET counts) can be extracted by specifying the*** `flux_col` ***parameter appropriately.***

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 three rows are selected. This restriction results from a limit to the number of keywords that can be used to store the dispersion relation.***

If the `imdir` ***parameter in your*** `login.cl` ***file is set to a directory that does not exist, then*** `tomultispec` ***will fail when trying to run*** `sarith`***. To avoid this error, set*** `imdir` ***equal to a directory that does exist or*** `HDR$`***, which will place files in the same directory as the header file.***

txtable

COS and STIS spectra are stored as data arrays within individual cells of FITS binary tables (see Section 2.2.2). 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 each spectral order. The **txtable** task 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:sporder=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 of 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 tabular 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
```

Similarly, for a file `cos_nuv.fits` which contains a COS NUV spectrum and you want to extract only the wavelength and flux arrays for all three stripes, type:

```
tt> txtable "cos_nuv.fits[1][c:WAVELENGTH,FLUX][r:row=(1:3)]" \
>>> out_table
```

3.6.2 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 arrays.

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 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, due to the different functional form of the prism-mode dispersion solution. For FOS prism spectra, use the header table mode of*** `mkmultispec` ***(see below) or create an STSDAS table using*** `imtab`***.***

There is another method by which **mkmultispec** can incorporate wavelength information into a flux file and that 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 necessary 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 them (thereby degrading the individual spectra to a common linear dispersion scale) before joining them. Instead, you could create separate tables for 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. **tedit** can be used to edit selected rows of a table.

3.6.3 Photometry

Photometric correction of COS, FOS, GHRS, and STIS spectra is done by the pipeline during spectral extraction, resulting in flux-calibrated spectra. For detailed information see the Data Handbooks specific to each of these instruments at:

http://www.stsci.edu/hst/HST_overview/documents/datahandbook.

3.6.4 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.twodspec** packages, and those who are not should consult the online help. Table 3.6 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 3.6: Spectral Analysis Tasks in **IRAF/STSDAS**

Task	Package	Input Format	Purpose
boxcar	images.imfilter	Image	Boxcar smooth a list of images
bplot	noao.onedspec	Multispec image[1]	Plot spectra non-interactively
calcspec	stsdas.hst_calib.synphot	N/A	Create a synthetic spectrum
continuum	noao.onedspec	Image	Continuum normalize spectra
fitprofs	noao.onedspec	Image	Non-interactive Gaussian profile fitting to features in spectra and image lines
fitspec	stsdas.hst_calib.synphot	table	Fit a model spectrum to an observed spectrum
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	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 Section 3.6.5)
ngaussfit	stsdas.analysis.fitting	Image, table	Interactive 1-D multiple Gaussian fitting (see Section 3.6.5)
poffsets	stsdas.hst_calib.ctools	GEIS image	Determine pixel offsets between shifted spectra
plspec	stsdas.hst_calib.synphot	table	Plot calculated and observed 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 Section 3.6.2)
sarith	noao.onedspec	Multispec image[1]	Spectrum arithmetic
scombine	noao.onedspec	Multispec image[1]	Combine spectra
sfit	noao.onedspec	Multispec image[1]	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 Section 3.5.3)
specalign	stsdas.hst_calib.ctools	GEIS image	Align and combine shifted spectra (see **poffsets**)
specplot	noao.onedspec	Multispec image[1]	Stack and plot multiple spectra
splot	noao.onedspec	Multispec image[1]	Plot and analyze spectra & image lines (see “splot” on page 74)

1. Multispec image is a spectrum created with **tomutispec** or **mkmultispec**.

splot

The **splot** task in the **IRAF noao.onedspec** package is a good general purpose 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 do not 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 "mkmultispec" on page 70).

The **splot** task has *many* available options described in detail in the online help. Table 3.7 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 "tomultispec" on page 67), 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 that the beam label, which indicates the spectral order, cannot be used for navigation. See the online help for details.

Table 3.7: Useful **splot** Cursor Commands

Command	Purpose
Manipulating spectra	
f	Arithmetic mode; add and subtract spectra
l	Convert spectrum from f_ν to f_λ
n	Convert spectrum from f_λ to f_ν
s	Smooth with a boxcar
u	Define linear wavelength scale using two cursor markings
Fitting spectra	
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)
Displaying and redrawing spectra	
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
General file manipulation commands	
?	Display help
g	Get another spectrum
i	Write current spectrum to new or existing image
q	Quit and go on to next input spectrum

3.6.5 STSDAS Fitting Package

The **STSDAS fitting** package contains several tasks, as listed in Table 3.8, for fitting and analyzing spectra and images. 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 CDELT, but these forms only apply to linear coordinate systems and therefore would require resampling of your data onto a linear wavelength scale first. 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 (see Section 3.6.2).

Table 3.8: Tasks in the **STSDAS fitting** Package

Task	Purpose
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 redraw the plot window, 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* commands[3] 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 have a good fit, and then turn the error estimation on for one last iteration.

Figure 3.6 and Figure 3.7 show the results of fitting the Hβ (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. To see the online help for details and a complete listing of cursor mode colon commands: type `help cursor`.

Figure 3.6: Fitting Hβ and [OIII] Emission Features in NGC 4449.

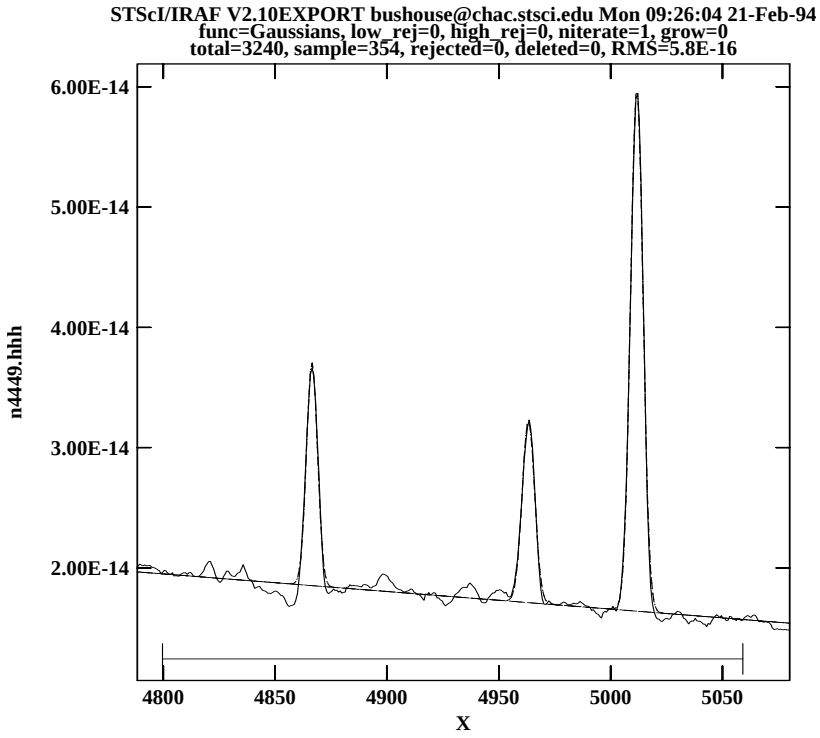


Figure 3.7: Coefficients and Error Estimates

```
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
```

3.6.6 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 at STScI by Gerard Kriss. 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 `CDELT1`. 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 flux 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.

3.7 References

3.7.1 Available from STScI

From the **STSDAS** Web page,

`http://www.stsci.edu/resources/software_hardware/stsdas/Documentation`

the following documents are available:

- *STSDAS User's Guide*, version 1.3, September 1994.
- *STSDAS Site Manager's Installation Guide and Reference*, version 3.9, November 2008.
- *Synphot User's Guide*, version 5.0, November 2005.
- *Synphot Data User's Guide*, version 1.2, June 2008.
- *IGI Reference Manual*, version 2.0, September 1998.

3.7.2 Available from NOAO

From the NOAO Web page, http://iraf.noao.edu/docs/photom.html, the following documents are available:

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

- *A Beginners Guide to Using IRAF,* 1993, J. Barnes.
- *Photometry Using IRAF*, version 2.10, 1994, L. Wells.
- *A User's Guide to Stellar CCD Photometry with IRAF*, 1992, P. Massy and L. Davis.

3.7.3 Other References Cited in This Chapter

- Busko, I., 1999, "A Java Graphics Package to Support specview", available at http://specview.stsci.edu/design/design6.ps
- Busko, I. 2002, "Specview: a Java Tool for Spectral Visualization and Model Fitting", Astronomical Data Analysis Software and Systems XI, ASP Conference Series, v.281, p.120, ed. Bohlender,D.A., Durand, D., and Handley,T.H.
- Fruchter, A. S. & Hook, R. N. 2002, PASP 114, 144.
- Fruchter, A. S. & Sosey, M. et al. 2009, "The MultiDrizzle Handbook", v3.0.
- 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.
- Koekemoer, A. M., Fruchter, A. S., Hook, R. N., & Hack, W. 2002, *HST* Calibration Workshop, ed. S. Arribas, A. M. Koekemoer, & B. Whitmore (STScI: Baltimore), p. 337.
- 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.
- www.stsci.edu/resources/software_hardware/pyraf
- www.stsci.edu/resources/software_hardware/pyfits
- numpy.scipy.org/

CHAPTER 4:

IRAF Primer

In this appendix...

4.1 IRAF Overview

The Image Reduction and Analysis Facility (**IRAF**), developed by the National Optical Astronomy Observatories (NOAO), forms the basis of the Space Telescope Science Data Analysis System (**STSDAS**). **IRAF** contains numerous packages of programs, called *tasks*, that perform a wide range of functions from reading data from storage media to producing plots and images. Most astronomers will already be familiar with **IRAF**, but we provide this tutorial for *HST* observers who are beginners in **IRAF**. It includes information on:

- How to set up **IRAF** the first time you use the software.
- How to start and stop an **IRAF** session.
- Basic concepts, such as loading packages, setting parameters, etc.
- How to use the online help facility.

Additional information on **IRAF**, in particular *A Beginner's Guide to Using **IRAF*** is available through the NOAO **IRAF** page at:

`http://iraf.noao.edu/iraf/ftp/pub/beguide.ps`

4.2 Initiating IRAF

This section explains:

- How to set up your **IRAF** working environment.
- How to start and logout of **IRAF**.

We assume that your site has IRAF and STSDAS installed. If not, you should obtain and install the software. See Section 4.4 for details.

4.2.1 Setting Up IRAF in Unix/Linux

Before running **IRAF** or **PyRAF** for the first time you need to:

1. Create an **IRAF** root directory.
2. Move to that directory and set the necessary environment variables or system logicals and symbols.
3. Run "`mkiraf`" to create a `login.cl` file and a `uparm` subdirectory.

Users generally name their **IRAF** home directory "`iraf`" (also referred to as your **IRAF** *root* directory) and put it in their user home directory (i.e., the default directory that you are in when you log in to the system). The **IRAF** home directory does not need to be in your home directory, nor does it need to be called "`iraf`", but you should *not* put it on a scratch disk that is periodically erased.

If you call your root **IRAF** directory "`iraf`", you can set up **IRAF** as follows

Under Unix:

```
% mkdir iraf
% cd iraf
% setenv iraf /usr/stsci/iraf/
% source $iraf/unix/hlib/irafuser.csh
% mkiraf
```

Can be placed in .login file

The directory name is site-dependent—check with your system staff

The "`mkiraf`" command initializes **IRAF** by creating a `login.cl` file and a subdirectory called `uparm`. After typing the "`mkiraf`" command, you will see something like the following:

```
% mkiraf
-- creating a new uparm directory
Terminal types: gterm=ttysw+graphics,vt640...
Enter terminal type:
```

Enter the type of terminal or workstation you will most often use with **IRAF**. Generic terminal types that will work for most users are:

- `xgterm` for sites that have installed X11 **IRAF** and **IRAF** v2.10.3 BETA or later.
- `xterm` for most workstations running under X-Windows.
- `vt100` for most terminals.

IRAF for other systems, like Mac OSX, can be obtained from the **IRAF** Web page at:

`http://iraf.noao.edu`

You can change your terminal type at any time by typing (*`set term=new_type`*) during an IRAF session. You can also change your default type by editing the appropriate line in your* `login.cl` *file.

After you enter your terminal type, you will see the following message before getting your regular prompt:

```
A new LOGIN.CL file has been created in the current ...
You may wish to review and edit this file to change ...
```

The `login.cl` file is the *startup file* used by the **IRAF** command language (CL). It is similar to the `.login` file used by Unix. Whenever **IRAF** starts, it looks for the `login.cl` file. You can edit this file to customize your **IRAF** environment. In fact, you should look at it to make sure that everything in it is correct. In particular, there is a line starting with "`set home =`" that tells **IRAF** where to find your **IRAF** home directory. You should verify that this statement does, in fact, point to your **IRAF** directory. If you will be working with standard **IRAF** format images you should also insert a line saying `set imdir = "HDR$"`. The `imdir` setting is ignored when working with GEIS and FITS format images.

The `uparm` directory will contain your own copies of **IRAF** task parameters. This directory allows you to customize your **IRAF** environment by setting certain parameter values as defaults. Once you set up **IRAF**, you should only have to do it again when you install an updated version of **IRAF**. It is also a good idea at this time to make sure your standard image size is set to the largest size of the images you may want to display. This is controlled by the `stdimage` keyword. **IRAF** will only display images in a visual tool up to this size. To enable image sizes up to 8192 × 8192, set `stdimage` equal to `imt7`.

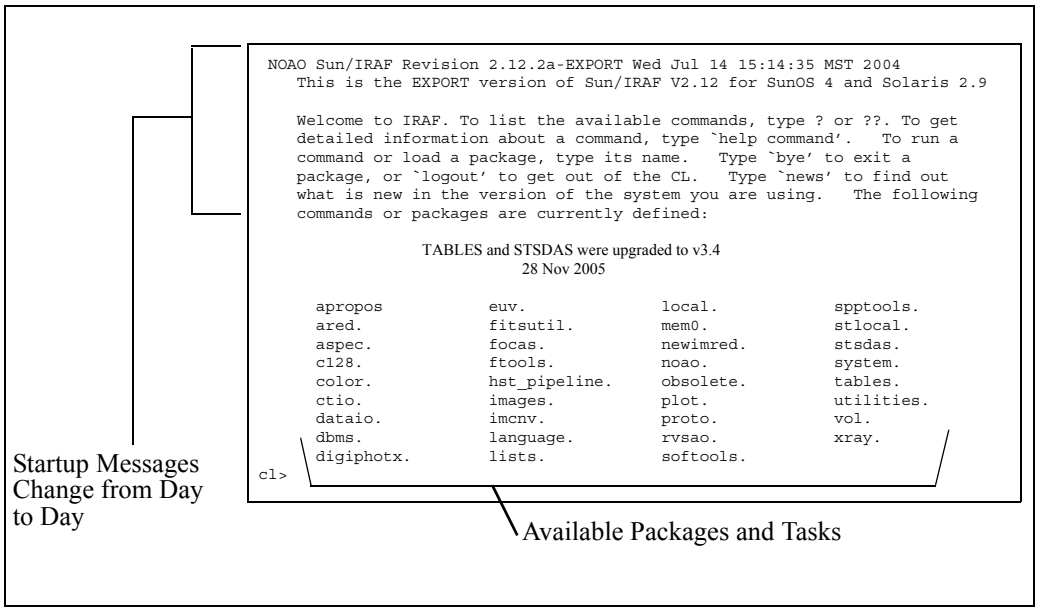
4.2.2 Starting and Stopping an IRAF Session

To start an IRAF session:

1. Move to your **IRAF** home directory.
2. Type `cl` at the command line.

IRAF starts by displaying several lines of introductory text and then puts a prompt at the bottom of the screen. Figure 4.1 is a sample **IRAF** startup screen.

Figure 4.1: **IRAF** Startup Screen



To quit an IRAF session:

Type "`logout`".

4.3 IRAF Basics

This section describes basic **IRAF** techniques such as:

- Loading packages
- Running tasks and commands
- Getting online help
- Viewing and setting parameters (see Section 4.3.4)
- Setting and using environment variables (see Section 4.3.5)
- File management
- Troubleshooting

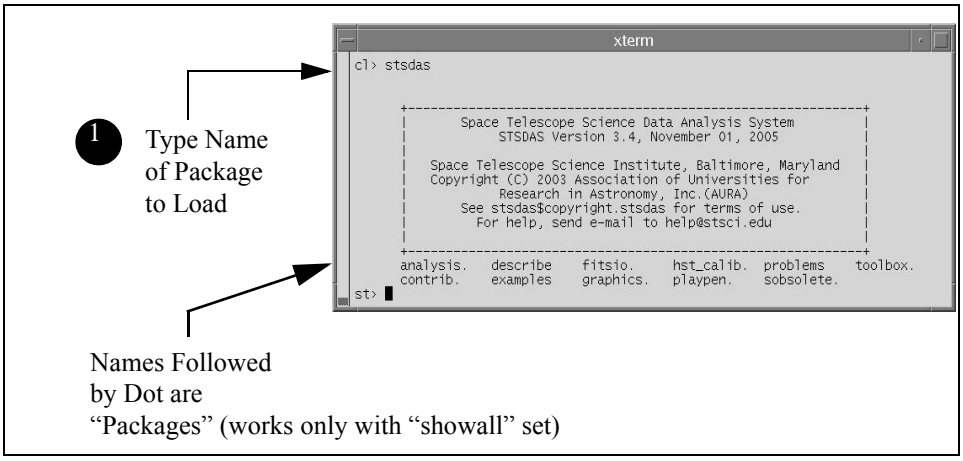
4.3.1 Loading Packages

In **IRAF** jargon, an application is called a *task* and logically related tasks are grouped together in a *package*. Before you can use a task, you must load the package containing that task. To load a package, type the name of the package. The prompt will then change to the first two letters of the package name, and the screen will display the names of all the newly available tasks and subpackages. Even though the prompt has changed, previously loaded packages remain loaded, and all their tasks remain available.

The standard way to specify a path through the **IRAF** package hierarchy to a task in a particular subpackage is to separate the package names with periods (e.g., **stsdas.hst_calib.foc.focgeom.newgeom**).

The most frequently used packages can be added to your `login.cl` file. List the packages, one per line in that file, and each will be loaded automatically when **IRAF** is started.

Figure 4.2: Loading Packages



Some helpful commands for managing packages are:

- `?` - Lists tasks in the most recently-loaded package
- `??` - Lists all tasks loaded
- `package` - Lists all packages loaded
- `bye` - Exits the current package

4.3.2 Running Tasks

This section explains how to run tasks and system-level commands, and how to use piping and redirection.

Running a Task

The simplest way to run a task is to type its name or any unambiguous abbreviation of it. The task will then prompt you for the values of any required *parameters*.

Alternatively, you can specify the values of the required *parameters* on the command line when you run the task. For example, if you want to print header information on `myfile.hhh`, type:

```
st> imhead myfile.hhh
```

IRAF does not require you to specify the complete command name—only enough of it to make it unique. For example, `dir` ***is sufficient for*** `directory`***.***

Escaping System-Level Commands

To run an operating system-level command (i.e., Unix command) from within the **IRAF** CL, precede the command with an exclamation point (!). This procedure is called *escaping* the command. For example:

```
st> !system_command
```

Piping and Redirection

You can run tasks in sequence if you desire, with the output of one task being used as the input for another. This procedure, called *piping*, is done by separating commands with a vertical bar (|), using the following syntax:

```
st> task1 filename | task2
```

For example, if a particular task prints a large volume of text to the screen, you may want to pipe it to `page`, which allows you to read the output one page at a time:

```
st> task1 filename | page
```

You can also *redirect* output from any task or command to a file by using the greater-than symbol (>) as follows:

```
st> command > outputfile
```

Background Tasks

To run a task as a background job, freeing your workstation window for other work, add an ampersand (&) to the end of the command line, like so:

```
st> taskname &
```

4.3.3 Getting Help

This section describes:

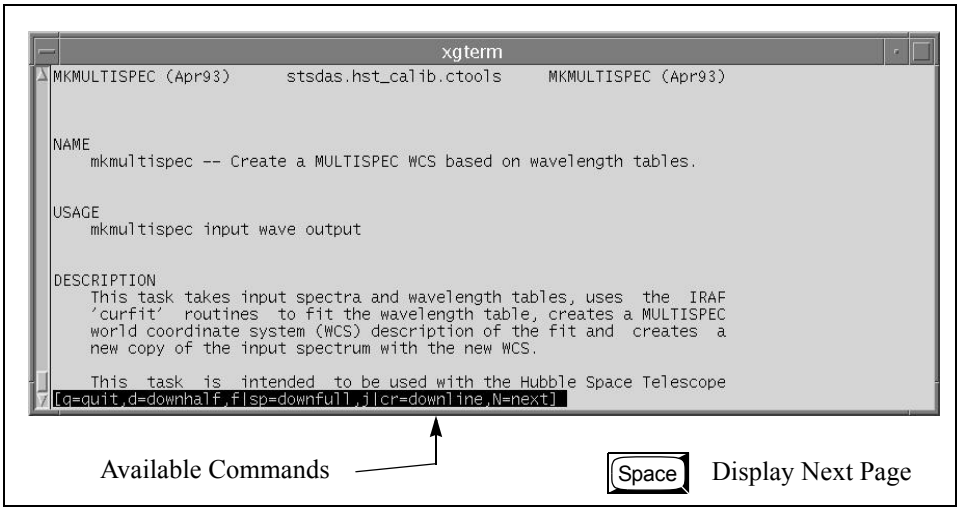
- How to use **IRAF**'s online help facility
- How to find a task that does what you want
- **IRAF** tutorials and CL tips

Online Help

You can get online help with any **IRAF** task or package by using the **help** command,[1] which takes as an argument the task or package name about which you want help. Wildcards are supported. To display the online help for the **STSDAS mkmultispec** task, type:

```
fi> help mkmultispec
```

Figure 4.3: Displaying Online Help



Two **STSDAS** tasks that display particular sections of the help file are also available:

- **examples** - Displays only the examples for a task.
- **describe** - Displays only the description of the task.

Typing `help` *package* will produce one-line descriptions of each task in the package.

Finding Tasks

There are several ways to find a task that does what you need:

- Use `help` *package* to search through the **IRAF**/**STSDAS** package structure.

1. There is an optional *paging* front-end for help called **phelp**. For more information, type `help phelp` from within **IRAF**.

- Use the **apropos** task as shown in Figure 4.4 to search the online help database. This task looks through a list of **IRAF** and **STSDAS** package menus to find tasks that match a specified keyword. Note that the name of the package containing the task is shown in parentheses.

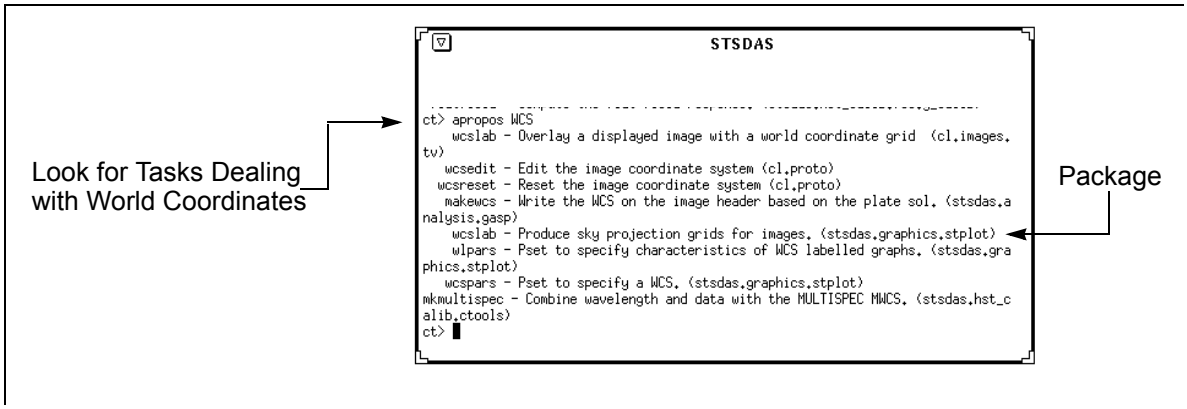
IRAF Tutorials and CL tips

Hints and tutorials are available on the NOAO Web site to help you become more familiar with **IRAF**. See the following Web sites:

`http://iraf.noao.edu/iraf/web/tutorials/tutorials.html`

`http://iraf.noao.edu/tips/cl.html`

Figure 4.4: Using the **apropos** Task



4.3.4 Setting Parameters

Parameters specify the input information for **IRAF** tasks. They can be the names of input or output files, particular pixel numbers, keyword settings, or many other types of information that control the behavior of the task.

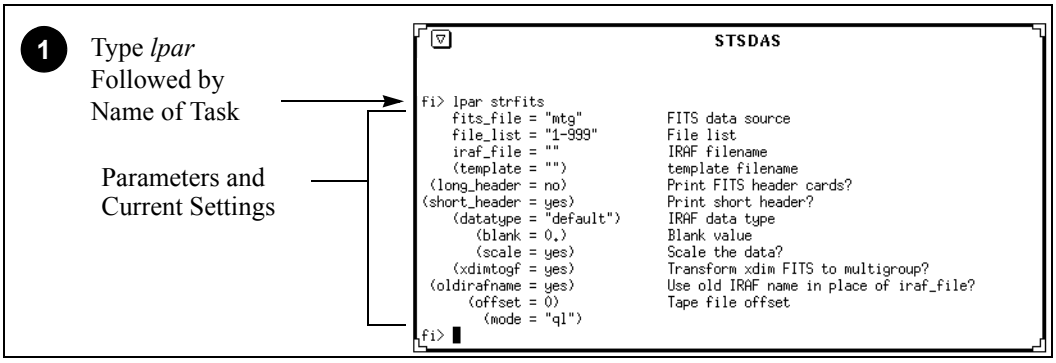
The two most useful commands for handling task parameters are:

- **lparam** to display the current parameter settings (abbreviated **lpar**)
- **eparam** to edit parameters (abbreviated **epar**)

Viewing Parameters with lparam

The **lpar** command lists the current parameter settings for a given task (see Figure 4.5).

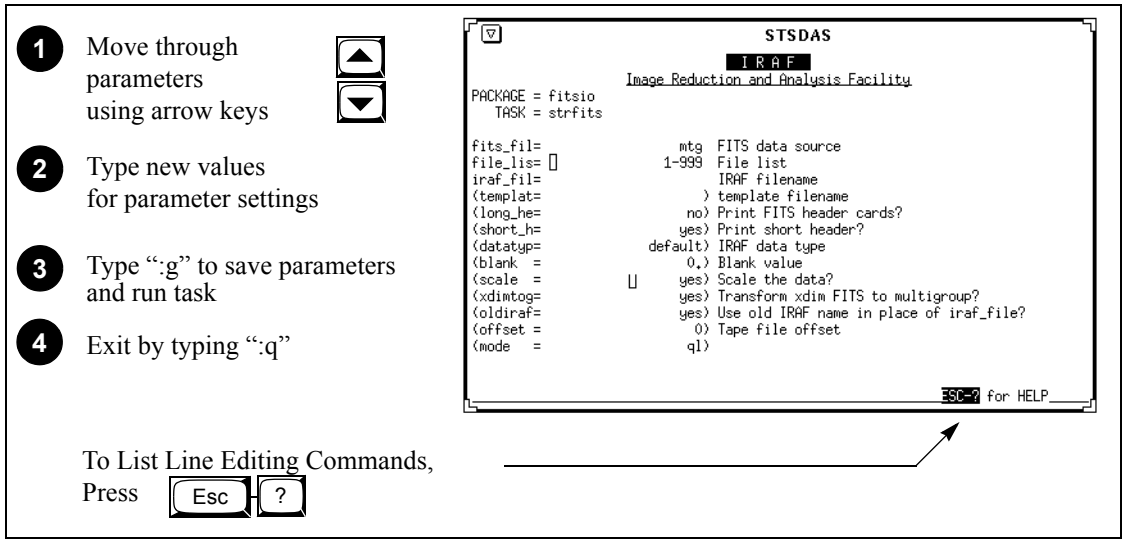
Figure 4.5: Displaying Parameter Settings with **lpar**



Setting parameters with eparam

Epar is an interactive parameter editor. It displays all of the parameters and their current settings. You can move around the screen using the arrow keys and type new settings for any parameters you wish to change. Figure 4.6 shows what you will see when typing `epar strfits` when using the CL interface.

Figure 4.6: Editing Parameters with **epar** in **IRAF**



Parameter Types—What to Specify

Parameters are either *required* or *hidden*, and each parameter expects information of a certain *type*. Usually, the first parameter is required, and very often it expects a file name. Parameters are described in the online help for each task. Hidden

parameters, shown in parentheses in the online help and the **lpar** and **epar** listings, need not be specified at each execution because their default values frequently suffice.

Wise IRAF users will check the values of hidden parameters, as they often govern important aspects of a task's behavior.

If you specify the wrong type of information for a parameter, **epar** will usually display an error message saying something like "Parameter Value is Out of Range." The message is displayed when you move to another parameter or if you press Return. Table 4.1 lists the different parameter types.

Table 4.1: Parameter Data Types

Type	Description
`File Name`	Full name of the file. Wild card characters (* and ?) are allowed. Some tasks accept special features when specifying file names, including "@" lists, **IRAF** networking syntax, and image section or group syntax (see Section 4.3.6).
`Integer`	Whole number. Often the task will specify minimum or maximum values (consult the help pages).
`Real`	Floating point numbers, can be expressed in exponential notation. Often will have minimum and maximum values.
`Boolean`	Logical "yes" or "no" values.
`String`	Any characters. Sometimes file names are specified as a string.
`Pset`	Parameter set.

Restoring Parameter Default Values

Occasionally, **IRAF** might get confused by your parameter values. You can restore the default parameters with the **unlearn** command. You can use **unlearn** either on a task or an entire package. Help on using **unlearn** can be found online:

`http://stsdas.stsci.edu/cgi-bin/gethelp.cgi?unlearn`

The `unlearn` ***command generally will restore the parameters to reasonable values, a big help if you are no longer sure which parameter values you have changed in a complicated task.***

4.3.5 Setting Environment Variables

IRAF uses *environment variables* to define which devices are used for certain operations. For example, your terminal type, default printer, and the disk and directory used for storing images are all defined with environment variables. Environment variables are defined using the **set** command and are displayed using the **show** command. Table 4.2 lists some of the environment variables that you might want to customize.

Table 4.2: Environment Variables

Variable	Description	Example of Setting
`printer`	Default printer for text	`set printer = lp2`
`terminal`	Terminal type	`set term = xgterm`
`stdplot`	Default printer for all graphics output	`set stdplot = ps2`
`stdimage`	Default terminal display setting for image output (Set this to the largest image you will mostly want to display. `imt7` will enable images up to 8192x8192.)	`set stdimage = imt1024`
`stdgraph`	Default graphics device	`set stdgraph = xgterm`
`clobber`	Allow or prevent overwriting of files	`set clobber = yes`
`imtype`	Default image type for output images. "`hhh`" is used for GEIS format and "`fits`" for FITS format. ("`imh`" is used for original **IRAF** format (OIF).)	`set imtype = "fits"`

You can set your environment variables automatically each time you login to **IRAF** by adding the appropriate commands to your `login.cl` file. Use your favorite text editor to specify each variable on its own line. The **show** command with no arguments prints the names and current values of all environment variables.

4.3.6 File Management

This section describes:

- File formats commonly used with **STSDAS** and **IRAF**.
- Specification of file names.
- Navigation through directories.

File Formats

IRAF recognizes a number of different file structures. Among them are the standard *HST* file formats known as GEIS and FITS (see Chapter 2), both of which differ from the original **IRAF** format, OIF.

GEIS is closer to OIF, in that two files are *always* used together as a pair:

- A *header file*, which consists of descriptive information. **IRAF** (OIF) header files are identified by the suffix `.imh`. GEIS header files are in ASCII text format and are identified by the suffix `.hhh` or another suffix ending in "`h`", such as `.c0h` or `.q1h`.
- A *binary data file*,[2] consisting of pixel information. **IRAF** data file names end with a `.pix` suffix. **STSDAS** data files end with a suffix of `.hhd` or another suffix that ends with "d", such as `.c0d` or `.q1d`.

STSDAS always expects both component files of a GEIS image to be kept together in the same directory.

A single FITS file contains both the header information and the data. See Chapter 2 for details on FITS files.

When working with IRAF (OIF) or STSDAS (GEIS) images, you need only specify the header file name—the tasks will automatically use the binary data file when necessary.

File Specification

Most tasks in **IRAF** and **STSDAS** operate on files and expect you to specify a file name for one or more parameters. Special syntax can be used with certain tasks when specifying file names. These syntax features include:

- **Wild card characters**, often called *templates*, which are used to specify multiple files using pattern matching techniques. The wild cards are:
  - * Matches any number of characters, e.g., `j91k10pt*.fits`
  - ? Matches any single character, e.g., `z01x23x.c?h`

When using wildcards to specify GEIS files with image-processing tasks, be sure to exclude the binary-pixel files by ending your file name specification with an "h", for example: `y*.??h`.

2. The binary data file format is host-dependent and may require translation before it can be moved to a computer using a different architecture.

- **List files**, often called *@-files*, are ASCII files that contain lists of file names, one per line. If your task supports the list file feature, you would type the name of your list file, preceded by the "@" character. For example: `@infiles.txt`
- **Image section** specification. Tasks that work with image data will allow you to specify a part of the image rather than the entire image. To extract a particular image section, specify each axis range in square brackets, for example: `image.hhh[10:200,20:200]`
- **IRAF networking** specification. **IRAF** is capable of reading and writing files to and from remote systems on a network. This feature is often used with tasks in the **fitsio** and **convfile** packages, or with image display tasks. The *STSDAS User's Guide* and the online help (type `help networking`) describe how to enable this feature. To specify that you want to use the **IRAF** networking feature, type the remote host name followed by an exclamation point (!), followed by the file or device name. For example: `nemesis!mta`

For example, when displaying from an **IRAF** session running on a remote machine back to your work station set the environment variable "node" by typing: `set node= my_workstation`!

Directory Navigation

To navigate through directories, you can use the following commands:

- **path** or **pwd** - Lists the current working directory.
- **cd** *directory* - Move to the named directory.
- **back** - Revert to directory last visited.

4.3.7 Troubleshooting

There are a couple of easy things you can do to make sure that you do not have a simple memory or parameter conflict—common causes of problems:

- Look at the parameter settings and make sure that you have specified reasonable values for every parameter.
- When you run an **IRAF** task for the first time in a session, **IRAF** stores the executable file in its *process cache*. If **IRAF** appears not to be running your tasks properly, you may need to use the **flprcache** command to clear the process cache. To do this, type `flpr`. Sometimes you will need to execute this command a few times.
- Occasionally, you may need to logout of the CL, restart **IRAF**, and try your command again.

If you still have a problem with an IRAF task, see the NOAO IRAF pages at iraf.noao.edu and iraf.net.

4.4 PyRAF Capabilities

PyRAF is a Python package that runs **IRAF** tasks from Python. All the features of Python are available when you run **PyRAF**. In addition, you can run **IRAF** tasks using the same syntax that you would use in the **IRAF** CL.

To start a PyRAF session:

1. Move to your **IRAF** home directory. (Note: If you have defined "~/iraf" as your home IRAF directory you can start up PyRAF from any directory).
2. Type "`pyraf`" at the command line.

Like **IRAF**, several lines of introductory text will be displayed, and a prompt will appear at the bottom of the screen.

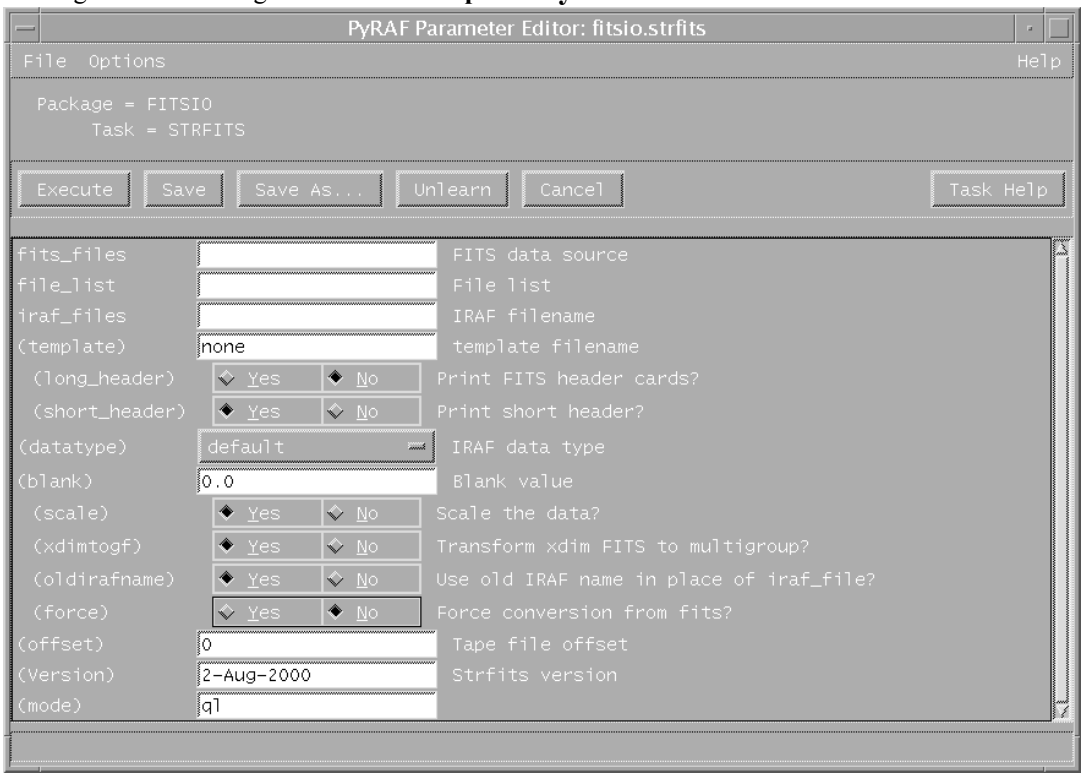
To quit a PyRAF session:

Type "`.exit`".

Setting parameters with eparam in PyRAF

In **PyRAF**, the **epar** editor uses a graphical user interface (GUI), as shown in Figure 4.7. The **PyRAF epar** includes features like file browsing and pop-up help, which are not available in the **IRAF** CL.

Figure 4.7: Editing Parameters with **epar** in **PyRAF**



Advantages of using PyRAF

There are many advantages of **PyRAF** over the **IRAF** CL. The error handling of **PyRAF** is superior. When using CL scripts and encountering an error, **PyRAF** reports the line number in the CL script where the error occurred. Python has excellent string handling capabilities and other features. Scripts can be written in Python, although you can still write and use CL scripts.

PyFITS can be used to read a FITS image or table to an in-memory array (a numpy array). Arithmetic manipulation can be performed with numpy arrays and results written to FITS images or tables. However, there are certainly cases where it would be difficult to duplicate the features of an **IRAF** task using simple in-memory array arithmetic, tasks such as **imcombine**.

IRAF tasks are easily run from **PyRAF**. Interactively, there is no noticeable difference from the **IRAF** CL, but in a Python script you would preface the name of *any* **IRAF** task with "`iraf.`" (including the period). For example: "`iraf.phot`".

The following Python script runs the **IRAF** task **imstatistics**. It is saved in a file called "`imstat_example.py`".

```
#! /usr/bin/env python

import sys
from pyraf import iraf

def run_imstat(input):
       iraf.images()
       for image in input:
             iraf.imstat(image)

if __name__ == "__main__":
      run_imstat(sys.argv[1:])
```

This script can be run in two ways:

- from Python or **PyRAF**:

```
--> import imstat_example
--> imstat_example.run_imstat(["im1.fits[1]","im2.fits[1]"])
```

- from the Unix shell and if the file "`imstat_example.py`" is executable:

```
%> imstat_example.py im1ex1.fits im2ex1.fits
```

In the first example, the argument to the script is a list of two different images, each specifying the first extension of each FITS file. In the second example, each of the files passed as a parameter to the script contains the first extension of the FITS images used in the first example. For further information, see the **PyRAF** Programmer's Guide, which can be downloaded from:

http://stsdas.stsci.edu/stsci_python_epydoc/

4.5 Accessing IRAF, PyRAF, and STSDAS Software

IRAF, PyRAF, and **STSDAS** are provided free of charge to the astronomical community. You can down load the **IRAF** software from http://iraf.noao.edu/. Instructions on how to download and install **PyRAF** can be found online:

http://www.stsci.edu/resources/software_hardware/pyraf/stsci_python/current/download

You must have **IRAF** or **PyRAF** installed on your system in order to run **STSDAS**. Detailed information about downloading and installing **STSDAS** is found at the following page:

http://www.stsci.edu/resources/software_hardware/stsdas/download

If you have any problems getting and installing **STSDAS**, **TABLES**, **PyRAF**, or any other STScI packages or data described in this handbook, please contact the Help Desk by sending e-mail to: help@stsci.edu.

When you download STSDAS, you should also download the TABLES package. TABLES must be installed prior to STSDAS. You must have IRAF or PyRAF installed on your system in order to install TABLES and STSDAS.

Bug fixes and new versions of this package in the process of being tested can be downloaded as the IRAFX environment from:

http://stsdas.stsci.edu/irafx/

These pre-release versions are updated weekly to include new code and bug fixes which have not been extensively tested or verified for accuracy by the Institute. They also require a very different installation procedure and can be installed without root access, but are only provided as binaries for the Mac OSX Leopard and Linux Red Hat Enterprise 4 operating systems.

4.5.1 Accessing the Synphot Data Set

This handbook sometimes refers to the **synphot** data set, which must be available in order to run tasks in the **STSDAS synphot** package (see Section 3.4.4). The package is included with the **STSDAS** installation, but the corresponding data set must be retrieved and installed separately. Instructions on how to obtain and install the **synphot** data set can be found at:

http://www.stsci.edu/resources/software_hardware/stsdas/synphot

Users should note that there are two separate synphot-related documents:

- the *Synphot User's Guide*, which includes instructions on how to use the tasks in the package, and
- the *Synphot Data User's Guide*, which includes instructions on how to set the different *HST* keywords.

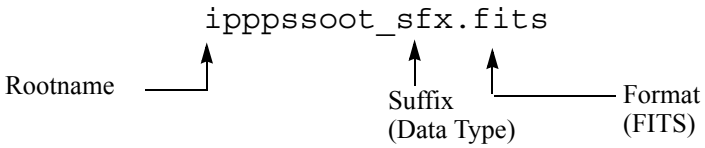
CHAPTER 5:

HST File Names

In this appendix...

5.1 File Name Format

HST data file names encode a large amount of information about the files themselves. Datasets retrieved from the HDA, as described in Chapter 2, consist of multiple files in Multi-Extension FITS format, each with a name that looks like this:



- ***Rootname***: The first part of the file name (`ipppssoot`) is the *rootname* of the dataset to which the file belongs. All files belonging to a given dataset share the same rootname.
- ***Suffix***: The part of the filename between the "_" and the ".fits" is called the *suffix (sfx)*, and it indicates the type of data the file contains. All science instruments except for COS, have data file suffixes with three characters. COS data file suffixes are between three and eleven characters long.
- ***Format***: The identifier `.fits` indicates that this file is in FITS format.

For example, a STIS data file named `o8v502010_x1d.fits` is a FITS file that belongs to the dataset with rootname "`o8v502010`", and its "`_x1d`" suffix indicates that it contains calibrated science spectra.

GEIS Files

In order to use **PyRAF/IRAF/STSDAS** tasks to work with data from the instruments FOC, FOS, GHRS, HSP, WF/PC-1, and WFPC2[1] you will want to convert these waivered FITS files into GEIS format. See Section 2.2 for instructions on how to convert FITS files to GEIS files using **strfits**. Like MEF format files, the names of GEIS files also derive from a file's rootname and suffix, and they have the form:

```
ipppssoot.sfx
```

Generally the suffixes of GEIS files end either in "`d`", indicating a binary data file, or "`h`", indicating an ASCII header file. The two GEIS files `x3l80101t.d0h` and `x3l80101t.d0d` together contain the same information as the single waivered FITS file `x3l80101t_d0f.fits`.

The identifier referred to here as a "suffix" has often been called an "extension" in the past. However, the individual pieces of FITS files are also known as "extensions" (see Section 2.2.1). For clarity, this handbook will use the term "extension" when referring to a component of a MEF format file and the term "suffix" when referring to the three character identifier in a filename.

5.2 Rootnames

Rootnames of *HST* data files follow the naming convention defined in Table 5.1.

1. Note that the HDA also distributes WFPC2 data in MEF format files and those files do not need to be converted to GEIS format.

Table 5.1: IPPPSSOOT Root Filenames

Character	Meaning
`I`	Instrument used, will be one of: ***e*** - Engineering data ***f*** - Fine Guidance Sensors ***i*** - Wide Field Camera 3 ***j*** - Advanced Camera for Surveys ***l*** - Cosmic Origins Spectrograph ***n*** - Near Infrared Camera and Multi-Object Spectrograph ***o*** - Space Telescope Imaging Spectrograph ***s*** - Engineering subset data ***t*** - Guide star position data ***u*** - Wide Field/Planetary Camera-2 ***v*** - High Speed Photometer ***w*** - Wide Field/Planetary Camera ***x*** - Faint Object Camera ***y*** - Faint Object Spectrograph ***z*** - Goddard High Resolution Spectrograph
`PPP`	Program ID; can be any combination of letters or numbers (46,656 possible combinations). There is a unique association between program ID and proposal ID.
`SS`	Observation set ID; any combination of letters or numbers (1,296 possible combinations).
`OO`	Observation ID; any combination of letters or numbers (1,296 possible combinations).
`T`	Source of transmission or association product number ***m*** - Merged real time and tape recorded ***n*** - Retransmitted merged real time and tape recorded ***o*** - Retransmitted real time (letter “O’) ***p*** - Retransmitted tape recorded ***q*** - Solid-state recorder ***r*** - Real time (not recorded) ***s*** - Retransmitted solid-state recorder ***t*** - Tape recorded ***0*** - Primary association product (number zero) ***1-8*** - *ACS, COS, NICMOS, or WFC3 association sub-product*

5.3 Suffixes of Files Common to All Instruments

The suffix of an *HST* file identifies the type of data that it contains. Several types of file suffixes are common to all instruments. These common suffixes are briefly described below. Please refer to the appropriate instrument’s “Data Structures” chapters (in Part II) for the definitions of the instrument-specific suffixes.

Trailer Files

Trailer files (suffix `trl`) are FITS ASCII tables that log the processing of your data by the OPUS pipeline.

Support Files

Support files (suffix `spt`) contain information about the observation and engineering data from the instrument and spacecraft recorded at the time of the observation.

Observation Logs

Observation Logs (suffix `ji* or cm*`, aka "obslogs") are FITS files that contain information describing how the *HST* spacecraft behaved during a given observation. More information on these files is given in Chapter 6.

5.3.1 Historical File Structures

Old Version Observation Logs

The engineering telemetry, which is the basis of the observation logs, does not get processed with OTFR. Thus any significant change in its processing results in a different type of observation log. Such a change occurred in February 2003, when the existing Observation Monitoring System (OMS) was streamlined to reduce the software maintenance costs. Obslogs produced prior to this time have `cm*` suffixes and contain science specific information, in addition to the pointing information. Obslogs produced after this date have `ji*` suffixes and do not contain any instrument specific information.

Obslogs headers, which you can read with the **IRAF** task **imheader** (see Section 2.3.3), are divided into groups of keywords that deal with particular topics such as spacecraft data, background light, pointing control data, and line-of-sight jitter summary. The headers themselves provide short descriptions of each keyword. Obs logs tables and images record spacecraft pointing information as a function of time. For more information on these files, consult Chapter 6 or the STScI Observation Logs Web pages at:

`http://www.stsci.edu/hst/observatory/pointing/obslog/OL_1.html`

PDQ Files

The suffix `pdq` denotes Post Observation Summary and Data Quality files which contain predicted as well as actual observation parameters extracted from the standard header and science headers. These files may also contain comments on any obvious features in the spectrum or image, as noted in the OPUS data assessment, or automatically extracted information about problems or oddities encountered during the observation or data processing. These comments may include correction to the keywords automatically placed in the obs logs files. PDQ files were discontinued on May 9, 2002.

OCX Files

The suffix `ocx` denotes Observer Comment Files which were produced by STScI personnel to document the results of real-time commanding or monitoring of the observation, along with keywords and comments. Prior to April 17, 1992, OCX files

were not always archived separately and, in some cases, were prepended to the trailer file.

After early February 1995, OCX files were produced only when an observation was used to locate the target for an Interactive Target Acquisition. At this time, mission and spacecraft information were moved to the PDQ reports and the Observation Logs (OMS jitter image and jitter table). OCX files were discontinued on May 9, 2002.

5.4 Associations

The ACS, NICMOS, STIS, and WFC3 calibration pipelines sometimes produce single calibrated images from *associations* of many exposures. For example, a NICMOS, ACS, or WFC3 observer might specify a dithering pattern in a Phase II proposal. Those instruments would then take several exposures at offset positions, and the pipeline would combine them into a single mosaic (suffix "`_mos`" for NICMOS; suffix "`_drz`" for ACS and WFC3). In this case, the original set of exposures constitutes the association, and the mosaic is the *association product*.

Similarly, a STIS observer might specify a `CR-SPLIT` sequence in a Phase II proposal. STIS would gather several exposures at the same pointing, and the STIS pipeline would process this association of exposures into a single image, free of cosmic rays, that would be the association product (suffix "`_crj`").

The COS calibration pipeline instead uses associations to process *all* of COS science data. Please refer to the Chapter 2 (in Part II of the COS Data Handbook) for more details.

When you search the HDA for observations involving associations of exposures, your search will identify the final association product. The rootnames of association products always end in zero (see Table 5.1 above). If you request both Calibrated and Uncalibrated data from the HDA, you will receive both the association product and the exposures that went into making it. The corresponding association table, stored in the file with suffix "`asn`" and the same rootname as the association product, lists the exposures or datasets belonging to the association. You can read this file using the **STSDAS tprint** or **tread** tasks (see Section 2.2.2). The exposure IDs in the association table share the same "`ipppss`" sequence as the association rootname, followed by a base 36 number "`nn`" (`n` = 0-9, A-Z) that uniquely identifies each exposure, and a character `t` that denotes the data transmission mode for that exposure (see Table 5.1). For association products and sub-products, the last character will be a number between 0-8.

In practice, STIS stores the exposures belonging to associations differently than the other instruments. The exposures belonging to a STIS association all reside in the same file, while the exposures belonging to an ACS, COS, NICMOS, or WFC3 association reside in separate data files. See the relevant Data Structures chapters (in Part II) for more details.

CHAPTER 6:

Observation Logs

In this appendix...

6.1 Observation Log Files

Observation Log Files, also known as *jitter* files, record pointing, jitter, and other Pointing Control System (PCS) data taken during an *HST* observation. You can use them to assess the behavior of the *HST* spacecraft during your observation, and in particular, to evaluate the jitter of the spacecraft while it was taking data. Here we describe the contents and structure of the observation log files, how to retrieve them from the Archive, and how to work with the data they contain.

These data files are produced by the Engineering Data Processing System (EDPS), an automated software system that interrogates the *HST* engineering telemetry and correlates the time-tagged engineering stream with *HST*'s Science Mission Schedule (SMS), the seven-day command and event list that drives all spacecraft activities. The EDPS replaced the Observatory Monitoring System (OMS) in February 2003. EDPS provides observers with information about guide star acquisition, pointing, and tracking that is not normally provided in the science headers.

The observation log files share the same rootname as the observation they are associated with, except for the final character, which for observation log files is always a "`j`" (see Chapter 5 for more on the names of *HST* data files). The `jit` table accompanies the `jif` header. The `jit` table is the three-second average pointing data. The `jif` header is a two-dimensional histogram of jitter excursions during the observation which includes the header plus some image related keywords.

A detailed description of the observation log files can be found online: http://www.stsci.edu/hst/observatory/pointing/obslog/OL_1.html. A summary of the latest jitter file format can be found at: http://www.ess.stsci.edu/projects/edps/jitter_format.html

Table 6.1: OMS Observation Log Files

Suffix	Contents
October 1994 to August 1995	
`cmh`	OMS header
`cmj`	High time resolution (**IRAF** table)
`cmi`	Three-second averages (**IRAF** table)
`_cmh.fits`	Archived FITS file of `cmh`
`_cmj.fits`	Archived FITS file of `cmj`
`_cmi.fits`	Archived FITS file of `cmi`
August 1995 to February 1997	
`jih/jid`	Two-dimensional histogram and header (GEIS)
`jit`	Three-second averages (**IRAF** table)[1]
`_jif.fits`	Archived FITS file which bundles the `jih/jid` files.
`_jit.fits`	Archived FITS file of `jit`.
February 1997 onward	
`_jif.fits`	Two-dimensional histogram (FITS)
`_jit.fits`	Three-second averages table (FITS)
`_jwf.fits`	Two-dimensional histogram for STIS wavecals.
`_jwt.fits`	Three-second averages for STIS wavecals.

1. After May 11, 1995, the `jit` tables for exposures shorter than 6 seconds contain higher-resolution, one-second average pointing data.

Pointing and tracking information prior to October 1994 is not routinely available. Interested observers with data from this epoch, can send e-mail to help@stsci.edu.

6.1.1 Jitter File Contents

The EDPS jitter files are limited to the engineering data that describe the performance of the Pointing Control System (PCS) including the Fine Guidance

Sensors that are used to control the vehicle pointing. The jitter files report on PCS engineering data for the duration of the observation. The EDPS jitter files contain:

- *rootname*`j_jif.fits`*:* The FITS header contains keywords providing information regarding the file structure, observation details, modeled background light, point control system, jitter summary, orbital geometry, and problem flags and warnings. The extension 0 header will contain parameters relating to the entire association or dataset, ending at the "ASSOCIATION KEYWORDS" block. The header values for extensions 1 and beyond will contain all the group 0 keywords as well as additional blocks of parameters relating to that particular exposure in the association.
- *rootname*`j_jit.fits`: This is the 3-second average jitter table. It contains the reconstructed pointing, guide star coordinates, derived jitter at the SI aperture, pertinent guiding-related flags, orbital data (e.g., latitude, longitude, limb angle, magnetic field values, etc.) and FGS_flags. There are examples of headers and tables in the online Observation Logs documentation located at:

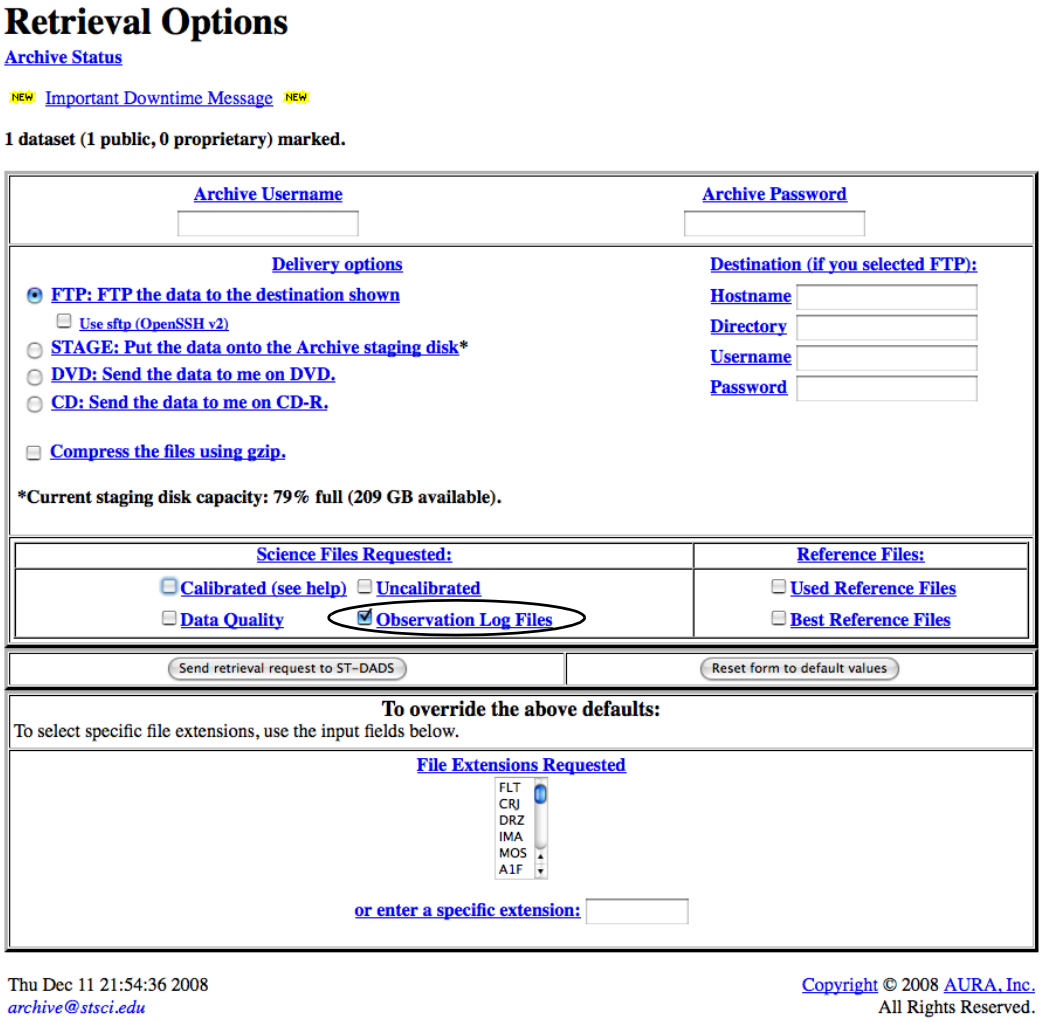
  `http://www.stsci.edu/hst/observatory/obslog/OL_3.html#HEADING2`

6.2 Retrieving Observation Logs

You can retrieve observation log files from the HDA, located at http://archive.stsci.edu/cgi-bin/dataset_input/. Unlike science data, which generally have a one-year proprietary period, observation log files become public as soon as they are archived.

The easiest way to get observation log files is to identify the observation of interest and then proceed with your request, as described in Chapter 2.9 of the ***STScI Archive Manual***, until you reach the "Retrieval Options" screen, shown in Figure 6.1. You then need to select "Observation Log Files" in the "Science Files Requested" section of the page. The HDA will then deliver the associated observation log files. You will receive the "`_jif.fits`" and "`_jit.fits`" files.

Figure 6.1: Choosing Observation Log Files in MAST



6.3 Using Observation Logs

Here are some simple examples of what can be learned from the observation log files. Note that for FITS format observation logs, current versions of **STSDAS** tools will handle the files with extensions properly. Keywords can be viewed with tools such as **imheader** or **hedit**, and data viewed, plotted, or displayed using the same tasks one might have for the GEIS files. For more information on FITS file structures, see Chapter 2.

6.3.1 Guiding Mode

Unless requested, all observations will be scheduled with FINE LOCK guiding, which may be one or two guide stars (dominant and roll). The spacecraft may roll

slightly during an observation if only one guide star is acquired. The amount of roll depends upon the gyro drift at the time of the observation, the location during an orbit, and the lever arm from the guide star to the center of the aperture.

There are three commanded guiding modes: FINE LOCK, FINE LOCK/GYRO, and GYRO. OMS header keywords GUIDECMD (commanded guiding mode) and GUIDEACT (actual guiding mode) will usually agree. If there was a problem, they will not agree and the GUIDEACT value will be the guiding method actually used during the exposure. If the acquisition of the second guide star fails, the spacecraft guidance, GUIDEACT, may drop from FINE LOCK to FINE LOCK/GYRO, or even to GYRO, which may result in a target rolling out of an aperture. Check the OMS header keywords to verify that there was no change in the requested guiding mode during the observation.

Until flight software version FSW 9.6 came online in September 1995, if the guide star acquisition failed, the guiding dropped to COARSE track. After September 1995, if the guide star acquisition failed, the tracking did not drop to COARSE track. Archival researchers may find older datasets that were obtained with COARSE track guiding.

The dominant and roll guide star keywords (GSD and GSR) in the OMS header can be checked to verify that two guide stars were used for guiding, or in the case of an acquisition failure, to identify the suspect guide star. The dominant and roll guide star keywords identify the stars that were scheduled to be used, and in the event of an acquisition failure, may not be the stars that were actually used. The following list of observation log keywords is an example of two star guiding. These keywords are found in the `jif` file or, for older data, the `cmh` file.

```
GSD_ID  = '0853601369        ' / Dominant Guide Star ID
GSD_RA  =            102.42595 / Dominant Guide Star RA (deg)
GSD_DEC =            -53.41362 / Dominant Guide Star DEC (deg)
GSD_MAG =               11.251 / Dominant Guide Star Magnitude
GSR_ID  = '0853602072        ' / Roll Guide Star ID
GSR_RA  =            102.10903 / Roll Guide Star RA (deg)
GSR_DEC =            -53.77683 / Roll Guide Star DEC (deg)
GSR_MAG =               12.426 / Roll Guide Star Magnitude
```

The guide star identifications, GSD_ID and GSR_ID, are different for the two Guide Star Catalogs: GSC2 IDs are 10-characters in length, like those of GSC1, but consist of both letters and numbers. GSC1 IDs consist entirely of numbers.

The GSC2 catalog is the default catalog since Cycle 15 (June 2006). The keyword REFFRAME in the primary science header indicates which catalog was in use for an observation. This keyword is included in all Cycle 15 and later observations, and is either "GSC1" for Guide Star Catalog 1, or "ICRS" for International Celestial Reference System, upon which GSC2 coordinates are based. The same information is added to the HST Archive catalog file "shp_refframe" of the "shp_data" database table since June 2006. For more information about the catalogs and their astrometric accuracy, see: http://www-gsss.stsci.edu/Catalogs/Catalogs.htm

If you suspect that a target has rolled out of the aperture during an exposure, you can quickly check the counts in each group of the raw science data. As an example, the following **IRAF** commands can be used to determine the counts in each group.

```
cl> grlist z2o4040dt.d0h 1-24 > groups.lis
cl> imstat @groups.lis
```

Some GHRS observations can span several orbits. If during a multiple orbit observation the guide star reacquisition fails, the observation may be terminated with possible loss of observing time, or switch to other less desirable guiding modes. The GSACQ keyword in the `cmh` header will state the time of the last successful guide star acquisition.

```
GSACQ   = '136:14:10:37.43   ' / Actual time of GS Acquisition Completion
```

6.3.2 Guide Star Acquisition Failure

The guide star acquisition at the start of the observation set could fail if the FGS fails to lock onto the guide star. The target may not be in the aperture, or maybe only a piece of an extended target is in the aperture. The jitter values will be increased because FINE LOCK was not used. The following list of observation log header keywords indicate that the guide star acquisition failed.

```
V3_RMS =                  19.3 / V3 Axis RMS (milli-arcsec)
V3_P2P =                 135.7 / V3 Axis peak to peak (milli-arcsec)
GSFAIL = '          DEGRADED' / Guide star acquisition failure!
```

The observation logs for all of the exposures in the observation set will have the "DEGRADED" guide star message. This is not a Loss-of-Lock situation but an actual

failure to acquire the guide star in the desired guiding mode. For the example above, the guiding mode dropped from FINE LOCK to COARSE TRACK.

```
GUIDECMD= 'FINE LOCK         ' / Commanded Guiding mode
GUIDEACT= 'COARSE TRACK      ' / Actual Guiding mode at end of GS acquisition
```

If the observation dataset spans multiple orbits, the guide star will be reacquired, but the guiding mode will not change from COARSE TRACK. In September 1995, the flight software was changed so that COARSE TRACK is no longer an option. The guiding mode drops from two guide star FINE LOCK to one guide star FINE LOCK, or to GYRO control.

6.3.3 Moving Targets and Spatial Scans

A type 51 slew is used to track moving targets (planets, satellites, asteroids, and comets). Observations are scheduled with FINE LOCK acquisition, i.e., with two or one guide stars. Usually, a guide star pair will stay within the pickle during the entire observation set, but if two guide stars are not available, a single guide star may be used, assuming the drift is small or the proposer says that the roll is not important for that particular observing program. An option during scheduling is to drop from FGS control to GYRO control when the guide stars move out of the FGS. Also, guide star handoffs (which are not a simple dropping of the guide stars to GYRO control) will affect the guiding and may be noticeable when the jitter ball is plotted.

In two-gyro mode, all observations are scheduled with two guide stars. Proposers cannot request the use of only one guide star.

Guide star handoffs are not allowed in two-gyro mode.

The jitter statistics are accumulated at the start of the observation window. Moving targets and spatial scan motion will be seen in the jitter data and image. Therefore, the OMS header keywords V2_RMS and V3_RMS values (the root mean square of the jitter about the V2- and V3-axes) can be quite large for moving targets. Also, a special anomaly keyword (SLEWING) will be appended to the OMS header indicating movement of the telescope occurred during the observation. This is expected when

observing moving targets. The following list of OMS header keywords is an example of expected values while tracking a moving target.

```
                       / LINE OF SIGHT JITTER SUMMARY
V2_RMS  =         3.2 / V2 Axis RMS (milli-arcsec)
V2_P2P  =        17.3 / V2 Axis peak to peak (milli-arcsec)
V3_RMS  =        14.3 / V3 Axis RMS (milli-arcsec)
V3_P2P  =        53.6 / V3 Axis peak to peak (milli-arcsec)
RA_AVG  =  244.01757 / Average RA (deg)
DEC_AVG =  -20.63654 / Average DEC (deg)
ROLL_AVG=  280.52591 / Average Roll (deg)
SLEWING = '        T' / Slewing occurred during this observation
```

6.3.4 High Jitter

The spacecraft may shake during an observation, even though the guiding mode is FINE LOCK. This movement may be due to a micro-meteorite hit, jitter at a day-night transition, or for some other unknown reasons. The FGS is quite stable and will track a guide star even during substantial spacecraft motion. The target may move about in an aperture, but the FGS will continue to track guide stars and reposition the target into the aperture. For most observations, the movement about the aperture during a spacecraft excursion will be quite small, but sometimes, especially for observations with the spectrographs, the aperture may move enough that the measured flux for the target will be less than a previous group. Check the OMS header keywords (V2_RMS, V3_RMS) for the root mean square of the jitter about the V2- and V3-axes. The following list of header keywords, found in the `jif` or older `cmh` files, is an example of typical guiding rms values.

```
                                / LINE OF SIGHT JITTER SUMMARY
V2_RMS  =                   2.6 / V2 Axis RMS (milli-arcsec)
V2_P2P  =                  23.8 / V2 Axis peak to peak (milli-arcsec)
V3_RMS  =                   2.5 / V3 Axis RMS (milli-arcsec)
V3_P2P  =                  32.3 / V3 Axis peak to peak (milli-arcsec)
```

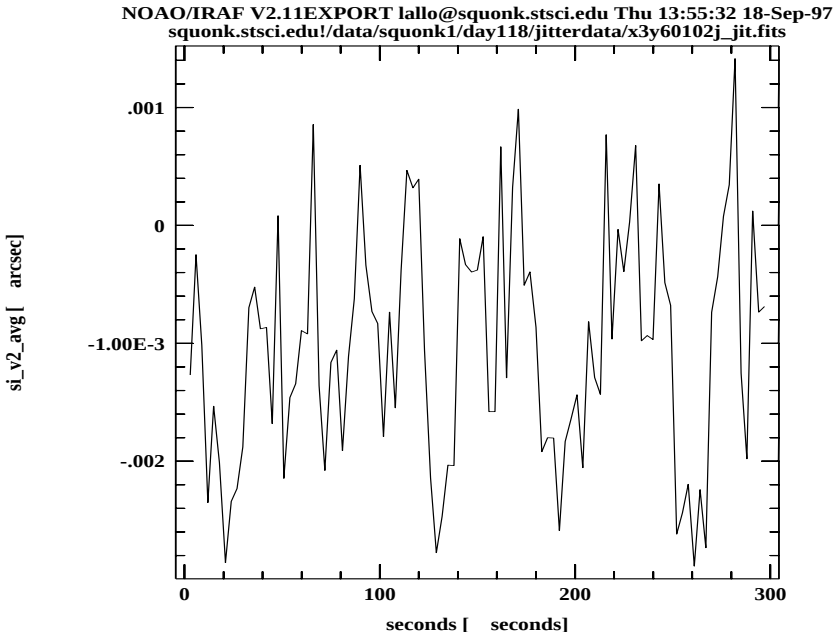
Recentering events occur when the spacecraft software decides that shaking is too severe to maintain lock. The FGS will release guide star control and within a few seconds reacquire the guide stars. It is assumed the guide stars are still within the FGS field of view. During the recentering time, INDEF will be written to the OMS table. Recentering events are tracked in the OMS header file.

Be careful when interpreting “Loss-of-Lock” and “Recentering” events that occur at the very beginning or at the end of the OMS window. The OMS window is larger than the observation window. These events might not affect the observation since the observation will start after the guide stars are acquired (or reacquired), and the observation may stop before the “Loss-of-Lock” or “Recentering” event that occurred at the end of an OMS window.

The **sgraph** command in the **stsdas.graphics.stplot** package can be used to plot time vs. jitter along the direction of *HST*'s V2-axis (see Figure 6.2):

```
cl> sgraph "x3y60102j_jit.fits seconds si_v2_avg"
```

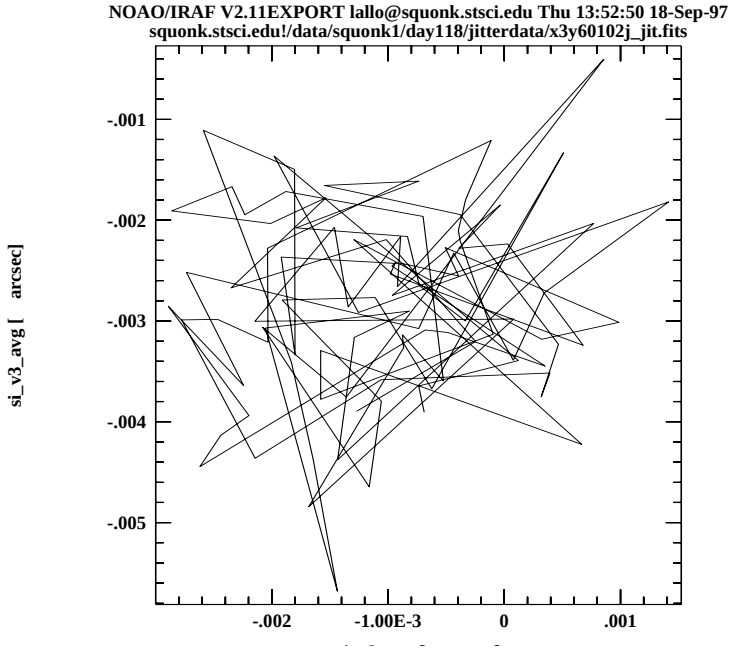
Figure 6.2: Plotting Jitter Along V3-axis



To get an idea of pointing stability, you can create a *jitter ball* by plotting jitter along the V2-axis vs. jitter along the V3-axis (see Figure 6.3):

```
st> sgraph "x3660102j_jit.fits si_v2_avg si_v3_avg"
```

Figure 6.3: Plotting V2 vs. V3 Jitter



The **tstatistics** task can be used to find the mean value of the `si_v3_avg` column—the amount of jitter (in arc seconds) in the direction of the V3. This value can be used to model jitter in a PSF. In this example, the mean jitter is ~3 mas, which is typical for *HST* data:

Figure 6.4: Averaging a Column with **tstatistics**

```
tt> tstat u26m0801j.cmi si_v3_avg
# u26m0801j.cmi  si_v3_avg
#
nrows            mean        stddev       median           min          max
   11   -0.003006443888  0.00362533  -7.17163E-4  -0.00929515  0.00470988
```

Understanding and interpreting the meaning of the table columns and header keywords is critical to understanding the observation logs. Please read the available documentation and contact the STScI Help Desk (help@stsci.edu) if you have any questions about the files. In particular please follow the following links for:

- ***Table content definitions.***
- ***Useful header keyword definitions.***
- ***Discussion on sources of HST pointing errors.***

Index