Version 2.0
December, 2003

ACS Data Handbook

SPACE TELESCOPE SCIENCE INSTITUTE

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

User Support

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

- **E-mail:** help@stsci.edu
- **Phone:** (410) 338-1082 or 1-800-544-8125

World Wide Web

Information and other resources are available on the ACS World Wide Web page:

- URL: http://www.stsci.edu/hst/acs/

ACS Revision History

Version	Date	Editor
2.0	November 2003	Jennifer Mack and Ron Gilliland, Editors, ACS Data Handbook
1.0	January 2002	Jennifer Mack, Editor, ACS Data Handbook Bahram Mobasher, Chief Editor, HST Data Handbook

Authorship

This document is written and maintained by the ACS+WFPC2 Group in the Instruments Division of STScI which consists of the following individuals: Biretta, J., Boffi, F., Bohlin, R., Cox, C., Fruchter, A., Giavalisco, M., Gilliland, R., Gonzaga, S., Heyer, I., Hook, R., Jogee, S., Koekemoer, A., Mack, J., Mutchler, M., Pavlovsky, C., Platais, V., Riess, A., Sirianni, M., van der Marel, R., and van Orsow, D. Other scientists who contributed to the writing of this handbook include Clampin, M., de Marchi, G., Hack, W., Hartig, G., Krist, J., Sparks, W., and Welty, A. at STScI or formerly at STScI, Blakeslee, J. and Meurer, G., of the ACS Science Team, and Pasquali, A. and Walsh, J. at the ST-ECF.

In publications, refer to this document as:

Mack, J., et al. 2003, "ACS Data Handbook", Version 2.0, (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

Part II: ACS Data Handbook

Part III: Appendixes

Preface

Introduction to Reducing HST Data

This data handbook provides an introduction to the process of retrieving and reducing Hubble Space Telescope (HST) data. The reduction procedures, calibrations, and sources of error specific to each active HST instrument (WFPC2, STIS, NICMOS and ACS) are explained in detail in their respective instrument data handbooks. However, we recommend a careful reading of this handbook before proceeding to the instrument data handbooks and before starting to work on your HST data. The present document is an updated version of chapters 1-3 of the HST Data Handbook, v.3.0, and is based on the information available as of December 2001. In particular, it is written under the assumption that the ACS and NICMOS instruments will be fully functional following HST Servicing Mission 3B (SM3B).

Many changes in the HST Data Archive and HST data reduction software have occurred since v. 3.0 of the Hubble Data Handbook. These differences are covered in this document and include, but are not limited to:

1. Expansion of the HST Data Archive into the Multimission Archive at Space Telescope (MAST), which currently includes 14 satellite mission archives as well as ground-based survey data.
2. The ability to retrieve MAST data using the World Wide Web.
3. New capabilities of the StarView program for searching for and retrieving HST and other MAST data.
4. The introduction of PyRAF, a new Python-based alternative to the IRAF cl shell, and
5. A new distinction between waiver FITS format, used to archive data from the older HST instruments such as WFPC and FOS, and the FITS extension format used for the newest instruments (STIS, NICMOS and ACS).

Future changes in this handbook are anticipated as MAST expands to cover additional missions, and as StarView and PyRAF evolve. The reader is advised to consult the STScI web site at http://resources.stsci.edu for the latest information. Moreover, as the present revision comes before SM3B, important revisions to the ACS file structure and data handling may be necessary after the installation of this instrument.

Bahram Mobasher (Chief Editor, HST Data Handbook)
Michael Corbin (Editor, Chapter 1)
Jin-chung Hsu (Editor, Chapters 2 and 3)

PART I:

Introduction to Reducing HST Data

The chapters in this part provides an introduction to the process of retrieving and reducing Hubble Space Telescope (HST) data.

CHAPTER 1:

Getting HST Data

In this chapter. . .

This chapter describes how to obtain Hubble Space Telescope (HST) data files. All 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 either of two ways:

- Electronically over the Internet from the HDA, where data are stored immediately after they pass through HST pipeline processing.
- On data storage media written at STScI from the HDA. The options are Exabyte and DAT tapes, and will include CDs and DVDs in the future.

To retrieve data electronically you must first register as a MAST user[2]. HST Principal Investigators (PIs) are ***not*** automatically registered. If you have not recently retrieved data, you should register or renew your registration before retrieving data from the HDA. PIs should register before their observations are made. GTO and GO observations normally remain proprietary for a period of one year, which means that during this period

1. MAST currently includes data from HST, FUSE, IUE, EUVE, ASTRO, HUT, UIT, WUPPE, ORFEUS, BEFS, IMAPS, TUES, Copernicus and ROSAT. Data from the FIRST radio survey, Digital Sky Survey (DSS) and Sloan Digital Sky Survey (SDSS) are also available.

2. By 2002, registration will no longer be required for public (non-proprietary) data.

other registered users cannot retrieve them without authorization from the PI. All calibration observations as well as observations made as part of the Public Parallel programs are immediately public. All observations made as part of the Treasury Programs begun in Cycle 11 will either be immediately public or have only a brief proprietary period. The HST section of MAST also contains several Prepared (fully reduced) data sets, including the Hubble Deep Fields, the Hubble Medium Deep Survey Fields, and a composite quasar spectrum, which are also public.

This chapter describes how to search the HDA, how to electronically retrieve files from it, and how to request and read tapes and disks containing HST data. As an aid to retrieving their data, PIs will automatically receive e-mail notification of the status of their observations two times: first, when the first datasets for their proposal are archived, and second, when all the datasets for their proposal and all necessary calibration files have been archived.

Note for Advanced Camera for Surveys (ACS) Users: Calibrated ACS images are approximately 168 MB in size, larger than those of any other HST instrument. Therefore, electronic retrieval of ACS data is enabled only for those with broadband (> 100 KB/s) Internet connections, in order to ensure uninterrupted transmission of individual files. Users retrieving large numbers of ACS files should also consider requesting them on tape or disk.

1.1 Archive Overview

The HDA contains all HST observations ever made. It also contains a database that catalogs and describes these observations. There are currently two ways to search and retrieve data from the HDA. The first is a program called StarView, which acts as an interface to the HDA. StarView currently runs as Java-based, stand-alone application that can be downloaded from the web site http://starview.stsci.edu. Previous versions of StarView, such as XStarView, are no longer available. The second search and retrieval method is through the HST section of the MAST web site, http://archive.stsci.edu. StarView is the more powerful of the two methods, and in particular allows an examination of the calibration files applied to a given data file. StarView also provides an interface to the Visual Target Tool (VTT) in the Astronomer’s Proposal Tool (APT) suite of programs. The VTT interface can display archive observations on a Digital Sky Survey (DSS) image alongside planned observations. StarView is thus recommended for observation planning, duplication checking, calibration file review, investigation of On-The-Fly Reprocessing flags and proprietary status. It is also recommended for those needing to retrieve large numbers of datasets, and those needing to examine calibration files. The MAST web site interface to the HDA has the same basic capabilities as StarView, and may be preferable for those requiring simple retrievals of datasets. Both StarView and the MAST web site allow simultaneous searches of the other MAST mission archives for all HDA searches. They also offer simple preview of the capabilities of HST datasets when available, as well as links to literature references citing a given dataset, using the Astrophysics Data System (ADS). In later sections we discuss StarView and the MAST web site in more detail.

1.1.1 Archive Registration

The simplest way to register and retrieve HST data is to complete the form on the Web page at: http://archive.stsci.edu/registration.html.

Registration requests may also be sent to the HDA hotseat, at: archive@stsci.edu.

The PI of each HST proposal must request proprietary access for their data, and for anyone else whom the PI wants to have access to it. PI retrieval permission is not granted automatically, for security reasons. PIs wishing to allow others access to their proprietary data should make that request to archive@stsci.edu.

When registration is granted, your account will be activated within two working days, and you will receive your username and password via e-mail.

1.1.2 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 to archive_news-request@stsci.edu and putting the single 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 Getting Data with StarView

1.2.1 Downloading and Setting Up StarView

The latest version of StarView runs under versions 1.2.2 and later of Java and may be downloaded from http://starview.stsci.edu.

This site also includes a FAQ page and news on releases and updates. StarView will automatically update itself to the latest version, so users do not have to worry about additional installations. Following its installation on computers running Unix and Linux, begin StarView by typing

```
> StarView
```

at the system prompt. Under Windows and MacIntosh systems, StarView will appear as an icon. The StarView session then begins, first with an Information window explaining navigation within StarView, and a request for the user to specify an object name resolver (SIMBAD or NED) for use in HDA searches. The first-time user are asked to supply their e-mail information in order to allow StarView to communicate the results of its attempts to retrieve the files requested from the HDA. This e-mail information includes the user's SMTP host, or the computer from which e-mail messages are routed. If unsure of your SMTP host, ask your system administrator. These queries can be turned off for future sessions once this information has been supplied.

1.2.2 Simple Use of StarView

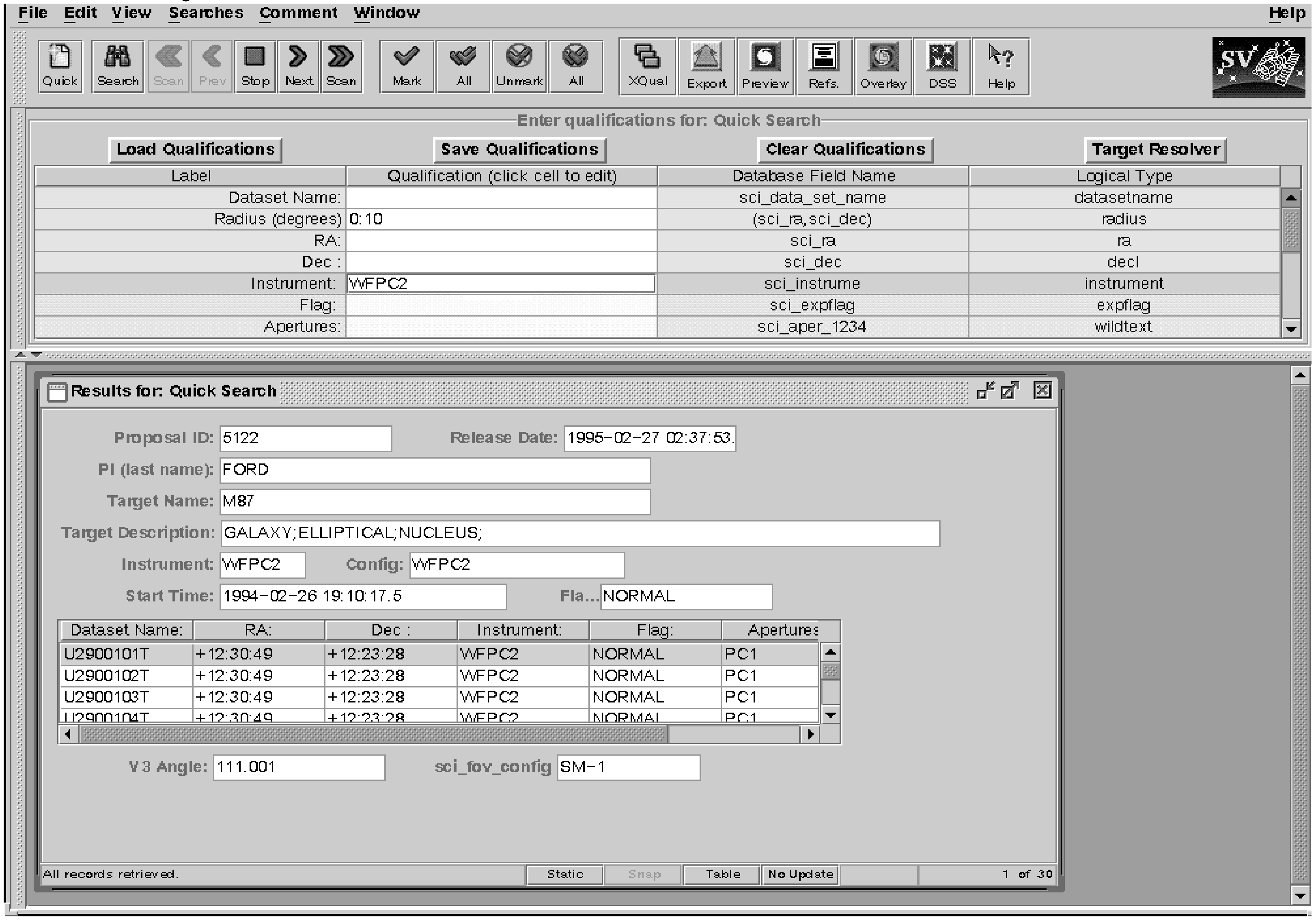
We now proceed to an introduction to the use of StarView. A more detailed description of its capabilities is provided at the web site above, which should also be consulted for more advanced topics such as its Table Exportation and Cross-Qualification functions.

The basic function of StarView is to enable the user to first search the HDA (and the other mission archives in MAST) for data files matching criteria such as object name, position, or proposal number, then allow the user to navigate through the set of files matching those criteria, and finally to let the user retrieve some or all of the files found in the search. Several options for the type of search that can be performed (e.g. by a particular instrument) will be discussed later.

The design of StarView is similar to that of a Web browser. At its top are pull-down menu bars including File, Searches, and Help. The Help menu offers links to documents including the StarView FAQ page. Beneath these menu bars is a row of buttons that run StarView's basic functions, such as searching, marking files for retrieval, and previewing images. A Help button allows users to display pop-up windows describing the function of the different StarView buttons and windows, by first clicking the Help button, then the item of interest. Beneath the row of buttons is the Qualifications table, which is displayed when a search is begun. It consists of several cells corresponding to the search parameters the user wishes to use, e.g., object name, proposal I.D., or instrument. Below this window will appear the Results table, displaying the datasets found to match a given set of search parameters entered into the Qualifications table. For the purpose of introduction, we will describe the use of the most basic search option, called “Quick Search,” which can be started by clicking the “Quick” button at the top left of StarView.

As an example of the use of the Quick Search option, we will request all available WFPC2 data for the galaxy M87. This is done by typing “WFPC2” and “M87” in the Instrument and Target Name cells of the Qualifications section, then clicking the “Search” button at the top left of the StarView window. The results of the search will then be displayed in the bottom panel of StarView, as shown in Figure 1.1. These results include the dataset name, instrument name, R.A. and Dec of the target, and the instrument aperture used. Note that these parameters could also have been specified in the Qualifications section, as can other parameters including proposal I.D. number, proposal P.I. name, and image central wavelength (corresponding to particular instrument filters or gratings).

Figure 1.1: Results of StarView Quick Search for WFPC2 files of M87



Enter qualifications for: Quick Search

Label	Qualification (click cell to edit)	Database Field Name	Logical Type
Dataset Name:		sci_data_set_name	datasetname
Radius (degrees)	0:10	(sci_ra,sci_dec)	radius
RA:		sci_ra	ra
Dec :		sci_dec	decl
Instrument:	WFPC2	sci_instrume	instrument
Flag:		sci_expflag	expflag
Apertures:		sci_aper_1234	wildtext

Results for: Quick Search

Proposal ID:	5122
Release Date:	1995-02-27 02:37:53.
PI (last name):	FORD
Target Name:	M87
Target Description:	GALAXY;ELLIPTICAL;NUCLEUS;
Instrument:	WFPC2
Config:	WFPC2
Start Time:	1994-02-26 19:10:17.5
Fla...	NORMAL

Dataset Name:	RA:	Dec :	Instrument:	Flag:	Apertures
U2900101T	+12:30:49	+12:23:28	WFPC2	NORMAL	PC1
U2900102T	+12:30:49	+12:23:28	WFPC2	NORMAL	PC1
U2900103T	+12:30:49	+12:23:28	WFPC2	NORMAL	PC1
U2900104T	+12:30:49	+12:23:28	WFPC2	NORMAL	PC1

V3 Angle:	111.001
sci_fov_config	SM-1

All records retrieved.

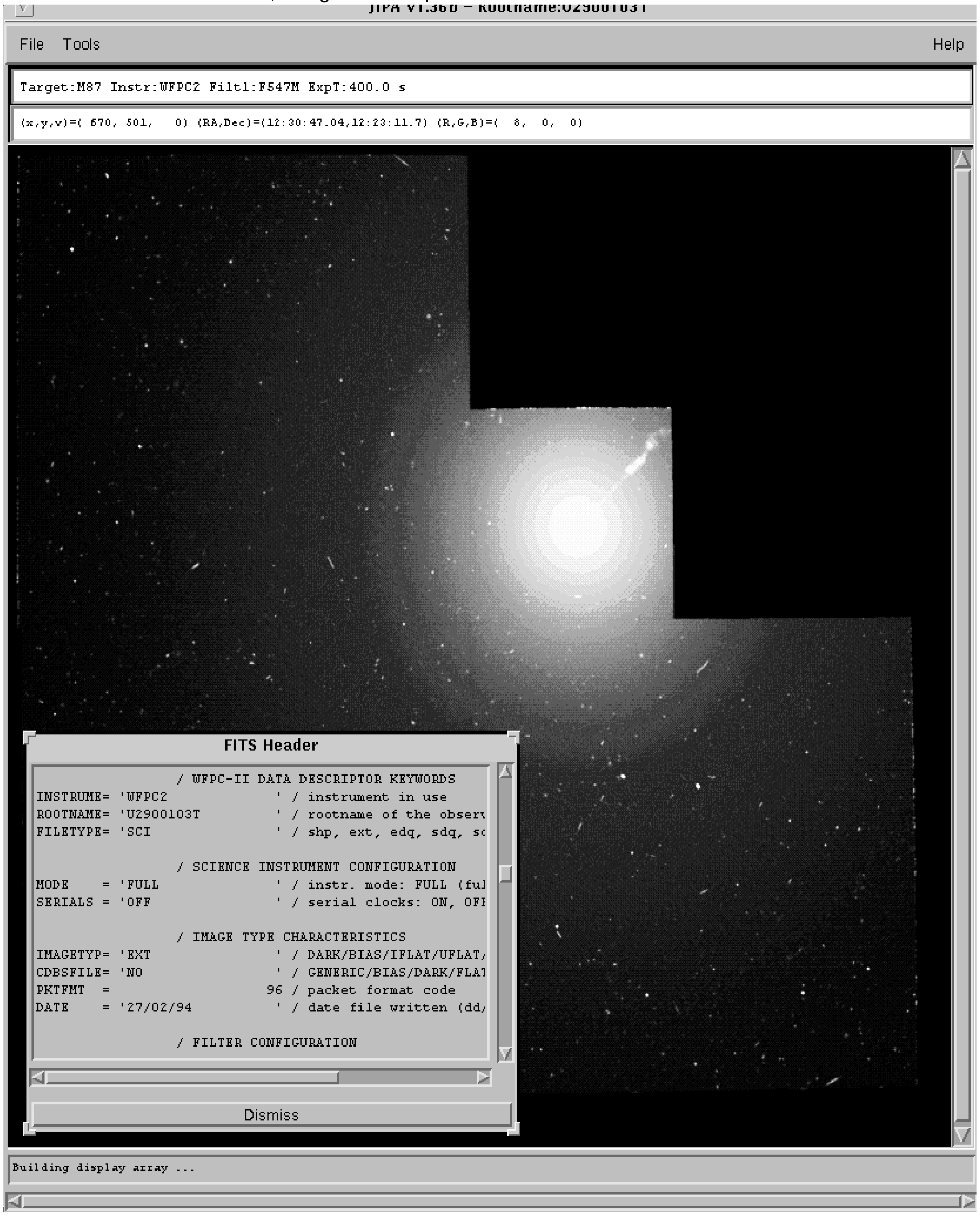
1 of 30

Clicking on a given dataset in the Results table will display the information shown in the cells above it (Proposal ID, Release Date, P.I., etc.). You may browse through the retrieved datasets either by using the mouse and scroll bar, or by using the navigation buttons (Scan, Previous, Next) in the top row of mouse buttons. The Scan option will automatically step through all of the files retrieved in the search, provided that the right most button at the bottom of the Results window is toggled to "Update." If this button is toggled to "No Update," the Scan option will go straight to the end of the list of files.

The ability to obtain a preview is available for many, but not all, of the datasets in the HDA (e.g., previews are not available for many FOC datasets). This is done with the "Preview" button, if it is enabled. For images, this will display a re-sampled version of the image using the Java Image Preview Application (JIPA) tool that is part of StarView. For spectra, a simple GIF image of the calibrated spectrum will be displayed. JIPA and VTT can also display an image's FITS header, under the JIPA "Tools" menu. The JIPA preview of the WFPC2 image U2900103T retrieved in the previous search for WFPC2 images of M87 is shown in Figure 1.2, along with the window displaying part of the FITS header file of this image.

Other display options with StarView include "DSS," which will display a 20' x 20' Digital Sky Survey image at the target coordinates, while the "Overlay" button will display the same DSS image with outlines of the HST instrument apertures at the target coordinates superimposed on it, at the orientation of the observation selected. The "References" button provides a link to any known published papers citing the dataset, as listed in ADS. Note that the HST images displayed by the Preview are of reduced quality compared to the actual data files, and cannot be downloaded. They are only meant to provide a quick check that the datasets found by the search met the search criteria, i.e., contained the object(s) of interest, and are of the desired quality.

Figure 1.2: JIPA preview of WFPC2 image U2900103T, along with image header file, using Preview option



```
JIPA V1.36b – Rootname:U2900103T
Target:M87 Instr:WFPC2 Filt1:F547M ExpT:400.0 s
(x,y,v)=( 670, 501,   0) (RA,Dec)=(12:30:47.04,12:23:11.7) (R,G,B)=(  8,  0,  0)
```

```
                 / WFPC-II DATA DESCRIPTOR KEYWORDS
INSTRUME= 'WFPC2             ' / instrument in use
ROOTNAME= 'U2900103T         ' / rootname of the obser
FILETYPE= 'SCI               ' / shp, ext, edq, sdq, sc

                 / SCIENCE INSTRUMENT CONFIGURATION
MODE    = 'FULL              ' / instr. mode: FULL (ful
SERIALS = 'OFF               ' / serial clocks: ON, OFF

                 / IMAGE TYPE CHARACTERISTICS
IMAGETYP= 'EXT               ' / DARK/BIAS/IFLAT/UFLAT/
CDBSFILE= 'NO                ' / GENERIC/BIAS/DARK/FLAT
PKTFMT  =                   96 / packet format code
DATE    = '27/02/94          ' / date file written (dd/

                 / FILTER CONFIGURATION
```

```
Building display array ...
```

1.2.3 Marking and Retrieving Data with StarView

Datasets are marked for retrieval by first clicking on them, then using the “Mark” button at the top of StarView. There is also the “All” button, which will mark all the datasets retrieved in the search. Marked datasets will be displayed in the Retrieval window. Datasets still within their proprietary period will be displayed in yellow, and users other than the proposal PI and those authorized by the PI will not able to retrieve them. The release date of files still within their proprietary period will also be indicated on the search results form.

If satisfied with the marked datasets, choose “Submit” in the Retrieval window to retrieve them. You will then be queried for both the type of data files associated with the dataset(s) to be retrieved, and for the method of delivery of these files. The options for type of file include files calibrated with the On-The-Fly-Recalibration (OTFR) pipeline for the WFPC2, NICMOS, STIS and ACS instruments. OTFR applies the best available calibration files (i.e., dark current and flat field images taken closest in time to the observations) to the uncalibrated data files. You may also request the uncalibrated (raw) files and calibration files separately. For some of the earlier instruments, e.g., WFPC and FOS, you may request both the calibration files actually applied to the images, as well as those that should provide the best calibration of them, if recalibration is desired. You may also request Data Quality and Observation Log files.

Options for data delivery include ftp transfer by the user from the HDA staging disk, automatic transfer from the HDA via the Internet to a directory specified by the user, and the mailing of tapes or disks. If Internet delivery is specified, you will be queried for the name of the computer and directory in which the files are to be placed, as well as your user name and password on that computer (these requests are encrypted, so there is no danger of your login information being stolen). Upon final submission of the request, you will receive an e-mail message acknowledging its receipt, and another message after all the requested files have been transferred. The status of the request, i.e., how many files have been transferred and any errors that have occurred, can be checked on a Web page that will be given in the acknowledgment message.

1.2.4 Using StarView to Retrieve Calibration Files and Proposal Information

StarView allows several additional types of searches of the HDA besides the Quick Search option described above. These can be selected from the Searches menu bar at the top of the StarView screen. One such search option is by instrument. This is necessary for identifying calibration reference files. As an example, selecting the option “WFPC2 OTFR” under the Instrument and WFPC2 sub-menus of the Searches menu, and then entering “M87” under Target Name in the qualifications box, brings up the screen shown in Figure 1.3. This screen shows all the calibration images and files applied by OTFR to the first file in the set of WFPC2 images of M87, as well as whether the application of these files was performed or omitted in the calibration pipeline. This is the same set of images found by the Quick Search query described above, and the same information for the other datasets from this search can be found using the Previous, Next and Scan buttons. Once these calibration images have been identified, further information on them can be obtained. For example, taking the name of the flat field file found in the above search and entering it into the “WFPC2 Calibration Data” Searches option will retrieve information on when and where this file was taken, and the date after which its use is recommended. This will help users decide if they would prefer to recalibrate their data using different files.

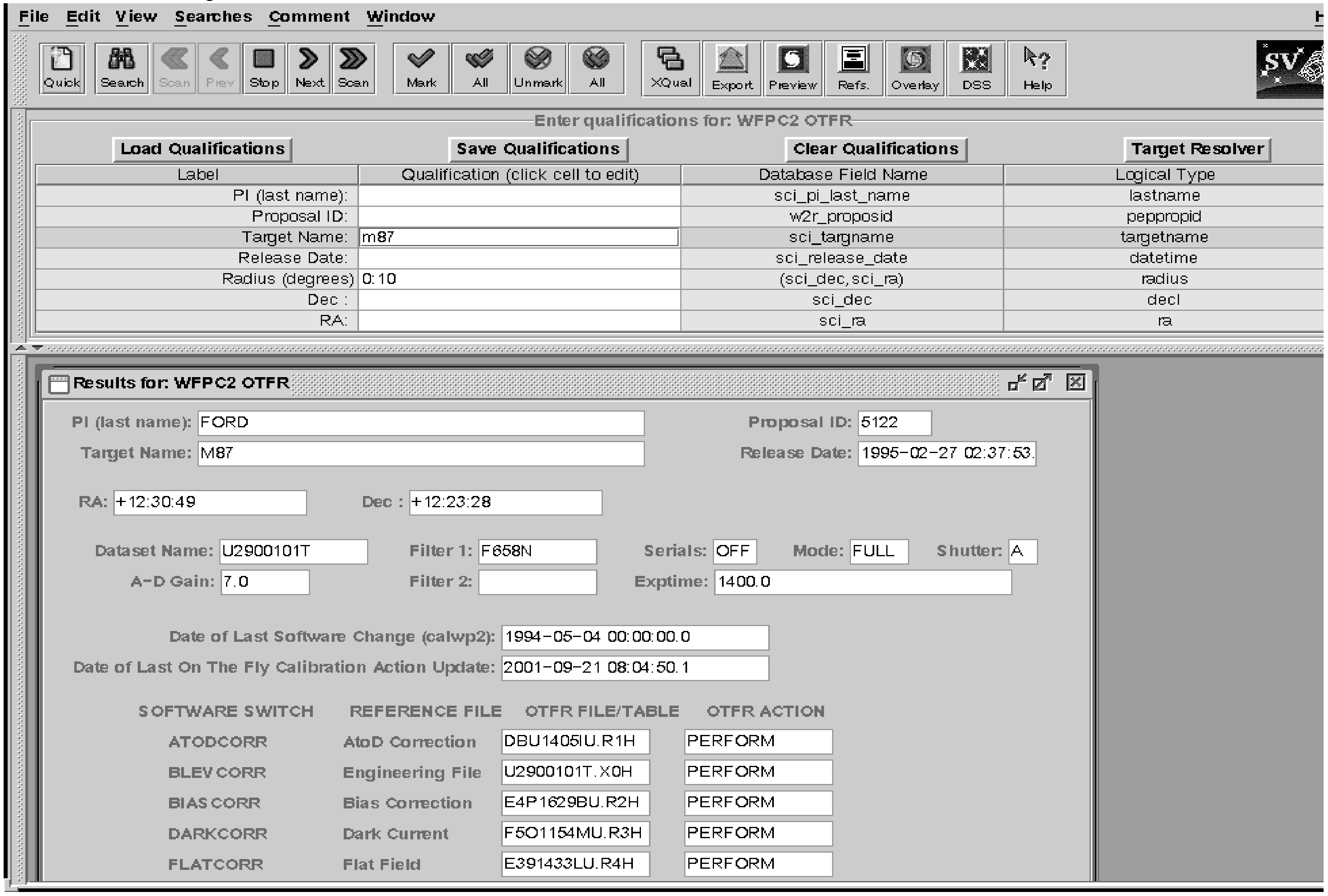
StarView can also be used to search for and view the abstracts of accepted HST proposals. Like the Preview capability of StarView, this provides additional information about a given dataset and whether it may be useful for your science goals. Viewing proposal abstracts is an option under the Searches menu, and an example is shown in Figure 1.4. The Qualifications window again offers several parameters by which this search can be constrained, including proposal I.D. number, HST cycle, P.I. name, and combinations thereof. In the example shown only the proposal I.D. number was used.

Finally, StarView can be used during the Phase I proposal process to see whether or not HST observations of a given object or object class have already been made, or else are scheduled for execution. Specifically, the Duplications option under the Searches menu allows users to check a database containing both HDA files and a list of queued observations in order to see if a given object has been or will be observed. Similarly, under Duplications the user may also query the database of proposal abstracts for a given object or object class, to check for archived or scheduled observations.

1.2.5 Advanced Features of StarView

In addition to its basic search and retrieval function, StarView allows users to cross-qualify results from separate searches of the HDA, and to export the results of searches to disk as ASCII files. These operations are performed with the “XQual” and “Export” buttons, respectively. As an example of cross-qualification, a user might want to identify all the spiral galaxies for which both WFPC2 images and STIS spectra have been obtained. This could be accomplished with the Cross-Qualification feature by first doing two separate Quick Searches, in which these respective instruments are specified in the query box, and in which “Galaxy;Spiral” is typed in the Target Description box for both searches. Clicking the XQual button, specifying “Target Name” as the common field in the two sets of search results (as shown in Figure 1.5), and clicking the “X-Qualify” button then identifies the galaxies occurring in both lists. StarView then places these galaxy names in the Target Name box of a new Quick Search window. Clicking the Search button with “WFPC2,STIS” entered for Instrument then gives a list of all the WFPC2 and STIS datasets for these galaxies. The Cross-Qualification function can also be performed on the files produced by the Export feature.

Figure 1.3: Results of StarView search for WFPC2 OTFR calibration files for M87



Enter qualifications for: WFPC2 OTFR

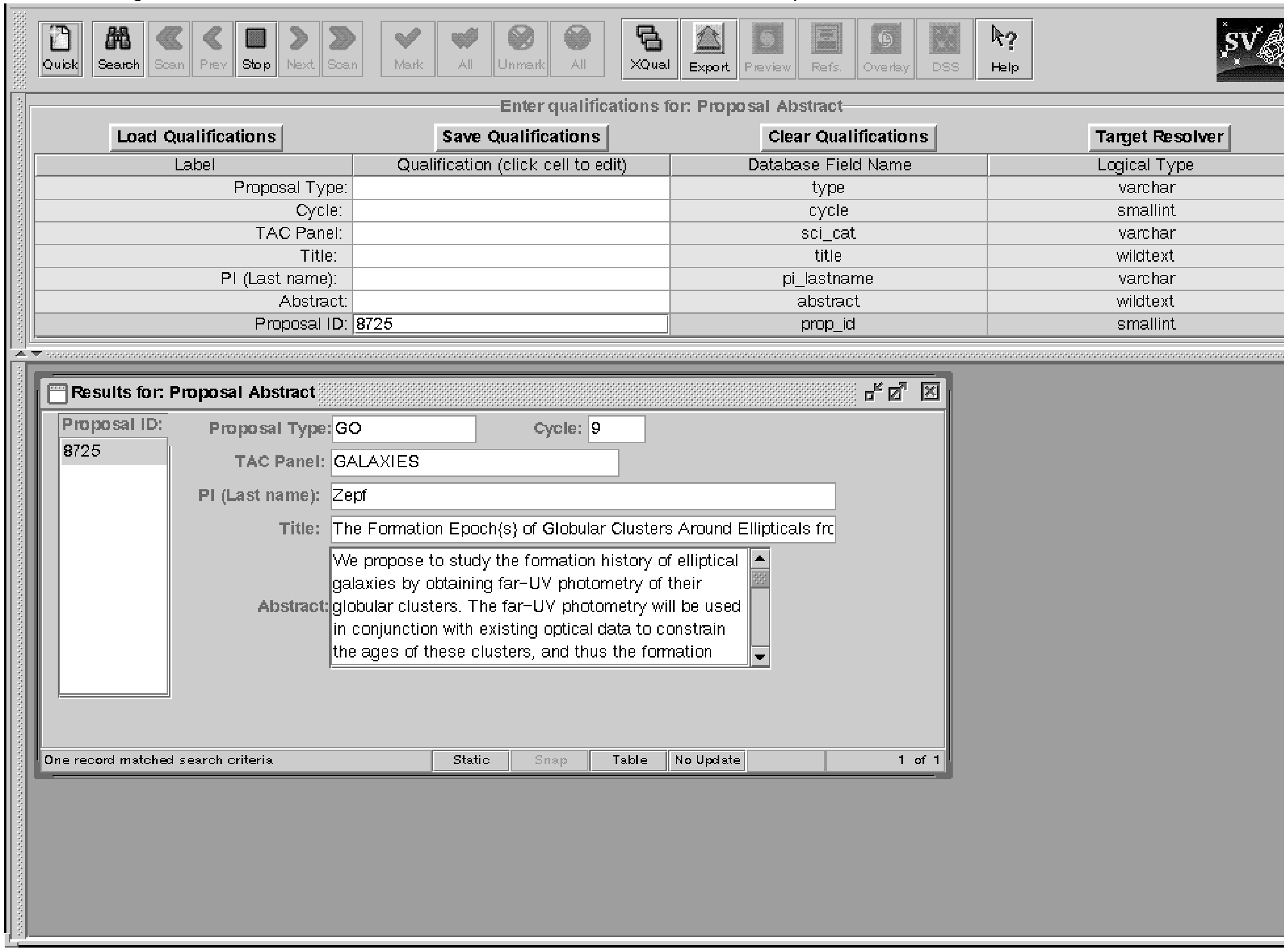
Label	Qualification (click cell to edit)	Database Field Name	Logical Type
PI (last name):		sci_pi_last_name	lastname
Proposal ID:		w2r_proposid	pepproprid
Target Name:	m87	sci_targname	targetname
Release Date:		sci_release_date	datetime
Radius (degrees)	0:10	(sci_dec,sci_ra)	radius
Dec :		sci_dec	decl
RA:		sci_ra	ra

Results for: WFPC2 OTFR

PI (last name): FORD
Proposal ID: 5122
Target Name: M87
Release Date: 1995-02-27 02:37:53.
RA: +12:30:49
Dec : +12:23:28
Dataset Name: U2900101T
Filter 1: F658N
Serials: OFF
Mode: FULL
Shutter: A
A-D Gain: 7.0
Filter 2:
Exptime: 1400.0
Date of Last Software Change (calwp2): 1994-05-04 00:00:00.0
Date of Last On The Fly Calibration Action Update: 2001-09-21 08:04:50.1

SOFTWARE SWITCH	REFERENCE FILE	OTFR FILE/TABLE	OTFR ACTION
ATODCORR	AtoD Correction	DBU1405IU.R1H	PERFORM
BLEVCORR	Engineering File	U2900101T.X0H	PERFORM
BIASCORR	Bias Correction	E4P1629BU.R2H	PERFORM
DARKCORR	Dark Current	F5O1154MU.R3H	PERFORM
FLATCORR	Flat Field	E391433LU.R4H	PERFORM

Figure 1.4: Results of the StarView search for the abstract of Proposal 8725



Enter qualifications for: Proposal Abstract

Label	Qualification (click cell to edit)	Database Field Name	Logical Type
Proposal Type:		type	varchar
Cycle:		cycle	smallint
TAC Panel:		sci_cat	varchar
Title:		title	wildtext
PI (Last name):		pi_lastname	varchar
Abstract:		abstract	wildtext
Proposal ID:	8725	prop_id	smallint

Results for: Proposal Abstract

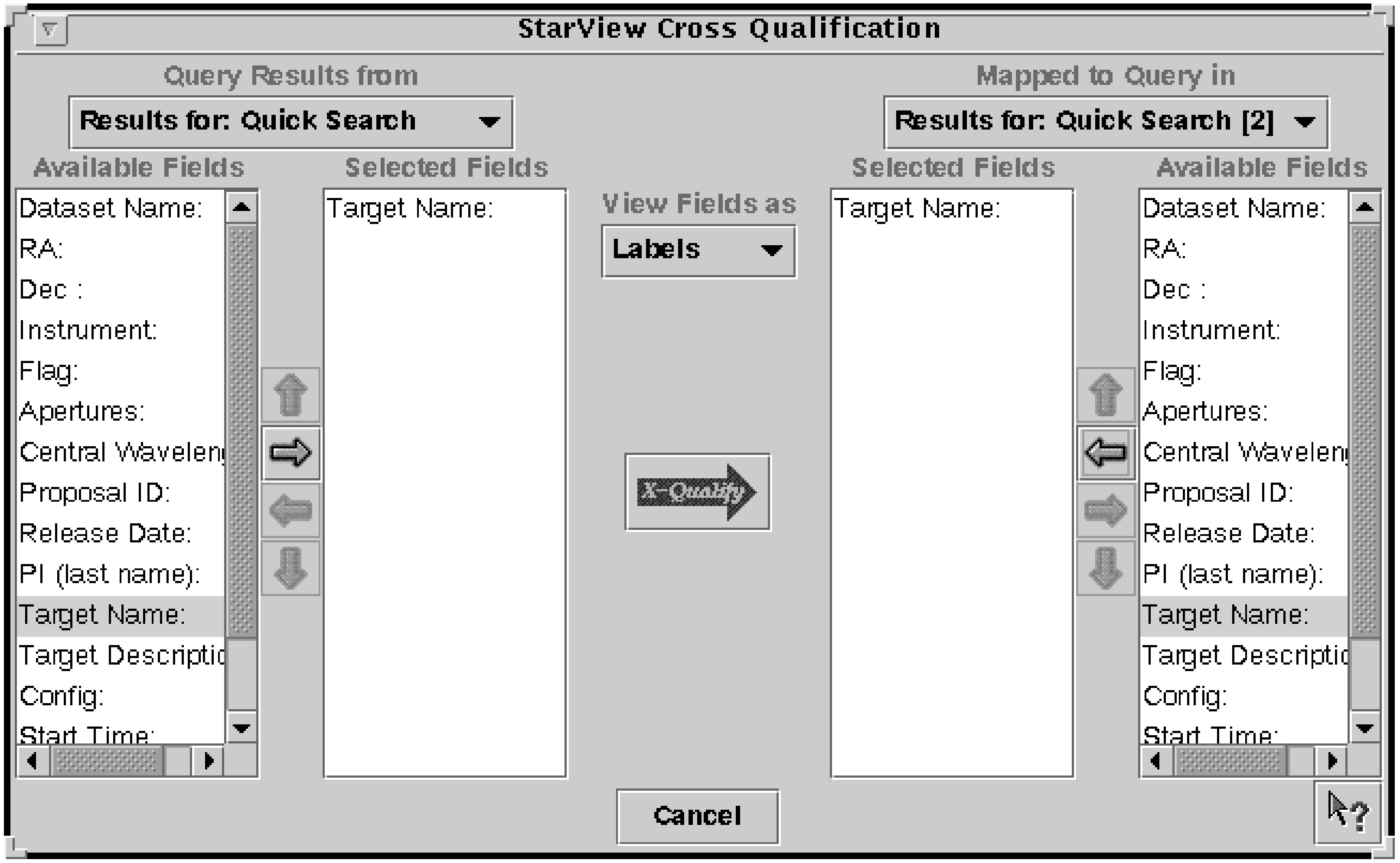
Proposal ID: 8725

Proposal Type: GO
Cycle: 9
TAC Panel: GALAXIES
PI (Last name): Zepf
Title: The Formation Epoch(s) of Globular Clusters Around Ellipticals frc
Abstract: We propose to study the formation history of elliptical galaxies by obtaining far-UV photometry of their globular clusters. The far-UV photometry will be used in conjunction with existing optical data to constrain the ages of these clusters, and thus the formation

One record matched search criteria

1 of 1

Figure 1.5: Example of Cross Correlation Feature, in which Target Name has been chosen as the common feature to search for in two Quick Search result lists.


StarView Cross Qualification

Query Results from: Results for: Quick Search

Mapped to Query in: Results for: Quick Search [2]

View Fields as: Labels

Selected Fields: Target Name:

Selected Fields: Target Name:

1.2.6 StarView and the Visual Target Tuner

The Visual Target Tuner (VTT) is part of the Astronomer's Proposal Tool (APT) package, which has been created to aid astronomers in planning their HST observations during the Phase I and Phase II proposal stages (see http://apt.stsci.edu). VTT is an image display tool which allows the user to display DSS images or local FITS images with proper World Coordinate System keywords in the headers. It offers more features than JIPA, which is the default StarView display tool. However, for a limited number of operating systems, the VTT can be used with StarView. VTT offers the particular advantage that it can overlay the instrument apertures of multiple observations on a single DSS image. Clicking on these apertures will also highlight the associated datasets in StarView.

Currently, to combine StarView and VTT requires downloading and installing APT from the above Web site. APT is only available for those operating systems with the Java Virtual Machine 1.3 (JVM 1.3). You can download the StarView/VTT package with JVM 1.3 included (a large download), or if you already have JVM 1.3 installed, you can get the smaller APT/VTT package. To make StarView use VTT, you must change your Viewer options from JIPA to VTT. Go to the Environment sub-menu of Edit in StarView, and change JIPA to VTT in the Viewers section. If VTT is not listed here, you should reinstall the two programs. Following this change, the Preview, DSS and Overlay buttons of StarView should all bring up VTT.

Once VTT has been installed, you can also bring up StarView from it. Enter in to StarView mode by clicking on the StarView button in the lower left hand menu of VTT. Clicking on a DSS image will then spawn a Quick StarView screen with the R.A. and Dec of the position you clicked loaded into the search fields. You can enter other constraints into these fields as usual. Search results can be displayed on the VTT screen by selecting the results in StarView, and pressing the Overlay button.

1.2.7 Quick Data Retrieval with StarView

The following steps summarize the basic process that PIs need to go through to retrieve their data with StarView. These steps follow registration as a MAST user, notification from STScI that the observations for a given proposal are complete, and providing StarView with your e-mail information. They are intended as a quick reference for this process.

1. Start StarView.
2. Click the “Quick” button.
3. Enter your PI name and/or proposal ID number in the appropriate cell.
4. Click the “Search” button.

5. Use the “Scan” button to step through the retrieved files, after toggling the right most button at the bottom of the Results window to “Update,” to verify that all datasets have been retrieved.
6. Preview some or all of the datasets if desired, to verify data quality and target acquisition.
7. Click “All” to mark all datasets for retrieval, or “Mark” to mark individual datasets for retrieval.
8. Click “Submit” in the window that will be spawned by marking the files.
9. Enter your MAST username and password and specify the means of data delivery. StarView remembers your name and password from past searches so it does not have to be entered each time.
10. Click “Done”, and your data are on their way. You will receive an e-mail message when your retrieval has been queued, and another when the transfer is complete.

1.3 Getting Data with the World Wide Web

HDA datasets can be searched for, previewed and retrieved via the World Wide Web in very much the same way as with StarView. As noted in Section 1.1, StarView offers more capabilities for this process, including cross-qualification, the use of VTT, and more information about instrument calibration files. However, Web retrievals may be preferable in some cases, particularly when information on calibration files is not needed, and the hypertext on the Results pages makes it easy to access all the information they contain. The starting point for Web-based searches of the HDA is the MAST web site at: http://archive.stsci.edu[3]

This web page is shown in Figure 1.6. A powerful feature of MAST is that all of its mission archives, including the HDA, can be searched simultaneously. This is done with the Cross-Correlation 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 “HST” on the MAST home page.

3. European archive users should generally use the ST-ECF Archive at http://archive.eso.org. 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.

The HST section of MAST offers tutorials about the HDA as well as a FAQ page and HDA news. It also provides links to HST Prepared datasets such as the Hubble Deep Field images. Clicking on the "Main Search Form" option of the HST section brings up the page shown in Figure 1.7. Here the user is queried for the same search parameters as requested by StarView, e.g., Object Name, Instrument and Proposal I.D. Once these are entered, clicking the Search button returns a page listing the datasets found, which can then be selectively marked for retrieval. The data type and retrieval options remain the same as those for StarView. Previews of GIF files of the datasets are also available.

1.4 Reading HST Data Tapes and Disks

If you request HDA files on tapes or disks, you will receive them within a few weeks of your request. The tapes will contain **tar** files containing the requested datasets. The datasets will all be in FITS (Flexible Image Transport System) format[4]. You should thus first create a directory where you want your data to reside, e.g., /home/myname/myhstdata, go to that directory, then read the tape or disk using the Unix/Linux **tar** command to read the FITS files into it.

Currently, datasets obtained with HST's original instruments (FGS, FOC, FOS, GHRS, HSP and WFPC) as well as WFPC2 must have their FITS files converted to GEIS (Generic Edited Information Set) format in order to work on them with IRAF/STSDAS. Further information on HST file formats is presented in Chapter 2. STSDAS is the package analysis software for HST data, and is discussed further in Chapter 3. Datasets obtained with the other current HST instruments (ACS, NICMOS and STIS) should be reduced in FITS format without conversion to GEIS. STSDAS support for the analysis of WFPC2 data in FITS format is currently planned.

The steps for reading and converting FITS files to GEIS files are as follows:

First bring up IRAF/STSDAS in your IRAF home directory by typing

```
> cl
```

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

This will start an IRAF session. IRAF and STSDAS are organized into *packages*. To load a package, type its name. To begin with, you must load the **stsdas** and **fitsio** (FITS Input/Output) packages:

```
cl> stsdas
st> fitsio
```

The IRAF prompt (such as st>) shows the first two letters of the most recently loaded package. The **fitsio** package contains the STSDAS programs (called *tasks* in the IRAF/STSDAS environment) required to read and write FITS files to and from tapes and disks. The two principle tasks are **strfits** for reading files, and **stwfits** for writing them.

Next, set the IRAF environment variable *imtype* to specify that your data files are to be written in GEIS format. This is done by typing

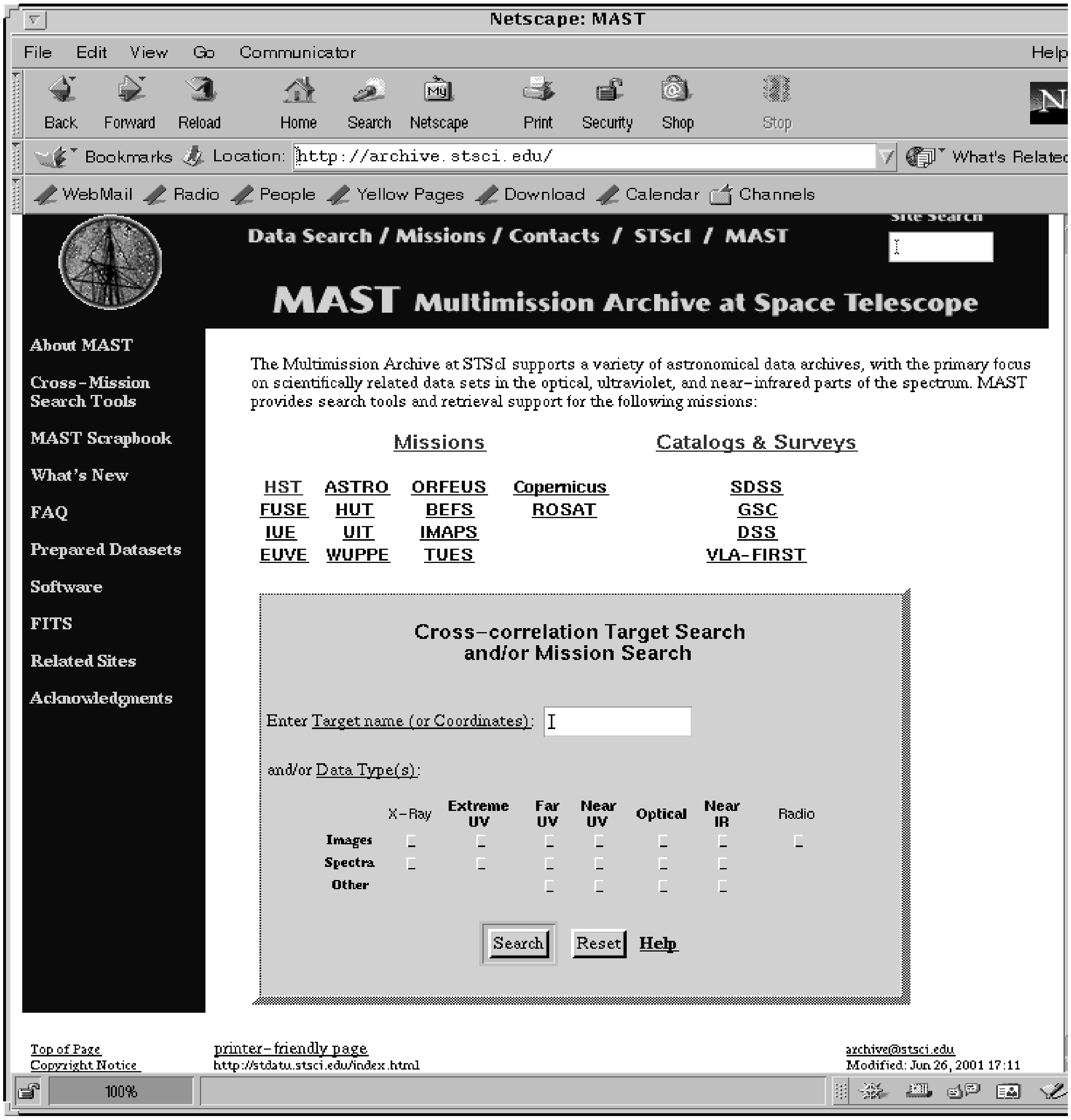
```
fi> set imtype="hhh"
```

You should then move to the directory containing the FITS files.

The last step is to use **strfits** to read the data. Like most IRAF/STSDAS tasks, **strfits** has several parameters that control its function. You can either edit these tasks using the IRAF "epar" command, or specify them on the command line. For the purpose converting FITS files to GEIS files, the important parameter is *oldirafname*, which needs to be set to "yes" in order to keep the file rootname the same. To convert all the FITS files in a directory to GEIS files, type

```
fi> strfits *.fits "  " oldirafname=yes
```

Figure 1.6: MAST Home Page


Netscape: MAST
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://archive.stsci.edu/ What's Related
WebMail Radio People Yellow Pages Download Calendar Channels
Site Search
Data Search / Missions / Contacts / STScI / MAST
MAST Multimission Archive at Space Telescope
About MAST
Cross-Mission Search Tools
MAST Scrapbook
What's New
FAQ
Prepared Datasets
Software
FITS
Related Sites
Acknowledgments

The Multimission Archive at STScI supports a variety of astronomical data archives, with the primary focus on scientifically related data sets in the optical, ultraviolet, and near-infrared parts of the spectrum. MAST provides search tools and retrieval support for the following missions:

Missions				Catalogs & Surveys
HST	ASTRO	ORFEUS	Copernicus	SDSS
FUSE	HUT	BEFS	ROSAT	GSC
IUE	UIT	IMAPS		DSS
EUVE	WUPPE	TUES		VLA-FIRST

Cross-correlation Target Search and/or Mission Search

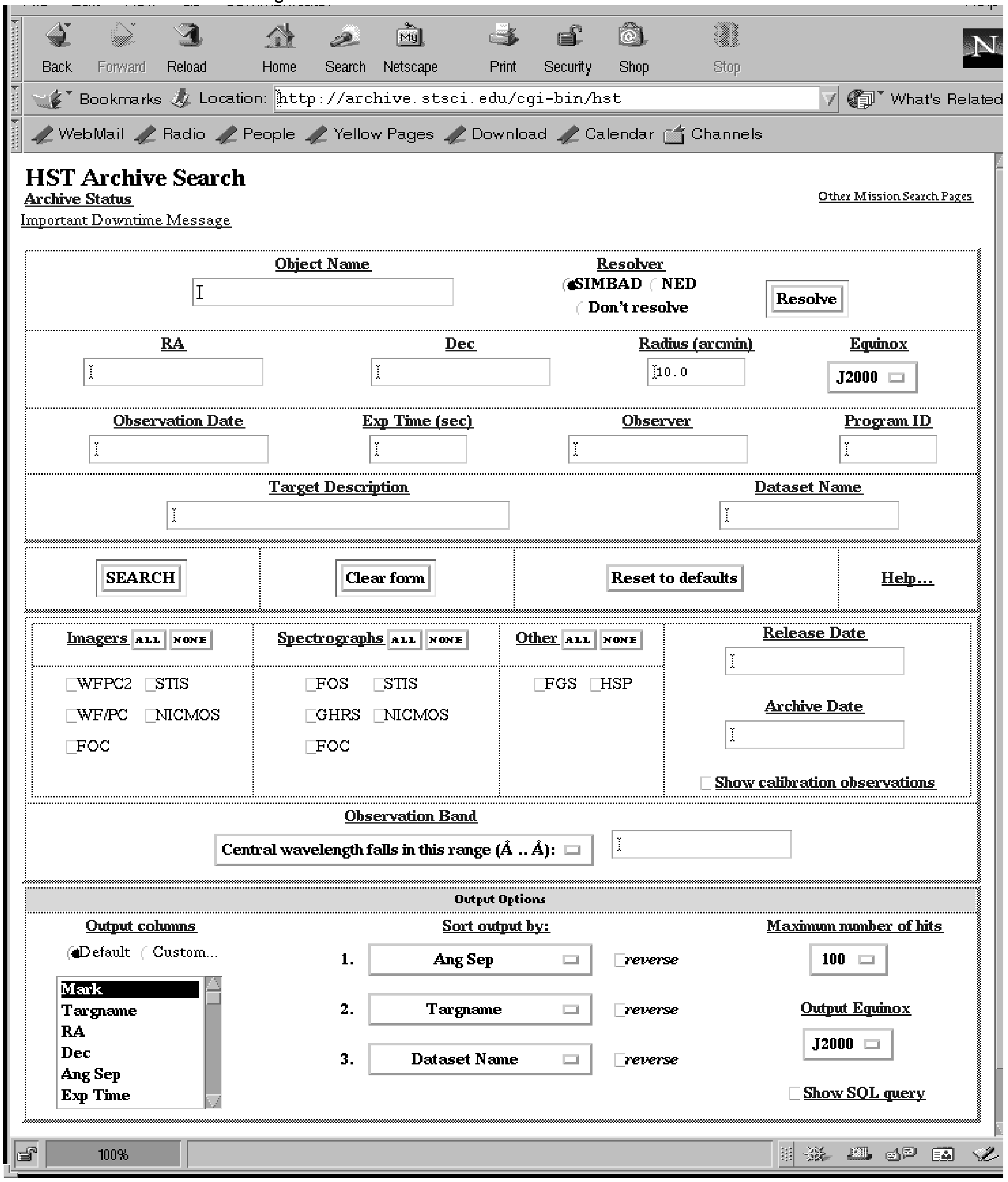
Enter Target name (or Coordinates):

and/or Data Type(s):

	X-Ray	Extreme UV	Far UV	Near UV	Optical	Near IR	Radio
Images	☐	☐	☐	☐	☐	☐	☐
Spectra	☐	☐	☐	☐	☐	☐	
Other			☐	☐	☐	☐	

Top of Page
Copyright Notice
printer-friendly page
http://stdatu.stsci.edu/index.html
archive@stsci.edu
Modified: Jun 26, 2001 17:11

Figure 1.7: HST Archive Web search Form



http://archive.stsci.edu/cgi-bin/hst

HST Archive Search

Archive Status

Important Downtime Message

Other Mission Search Pages

Object Name

Resolver: [✓ SIMBAD] NED Don't resolve

Resolve

RA | Dec | Radius (arcmin): 10.0 | Equinox: J2000

Observation Date | Exp Time (sec) | Observer | Program ID

Target Description | Dataset Name

SEARCH | Clear form | Reset to defaults | Help...

Imagers ALL NONE: WFPC2, STIS, WF/PC, NICMOS, FOC

Spectrographs ALL NONE: FOS, STIS, GHRS, NICMOS, FOC

Other ALL NONE: FGS, HSP

Release Date

Archive Date

Show calibration observations

Observation Band

Central wavelength falls in this range (Å .. Å):

Output Options

Output columns: [✓ Default] Custom...

Mark
Targname
RA
Dec
Ang Sep
Exp Time

Sort output by:

1. Ang Sep reverse
2. Targname reverse
3. Dataset Name reverse

Maximum number of hits: 100

Output Equinox: J2000

Show SQL query

100%

This command will make GEIS format copies (having extension ".hhh") of all the FITS files in the directory, with the same rootname. Following reduction and analysis of the GEIS files with the IRAF/STSDAS tasks, they may be written back into FITS format, on hard disk or to a tape or other storage media, with the **stwfits** task.

CHAPTER 2:

HST File Formats

In this chapter. . .

STScI automatically processes and calibrates all the data received from HST. The suite of software programs that performs this processing—part of a system known as OPUS—is frequently called the *pipeline*, and its purpose is to provide data to observers and to the HST Data Archive in a form suitable for most scientific analyses. Pipeline processing assembles data received from HST into *datasets*, calibrates the data according to standard procedures described in the instrument sections of this handbook, and stores both calibrated and uncalibrated datasets in the Archive.

Pipelines of older instruments (FOC, FOS, FGS, GHRS, HSP, WF/PC-1, and WFPC2) generate files in the so-called GEIS (stands for Generic Edited Information Set) format. Since GEIS is a machine dependent format, these files are converted to a specific kind of FITS file format, sometimes referred as “waiver” FITS, before being archived. We'll explain the structure of this “waiver” FITS format later in this chapter. Since the “waiver” FITS format is only designed for archival purpose, it is necessary to convert it back to the GEIS format before further data processing and analysis using IRAF/STSDAS tasks.

Instruments installed after the 1997 servicing mission (STIS, NICMOS, ACS, and most likely all future instruments) have pipelines which generate FITS files directly. They are ready to be used by relevant IRAF/STSDAS tasks and, unlike the “waiver” FITS files, do NOT need to (and indeed, should not) be converted to GEIS format. Sometimes FITS files for the newer instruments are referred to as “FITS with extension” or “extended”

FITS files. But this can be misleading, since a “waiver” FITS file also has one (ASCII table) extension.

Much confusion has occurred about the two kinds of FITS files been archived at STScI. So we like to repeat one more time:

Older instruments (FOC, FOS, FGS, GHRS, HSP, WF/PC-1, and WFPC2) generate files in GEIS formats, but are stored and delivered as “waiver” FITS format in the archive, and need to be converted back to GEIS format before processing. Newer instruments (STIS, NICMOS, ACS) generate and store files in FITS format and should not be converted to GEIS.

This chapter describes these two HST file formats, first giving some historical perspective on the reasons why they were selected, then explaining the FITS and GEIS formats in more detail. STIS,ACS, and NICMOS observers should pay particular attention to the section on FITS files, which shows how to identify and access the contents of these files and covers some important conventions regarding header keywords. Veteran observers with the other instruments will find little new in the section on GEIS files, but newcomers to the older HST instruments should consult the material on data groups and conversion from FITS to GEIS before proceeding to chapter 3 of the HST Introduction.

2.1 Historical Perspective

In the early 1980’s, when GEIS was selected as the standard format for HST data files, it held several advantages over both FITS and the original IRAF format (OIF):

- GEIS allows floating-point data. The early incarnations of FITS accommodated only integer data, and this restriction to integers would have made data reduction and storage of calibrated data rather cumbersome.
- GEIS files can hold multiple images, each with associated parameters. This feature allowed the packaging of images from the four WF/PC-1 chips into a single unit, as well as the packaging of multiple FOS or GHRS readouts into single files. OIF files and early FITS files could contain only single images.

- GEIS data are stored in two parts, an ASCII header and a binary data file. The separation of these two pieces and the restriction of the header to ASCII made these headers easier to read and print in the days when computers were less powerful and tasks for reading header information were less numerous. OIF headers combine ASCII and binary information, and FITS headers come packaged with the data in a single file.

GEIS was also the standard format for archiving and distribution of HST data until September 1994, when the Space Telescope Data Archive and Distribution Service (ST-DADS) came online. This new system stores and distributes HST data files in machine-independent FITS format, but observers with FOC, FOS, FGS, GHRS, HSP, WF/PC-1, and WFPC2 still must convert their files to machine-dependent GEIS format as described in Section 2.3.1 before using IRAF/STSDAS software (see chapter 3 in the HST Introduction) to reduce their data.

Since the selection of GEIS as HST's standard data format, FITS has added features that have dramatically increased its flexibility. In particular, FITS files can now contain multiple *image extensions*, each with its own header, size, and datatype, that allow multiple exposures to be packaged into the same file, along with associated error and data quality information. The FITS image kernel in IRAF version 2.11, released in August 1997, enables users to access FITS image extensions in ways similar to how they would access GEIS data groups.

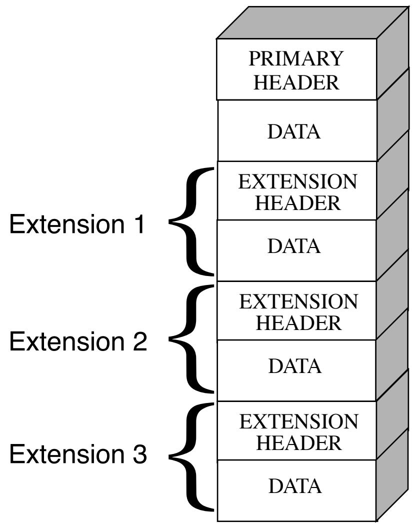
Because of these advantages, FITS was chosen as the standard reduction and analysis format for STIS and NICMOS. The STSDAS tasks written for these instruments expect FITS files as input and produce FITS files as output. *You cannot convert STIS and NICMOS files to GEIS format*. Observers using these instruments should therefore read the following section, which explains how to work with these new FITS files.

2.2 FITS File Format

Flexible Image Transport System (FITS) is a standard format for exchanging astronomical data between institutions, independent of the hardware platform and software environment. A data 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 *header keywords* that describe the data in a particular HDU and the data component immediately follows the header.

The first header in a FITS file is known as the *primary header*, and any number of *extensions* can follow the primary HDU. The data unit following the primary header must contain either an image or no data at all, but each extension can contain one of several different data types, including images, binary tables, and ASCII text tables. The value of the XTENSION keyword in the extension's header identifies the type of data the extension contains. Figure 2.1 schematically illustrates the structure of a FITS file and its extensions.

Figure 2.1: FITS File Structure



The three-letter identifier (e.g., d0h) that follows the rootname of an HST data file (see appendix B 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."

2.2.1 Working with FITS Image Extensions

The FITS image kernel included in IRAF version 2.11 is designed to read and write the images in FITS extensions and their associated headers. Once IRAF has ingested a FITS image and its header, it treats the header-data pair like any other IRAF image. 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 2.11 or higher. It covers how to:

- List a FITS file's extensions.
- Access data in particular FITS extensions.

- Inherit keywords from the primary header.
- Append new extensions to existing FITS files.

Retaining the .fits 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.

If you want to work with STIS and NICMOS data, you will need to upgrade to IRAF 2.11 or higher and STSDAS 2.0.

Generating a FITS File Listing

Once you have downloaded STIS,ACS, or NICMOS FITS files from the Archive, 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. The following example, in Table 2.1, illustrates the first 11 lines generated by **catfits** from a NICMOS MULTIACCUM FITS file containing images only.

The first column of a **catfits** listing gives the extension numbers. Note that the primary HDU is labeled extension number zero. The second column lists the extension type, given by the keyword XTENSION (IMAGE = image, BINTABLE = binary table, TABLE = ASCII table). The third column lists the extension name, given by the keyword EXTNAME. In STIS, ACS, and NICMOS image files, the EXTNAME values SCI, ERR, and DQ indicate science, error, and data quality images, respectively. NICMOS image files contain samples and exposure time images as well, with EXTNAME values SAMP and TIME.

Each STIS or NICMOS readout generates an image set or *imset*. STIS and ACS imsets comprise three images (SCI, ERR, DQ), while NICMOS imsets comprise five (SCI, ERR, DQ, SAMP, TIME). All images belonging to the same imset share the same integer value of the EXTVER keyword, given in the fourth column of a **catfits** listing. Several STSDAS tasks can work with entire imsets (see Section 3.3.3), but most operate on individual images. See the Data Structure chapters of STIS, ACS, and NICMOS Data Handbooks for more information on the contents of imsets.

Table 2.1: 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*][*keyword options*][*image section*]

Note that all the bracketed information is optional. However, the only time it is *valid* to provide only a file name without further specification is when the file is a simple FITS file that contains a single image in the primary HDU.

Designation of the extension number is the most basic method of access, but it is not necessarily the most helpful. 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 on the previous page, you can specify it in two equivalent ways:

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

Designations giving an EXTNAME without an EXTVER refer 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 adheres to 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 extract a 100 by 200 pixel subsection of the same science image in

```
fitsfile.fits.
```

Header Keywords and Inheritance

STIS, ACS, and NICMOS data files use an IRAF image kernel convention regarding the relationship of the primary header keywords to image extensions in the same file. In particular, IRAF allows image extensions to *inherit* keywords from the primary header under certain circumstances. 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. The FITS standard does not cover or imply keyword inheritance, and while the idea itself is simple, its consequences are often complex and sometimes surprising to users.

Generally 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 keyword inheritance by using the NOINHERIT keyword in the file specification. For example:

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

Both of the preceding commands will create an output file whose header contains only those keywords that were present in the original extension header. Note that in the second command, the `noinherit` specification is bracketed with the EXTNAME and EXTVER keywords and not in a separate bracket of its own, as in the first command where an absolute extension number is used. 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 STIS, ACS, or NICMOS data:

If the specified output file does not yet exist, a new output file is created containing only a primary HDU if no specification is appended to the output file name. For example, to copy the contents of the primary header of `fitsfile.fits` into the primary HDU of the FITS file `outfile.fits`, type the command:

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

If the specified output file already exists and you want to append a new extension to it, you need to include the APPEND option in the output file specification. The following command appends extension `[sci,2]` of `fitsfile.fits` onto the existing file `outfile.fits`, while retaining the original EXTNAME and EXTVER of the extension—the `noinherit` specification inhibits 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]
```

If you want to change the EXTNAME or EXTVER of the appended extension, you can specify the new values of these keywords in the output extension, like this:

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

For obvious reasons, it is not generally 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

STIS and NICMOS use FITS tables in two basic ways. Both instruments produce association tables (see appendix B.3) listing the exposures that go into constructing a given association product. In addition, STIS provides certain spectra, calibration reference files, and time-tagged data in tabular form . Here we describe:

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

This discussion assumes you are using STSDAS 2.0 or later. (The IRAF FITS kernel deals only with FITS images. The **tables** package installed with 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, and 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 cannot reside in the primary HDU of a FITS file. They must reside instead in a FITS table extension, in either ASCII form (XTENSION=TABLE) or binary form (XTENSION=BINTABLE).
- If the first extension in a FITS file is a TABLE or a BINTABLE, you can access it by typing the file name with no extension specified. It is not sufficient for the table to be just the first BINTABLE or TABLE; *it must actually be the first extension.*

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 (i.e., **tedit**) can modify an existing FITS extension, and tasks that create a new table (i.e., **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, your 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 spectra take advantage of this feature. Each column of a STIS spectral table holds data values corresponding to a particular physical attribute, such as wavelength, net flux, or background flux. 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 the physical attribute and spectral order of the cell.

In order to analyze tabular spectral data with STSDAS tasks other than the **sgraph** task and the **igi** package, which have been appropriately modified, you will need to extract the desired arrays from the three-dimensional table. Two new 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. 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). If you need a list of the column names, you can run the **tlcol** task (type `tlcol infile.fits`).

Rows are selected according to a *filter*. The filter is evaluated at each table row, and the row is selected if the filter is true. 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 the FITS file, `infile.fits`, specifically the data from the columns labelled 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 of SCI and EXTVER of 2. The data will come from the column FLUX and be restricted to the row numbers 20–30, inclusive. Eventually, all STSDAS and TABLES tasks will be able to use row and column selection. For a complete explanation of the table selector syntax, `type help selectors`.

2.3 GEIS File Format

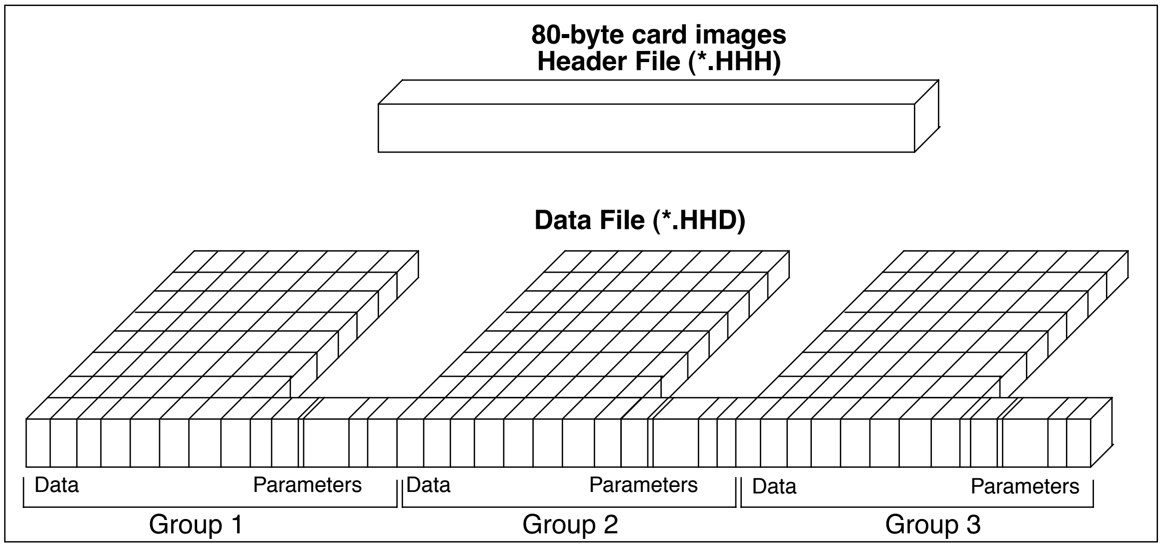
The HST-specific Generic Edited Information Set (GEIS) format[1] is the standard format for reducing data from FOC, FOS, FGS, GHRS, HSP, WF/PC-1, and WFPC2. All HST images in GEIS format consist of two components: a *header file* and a separate *binary data file*, both of which should reside in the same directory. GEIS header files, whose suffixes end in "h" (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, whose suffixes end in "d" (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). The sizes and datatypes of the data arrays and group parameters in

1. GEIS files are also commonly referred to as STSDAS images.

each group of a GEIS file are identical. Figure 2.2 depicts the structure of a GEIS data file graphically.

The binary content of GEIS files is machine dependent. Copying GEIS files directly from one platform to another (e.g., from a VAX to a Sun) may result in unreadable data.

Figure 2.2: GEIS File Structure



2.3.1 Converting FITS to GEIS

The STScI archive stores and distributes datasets from FOC, FOS, FGS, GHRS, HSP, WF/PC-1, and WFPC2 in a special archival FITS format. *We highly recommend that users convert these datasets back into their native GEIS format before working with them.* Your data must be in GEIS format for you to use many of 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 archival FITS format to the GEIS format because the data-processing pipeline employs a special convention for mapping GEIS files to 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 multigroup 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
```

2.3.2 GEIS Data Groups

One of the original advantages of GEIS format noted in Section 2.1 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, the FOS and GHRS obtain data in a time-resolved fashion so that a single FOS or GHRS dataset comprises many spectra—one corresponding to each readout. The data corresponding to each sub-image (for the WF/PC-1 or WFPC2) or each sub-integration (for the FOS or 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 2.2 lists the *contents* and the number of groups in the final calibrated image for the most commonly-used modes of each instrument which uses the GEIS data format.

Table 2.2: Groups in Calibrated Images, by Instrument and Mode

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.

Instrument	Mode	Number of Groups	Description
HSP	All	1	HSP datasets always have only a single group that represents either digital star (`.d0h`, `.c0h`), digital sky (`.d1h`, `.c1h`), analog star (`.d2h`, `.c2h`), or analog sky (`.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 header file itself and the parameter block associated with the group. Because GEIS header files are composed solely of ASCII text, they are easy to print using standard Unix or VMS text-handling facilities. However, the group parameters are stored in the binary data file. To access them you need to use a task such as **imheader**, as shown in section“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 and VMS 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, described in more detail in the Calibration chapters for each instrument’s data handbook, is useful for updating header keywords containing calibration switches and calibration reference files.

Always edit headers using tasks like hedit, eheader, and 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 (see chapter 3 of the HST Introduction). Most of these tasks operate on only one image at a time, so you usually need to specify which group of a GEIS file is to be processed. If you do not specify a group, your task will choose 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 (always refer to a GEIS file by its header file name, i.e. *.??h, even though mathematically you are operating on the data portion):

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

This command 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 multigroup GEIS file results in an output file containing a single group.

Specifying an Image Section

If you wish to process only a portion 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 this data 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.

Printing Header Information

As discussed in the previous section, the task **imheader** extracts and prints information about the GEIS image. This task reports 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 multigroup GEIS file which contains 64 groups of images, each consisting of a spectral 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
```

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

Other Group-Related Tasks

Currently, IRAF or STSDAS tasks cannot process all the groups in an input image and write the results to corresponding groups in an output image. However, there are several STSDAS tasks, particularly in the **toolbox.imgtools** and **hst_calib.ctools** packages, that simplify working with group format data. Please refer to chapter 3 and the *STSDAS User's Guide* for more details about working with GEIS images.

2.3.4 The "waiver" FITS format

Although “waiver” is not quite the accurate or good word for the intended purpose, for historic reasons it has stuck and will be reluctantly adopted. However, in the past, a grammatically incorrect word “waivered” had been used.

The “waiver” FITS format was developed when the HST archive needed a format to store and distribute the data products in a machine-independent medium for the community, at a time before FITS image extension was standardized. As a result, the “waiver” FITS format was adopted as a compromise.

Since, at the time, FITS could only have a single image while the HST data (in GEIS format) may have several images as multiple groups in one file, the idea is to stack the images of different groups together as a new dimension in the FITS image. As for group parameters, they are put in an ASCII table and the table becomes the first (and only) extension of the FITS file.

For example, the WFPC2 pipeline generates the science data as a GEIS file of 4 groups, each is an 800x800 image corresponding to one of the 4 detectors. When this GEIS file is converted to the “waiver” FITS file, the FITS file has an image of 800x800x4 (a three-dimensional image!) at its primary HDU. Similarly, an FOS GEIS file may have 40 groups, each group is a 1-D image (spectrum) of the size 2064. The waiver FITS file then will have one 2-D image of the size 2064x40, at its primary HDU. In the case of WFPC2, the first extension of the waiver FITS file will be an ASCII table containing 4 rows; each row corresponds to a group. The value of each group parameter is under a column named after the group parameter, i. e. the value of the group parameter CRVAL1 of the 2nd group will be at the 2nd row, under the column named “CRVAL1”. In other words, the ASCII table has as many rows as there are groups in the original GEIS file, and as many columns as group parameters.

Although, *in theory, certain* IRAF/STSDAS tasks can directly access the data in the “waiver” FITS file, e.g. to display the 2nd “group” of a WFPC2 image:

```
st.> display u67m0206r_c0f.fits[0][*,*,2]
```

will work, while *most tasks, especially those specific to HST instruments, can not*. It is therefore HIGHLY recommended that all waiver FITS files are converted back to the GEIS format, by using the task **strfits**, before further processing and analysis with IRAF/STSDAS tasks.

CHAPTER 3: STSDAS Basics

In this chapter. . .

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

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

3.1 Navigating STSDAS

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

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

3.1.1 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** pack. Figure 3.1 shows the STSDAS package structure. Note that IRAF version 2.11 must be installed on your system in order for you to use STSDAS 2.0 and TABLES version 2.0 or higher

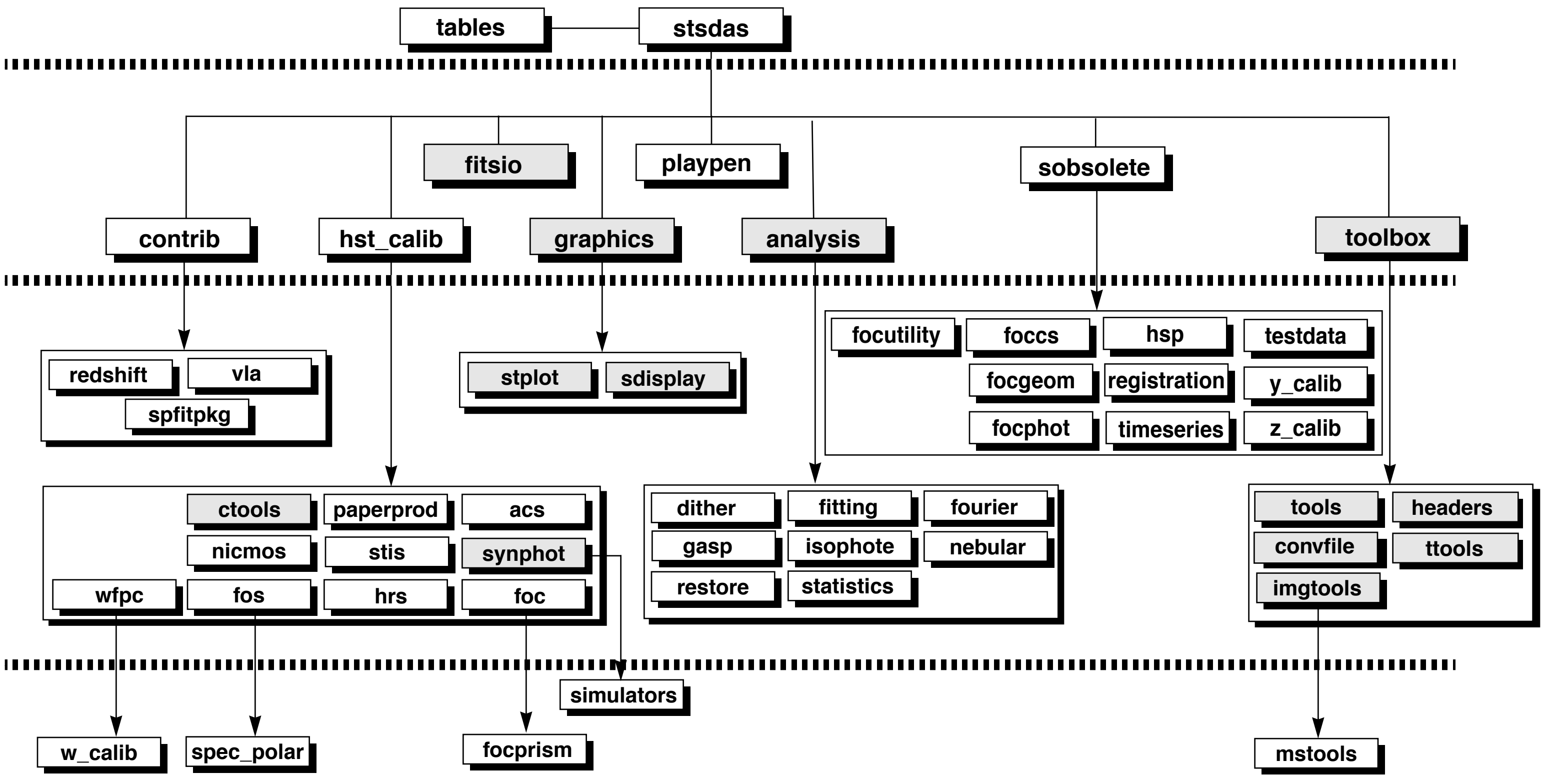
3.1.2 Packages of General Interest

Images

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

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

Figure 3.1: STSDAS Version 2.3 Package Structure



▨ - Implicitly Loaded

- **stsdas.toolbox.imgtools.mstools**: This package contains tools for working with FITS image extensions, in particular STIS and NICMOS image sets (imsets).
- **stsdas.analysis**: This package contains general tasks for image analysis, such as Fourier analysis and dither.

Tables

Several of the analysis packages in STSDAS, including calibration pipeline tasks, create output files in STSDAS table format, which is a binary row-column format, or in FITS binary table format. (ASCII-format tables are also supported, for input only.) The *STSDAS User's Guide* describes the STSDAS table format in detail. Tasks in the **ttools** package or in the external **tables** package can be used to read, edit, create, and manipulate tables. For example:

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

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

3.2 Displaying HST Images

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

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

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

3.2.1 The Display Task

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

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

```
saoimage &
```

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

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

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.ed u/RD/ds9/. Ximtool may be retrieved from ftp://iraf.noao.edu/iraf/x11iraf/.

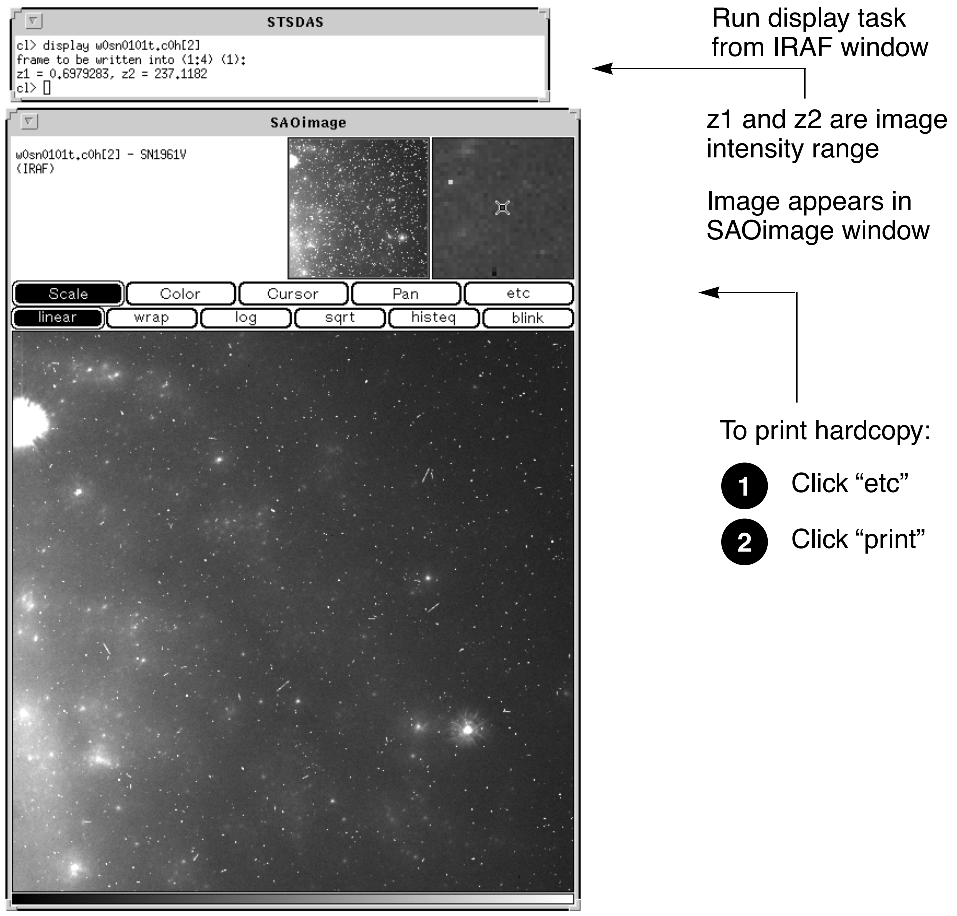
3. Display the image with the IRAF **display** task, using the syntax appropriate for the file format (Chapter 2 explains how to specify GEIS groups and FITS extensions):

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

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

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

Figure 3.2: Displaying an Image



```
cl> display w0sn0101t.c0h[2]
frame to be written into (1:4) (1):
z1 = 0.6979283, z2 = 237.1182
cl> 
```

```
w0sn0101t.c0h[2] - SN1961V
(IRAF)
```

Modifying the Display

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

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

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

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

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

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

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

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

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

Otherwise, the user can simply do:

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

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

The image display buffer can also be adjusted in IRAF by setting the stdimage parameter. For example,

```
im> set stdimage = imt 2048
```

will allow a larger image to be displayed without losing the borders.

3.2.2 Working with Image Sections

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

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

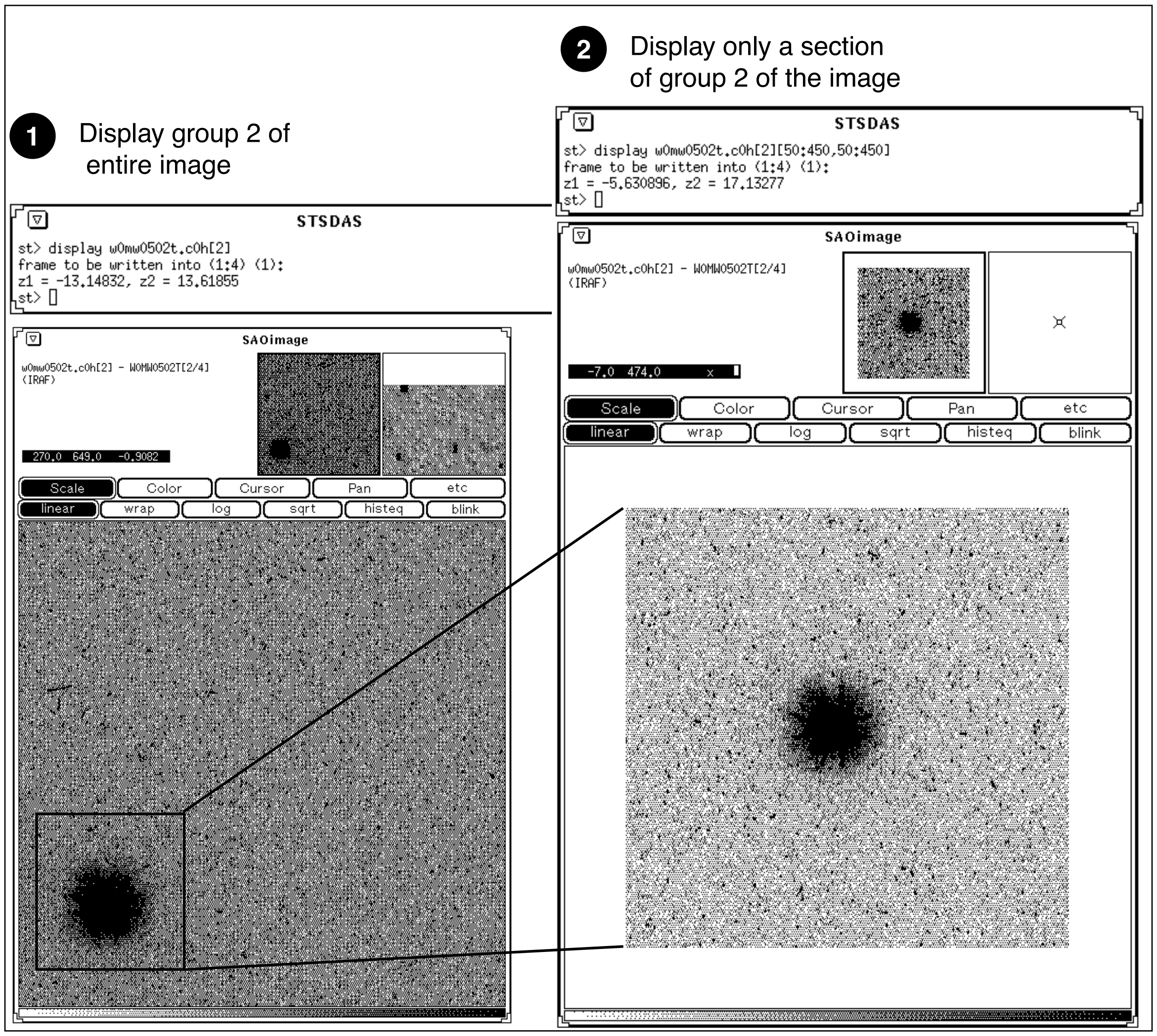
To specify the same pixel range in the second SCI extension of a NICMOS FITS image:

```
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, which also comes before the image section

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

Figure 3.3: Displaying Sections and Groups of an Image


1 Display group 2 of entire image
2 Display only a section of group 2 of the image
STSDAS
SAOimage
Scale
Color
Cursor
Pan
etc
linear
wrap
log
sqrt
histeq
blink

```
st> display w0mw0502t.c0h[2]
frame to be written into (1:4) (1):
z1 = -13.14832, z2 = 13.61855
st> ▯
```

```
w0mw0502t.c0h[2] - W0MW0502T[2/4]
(IRAF)
270.0  649.0  -0.9082
```

```
st> display w0mw0502t.c0h[2][50:450,50:450]
frame to be written into (1:4) (1):
z1 = -5.630896, z2 = 17.13277
st> ▯
```

```
w0mw0502t.c0h[2] - W0MW0502T[2/4]
(IRAF)
-7.0  474.0      x
```

3.3 Analyzing HST Images

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

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

3.3.1 Basic Astrometry

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

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

Positional Information

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

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

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

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

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

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

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

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

Table 3.1: Additional IRAF and STSDAS Astrometry Tasks

Task	Purpose
compass	Plot north and east arrows on an image.
north	Display the orientation of an image based on keywords.
rimcursor	Determine RA and Dec of a pixel in an image.
wcscoords	Use WCS[1] to convert between IRAF coordinate systems.
wcslab	Produce sky projection grids for images.

1. World Coordinate System (WCS). Type "help specwcs" at the IRAF prompt for details.

Improving Astrometric Accuracy

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

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

3.3.2 Examining and Manipulating Image Data

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

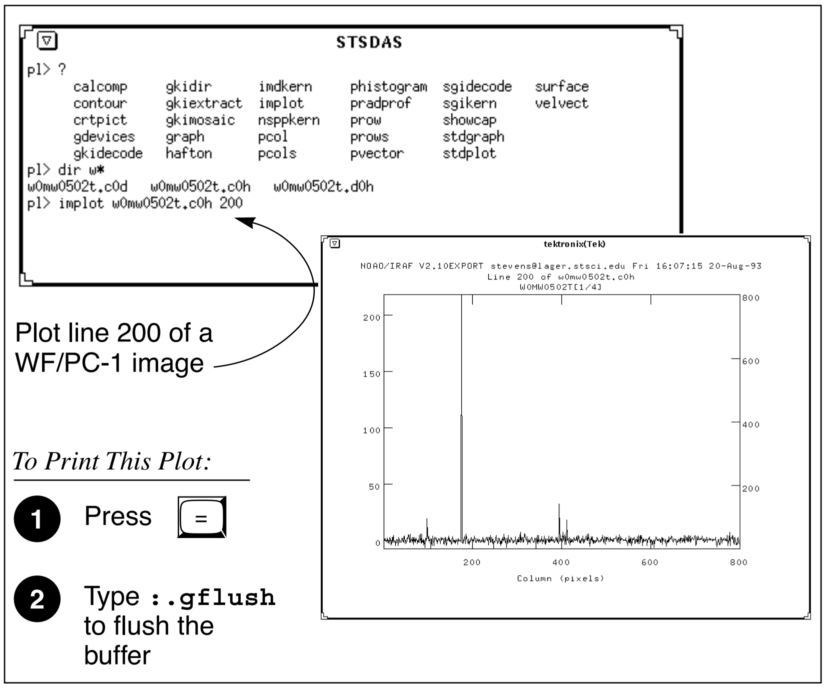
implot

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

Table 3.2: Basic implot Commands

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

Figure 3.4: Plotting Image Data with implot


STSDAS
tektronix(Tek)
NOAO/IRAF V2.10EXPORT stevens@lager.stsci.edu Fri 16:07:15 20-Aug-93
Line 200 of w0mw0502t.c0h
W0MW0502T[1/4]
Column (pixels)
=

```
pl> ?
      calcomp     gkidir      imdkern     phistogram  sgidecode   surface
      contour     gkiextract  implot      pradprof    sgikern     velvect
      crtpict     gkimosaic   nsppkern    prow        showcap
      gdevices    graph       pcol        prows       stdgraph
      gkidecode   hafton      pcols       pvector     stdplot
pl> dir w*
w0mw0502t.c0d   w0mw0502t.c0h   w0mw0502t.d0h
pl> implot w0mw0502t.c0h 200
```

Plot line 200 of a WF/PC-1 image

To Print This Plot:

1. Press =
2. Type **`:.gflush`** to flush the buffer

imexamine

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

Table 3.3: Image Manipulation Tasks

Task	Package	Purpose
boxcar	images.imfilter	Boxcar smooth a list of images
gcombine	stsdas.toolbox.imgtools	Combine images using various algorithms and rejection schemes
gcopy	stsdas.toolbox.imgtools	Copy GEIS multigroup images
geomap	images.immatch	Compute a coordinate transformation
geotran	images.immatch	Resample an image based on geomap output
grlist	stsdas.graphics.stplot	List of file names of all groups of a GEIS image (to make @lists)
gstatistics	stsdas.toolbox.imgtools	Compute image statistics[1]
imcalc	stsdas.toolbox.imgtools	Perform general arithmetic on GEIS images[a]
imedit	images.tv	Fill in regions of an image by interpolation
imexamine	images.tv	Examine images using display, plots, and text (see "imexamine" on page 3-13)
implot	plot	Plot lines and columns of images (see "implot" on page 3-12)
magnify	images.imgeom	Magnify an image
msarith	stsdas.toolbox.mstools	Performs basic arithmetic on STIS and NICMOS imsets
mscombine	stsdas.toolbox.mstools	Extension of **gcombine** for STIS and NICMOS imsets
msstatistics	stsdas.toolbox.mstools	Extension of **gstatistics** for STIS and NICMOS imsets
newcont	stsdas.graphics.stplot	Draw contours of two-dimensional data
pixcoord	stsdas.hst_calib.wfpc	Compute pixel coordinates of stars in a GEIS image
plcreate	xray.ximages	Create a pixel list from a region file (e.g., from SAOimage)
rotate	images.imgeom	Rotate an image
saodump	stsdas.graphics.sdisplay	Make image and colormap files from SAOimage display
siaper	stsdas.graphics.stplot	Plot science instrument apertures of HST

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

3.3.3 Working with STIS, ACS, and NICMOS Imsets

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

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 appear with an associated error, which will propagate into the error array(s) of the output file. Table 3.4 below shows how this task operates on the SCI, ERR, and DQ images in a STIS, ACS, or NICMOS imset, as well as the additional TIME and SAMP images belonging to NICMOS imsets:

Table 3.4: Task msarith Operations

Operation	Operand2	SCI	ERR	DQ	TIME	SAMP
ADD	file	op1+op2	$\sqrt{\sigma 1^2 + \sigma 2^2}$	OR	T1+T2	S1+S2
SUB	file	op1-op2	$\sqrt{\sigma 1^2 + \sigma 2^2}$	OR	T1	S1
MULT	file	op1*op2	$(\mathrm{op1} \times \mathrm{op2})\sqrt{(\sigma 1/\mathrm{op1})^2 + (\sigma 2/\mathrm{op2})^2}$	OR	T1	S1
DIV	file	op1/op2	$(\mathrm{op1}/\mathrm{op2})\sqrt{(\sigma 1/\mathrm{op1})^2 + (\sigma 2/\mathrm{op2})^2}$	OR	T1	S1
ADD	constant	op1+op2	$\sqrt{\sigma 1^2 + \sigma 2^2}$	...	...	...
SUB	constant	op1-op2	$\sqrt{\sigma 1^2 + \sigma 2^2}$	...	...	...
MULT	constant	op1*op2	$(\mathrm{op1} \times \mathrm{op2})\sqrt{(\sigma 1/\mathrm{op1})^2 + (\sigma 2/\mathrm{op2})^2}$	...	T1*op2	...
DIV	constant	op1/op2	$(\mathrm{op1}/\mathrm{op2})\sqrt{(\sigma 1/\mathrm{op1})^2 + (\sigma 2/\mathrm{op2})^2}$	...	T1*op2	...

In Table 3.4, the first operand (op1) is always a file, and the second operand (op2) can be either a constant or a file. The ERR arrays of the input files ($\sigma 1$ and $\sigma 2$) are added in quadrature. If the constant is given with an error ($\sigma 2$), the latter is added in quadrature to the input ERR array. Note that in Table 3.4 the pixels in the SCI images are in counts, but **msarith** can also operate on count rates.

mscombine

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

Once **gcombine** has finished, **mscombine** then reassembles the individual output images into imsets and outputs them as one STIS, ACS, or NICMOS data file. The output images and error maps from **gcombine** form the SCI and ERR extensions of the output imset. The DQ extension will be a combination of the masking operations and the rejection algorithms executed in **gcombine**. For NICMOS, the TIME extension will be the sum of the TIME values from the input files minus the rejected values, divided on a pixel-by-pixel basis by the number of valid pixels in the output image. The final TIME array will be consistent with the output SCI image (average or median of the science data). The SAMP extension for NICMOS is built from all the input SAMP values, minus the values discarded by masking or rejection.

msstatistics

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

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

and:

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

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

mssplit and msjoin

The **mssplit** task extracts user-specified imsets from a STIS, ACS, or NICMOS data file and copies them into separate files. Each output file contains a single imset along with the primary header of the original file. You might find this task useful for reducing the size of a STIS, ACS, or

NICMOS file containing many imsets or for performing analysis on a specific imset. The **msjoin** task inverts the operation of **mssplit**: it assembles separate imsets into a single data file.

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

3.3.4 Photometry

Included in this section are:

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

IRAF and STSDAS Photometry Tasks

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

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

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

http://iraf.noao.edu/docs/photom.html

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

Converting Counts to Flux or Magnitude

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

For all instruments other than NICMOS, PHOTFLAM is defined to be 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. If the F_λ spectrum of your source is significantly sloped across the bandpass or contains prominent features, such as strong emission lines, you may wish to recalculate the inverse sensitivity using **synphot**, described below. WF/PC-1 observers should note that the PHOTFLAM value calculated during pipeline processing does not include a correction for temporal variations in throughput owing to contamination buildup. Likewise, FOC observers should note that PHOTFLAM values determined by the pipeline before May 18, 1994 do not account for sensitivity differences in formats other than 512 x 512.

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

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

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

2. Except for 2-D rectified STIS images, which are in units of I_λ.

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

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

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

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

synphot

The STSDAS synthetic photometry package, called **synphot**, can simulate HST observations of astronomical targets with known spectra. It contains throughput curves of all HST optical components, such as mirrors, filters, gratings, apertures, and detectors, and can generate passband shapes for any combination of these elements. It can also generate synthetic spectra of many different types, including stellar, blackbody, power-law and H II region spectra, and can convolve these spectra with the throughputs of HST's instruments. You can therefore use it to compare results in many different bands, to cross-calibrate one instrument with another, or to relate your observations to theoretical models.

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

```
sy> calcphot foc,f/96,x96zlrg,f501n ‘unit(1,flam)’ counts
```

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

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

3.4 Displaying HST Spectra

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

3.4.1 FOS and GHRS Spectra

Before you work with FOS and GHRS data within STSDAS, you will want to convert the FITS files you received from the Archive into GEIS format (see 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 subintegration. FOS readouts taken in ACCUM mode are cumulative, so the last group contains the results of the entire integration. In contrast, GHRS readouts and FOS readouts in RAPID mode are independent. If you want to see the results of an entire GHRS FP-SPLIT integration, you will need to align and coadd the spectra in the groups of the GHRS file. You can also combine all the groups in an FOS or GHRS data file, without wavelength alignment, using the **rcombine** task in the **hst_calib.ctools** package. See online help for details.

The STSDAS task **sgraph** (in the **graphics.stplot** package) can plot the contents of a single GEIS group. For example, if you want to see group 19 of the calibrated FOS spectrum with rootname `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 a separate parameter, rather than enclosed in the standard square brackets.

3.4.2 STIS Spectra

STIS data files retrieved from the Archive can contain spectra in two different forms: as long-slit spectral images in FITS IMAGE extensions or as extracted echelle spectra in FITS BINTABLE extensions.

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

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

Displaying the long-slit spectral image using the **display** task (see Section 3.2.1 in the HST Introduction) 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.

To plot STIS spectra in BINTABLE extensions, you first need to understand how STIS spectra are stored as binary arrays in FITS table cells. Chapter 2 (Section 2.2.2) discusses this format and describes the *selectors* syntax used to specify these data arrays. Each row of a STIS echelle table contains a separate spectral order, and each column contains data of a certain type, such as WAVELENGTH data or FLUX data. To specify a particular array, you must first type the file name, then the extension containing the BINTABLE, then the column selector, then the row selector. For example, to select the WAVELENGTH array corresponding to spectral order 80 of the echelle spectrum in extension 4 of `stis.fits`, you would specify the file as:

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

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

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

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

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

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

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

3.4.3 Producing Hardcopy

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

Direct Hardcopies

To print a quick copy of the displayed plot:

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

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

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

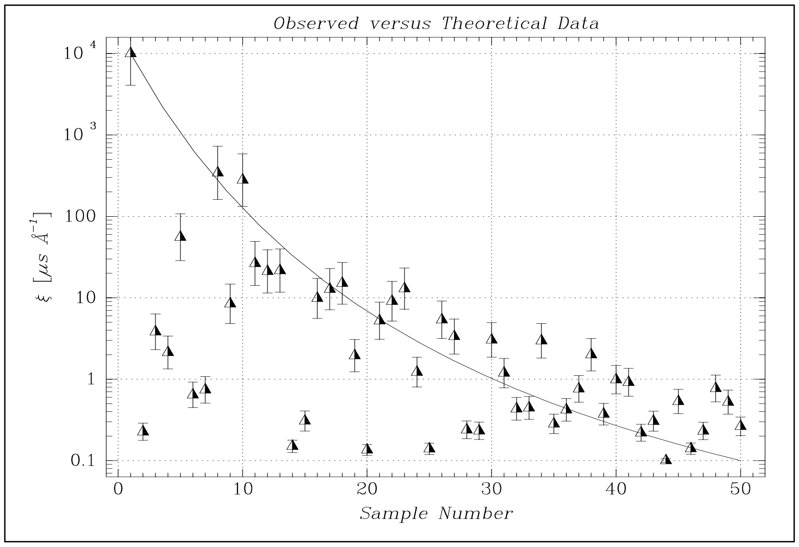
```
st> fwplot y3bl0104t.c1h[19] device=psi_land
st> gflush
/tmp/pskxxxx
```

The above commands would write a plot of flux vs. wavelength in landscape-mode into a temporary PostScript file, named `/tmp/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.



3.5 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 special preparations of data other than STIS two-dimensional spectra. Before discussing these tasks we will first show how to recast your data into forms that are more generally accessible.

3.5.1 Preparing FOS and GHRS Data

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

Several options for combining flux and wavelength information are available:

- **resample**: This simple task resamples your flux data onto a linear wavelength scale, creating a new flux file containing the starting wavelength of the new grid in the CRVAL1 keyword and the wavelength increment per pixel in the CD1_1 keyword. Encoding the wavelength information into these standard FITS header keywords makes this format quite portable, but the resampling process loses some of the original flux information. In addition, the error (`.c2h`) and data quality (`.cqh`) files cannot be similarly resampled, limiting the usefulness of this technique.

- **mkmultispec**: This task writes wavelength information into the header of a flux file while preserving all the original information. It is therefore a better choice than **resample** for most applications, and we describe it in more detail below.
- **imtab**: An alternative to writing wavelength information into the header is to use the **imtab** task to create a table recording the wavelength, flux, and if desired, the error data corresponding to each pixel. Many STSDAS tasks, such as those in the STSDAS **fitting** package, can access data in tabular form, so we describe this approach in more detail as well.

mkmultispec

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

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

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

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

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

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

imtab

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

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

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

Constructing tables is 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, and therefore degrade 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.

3.5.2 Preparing STIS Spectra for Analysis

Calibrated STIS spectra emerge from the pipeline either as two-dimensional images (`_x2d` files) or as one-dimensional spectra in tabular form (`_x1d` files.) You can analyze calibrated two-dimensional STIS spectra in IRAF as you would with any other long-slit spectral image, because their headers already contain the necessary wavelength information. Tabulated STIS spectra can be analyzed directly using STSDAS tasks that understand the *selectors* syntax described in Section 2.2.2. However, to use IRAF tasks, such as **splot**, that rely on the multispec WCS or to use STSDAS tasks that do not understand three-dimensional tables, you will have to prepare your data appropriately. This section describes two useful tasks for putting your data in the proper form:

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

tomultispec

The **tomultispec** task in the **stsdas.hst_calib.ctools** package extracts one or more spectral orders from a STIS table, fits a polynomial dispersion solution to each wavelength array, and stores the spectra in an output file in original IRAF format (OIF), using the multispec WCS. This task is layered upon the **mkmultispec** task, which performs a similar operation for FOS and GHRS calibrated spectra (see “mkmultispec” on page 3-25). Most of the parameters for **tomultispec** echo those for **mkmultispec**. As a helpful navigational aid, the STIS spectral order numbers are written to the corresponding *beam* numbers in the multispec image; the aperture numbers are indexed sequentially starting from one. You can choose to fit the dispersion solution interactively, but the default fourth-order Chebyshev polynomial will likely suffice for all STIS spectral orders, except for prism-dispersed spectra. However, you cannot use the interactive option if you are selecting more than one order from the input file.

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

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

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

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

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

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

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

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

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

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

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

Column selectors cannot be used with tomultispec

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

txtable

Tabulated STIS spectra are stored as data arrays within individual cells of FITS binary tables (see 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 the order. The **txtable** in the **tables.ttools** package extracts these data arrays from the cells specified with the selectors syntax and stores them in the columns of conventional two-dimensional binary tables.

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

```
tt> txtable “data.fits[1][c:WAVELENGTH,FLUX][r: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 in a tabulated echelle spectrum, you would type:

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

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

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

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

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

3.5.3 General Tasks for Spectra

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

Table 3.5: Tasks for Working with Spectra

Task	Package	Input Format	Purpose
boxcar	images.imfilter	Image	Boxcar smooth a list of images
bplot	noao.onedspec	Multispec image	Plot spectra non-interactively
continuum	noao.onedspec	Image	Continuum normalize spectra
fitprofs	noao.onedspec	Image	Non-interactive Gaussian profile fitting to features in spectra and image lines
gcopy	stsdas.toolbox.imgtools	GEIS image	Copy multigroup images
grlist	stsdas.graphics.stplot	GEIS image	List file names for all groups in a GEIS image; used to make lists for tasks that do not use group syntax
grplot	stsdas.graphics.stplot	GEIS image	Plot arbitrary lines from 1-D image; overplots multiple GEIS groups; no error or wavelength information is used
grspec	stsdas.graphics.stplot	Multispec GEIS image	Plot arbitrary lines from 1-D image; stack GEIS groups
magnify	images.imgeom	Image	Interpolate spectrum on finer (or coarser) pixel scale
nfit1d	stsdas.analysis.fitting	Image, table	Interactive 1-D non-linear curve fitting (see Section 3.5.4)
ngaussfit	stsdas.analysis.fitting	Image, table	Interactive 1-D multiple Gaussian fitting (see Section 3.5.4)
poffsets	stsdas.hst_calib.ctools	GEIS image	Determine pixel offsets between shifted spectra
rapidlook	stsdas.hst_calib.ctools	GEIS image	Create and display a 2-D image of stacked 1-D images
rcombine	stsdas.hst_calib.ctools	GEIS image	Combine (sum or average) GEIS groups in a 1-D image with option of propagating errors and data quality values
resample	stsdas.hst_calib.ctools	GEIS image	Resample FOS and GHRS data to a linear wavelength scale (see Section 3.5.1)
sarith	noao.onedspec	Multispec image	Spectrum arithmetic
scombine	noao.onedspec	Multispec image	Combine spectra
sfit	noao.onedspec	Multispec image	Fit spectra with polynomial function
sgraph	stsdas.graphics.stplot	Image, table	Plot spectra and image lines; allows overplotting of error bars and access to wavelength array (see Section 3.4.1)
specalign	stsdas.hst_calib.ctools	GEIS image	Align and combine shifted spectra (see **poffsets**)
specplot	noao.onedspec	Multispec image	Stack and plot multiple spectra
splot	noao.onedspec	Multispec image	Plot and analyze spectra & image lines (see “splot” on page 3-31)

splot

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

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

If you don't specify a group in brackets, **splot** will assume you want the first group. In order to use **splot** to analyze your FOS or GHRS spectrum, you will first need to write the wavelength information from your `.c0h` file to the header of your `.c1h` files in WCS, using the **mkmultispec** task (see "mkmultispec" on page 3-25).

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

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

If you have used **tomultispec** to transform a STIS echelle spectrum into `.imh/.pix` OIF files with WCS wavelength information (see "tomultispec" on page 3-27), you can step through the spectral orders stored in image lines using the ")", "(", and "#" keys. To start with the first entry in your OIF file, type:

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

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

Table 3.6: Useful splot Cursor Commands

Command	Purpose
Manipulating spectra	
f	Arithmetic mode; add and subtract spectra
l	Convert spectrum from f_ν to f_λ (invert transformation with “n”)
n	Convert spectrum from f_λ to f_ν
s	Smooth with a boxcar
u	Define linear wavelength scale using two cursor markings
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

3.5.4 STSDAS Fitting Package

The STSDAS **fitting** package contains several tasks, as listed in Table 3.7, 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 apply only 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 “imtab” on page 3-26).

Table 3.7: 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 rewindow the plot, resticting 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've got a good fit, and then turn the error estimation on for one last iteration.

Figure 3.6 and Figure 3.7 shows 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

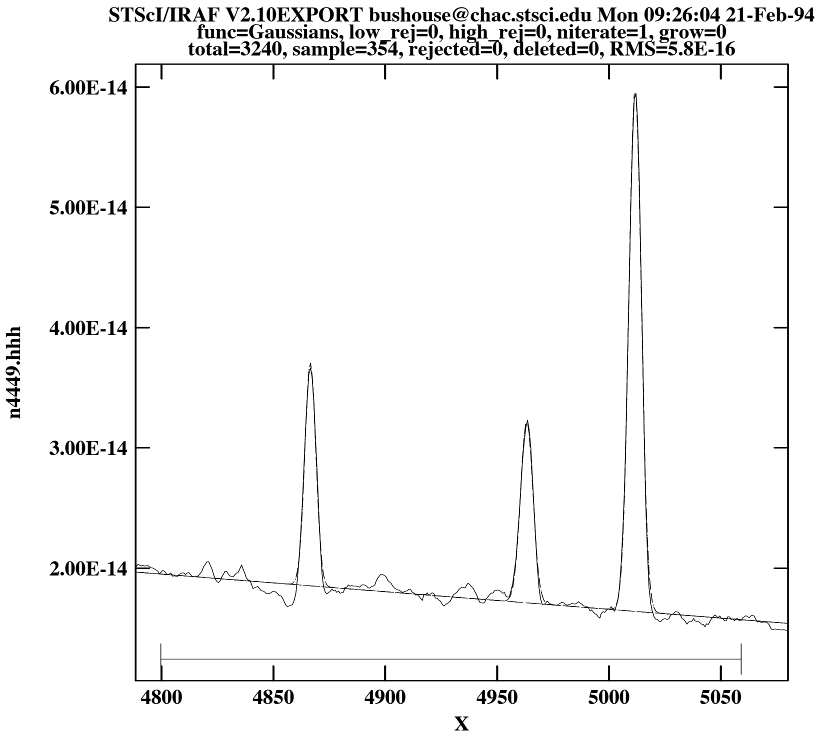


Figure 3.7: Coefficients and Error Estimates

```
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
```

3.5.5 Specfit

The **specfit** task, in the STSDAS **contrib** package, is another powerful interactive facility for fitting a wide variety of emission-line, absorption-line, and continuum models to a spectrum. This task was written by Gerard Kriss. 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 data value. The online help explains a few pieces of additional information that must be included as header lines in an input text file.

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

3.6 References

3.6.1 Available from STScI

(http://stsdas.stsci.edu/STSDAS.html

- *STSDAS Users Guide* , version 1.3, September 1994.
- *STSDAS Installation and Site Managers Guide*, version 2.3, June 2001.
- *Synphot Users Guide*, December 1998.
- *IGI Reference Manual* , version 1.3, October 1992.

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

3.6.2 Available from NOAO

(http://iraf.noao.edu/docs/docmain.html)

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

3.6.3 Other References Cited in This Chapter

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

PART II:

ACS Data Handbook

This handbook is designed to help users manipulate, process and analyze data from the Advanced Camera for Surveys (ACS) which will be installed on-board the Hubble Space Telescope (HST) during the 2002 servicing mission (SM3B).

Part II:ACS Data Handbook

ACS Introduction

How to Use this Handbook

This handbook is designed to help users manipulate, process and analyze data from the Advanced Camera for Surveys (ACS) which was installed on-board the Hubble Space Telescope (HST) during the 2002 servicing mission (SM3B). It is presented as an independent and self-contained document and is designed for users familiar with HST data but new to ACS. A detailed discussion of ACS calibration and drizzling techniques are provided in this document. Users who wish to find more general information, including instructions for retrieving HST data from the archive, a description of HST file formats, and a discussion of general IRAF/STSDAS software for displaying and processing these data, are referred to a companion volume, the Introduction to the *HST Data Handbook* at:

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

While the present handbook provides comprehensive information for the treatment of ACS data, it also includes a brief discussion of the capabilities and design of the ACS and describes the basic instrument operations (Chapter 1). For a detailed discussion of the instrument and detectors, the reader is referred to the ACS Instrument Handbook. One important concern for users will be the large volume of ACS data compared to other instruments on-board the HST. This subject is addressed in Chapter 2, with a discussion of the ACS file structure. Chapter 3 gives a detailed description of the ACS calibration pipeline software, the ACS reference files, and instructions for manually recalibrating data. Given the large field of view of the ACS, and hence the importance of geometric distortion, Chapter 4 is devoted to a thorough discussion of the ACS distortion. A new Python package (**PyDrizzle**) has been added to the calibration pipeline for correcting distortion in ACS images and for drizzling multiple dithered images. We also recommend using the new **MultiDrizzle** software for users who wish to improve or fine-tune the drizzling process, choosing parameters specific to their scientific goals.

The present handbook does not address the results of in-flight ACS calibration programs, data analysis techniques, and error sources. This information will comprise two additional chapters to be included in version 3.0 of the Data Handbook, scheduled for release in early 2004.

Since many of the instrument characteristics may be revised over a short time frame, readers are advised to consult the ACS web pages (www.stsci.edu/hst/acs) for the latest information regarding ACS performance and calibration.

Jennifer Mack and Ron Gilliland (Editors, ACS Data Handbook)

CHAPTER 1:

ACS Overview

In this chapter. . .

This chapter provides an overview of the capabilities and design of the Advanced Camera for Surveys (ACS) and describes the basic instrument operations. For more information we refer you to the *ACS Instrument Handbook* which gives a technical description of the instrument's properties, expected performance, operations, and calibration.

1.1 Instrument Design and Capabilities

The ACS camera is designed to provide HST with a deep, wide-field survey capability from the visible to near-IR, high resolution imaging from the near-UV to the near-IR, and solar blind far-UV imaging. The primary design goal of the ACS Wide-Field Channel is to achieve a factor of 10 improvement in discovery efficiency compared to WFPC2, where discovery efficiency is defined as the product of imaging field of view (FOV) and instrument throughput.

ACS comprises three channels, each optimized for a specific goal:

- Wide field channel (WFC): ~202 × 202 arcsecond field of view from 3700–11,000 Å, and peak efficiency of 48% (including the Optical Telescope Assembly). The plate scale of ~0.049 arcsec/pixel provides critical sampling at 11,600 Å.

- High resolution channel (HRC): 29 × 25 arcsecond field of view from 2000–11,000 Å, and peak efficiency of 29%. The plate scale of ~0.027 arcsec/pixel provides critical sampling at 6300 Å.
- Solar Blind Channel (SBC): 35 × 31 arcsecond field of view from 1150-1700 Å, and peak efficiency of 7.5%. The plate scale of ~0.032 arcsec/pixel provides a good compromise between resolution and field of view.

In addition to the these three prime capabilities, ACS also provides:

- Grism spectroscopy: Low resolution (R~100) wide field spectroscopy from 5500–11,000 Å available in both the WFC and the HRC.
- Objective prism spectroscopy: Low resolution (R~100 @ 2000 Å) near-UV spectroscopy from 2000–4000 Å available in the HRC.
- Objective prism spectroscopy: Low resolution (R~100 @ 1216 Å) far-UV spectroscopy from 1150–1700 Å available in the SBC.
- Coronagraphy: Aberrated beam coronagraphy in the HRC from 2000–11,000 Å with 1.8 arcsecond and 3.0 arcsecond diameter occulting spots.
- Imaging Polarimetry: Polarimetric imaging in the HRC and WFC with relative polarization angles of 0°, 60° and 120°.

ACS is a versatile instrument that can be applied to a broad range of scientific programs. For example, the high sensitivity and wide field of the WFC in the visible and near-infrared will make it the instrument of choice for deep imaging programs in this wavelength region. The HRC, with its excellent spatial resolution, provides full sampling of the HST PSF at λ>6000 Å and can be used for high precision photometry in stellar population programs. The HRC coronagraph can be used for the detection of circumstellar disks and QSO host galaxies.

1.1.1 Detectors

ACS uses one or more large-format detectors in each channel:

- The WFC detector, called `ACS/WFC,` employs a mosaic of two 2048 × 4096 Scientific Imaging Technologies (SITe) CCDs, with ~0.049 arcsecond pixels, covering a nominal ~202 × 202 arcsecond field of view, and a spectral response from ~3700 to 11,000 Å.
- The HRC detector, called `ACS/HRC,` is a 1024 × 1024 SITe CCD, with ~0.028 × 0.025 arcsecond pixels, covering a nominal 29 × 25 arcsecond field of view, and spectral response from ~2000 to 11,000 Å.

- The SBC detector, called the `ACS/SBC,` is a solar-blind CsI Multi-Anode Microchannel Array (MAMA), with 1024 × 1024 ~0.034 × 0.030 arcsecond pixels, and a nominal 35 × 31 arcsecond field of view, with far-UV spectral response from ~1150 to 1700 Å.

The WFC & HRC CCDs

The ACS CCDs are thinned, backside-illuminated devices cooled by thermo-electric cooler (TEC) stacks and housed in sealed, evacuated dewars with fused silica windows. The spectral response of the WFC CCDs is optimized for imaging at visible to near-IR wavelengths. The HRC CCD covers wavelengths similar to the WFC but the spectral response has been additionally optimized for the near-UV. Both CCD cameras produce a time-integrated image in the `ACCUM` data-taking mode. The HRC can also be operated in target acquisition (ACQ) mode for coronagraphic observations. As with all CCD detectors, there is noise (*readout noise*) and time (*read time*) associated with reading out the detector following an exposure. The minimum exposure time is 0.1 *sec* for HRC, and 0.5 *sec* for WFC, and the minimum time between successive identical exposures is 45 *sec* (HRC) or ~135 *sec* (WFC) for full-frame and can be reduced to ~36 *sec* for subarray readouts. The dynamic range for a single exposure is ultimately limited by the depth of the CCD full well (~85,000 e^- for the WFC and 155,000 e^- for the HRC), which determines the total amount of charge that can accumulate in any one pixel during an exposure without saturation. Cosmic rays will affect all CCD exposures: CCD observations should be broken into multiple exposures whenever possible, to allow removal of cosmic rays in post-observation data processing.

The SBC MAMA

The SBC MAMA is a *photon-counting* detector which provides a two-dimensional ultraviolet capability, optimized for the far-UV. It can only be operated in `ACCUM` mode. The ACS MAMA detector is subject to both *scientific* and *absolute* brightness limits. At high local (≥50 *counts* sec^{-1} $pixel^{-1}$) and global (>285,000 *counts* sec^{-1}) illumination rates, counting becomes nonlinear in a way that is not correctable. At only slightly higher illumination rates, the MAMA detectors are subject to damage. We have therefore defined absolute local and global count-rate limits, which translate to a set of configuration-dependent bright-object screening limits. Sources which violate the absolute count rate limits in a given configuration *cannot be observed in those configurations*.

1.1.2 ACS Optical Design

The ACS design incorporates two main optical channels: one for the WFC and one which is shared by the HRC and SBC. These channels are illustrated in figures 3.1 and 3.2 of the *ACS Instrument Handbook*. Each channel has independent corrective optics to compensate for HST's spherical aberration. The WFC has three optical elements, coated with silver, to optimize instrument throughput in the visible. The silver coatings cut off at wavelengths shortward of 3700 Å. The WFC has two filter wheels which it shares with the HRC, offering the possibility of internal WFC/HRC parallel observing for some filter combinations. The HRC/SBC optical chain comprises three aluminized mirrors, overcoated with MgF_2. The HRC or SBC channels are selected by means of a plane fold mirror. The HRC is selected by inserting the fold mirror into the optical chain so that the beam is imaged onto the HRC detector through the WFC/HRC filter wheels. The SBC channel is selected by moving the fold mirror out of the beam to yield a two mirror optical chain which images through the SBC filter wheel onto the SBC detector. The aberrated beam coronagraph is accessed by inserting a mechanism into the HRC optical chain. This mechanism positions a substrate with two occulting spots at the aberrated telescope focal plane and an apodizer at the re-imaged exit pupil.

While there is no mechanical reason why the coronagraph could not be used with the SBC, for health and safety reasons (due to brightness limits of the MAMA detector) **use of the coronagraph is forbidden with the SBC**.

1.1.3 ACS Geometric Distortion

The ACS detectors exhibit significantly more distortion than previous HST instruments. All ACS observations must be corrected for distortion before any photometry or astrometry is derived. For a thorough discussion of ACS Geometric Distortion, we refer the reader to Chapter 4.

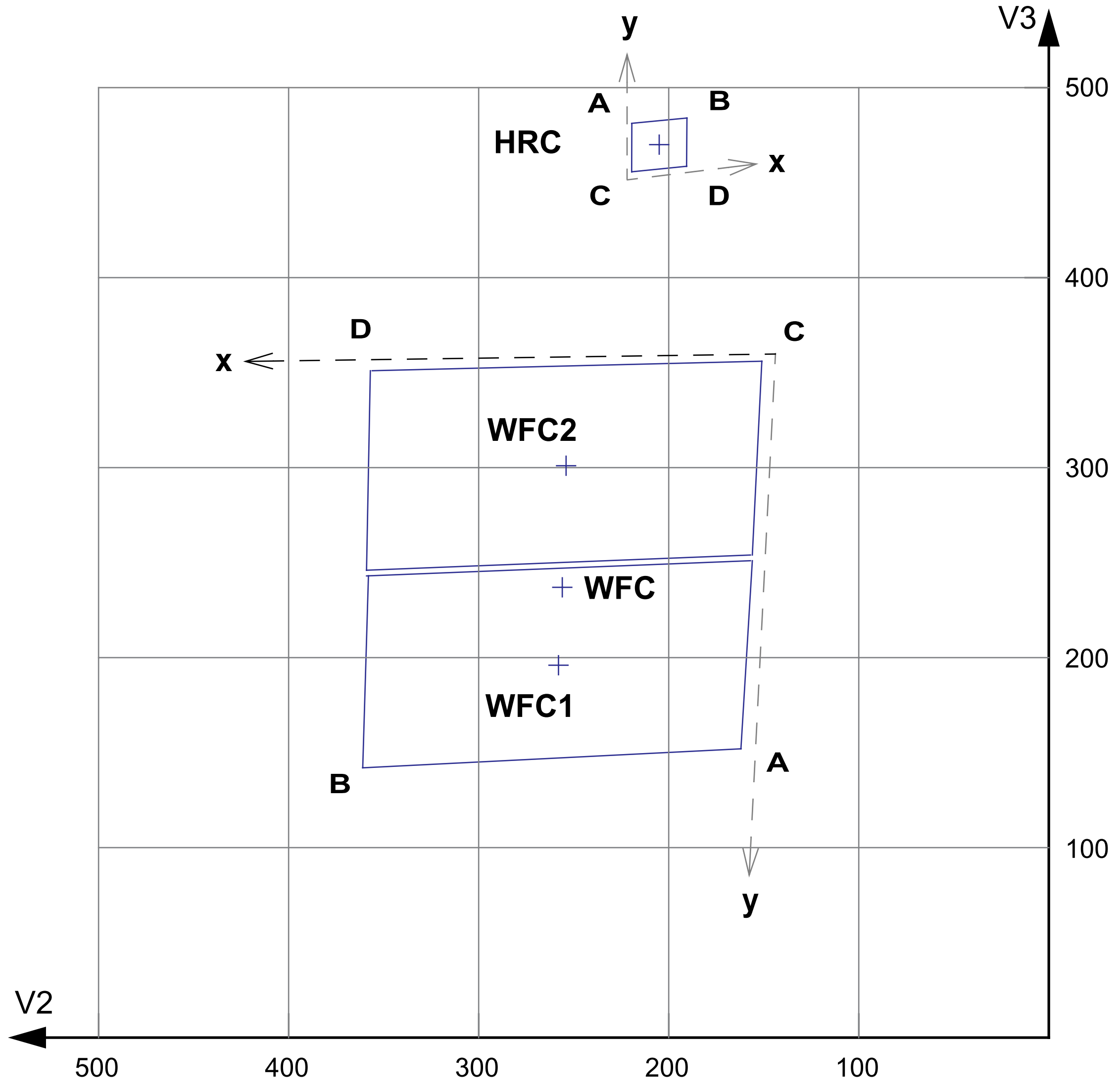
The principal cause of the ACS distortion is that the optics have been designed with a minimum number of components, consistent with correcting for the spherical aberration induced by the Optical Telescope Assembly (OTA), without introducing coma. The result is a high throughput, but focal surfaces far from normal to the principal rays. The WFC detector is tilted at 22 degrees giving an elongation of 8% along the diagonal. The HRC and SBC detectors have a 25 degree tilt giving an elongation of 12%. In each case, the scale in arcseconds per pixel is smaller along the radial direction of the OTA field of view than along the tangential direction. When projected on the sky, this causes each detector to appear "rhombus-shaped" rather than square. The angle on the sky between the x and y axes is 84.9 degrees for the WFC1, 86.1 for the WFC2 and 84.2 degrees for the HRC.

The orientations of the ACS detector edges are approximately in line with the V2 and V3 coordinate axes of the telescope. Consequently, the eigenaxes of the scale transformation are along the diagonals for WFC and the apertures and pixels appear non-rectangular in the sky projection. For the HRC and SBC the situation is even more irregular because the aperture diagonals do not lie along a radius of the HST field of view. Figure 1.1 shows the ACS apertures in the telescope's V2/V3 reference frame and illustrates the "rhombus" shape of each detector. A telescope roll angle of zero degrees would correspond to an on-sky view with the V3 axis aligned with North and the V2 with East. The readout amplifiers are also indicated for each detector.

If these were the only distortions present, their impact on photometry and mosaicing/dithering could be simply computed. A more problematic effect is the variation of scale and pixel area across each detector. For the WFC this amounts to a change of ~10% in scale from corner to corner. For the HRC and SBC this variation is only about 1%, since these detectors cover much smaller fields of view. The area on the sky covered by a WFC pixel varies by ~18% from corner to corner, allowance for which *must* be made in photometry.

Dithering and mosaicing are complicated by the fact that an integral pixel shift near the center of the detector will translate into a non-integral displacement for pixels near the edges. This is not a fundamental limitation, but will imply some computational complexity in registering images and will depend on an accurate measurement of distortions.

Figure 1.1: ACS apertures compared with the V2/V3 reference frame. The read-out amplifiers (A,B,C,D) are indicated on the figure. The WFC data products from the calibration pipeline are oriented so that WFC1 (chip 1, which uses amps A,B) is on top. The HRC data products are also oriented such that amps A,B are on top, but they are inverted from the WFC images with respect to the sky.



1.2 Basic Instrument Operations

1.2.1 Target Acquisitions

For the majority of ACS observations target acquisition is simply a matter of defining the appropriate aperture for the observation. Once the telescope acquires its guide stars, the target will be within ~1–2 arcseconds of the specified pointing. For observations with the ramp filters, one must specify the desired central wavelength for the observation. For the special case of coronagraphic observations, an onboard target acquisition will need to be specified. The nominal accuracy of the combined target acquisition and slew procedure is ~0.03 arcseconds, comparable to that achieved by STIS.

1.2.2 Typical ACS Observing Sequence

An important issue for observers to consider is the "packaging" of their observations, i.e. how observations are `CR-SPLIT` to mitigate the impact of cosmic rays, whether sub-stepping or "dithering" of images is required, and how, if necessary, to construct a mosaic pattern to map the target. For an online library of pointing patterns, refer to the ACS dither webpage:

http://www.stsci.edu/hst/acs/proposing/dither.

HRC observations and narrowband observations with the WFC are more likely to be read-noise limited, requiring consideration of the optimum `CR-SPLIT` times. Observations with the MAMA detectors do not suffer from cosmic rays or read noise, but long integration times are often needed to obtain sufficient signal-to-noise in the photon-starved ultraviolet.

A typical ACS observing sequence consists of a series of `CR-SPLIT` and dithered ~10–20 minute exposures for each program filter. Coronagraphic observations require an initial target acquisition observation to permit centering of the target under the occulting mask. Observers generally need not take their own calibration exposures.

1.2.3 Data Storage and Transfer

At the conclusion of each exposure, the science data are read out from the detector and placed in ACS's internal buffer memory, where they are stored until it can be transferred to the HST solid state data recorder (and thereafter to the ground). The internal buffer memory is large enough to hold one WFC image, or sixteen HRC or SBC images, and so the buffer will typically need to be dumped during the following WFC exposure, assuming it is longer than 340 seconds. For shorter exposures an extra overhead of this length is imposed.

ACS's internal buffer stores the data in a 16 bit-per-pixel format. This structure imposes a maximum of 65,535 counts per pixel. For the MAMA detectors this maximum is equivalent to a limit on the total number of detected photons per pixel which can be accumulated in a single exposure. For the WFC and HRC, the full well (and not the 16 bit buffer format) limits the photons per pixel which can be accumulated without saturating in a single exposure when `GAIN` > 1 and `GAIN` > 2, respectively, are selected.

1.2.4 Parallel Operations

Parallel observations with the WFC and HRC are possible with ACS for certain filter combinations. ACS can be used in parallel with any of the other science instruments on HST within restrictions described in detail in the *ACS Instrument Handbook*. There are significant constraints on the use of the MAMA detectors in parallel. The policy for applying for parallel observing time is described in the *HST Call for Proposals*.

CHAPTER 2:

ACS Data Structure

In this chapter. . .

This chapter describes the ACS data products, including a discussion of file suffixes, association tables, and trailer files. It describes the FITS file structure and data storage requirements for ACS images. Finally, a description of the FITS header keywords is provided for both primary and extension headers.

2.1 Types of ACS Files

2.1.1 Data Files and Suffixes

The file suffixes given to ACS data products are described in Table 2.1 and closely mimic the suffixes used by STIS. The initial input files to the calibration pipeline are the raw (RAW) files from Generic Conversion and the association (ASN) table, if applicable, for the complete observation set.

For CCD images, a temporary file, with the suffix BLV_TMP, is created once bias levels are subtracted and the overscan regions are trimmed. This file is renamed with the FLT extension after the standard calibrations (flat fielding, dark subtraction, etc.) are complete. The FLT files will serve as input for cosmic ray rejection (if required). For `CR-SPLIT` exposures, a temporary '*cr-combined image*' (CRJ_TMP) is created and then renamed with the CRJ extension once basic calibrations are complete. Single MAMA images are given the FLT suffix once calibrations are complete. By definition, these images do not have an overscan region and are not affected by cosmic rays. The calibrated products of a `REPEAT-OBS` association will be several individually calibrated FLT exposures and a summed flat-fielded (but not cosmic-ray cleaned) SFL image.

Table 2.1: ACS File Suffixes:

File Suffix	Description	Units
_RAW	Raw data	*DN*
_ASN	Association file for observation set	
_SPT	Telemetry and engineering data	
_TRL	Trailer File with processing comments	
_BLV_TMP	Overscan-trimmed individual exposure (renamed to _FLT)	*DN*
_CRJ_TMP	Uncalibrated, CR-rejected Combined image (renamed to _CRJ)	*DN*
_FLT	Calibrated, Flat fielded individual exposure	*electrons*
_CRJ	Calibrated, CR-rejected, Combined image	*electrons*
_SFL	Calibrated, Repeat-Obs, Combined image	*electrons*
_DRZ	Calibrated, Geometrically Corrected, Dither-Combined image	*electrons/sec*

2.1.2 Association Tables

Association tables are useful for keeping track of the complex set of relationships that can exist between exposures taken with ACS, especially with `REPEAT-OBS`, `CR-SPLIT,` and dithered exposures. Images taken at a given dither position may be additionally `CR-SPLIT` into multiple exposures. In these cases, associations are built to describe how each exposure relates to the desired final product. As a result, ACS associations will create one or more science products from the input exposures, unlike NICMOS or STIS associations. The relationships defined in the association table determine how far through the calibration pipeline the exposures are processed and when the calibrated exposures are combined into sub-products for further calibration.

ACS data files are given the following definitions:

- An *exposure* is a single image, the “atomic unit” of HST data.
- A *dataset* is a collection of files having a common root name (first 9 characters).
- A *sub-product* is a dataset created by combining a subset of the exposures in an association.
- A *product* is a dataset created by combining sub-products of an association.

ACS association tables were designed to closely resemble the NICMOS association format, with three primary columns: `MEMNAME, MEMTYPE,` and `MEMPRSNT`. The column `MEMNAME` gives the name of each exposure making up the association and output product name(s). The column `MEMTYPE` specifies the role the file has in the association. A unique set of `MEMTYPES` specific to ACS were adopted to provide the support for multiple products. These types are summarized in Table 2.2.

Table 2.2: Exposure types in ACS associations. The suffix "*n*" is appended to the `MEMTYPE` when multiple sets are present within a single association.

MEMTYPE	Description
EXP-CRJ	Input CR-SPLIT exposure (single set)
EXP-CR*n*	Input CR-SPLIT exposure for CR-combined image *n* (multiple sets)
PROD-CRJ	CR-combined output product (single set)
PROD-CR*n*	CR-combined output product *n* (multiple sets)
EXP-RPT	Input REPEAT-OBS exposure (single set)
EXP-RP*n*	Input REPEAT-OBS exposure for repeated image *n* (multiple sets)
PROD-RPT	REPEAT-OBS combined output product (single set)
PROD-RP*n*	REPEAT-OBS combined output product *n* (multiple sets)
EXP-DTH	Input dither exposure
PROD-DTH	Dither-combined output product

A sample association table for a two-position dithered observation with `CR-SPLIT`=2 is presented in Table 2.3. This example shows how both `MEMNAME` and `MEMTYPE` are used to associate input and output products. The `MEMTYPE` for each component of the first `CR-SPLIT` exposure, `JxxxxxECM` and `JxxxxxEGM`, are given the type `EXP-CR1`. The sub-product `Jxxxxx011` is designated in the table with a `MEMTYPE` of `PROD-CR1`. The last digit of the product filename corresponds to the output product number in the `MEMTYPE`. A designation of zero for the last digit in the filename is reserved for the dither-combined product.

The column `MEMPRSNT` indicates whether a given file already exists. For example, if cosmic ray rejection has not yet been performed by **CALACS**, the `PROD-CRn` files will have a `MEMPRSNT` value of "no". The sample association table in Table 2.3 shows the values of `MEMPRSNT` prior to **CALACS** processing.

Table 2.3: Sample Association Table `Jxxxxx010_ASN`

MEMNAME	MEMTYPE	MEMPRSNT
JxxxxxECM	EXP-CR1	yes
JxxxxxEGM	EXP-CR1	yes
Jxxxxx01<u>1</u>	PROD-CR1	no
JxxxxxEMM	EXP-CR2	yes
JxxxxxEOM	EXP-CR2	yes
Jxxxxx01<u>2</u>	PROD-CR2	no
Jxxxxx010	PROD-DTH	no

2.1.3 Trailer Files

Each task in the **CALACS** package creates messages during processing which describe the progress of the calibration and which are sent to STDOUT. In the pipeline processing for other HST instruments, trailer files were created by simply redirecting the STDOUT to a file. Because multiple output files can be processed in a single run of **CALACS**, creating trailer files presents a unique challenge. Each task within the **CALACS** package must decide which trailer file should be appended with comments and automatically open, populate, and close each trailer file.

CALACS will *always overwrite* information in trailer files from previous runs of **CALACS** while preserving any comments generated by Generic Conversion. This ensures that the trailer files accurately reflect the most recent processing performed. The string *CALACSBEG* will mark the first comment added to the trailer file. If a trailer file already exists, **CALACS** will search for this string to determine where to append processing comments. If it is not found, the string will be written to the end of the file and all comments will follow. Thus any comments from previous processing are overwritten and only the most current calibrations are recorded.

As each image is processed**,** an accompanying trailer file with the `'.trl'` ending will be created. Further processing will concatenate all trailer files associated with the output product into a single file. Additional messages will then be appended to this concatenated file. Thus, some information is duplicated across multiple trailer files, but for any product processed within the pipeline, the trailer file is ensured to contain processing comments from each input file.

Linking trailer files together can result in multiple occurrences of the *CALACSBEG* string. Only the first, however, determines where **CALACS** will begin overwriting comments if an observation is reprocessed.

2.2 ACS File Structure

The ACS calibration pipeline assembles data received from HST into datasets, applies the standard calibrations, and stores the uncalibrated datasets in the HST Data Archive. The structure of these data products is based on the STIS and NICMOS file format and consists of multi-extension FITS files which store science (SCI), data quality (DQ) and error (ERR) arrays, as shown in Figure 2.1.

ACS WFC data come from two CCD chips and are treated as separate observations, with a SCI, DQ and ERR array for each chip, and with both chips being stored in the same FITS file. The result is a FITS file with 6 data extensions plus a global header for a single WFC exposure. The WFC apertures are plotted with respect to the V2/V3 reference frame in Figure 2.2 and are oriented such that the x-axis is approximately to the right and the y-axis is approximately straight up. For pipeline data products, chip 2 is below chip 1 in y-pixel coordinates and was therefore defined as extension 1. Thus, the chip/extension notation is counterintuitive. To display the science image for chip 1, the user must specify the extension 'file.fits[sci,2]'. Similarly, the data quality and error array for chip 1 are specified as [dq,2] and [err,2].

A single HRC or SBC exposure comes from a single chip and has only 3 data extensions plus a global header. While the raw and calibrated WFC images contain 6 data extensions, the drizzled product will always contain 3 data extensions, no matter which detector was used. When WFC data is drizzled, both chips are included in a single FITS extension.

Figure 2.1 illustrates the basic format for storing ACS images. An uncalibrated (RAW) exposure contains a primary header plus, for each chip, a SCI extension (in 16-bit integer format), an empty ERR extension, and an empty DQ array. The calibrated product from **CALACS** contains a primary header plus, for each chip, a SCI extension (in 32-bit float format), an ERR extension (32-bit float), and a DQ extension (16-bit integer).

PyDrizzle removes the effects of the geometric distortion and produces a drizzled (DRZ) product which has the multi-extension FITS file shown in Figure 2.1. The SCI extension contains the distortion-corrected data (32-bit float), the WHT extension contains the weight mask (32-bit float), and the CTX extension contains the context image (32-bit integer). For more information on these DRZ product extensions, refer to "Data Products" in Section 4.5.

Figure 2.1: Data format for calibrated and drizzled ACS modes. Note that for calibrated science data, WFC1 (chip 1) corresponds to extension [sci,2].

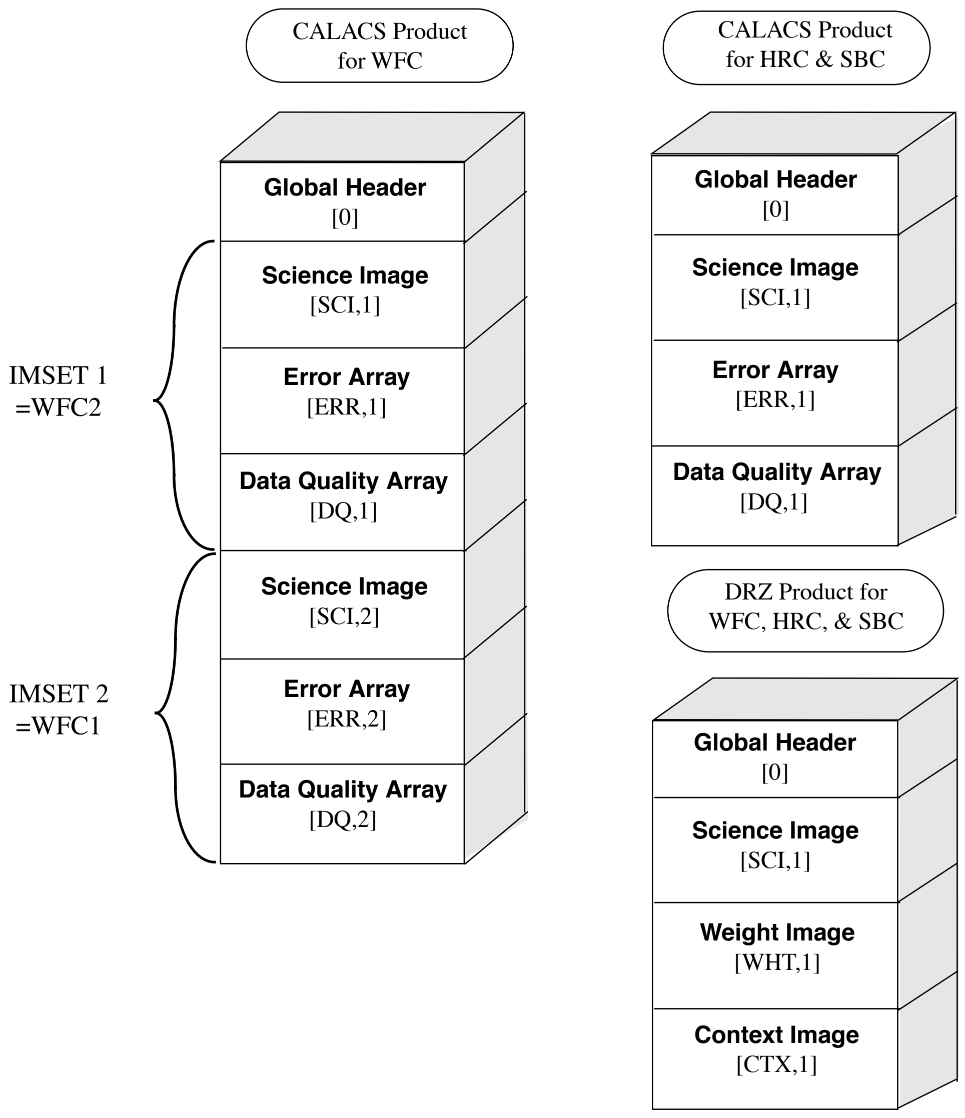
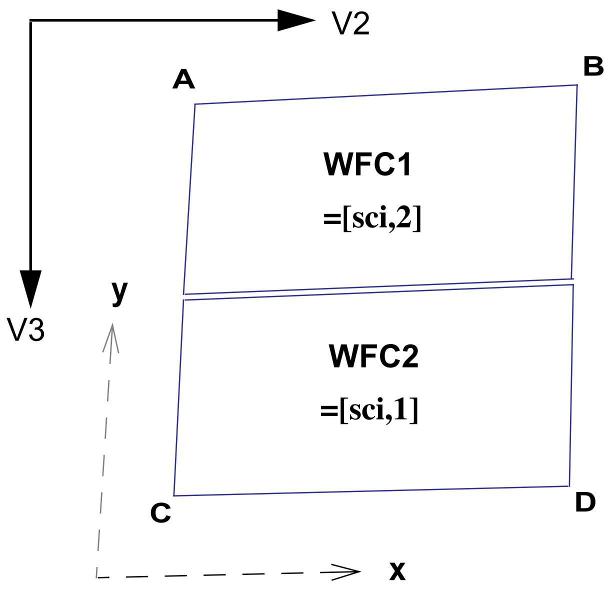


Figure 2.2: WFC apertures compared with the V2/V3 reference frame. The readout amplifiers (A,B,C,D) are indicated on the figure. The WFC data products from the calibration pipeline are oriented so that WFC1 (chip 1, which uses amps A,B) is on top. The science image for chip 1 may be specified as [sci,2].



2.3 Data Storage Requirements

Users are reminded to consider the large size of WFC exposures when allocating disk space for storing and reprocessing ACS data. Raw images serve as input to the calibration pipeline and have the file sizes (in MB) given in Table 2.4. The WFC image includes two SCI arrays, while the HRC and SBC detectors have a single array. The raw image sizes presume that both the SCI and DQ arrays are populated with short integer values, but that the ERR arrays are NULL (all pixels have a value of zero).

During **CALACS** processing, the SCI arrays are converted from integer to floating point values. The null ERR array is populated with floating point values, and the null DQ array is populated with integer values. As a result, the size of calibrated images (in MB) is much larger. The image size in pixels is given in Table 2.5. Calibrated images taken with the WFC and HRC detectors are smaller than raw images because the overscan regions have been trimmed during processing.

PyDrizzle combines the science, data quality, and error arrays for each detector/array into a single science, weight, and context extension. Once the distortion is corrected, the size of the drizzled images (in pixels and therefore also in MB) will be larger than the calibrated images due to the fact that their projection on the sky is rhombus-shaped, rather than square. The specific dimensions of the drizzled image will depend on which distortion model is currently in use in the pipeline and will vary slightly (~1-2 pixels) due to the effects of velocity aberration.

Table 2.4: Size (in MB) of raw, calibrated, and drizzled ACS images for each detector. The drizzled image sizes are distortion-corrected only and assume no dither offset or scale change.

Detector	Size of FITS file (S_{raw})	Size of FITS file (S_{cal})	Size of FITS file (S_{drz})
WFC (2 Chips)	68.7 MB	168 MB	215 MB
HRC (1 Chip)	4.4 MB	10.5 MB	16 MB
SBC (1 Chip)	4.2 MB	10.5 MB	16 MB

Table 2.5: Size (in pixels) for raw, calibrated, and drizzled ACS images for each detector. The drizzled image sizes are distortion-corrected only and assume no dither offset or scale change. The size of drizzled images will vary slightly due to velocity aberration effects.

Detector	X_{raw}	Y_{raw}	X_{cal}	Y_{cal}	X_{drz}	Y_{drz}
WFC (2 Chips)	4144	2068	4096	2048	4213	4240
HRC (1 Chip)	1062	1044	1024	1024	1164	1138
SBC (1 Chip)	1024	1024	1024	1024	1397	1341

While the size of calibrated, drizzled HRC and SBC images is comparable to that of STIS or WFPC2, the ACS WFC images are over 16 times as large. The following equation can be used to estimate the minimum amount of free storage required for processing associated ACS data:

$$D_{min} = n \cdot S_{raw} + (1+n) \cdot S_{cal} + (2.1 + p^2) \cdot S_{drz}$$

where:

- D_{min} is the minimum free disk space required for processing,
- n is the number of exposures in each `CR-SPLIT` set or `REPEAT-OBS` set,
- S_{raw} is the size of the raw exposure (from Table 2.4),
- S_{cal} is the size of the calibrated exposure (from Table 2.4),
- S_{drz} is the size of the distortion-corrected exposure (from Table 2.4),
- p is the percentage shift (in pixels) across all dither positions

2.4 Headers and Keywords

While the ACS image headers for the WFC, HRC, HRC-ACQ, and SBC observations have very similar formats, two main differences are apparent. The MAMA (SBC) headers require a slightly modified structure from the CCD (WFC, HRC) headers, and the HRC-ACQ headers require several unique keywords.

In Table 2.6 and Table 2.7, we describe the ACS header keywords, with a note indicating to which detector those keywords refer. The primary header keywords (FITS extension 0) are given in Table 2.6, and the extension header keywords (FITS extensions > 0) are given in Table 2.7. For more information about primary and extension FITS headers, refer to Section 2.2.1 of the HST Data Handbook Introduction.

Table 2.6: ACS Primary Header Keywords (FITS extension 0)

KEYWORD	DESCRIPTION
SIMPLE	data conform to FITS standard (T/F)
BITPIX	bits per data value
NAXIS	number of data axes
EXTEND	file may contain standard extensions (T/F)
NEXTEND	number of standard extensions
GROUPS	image is in group format? (T/F)
DATE	date this file was written (yyyy-mm-dd)
FILENAME	name of file
FILETYPE	type of data found in data file
TELESCOP	telescope used to acquire data
INSTRUME	identifier for instrument used to acquire data
EQUINOX	equinox of celestial coordinate system
DATA DESCRIPTION KEYWORDS (All Detectors)	
ROOTNAME	rootname of the observation set
IMAGETYP	type of exposure identifier
PRIMESI	instrument designated as prime
TARGET INFORMATION (All Detectors)	
TARGNAME	proposer's target name
RA_TARG	right ascension of the target (deg) (J2000)
DEC_TARG	declination of the target (deg) (J2000)
PROPOSAL INFORMATION (All Detectors)	
PROPOSID	PEP proposal identifier
LINENUM	proposal log line number
PR_INV_L	last name of principal investigator
PR_INV_F	first name of principal investigator

EXPOSURE INFORMATION (All Detectors)	
SUNANGLE	angle between sun and V1 axis
MOONANGL	angle between moon and V1 axis
SUN_ALT	altitude of the sun above Earth's limb
FGSLOCK	commanded FGS lock (fine, coarse, gyros, unknown)
DATE-OBS	UT date of start of observation (yyyy-mm-dd)
TIME-OBS	UT time of start of observation (hh:mm:ss)
EXPSTART	exposure start time (modified Julian date)
EXPEND	exposure end time (modified Julian date)
EXPTIME	exposure duration (seconds)--calculated
EXPFLAG	exposure interruption indicator
POINTING INFORMATION (All Detectors)	
PA_V3	position angle of V3-axis of HST (deg)
TARGET OFFSETS (POSTARGS) (All Detectors)	
POSTARG1	POSTARG in axis 1 direction
POSTARG2	POSTARG in axis 2 direction
DIAGNOSTIC KEYWORDS (All Detectors)	
OPUS_VER	OPUS software system version number
CAL_VER	CALACS code version
PROCTIME	pipeline processing time (MJD)
SCIENCE INSTRUMENT CONFIGURATION (All Detectors)	
OBSTYPE	observation type (imaging, spectroscopic, coronagraphic)
OBSMODE	operating mode
CTEIMAGE	type of charge transfer image, if applicable
SCLAMP	lamp status: NONE or name of lamp which is on
NRPTEXP	number of repeat exposures in set (default=1)
SUBARRAY	data from a subarray (T) or full frame (F)
DETECTOR	detector in use: WFC, HRC or SBC
FILTER1	element selected from filter wheel 1
FILTER2	element selected from filter wheel 2
FWOFFSET	computed filter wheel offset (0, +1, -1)
FWERROR	filter wheel position error flag (T, F)
LRFWAVE	proposed linear ramp filter wavelength
APERTURE	aperture name
PROPAPER	proposed aperture name
CRSPLIT	number of cosmic ray split exposures
DIRIMAGE	direct image for grism or prism exposure
MAMA OFFSETS (SBC Only)	
MOFFSET1	axis 1 MAMA offset
MOFFSET2	axis 2 MAMA offset

LOCAL RATE CHECK IMAGE (SBC Only)	
LRC_XSTS	local rate check image exists (T/F)
LRC_FAIL	local rate check failed (T/F)
CALIBRATION SWITCHES: PERFORM, OMIT (All Detectors)	
STATFLAG	calculate statistics? (T/F)
WRTERR	write out error array extension? (T/F)
DQICORR	data quality initialization?
ATODCORR	correct for A to D conversion errors?
BLEVCORR	subtract bias level computed from overscan image?
BIASCORR	subtract bias image?
FLSHCORR	subtract post flash image?
CRCORR	combine observations to reject cosmic-rays?
EXPSCORR	process individual observations after cr-reject?
SHADCORR	apply shutter shading correction?
GLINCORR	correct for global detector non-linearities? (SBC only)
LFLGCORR	flag pixels for local and global nonlinearities? (SBC only)
DARKCORR	subtract dark image?
FLATCORR	apply flat field correction?
PHOTCORR	populate photometric header keywords?
RPTCORR	add individual repeat observations?
DRIZCORR	process dithered images?
CALIBRATION REFERENCE FILES (All Detectors)	
BPIXTAB	bad pixel table
CCDTAB	CCD calibration parameter table
ATODTAB	analog-to-digital correction file
OSCNTAB	CCD overscan table
BIASFILE	bias image file name
FLSHFILE	post flash image file name
CRREJTAB	cosmic-ray rejection parameter table
SHADFILE	shutter shading correction file
MLINTAB	MAMA linearity correction table (SBC only)
DARKFILE	dark image file name
PFLTFILE	pixel-to-pixel flat field file name
DFLTFILE	delta-flat field file name
LFLTFILE	low order flat
PHOTTAB	photometric throughput table (not used)
GRAPHTAB	HST graph table
COMPTAB	HST components table
IDCTAB	image distortion correction table

COSMIC RAY REJECTION ALGORITHM PARAMETERS (WFC, HRC, HRC ACQ)	
MEANEXP	reference exposure time
SCALENSE	multiplicative scale factor applied to noise
INITGUES	initial guess method (min, med)
SKYSUB	sky value subtracted (mode, none)
SKYSUM	sky level from the sum of all constituent images
CRSIGMAS	statistical rejection criteria
CRRADIUS	rejection propagation radius (pixels)
CRTHRESH	rejection propagation threshold
BADINPDQ	data quality flag used for rejection
REJ_RATE	rate at which pixels are affected by cosmic rays
CRMASK	flag CR-rejected pixels in input files? (T/F)
PATTERN KEYWORDS (WFC, HRC, SBC)	
PATTERN1	primary pattern type
P1_SHAPE	primary pattern shape
P1_PURPS	primary pattern purpose
P1_NPTS	number of points in primary pattern
P1_PSPAC	point spacing for primary pattern (arcsec)
P1_LSPAC	line spacing for primary pattern (arcsec)
P1_ANGLE	angle between sides of parallelogram patt (deg)
P1_FRAME	coordinate frame of primary pattern
P1_ORINT	orientation of pattern to coordinate frame (deg)
P1_CENTR	center pattern relative to pointing (yes/no)
PATTSTEP	position number of this point in the pattern
POST FLASH PARAMETERS (WFC, HRC, HRC ACQ)	
FLASHDUR	exposure time in seconds: 0.1 to 409.5
FLASHCUR	post-flash current (off, low, med, high)
FLASHSTA	status (successful, aborted, not performed)
SHUTRPOS	shutter position (A or B)
ENGINEERING PARAMETERS (WFC, HRC, HRC ACQ)	
CCDAMP	CCD amplifier read out configuration
CCDGAIN	commanded gain of CCD (electrons/DN)
CCDOFSTA	commanded CCD bias offset for amplifier A (electrons)
CCDOFSTB	commanded CCD bias offset for amplifier B (electrons)
CCDOFSTC	commanded CCD bias offset for amplifier C (electrons)
CCDOFSTD	commanded CCD bias offset for amplifier D (electrons)

CALIBRATED ENGINEERING PARAMETERS (WFC, HRC, HRC ACQ)	
ATODGNA	calibrated gain for CCD amplifier A (electrons/DN)
ATODGNB	calibrated gain for CCD amplifier B (electrons/DN)
ATODGNC	calibrated gain for CCD amplifier C (electrons/DN)
ATODGND	calibrated gain for CCD amplifier D (electrons/DN)
READNSEA	calibrated read noise for amplifier A (electrons)
READNSEB	calibrated read noise for amplifier B (electrons)
READNSEC	calibrated read noise for amplifier C (electrons)
READNSED	calibrated read noise for amplifier D (electrons)
BIASLEVA	bias level for amplifier A (electrons)
BIASLEVB	bias level for amplifier B (electrons)
BIASLEVC	bias level for amplifier C (electrons)
BIASLEVD	bias level for amplifier D (electrons)
TARGET ACQUISITION PARAMETERS (HRC only)	
ACQTYPE	type of acquisition
ACQNAME	rootname of acquisition exposure
TARGET ACQUISITION PARAMETERS (HRC ACQ only)	
ACQTYPE	type of acquisition
CENTMETH	target acquisition centering method
CRELIM	perform cosmic ray rejection in acquisition
CCDBIASS	CCD bias subtracted from target acq image? (y/n)
BIASLEV	CCD bias level used to process acquisition exp.
CHECKBOX	size of checkbox for finding algorithms
TARGAREA	area of target (detector pixels)
ASSOCIATION KEYWORDS (All Detectors)	
ASN_ID	unique identifier assigned to association
ASN_TAB	name of association table
ASN_MTYP	role of exposure in association

Table 2.7: ACS Extension Header Keywords (FITS extensions > 0)

KEYWORD	DESCRIPTION
XTENSION	extension type
BITPIX	bits per data value
NAXIS	number of data axes
NAXIS1	length of first data axis
NAXIS2	length of second data axis
PCOUNT	number of group parameters
GCOUNT	number of groups
INHERIT	inherit the primary header
EXTNAME	extension name
EXTVER	extension version number
ROOTNAME	rootname of the observation set
EXPNAME	9 character exposure identifier
DATAMIN	minimum data value (electrons)
DATAMAX	maximum data value (electrons)
BUNIT	brightness units (counts, electrons, electrons/sec)
BSCALE	scale factor for array value to physical value
BZERO	physical value for an array value of zero
WORLD COORDINATE SYSTEM AND RELATED PARAMETERS (All Detectors)	
CRPIX1	x-coordinate of reference pixel
CRPIX2	y-coordinate of reference pixel
CRVAL1	first axis value at reference pixel
CRVAL2	second axis value at reference pixel
CTYPE1	the coordinate type for the first axis
CTYPE2	the coordinate type for the second axis
CD1_1	partial of first axis coordinate w.r.t. x
CD1_2	partial of first axis coordinate w.r.t. y
CD2_1	partial of second axis coordinate w.r.t. x
CD2_2	partial of second axis coordinate w.r.t. y
LTV1	offset in X to subsection start
LTV2	offset in Y to subsection start
LTM1_1	reciprocal of sampling rate in X
LTM2_2	reciprocal of sampling rate in Y
RA_APER	RA of aperture reference position
DEC_APER	Dec of aperture reference position
PA_APER	position angle of aperture reference position (deg)
ORIENTAT	position angle of image y axis (degrees E of N)
VAFACTOR	velocity aberration plate scale factor

WFC CCD CHIP IDENTIFICATION (WFC only)	
CCDCHIP	CCD chip (1 or 2)
READOUT DEFINITION PARAMETERS (All Detectors)	
CENTERA1	subarray axis1 center point in unbinned detector pixels
CENTERA2	subarray axis2 center point in unbinned detector pixels
SIZAXIS1	subarray axis1 size in unbinned detector pixels
SIZAXIS2	subarray axis2 size in unbinned detector pixels
BINAXIS1	axis1 data bin size in unbinned detector pixels
BINAXIS2	axis2 data bin size in unbinned detector pixels
PHOTOMETRY KEYWORDS (All Detectors)	
PHOTMODE	observation configuration for photometric calibration
PHOTFLAM	inverse sensitivity (erg s^{-1} cm^{-2} A^{-1} for 1 electron/sec)
PHOTZPT	ST magnitude zero point
PHOTPLAM	pivot wavelength
PHOTBW	rms bandwidth of filter plus detector
REPEATED EXPOSURES INFO (WFC, HRC, HRC ACQ)	
NCOMBINE	number of image sets combined during CR rejection
DATA PACKET INFORMATION (All Detectors)	
FILLCNT	number of segments containing fill
ERRCNT	number of segments containing errors
PODPSFF	podps fill present? (T/F)
STDCFFF	ST DDF fill present? (T/F)
STDCFFP	ST DDF fill pattern (hex)
ON-BOARD COMPRESSION INFORMATION (WFC Only)	
WFCMPRSD	WFC data compressed? (T/F)
COMPTYP	compression type performed? (partial/full)
CBLKSIZ	size of compression block in bytes
LOSTPIX	number of pixels lost due to buffer overflow
TARGET ACQUISITION PARAMETERS (HRC ACQ Only)	
TARGA1	fsw located subarray axis1 coord. of target
TARGA2	fsw located subarray axis2 coord. of target
APERA1	fsw located subarray axis 1 coord. of aperture
APERA2	fsw located subarray axis 2 coord. of aperture
APERLKA1	axis 1 detector pixel of acq aperture center
APERLKA2	axis 2 detector pixel of acq aperture center
MAXCHCNT	counts in the brightest checkbox
BOPOFFA1	axis 1 offset (arcs) object moved off aperture
BOPOFFA2	axis 2 offset (arcs) object moved off aperture

ENGINEERING PARAMETERS (SBC Only)	
GLOBRATE	global count rate
GLOBLIM	was global linearity level exceeded?
IMAGE STATISTICS AND DATA QUALITY FLAGS (All Detectors)	
NGOODPIX	number of good pixels
SDQFLAGS	serious data quality flags considered ‘bad’ by CALACS
GOODMIN	minimum value of good pixels (electrons)
GOODMAX	maximum value of good pixels (electrons)
GOODMEAN	mean value of good pixels (electrons)
SNRMIN	minimum signal to noise of good pixels
SNRMAX	maximum signal to noise of good pixels
SNRMEAN	mean signal to noise of good pixels
SOFTERRS	number of soft error pixels (DQF=1)
MEANBLEV	average of the subtracted bias levels (electrons)
MEANDARK	average of the subtracted dark values (electrons)
MEANFLSH	average of the subtracted post flash values (electrons)

CHAPTER 3:

ACS Calibration Pipeline

In this chapter. . .

This chapter describes the ACS calibration pipeline and the On-the-Fly-Reprocessing, which is standard for all HST data requested from the STScI archive. The pipeline software consists of two separate packages: **CALACS** and **PyDrizzle**. **CALACS** removes various instrumental signatures, combines multiple exposures, and updates header keyword values. **PyDrizzle** corrects images for geometric distortion and combines observations which were taken as part of a dither pattern.

The data products from OTFR are described, as well as considerations and requirements for manual recalibration. The remaining bulk of the chapter is devoted to a detailed description of **CALACS**. While **PyDrizzle** is also used during pipeline processing, a detailed discussion of this software is deferred to Chapter 4.

3.1 On The Fly Reprocessing (OTFR)

OTFR is now the standard way of processing data which has been requested by the user from the STScI archive. It provides the best calibrated products by reprocessing the raw telemetry files “on-the-fly” for distribution each time data is requested. Each observation requires calibrations suited to the particular mode used for taking data. Calibration data specific to the various modes of ACS operation are prepared and archived in the Calibration Data Base System (CDBS).

The STScI calibration pipeline consists of two main software systems: the Operations Pipeline Unified System (OPUS) and the Data Archive and Distribution System (DADS). Raw spacecraft telemetry data from Goddard Spaceflight Center are transmitted to STScI in the form of POD files. When a user requests data from the HST Archive, OTFR uses the POD files as input to the OPUS step called Generic Conversion and generates the uncalibrated “raw” data. At the same time, Generic Conversion queries the CDBS to determine the most up-to-date reference files for the observation configuration. OPUS then runs **CALACS** to process the uncalibrated data, using specific ACS reference images and tables from the CDBS. With these data, DADS populates a database which is accessible to users via StarView. DADS then distributes any data (both calibrated and uncalibrated) requested for download to the user. (See Chapter 1 of the HST Introduction for more information.)

The most current ACS reference files are used by **CALACS** each time OTFR is run. The calibration reference files (e.g. flat fields, bad pixel tables) are also available from the HST Archive. Since reference files such as CCD biases and darks are frequently updated, OTFR may use different reference files depending on the date of reprocessing. Previously, the archived data from DADS had to be manually recalibrated by running the calibration software on the user’s home workstation with the updated reference files. OTFR replaces this step by automatically recalibrating with the most current reference data available. The user simply waits until the contemporaneous reference files are in place and requests the data via StarView or the HST web-based archive request form. The image header keywords are then updated with the names of the reference files used during that OTFR run. The PROCTIME keyword is updated to reflect the pipeline processing date and time in MJD.

OTFR allows the user to avoid downloading archived data which is outdated because of software changes made for bug fixes, improved algorithms, new capabilities, or header keyword changes. An archive request for calibrated data will result in OTFR reprocessing and distributing data which uses the latest software versions available.

Currently OTFR will distribute *all* files associated with an ACS observation, including both the uncalibrated and calibrated files. Future versions could enable users to select only certain parts of the dataset, for example the final calibrated image. Future versions could also enable users to set calibration parameters for a particular OTFR run. Until then, OTFR will process with default values. Of course, the option of recalibrating aside from OTFR always exists. (See "Manual Recalibration of ACS Data" in Section 3.5.)

3.2 Pipeline Overview

Pipeline processing is carried out by 2 separate packages: **CALACS**, which corrects for instrumental effects and produces calibrated products, and **PyDrizzle**, which corrects for distortion and combines associated dithered data.

3.2.1 CALACS: Image Calibration

The **CALACS** software consists of a series of individual tasks that:

- Orchestrate the flow of data through the pipeline.
- Perform the initial tasks of basic two-dimensional image reduction (e.g., overscan subtraction, bias subtraction) for CCD data.
- Reject cosmic rays from CR-SPLIT CCD data.
- Perform the remaining tasks of basic two-dimensional image reduction (e.g., dark subtraction, flat fielding) for CCD and MAMA data.
- Sum any REPEAT-OBS exposures.

The calibrated products from the pipeline may still contain artifacts such as hot pixels and cosmic-rays. If hot pixels are to be detected and flagged by **CALACS**, dithered exposures are required. Similarly, cosmic ray rejection by the pipeline requires the specification of CR-SPLIT exposures in the observing program. In the future, improved pipeline software will allow all associated data (including REPEAT-OBS exposures and dithered exposures) to be automatically corrected for cosmic-rays.

While intermediate steps in **CALACS** make use of sky subtraction, for example for identifying cosmic-rays, all data products created by the pipeline will *not* be sky subtracted. Fully calibrated data products (with suffixes FLT, CRJ, SFL) will be in units of *electrons*.

CALACS and Single exposures

A single raw ACS exposure is processed by **CALACS** following the steps outlined in Section 3.3. These steps include the standard detector calibrations: bias subtraction, dark subtraction, flat fielding, etc. When

CALACS processes a single raw exposure, the calibrated product is given the FLT suffix. The data in the calibrated SCI and ERR extensions are in units of *electrons*.

CALACS and Multiple exposures

The same processing steps performed on single images are performed for multiple images which are part of an association. The result is a single calibrated product in units of *electrons*.

CALACS will recognize and correctly process `CR-SPLIT` or `REPEAT-OBS` exposures by interpreting the association table and determining which exposures should be combined during calibration. `REPEAT-OBS` exposures are individually run through all the calibration steps to create calibrated, flat-fielded (FLT) product for each input file. The FLT images are then summed to create the summed flat-fielded (SFL) product. No cosmic-ray rejection is performed for `REPEAT-OBS` images.

Observations taken as part of a CR-SPLIT association, on the other hand, are combined during cosmic-ray rejection. First, the bias and dark subtraction is performed. Then, the images are combined while at the same time rejecting cosmic rays. The product is then flat fielded to create a single calibrated, cosmic-ray rejected, flat-fielded (CRJ) image.

CALACS and Dithered exposures

Observations which use the dither patterns provided in the proposal instructions will be automatically associated for combining in the pipeline. Any pattern designed with POS-TARGs will not be associated. When processing observations which are part of a dither pattern, **CALACS** will produce a calibrated FLT, CRJ, or SFL file at each dither position. It will not, however, combine images from multiple positions. Further pipeline processing by **PyDrizzle** will correct for geometric distortion and combine dithered images.

Table 3.1: The input and output image suffixes from **CALACS** and **PyDrizzle** are given for various observing modes.

Image Type	CALACS Input	CALACS Output	PyDrizzle Input	PyDrizzle Output	Cosmic Ray Rejected?
Single	RAW	FLT	FLT	DRZ	No
RPT-OBS	ASN+RAW	FLT+SFL	ASN+SFL	DRZ	No
CR-SPLIT	ASN+RAW	FLT+CRJ	ASN+CRJ	DRZ	Yes
DITH-PATTERN	ASN+RAW	FLT	ASN+FLT	DRZ	No

3.2.2 PyDrizzle: Distortion Correction and Dither Combination

All ACS data will be automatically corrected for distortion during pipeline processing using a task called **PyDrizzle**. This task relies on the IDCTAB reference file for a description of the distortion model. **PyDrizzle** understands the ACS association tables which allow the pipeline to combine dithered observations. Only observations which use the dither patterns provided in the proposal instructions will be automatically associated for combining in the pipeline. Programs which rely on explicit POS-TARG commands will NOT be associated, resulting in separately calibrated images for each position. **PyDrizzle** will automatically produce images which are both astrometrically and photometrically accurate regardless of whether they were taken as a single exposure or a set of dithered exposures.

The flat-fielded, calibrated products from **CALACS** are listed in Table 3.1. The calibrated product of a single ACS exposure is an FLT file. The product of a CR-SPLIT or REPEAT-OBS association is a CRJ or SFL file, respectively. The product of a dither pattern is an FLT, CRJ or SFL file at *each* dither position. Using these files as input, **PyDrizzle** performs the geometric correction on *all* data (dithered or not) and combines multiple dithered images into a single output image with the DRZ suffix. For WFC observations, both chips are combined into a single extension. **PyDrizzle** then converts the data to units of count rate (*electrons/sec).*

It is important to note that **PyDrizzle**, when combining dithered images, does *not* remove cosmic-rays. This can be done offline with a new task called **MultiDrizzle** using the individual FLT files as input. Subsequent reprocessing with **PyDrizzle** or **MultiDrizzle** offline may be required to obtain the desired scientific result. For information on obtaining this software, we refer the reader to Section 3.5.1.

3.2.3 When is OTFR *not* Appropriate?

The goal of the ACS pipeline is to provide data calibrated to a level suitable for initial evaluation and analysis for all users. Observers frequently require a detailed understanding of the calibrations applied to their data and the ability to repeat, often with improved products, the calibration process at their home institution. There are several occasions when OTFR is not ideal and when off-line interactive processing by the user is required:

- when running **CALACS** with personal versions of reference files,
- when running **CALACS** with non-default calibration switch values,
- when images must be cleaned of artifacts such as new hot pixels or cosmic-rays.

Many ACS datasets do not contain multiple exposures at the same pointing and hence will not be cleaned of cosmic rays by OTFR processing. In addition, the increasing prevalence of hot pixels, particularly in the WFC mode, makes dithered observations highly desirable. In these cases the pipeline will provide products which are geometrically corrected but which still contain cosmic rays.

To address this problem, the **MultiDrizzle** script has been developed to be run interactively outside the pipeline. **MultiDrizzle** provides a single-step interface to the complex suite of tasks in the STSDAS **dither** package. It makes use of the **PyDrizzle** software and can only be executed in the PyRAF environment. For more information, we refer the reader to Chapter 4.

3.3 Structure of CALACS

The **CALACS** package consists of four individual tasks listed in Table 3.2. These tasks are called automatically by **CALACS**, but each may be run separately should some variation in the normal processing be desired by the user.

Table 3.2: Tasks in the **CALACS** pipeline

ACSCCD	CCD specific calibrations
ACSREJ	Cosmic-ray rejection task
ACS2D	Basic MAMA and CCD calibrations
ACSSUM	Repeat-obs summing task

The flow of data through the ACS calibration pipeline and the decisions made while working with associated data are diagrammed in Figure 3.1. These calibration steps are also outlined below:

- Calculate a noise model for each pixel and record in the error array
- Flag known bad pixels and saturated pixels in the data quality array
- Subtract bias-level determined from overscan regions (CCD only)
- Subtract bias image (CCD only)
- Subtract post-flash image, if required (CCD only)
- Perform cosmic-ray rejection and combining of `CR-SPLIT` data (CCD only)
- Perform global linearity corrections (MAMA only)
- Scale and subtract dark image and calculate mean dark value
- Divide by flat field and multiply by gain
- Calculate photometry header keywords for flux conversion
- Calculate image statistics

Additional calibration steps are performed by **PyDrizzle**, including:

- Correct the geometric distortion of non-dithered images
- Convert the data units from counts to count rates
- Combine calibrated dithered observations into a single product

As indicated in Figure 3.1, the calibration tasks have been split to handle CCD-specific calibrations (**ACSCCD**) separately from those steps which can be applied to any ACS data (**ACS2D**). For example, the MAMA data obtained with the SBC do not have the overscan regions found in CCD data.

The initial processing performed on CCD data alone is shown in Figure 3.2. The reference files appropriate for each processing step and the calibration switches controlling them are also given beside the name of the task they control. The output (overscan-trimmed image) from **ACSCCD** is then sent through **ACS2D** as shown in Figure 3.3. Raw MAMA data are processed from the start with **ACS2D**, which initializes the error and DQ arrays (already performed for CCD data) and which performs linearity corrections.

Figure 3.1: Flow Diagram for ACS data with **CALACS** task names in bold.

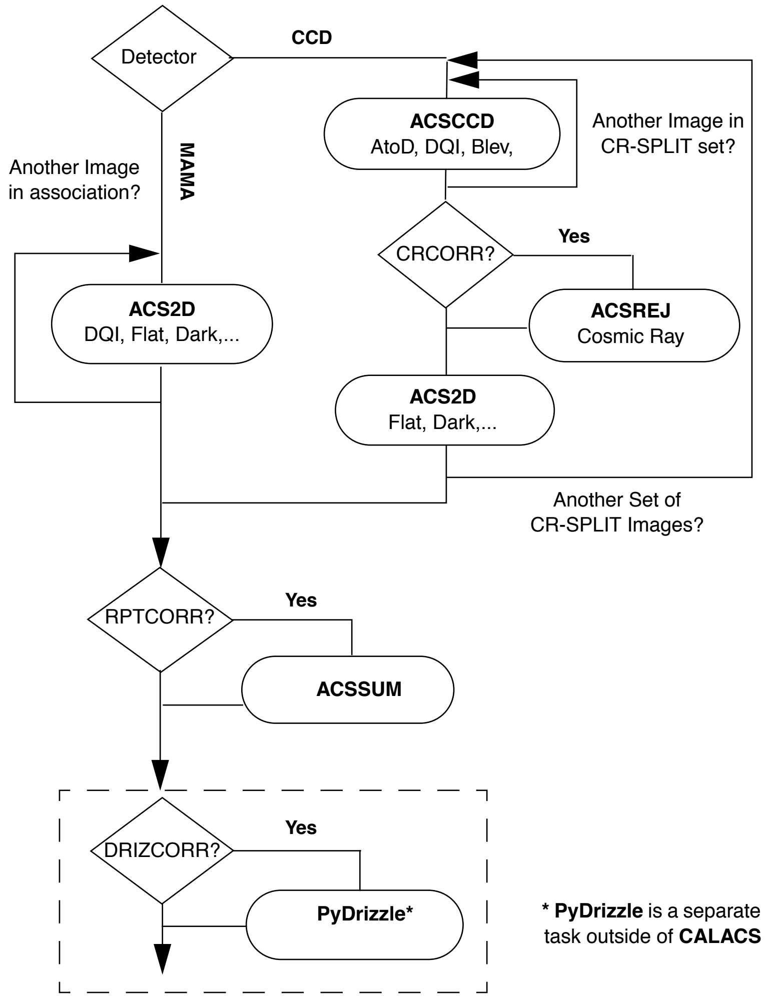


Figure 3.2: Flow diagram for CCD data using **ACSCCD** in **CALACS**.

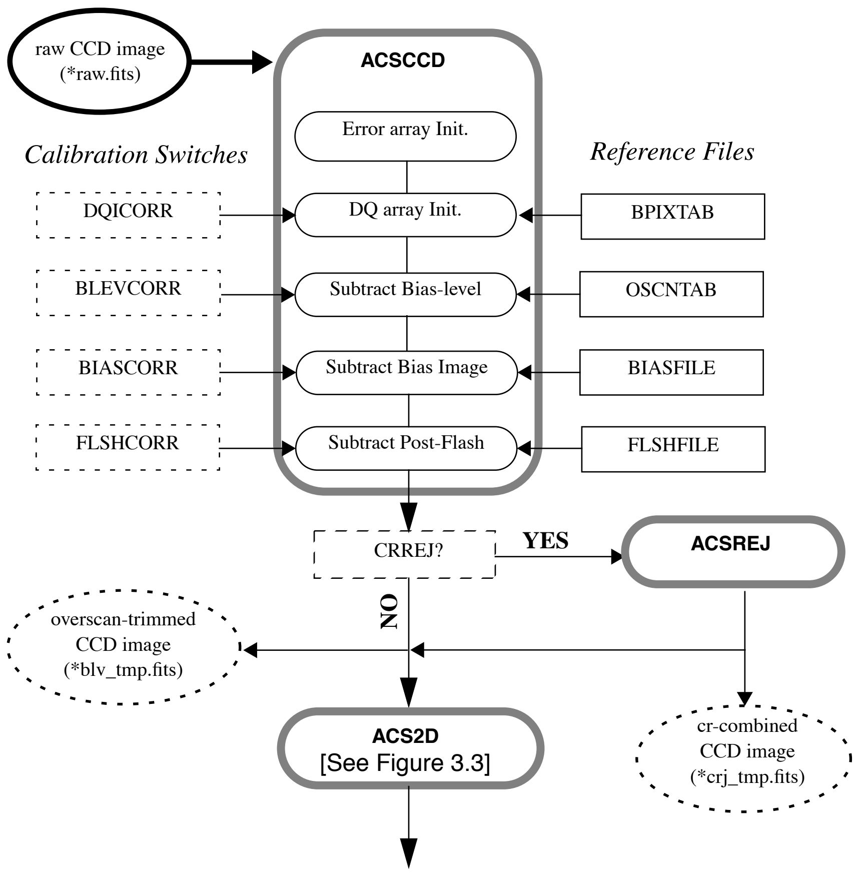
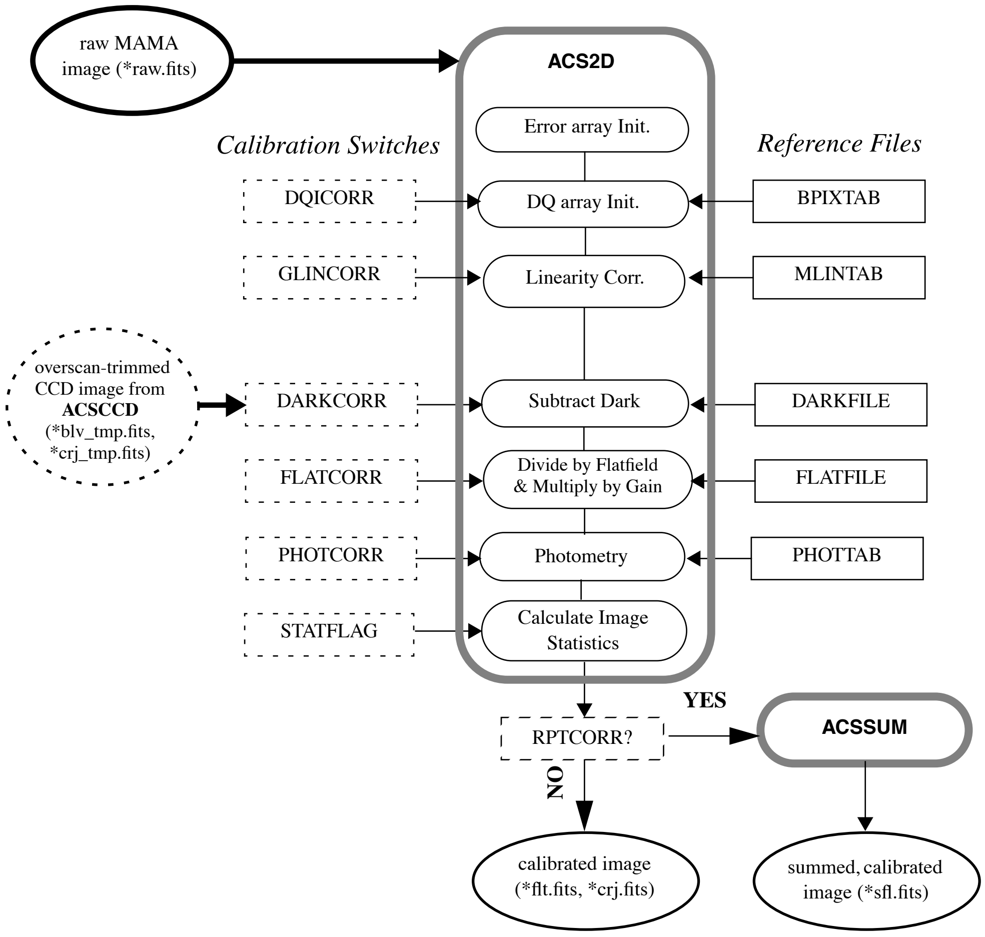


Figure 3.3: Flow diagram for MAMA and overscan-trimmed CCD data using **ACS2D** in **CALACS**.


raw MAMA image (*raw.fits)
ACS2D
Calibration Switches
Reference Files
Error array Init.
DQICORR
DQ array Init.
BPIXTAB
GLINCORR
Linearity Corr.
MLINTAB
overscan-trimmed CCD image from ACSCCD (*blv_tmp.fits, *crj_tmp.fits)
DARKCORR
Subtract Dark
DARKFILE
FLATCORR
Divide by Flatfield & Multiply by Gain
FLATFILE
PHOTCORR
Photometry
PHOTTAB
STATFLAG
Calculate Image Statistics
RPTCORR?
YES
NO
ACSSUM
calibrated image (*flt.fits, *crj.fits)
summed, calibrated image (*sfl.fits)

3.4 CALACS Processing Steps

The **CALACS** pipeline consists of four individual calibration tasks: **ACSCCD**, **ACSREJ, ACS2D**, and **ACSSUM**. These tasks are listed in Table 3.2 and diagrammed in Figure 3.1. **CALACS** is responsible for controlling the processing rather than actually calibrating the data. The individual tasks apply the desired calibration steps to the data and create the output products, including the trailer files.

In the following four sections, we describe each **CALACS** task, give a detailed description of the calibrations steps performed within each task, and give a brief description of the reference files used for each step.

3.4.1 ACSCCD

This routine contains the initial processing tasks for all ACS CCD data. These tasks are listed in operational order in Table 3.3. The bulk of this routine was based on the code for **CALSTIS1**. **ACSCCD** processes each image in the input list one at a time, using the header keywords to determine which calibrations are performed. Upon completion of **ACSCCD**, the overscan regions will be trimmed from the image and an output image with the file extension '_blv_tmp.fits' is created.

Table 3.3: **ACSCCD** processing tasks

doNoise	Initialize error array
doDQI	Initialize data quality array
doAtoD	Perform AtoD correction (currently skipped)
doBlev	Subtract bias level from overscan region
doBias	Subtract bias image
doFlash	Subtract post-flash image (if required)
Final Output	Output overscan-trimmed or full image

doNoise - Error Array Initialization

- Header Switch: None
- Header Keywords Updated: None
- Reference File: CCDTAB (*_ccd.fits)

The first processing step is initializing the image error array. The **doNoise** function examines the ERR extension of the input data to determine the state of the array. Input '_raw.fits' images delivered by Generic Conversion will contain a NULL ERR array, defined by the keywords NPIX1, NPIX2 and PIXVALUE, where PIXVALUE=0. If the ERR array has already been expanded and contains values other than zero, then this function does nothing. Otherwise, **CALACS** will initialize the array and **doNoise** will assign a simple noise model.

The noise model reads the science array and for each pixel calculates the error value σ (in DN for the CCD, in counts for the MAMA):

$$\sigma_{\mathrm{CCD}} = \sqrt{(DN - bias)/(gain) + (readnoise/gain)^2}$$

$$\sigma_{\mathrm{MAMA}} = max(1, \sqrt{counts})$$

Because MAMA data is not processed with **ACSCCD**, the **doNoise** function is later called as part of **ACS2D**.

The CCDTAB reference file, the CCD Characteristics Table, is used to initialize the ERR array by determining the bias, gain, and readnoise for each observation. The table contains one row for each amp configuration which can be used during readout. Each configuration is uniquely identified by the list of amps possible (CCDAMP), the particular chip being read out (CCDCHIP), the commanded gain (CCDGAIN), the commanded bias level

(CCDBIAS), and the pixel bin size (BINAXIS). CCDTAB uses these commanded values to determine each amp's physical read-out characteristics, including readnoise (READNSE), A-to-D gain (ATODGN), and bias level (CCDOFST). Note that by using the bias level from the CCDTAB, instead of from a full bias image, to create the ERR array, slight errors may be introduced which could be relevant for faint sources.

doDQI - Bad Pixel Determination

- Header Switch: DQICORR
- Header Keywords Updated: None
- Reference File: BPIXTAB (*_bpx.fits), CCDTAB (*_ccd.fits)

The function **doDQI** initializes the data quality (DQ) array by combining it with a table of known bad pixels for the detector, stored in the Bad-Pixel reference table (BPIXTAB). The type of bad pixel flags are listed in Table 3.4.

The DQ array may have already been partially populated to reflect pixels which were affected by telemetry problems or compression buffer overflow. Other DQ flags will be added in further processing steps (such as cosmic-ray rejection). For CCD data, the values in the SCI extension are also checked for saturation by comparison with the value of the SATURATE column in the CCD parameters table CCDTAB.

The **doDQI** function combines the DQ flags from pre-processing, from the BPIXTAB, and from saturation tests into a single result for the particular observation. These values are combined using a "bitwise/logical OR operator" for each pixel. Thus, if a single pixel is affected by two DQ flags, those flag values will be added in the final DQ array. This array then serves as a mask so that **CALACS** can ignore bad pixels during processing.

The BPIXTAB reference file maintains a record of the (x,y) position and DQ value for all known bad pixels in each CCD. These pixels may change as some hot pixels are annealed and others appear over time, for example. Permanently bad pixels due to chip defects may be flagged during Generic Conversion, and it is the job of the BPIXTAB to maintain the list of bad pixels applicable for a given time period.

Table 3.4: Flags for the DQ array

Flag Value	Definition
0	good pixel
1	Reed-Solomon decoding error
2	data replaced by fill value
4	bad detector pixel or beyond aperture
8	masked by aperture feature
16	new hot pixel
32	“blob” at end of hot column
64	pre-existing hot pixel
128	bias level pixel
256	saturation (full well or a-to-d)
512	bad pixel in reference file
1024	weak trap
2048	a-to-d saturation*
4096	*reserved*
8192	cosmic ray rejected during image combination

*Note: The ability to flag pixels affected by a-to-d situation as 2048 will be implemented in the pipeline in Spring 2004.

doAtoD - A-to-D Correction (skipped)

- Header Switch: ATODCORR
- Header Keywords Updated: None
- Reference File: ATODTAB (*_a2d.fits)

The **doAtoD** function converts pixel values in the raw input image from units of DN (16-bit integer) to counts (32-bit float). An analog-to-digital correction would be applied if the CCD electronic circuitry, which performs the analog-to-digital conversion, were biased toward the assignment of certain DN (data number) values. Ground test results show that this correction is not currently needed, so the ATODCORR switch will always be set to OMIT.

doBlev - Bias-level Correction

- Header Switch: BLEVCORR
- Header Keywords Updated: BIASLEV[A,B,C,D], MEANBLEV, CRPIX[1,2], LTV[1,2]
- Reference File: OSCNTAB (*_osc.fits)

The **doBlev** function fits the bias level from the overscan region and subtracts it from the image data. The definition of the overscan region is taken from the OSCNTAB reference file. With these regions defined, the overscan level for each row of the input image is measured and a straight line is fit as a function of image line number. The fit will then be evaluated for each row of the input image, and that value will be subtracted from each pixel of that row. If the measured overscan level for a given row is greater than 3 sigma times the mean of all rows, that row will be excluded from the linear fit. If the overscan level cannot be determined, or if the overscan region is not present in the input image, a default value (the CCDBIAS from the CCD parameters table) will be subtracted instead. In this case, an error message will be written to the image trailer file.

In **CALSTIS**, the overscan regions were removed from the image just after the bias-level subtraction, since no subsequent processing was required. **CALACS**, on the other hand, retains the output image in memory until the completion of all the processing steps in **ACSCCD**. The overscan regions will not be trimmed until the image is written to disk at the completion of **ACSCCD**.

The positional keywords CRPIX* and LTV* will be updated to reflect the offset due to removing the overscan. In addition, the mean value of all overscan levels will be computed and written to the SCI extension header as MEANBLEV. The individual bias level measured for each amplifier will be written to the SCI header as BIASLEV*.

The OSCNTAB reference file (Overscan Region Table) describes the overscan regions to be used for measuring the bias level of the observation. Each row corresponds to a specific configuration, given by the CCD amp and chip used. The columns TRIMX* give the number of columns to trim off the beginning and end of each line (the physical overscan region), while the TRIMY* columns give the number of rows to trim off the top and bottom of each column (the virtual overscan region). The result of trimming (TRIMX1 + TRIMX2) columns and (TRIMY1 + TRIMY2) rows gives the final calibrated image sizes, 4096x2048 for a full WFC image and 1024x1024 for a full HRC image. The OSCNTAB columns BIASSECTA* and BIASSECTB* give the range of image columns to be used for determining the bias level in the leading and trailing regions, respectively, of the physical overscan region. The virtual overscan region is also defined in the OSCNTAB, where (VX1, VY1) indicates the beginning of the upper or lower overscan region, extending to the position (VX2, VY2).

To determine which overscan regions were actually used for determining the bias level, users are encouraged to check the OSCNTAB reference file. If desired for manual calibration, users may modify the bias section and virtual overscan region definitions in the reference table, but the TRIMX*, TRIMY* columns must not be changed.

doBias - Bias Image Subtraction

- Header Switch: BIASCORR
- Header Keywords Updated: None
- Reference File: BIASFILE (*_bia.fits)

Subtraction of the bias image is performed prior to cosmic-ray rejection using the function **doBias**. The appropriate reference image, BIASFILE, is selected using the DETECTOR, CCDAMP, and CCDGAIN keywords. The dimensions of the science image are used to distinguish observations which use the entire chip from sub-array images. Because the bias image has already been scaled by the gain and had the overscan values subtracted, the bias image may have a mean pixel value which is less than one.

Due to the way the bias reference image is created, part of the dark subtraction is also included in this step. Dark counts accumulate for an additional time beyond the exposure time, primarily the time required to read out the detector, and this portion of the dark current is subtracted along with the bias. This is described further in the section on Dark Image Subtraction.

The bias reference files are updated frequently (~weekly). The BIASFILE has the same dimensions as a full-size science image, 1062x1044 for HRC and 4144x2068 for WFC, allowing for simple image arithmetic on arrays of equal size. Only after the completion of **ACSCCD** are the images trimmed to 1024x1024 (HRC) and 4096x2048 (WFC). **CALACS** assumes that CCD data will not be read-out in binned mode, so any image which is not the full size is assumed to be a sub-array.

For sub-array images, **CALACS** uses the LTV[1,2] keywords to extract the appropriate region from the reference file and apply it to the sub-array input image. For users who take their own bias files, the BIASFILE keyword will need to be updated manually before recalibrating. This is described in detail in Section 3.5.2.

doFlash - Post-Flash Subtraction

- Header Switch: FLSHCORR
- Header Keywords Updated: MEANFLSH
- Reference File: FLSHFILE (*_fls.fits)

ACS has a post-flash capability to provide the means of mitigating the effects of Charge Transfer Efficiency degradation. The proposer controls whether to use this capability via the FLASHEXP optional parameter in the Phase II proposal.

The function **doFlash** will subtract the Post-Flash reference file, FLSHFILE, from the science image. This file has the same dimensions as a full-size science image, 1062x1044 for HRC and 4144x2068 for WFC, allowing for simple image arithmetic on arrays of equal size. The appropriate FLSHFILE is selected using the following keywords from the image header: DETECTOR, CCDAMP, CCDGAIN, FLASHCUR and SHUTRPOS.

First, the success of the Post-Flash exposure is verified by checking the keyword FLASHSTA. If any problems were encountered, a comment will be added to the history comments in the SCI extension header. The FLSHFILE is then normalized to one second for the appropriate post-flash current level (LOW, MED, HIGH), given by the FLASHCUR keyword.

The FLSHFILE is then multiplied by the flash duration in seconds (FLASHDUR) and is subtracted from the science image. Finally, the mean value of the scaled Post-Flash image is written to the output SCI extension header as the keyword MEANFLSH.

Final Output

Upon completion of **ACSCCD**, the overscan regions will be trimmed from the image when it is written out, but only if **doBlev** is performed successfully. Otherwise, the full image array will be written out.

3.4.2 ACSREJ

- Header Switch: CRCORR
- Header Keywords Updated: TEXPTIME, SKYSUM, EXPEND, REJ_RATE, EXPTIME, NCOMBINE, ROOTNAME
- Header Keywords added as 'HISTORY' Comments: INITGUES, SKYSUB, CRSIGMAS, MEANEXP, CRRADIUS, CRTHRESH, SCALENSE, CRMASK, NEXTEND
- Reference File: CRREJTAB (*_crr.fits)

ACSREJ, the cosmic-ray rejection task in **CALACS**, combines CR-SPLIT exposures into a single image, first detecting and then replacing flagged pixels. The task uses the same statistical detection algorithm developed for STIS data (**OCRREJ**) and WFPC2 data (**CRREJ**), providing a well-tested and robust procedure.

First, **ACSREJ** will compute the sky background using the mode of each image. Sky subtraction is performed before any statistical checks are made for cosmic rays. Next, **ACSREJ** constructs an initial comparison image from each input exposure. The comparison image can either be a median- or minimum-value sky-subtracted image constructed from all the input images and represents the 'first guess' of a cosmic-ray free image. The comparison image serves as the basis for determining the statistical deviation of each pixel from the input image.

A detection threshold is then calculated for each pixel based on the comparison image. This threshold is equal to a constant times sigma squared, given in the equation below:

$$\tau_n = \sigma^2 \times (noise + value(x,y)/gain + scale(value(x,y) - sky_n)^2)/T_n^{\;2}$$

where:

- σ is the sigma value used as the detection limit,
- *noise* is the readnoise squared (in DN) and *gain* is the CCD gain (in e$^-$/DN) for the amp used to read the pixel,
- *scale* is the scale factor for the noise model,
- T_n is the exposure time (in seconds) for the input image, and
- *value* is the pixel value (in DN) from the median or minimum combined comparison image.

The actual cosmic-ray detection criteria at each pixel is determined as:

$$\Delta = ((pix_n(x,y) - sky_n)/T_n - median(x,y))^2$$

where:

- $pix_n(x,y)$ is the pixel value (in DN) from input image *n,*
- *sky*$_n$ is the sky background (in DN) of image *n*, and
- $median_n(x,y)$ is the median or minimum pixel value (in DN/sec) from the comparison image.

If $\Delta > \tau_n$, the pixel is flagged as a cosmic-ray in the input image's DQ array and is ignored when images are summed together. Surrounding pixels within a given expansion radius are marked as 'SPILL' pixels and are given less stringent detection thresholds.

When all input images have been processed, the values of the acceptable pixels are summed over all input images. Each pixel in the summed output array is then scaled by the total exposure time:

$$pixout(x,y) = T \cdot \frac{\Sigma_n (pix_n(x,y) - sky_n) m_n(x,y)}{\Sigma_n T_n m_n(x,y)} + \Sigma_n sky_n$$

where:

- T_n is the exposure time for image *n*,
- $m_n(x,y)$ is the mask value (0 for cr-rejected pixels, 1 for good data) for the *n*'th image at pixel (*x, y*),
- *T* is the total exposure time (regardless of whether all input images were used for that particular pixel). This corresponds to the header keywords TEXPTIME, EXPTIME.

The following keywords are derived from the variables in this computation:

- TEXPTIME = EXPTIME = T
- SKYSUM = $\Sigma_n sky_n$
- REJ_RATE = $(\Sigma_n T_n m_n(x,y)/T)$ averaged over all pixels
- NCOMBINE = *n*

The remaining keywords EXPSTART, EXPEND are updated based on the input image headers.

In summary, the cosmic ray rejection task sums all accepted pixel values, computes the true exposure time for that pixel, and scales the sum to correspond to the total exposure time. The final scaled, cleaned pixel is written to the comparison image to be used for the next iteration. This process is then repeated with successively less stringent detection thresholds, as specified by CRSIGMAS. Further processing by **CALACS** will scale the *pixout(x,y)* array by the gain, resulting in the summed, cosmic-ray eliminated, but not sky-subtracted product (*_crj.fits) in units of *electrons*.

Cosmic-ray Rejection Table

ACSREJ uses the Cosmic-ray Rejection Parameter Table (CRREJTAB) to determine the number of iterations for cosmic-ray rejection, the sigma levels to use for each iteration, and the spill radius to use during detection. This allows the rejection process to be tuned to each detector, with suitable defaults being applied during pipeline processing. Observers may fine-tune the cosmic-ray rejection parameters when manually reprocessing data with **ACSREJ** by editing the CRREJTAB.

The CRREJTAB reference file contains the basic parameters necessary for performing cosmic-ray rejection. The column names and default values for the CRREJTAB are given in Table 3.5. The appropriate row is selected based on the chip being processed (CCDCHIP), the number of images into which the exposure was split (CR-SPLIT), and the exposure time of each CR-SPLIT image (MEANEXP). The sky fitting algorithm is controlled by the parameter SKYSUB which can have values of 'mode' or 'none'. The 'first guess' CR-combined image is then created using the median or minimum value of the input exposures, as specified by the column INITGUES.

Table 3.5: Columns in Cosmic-Ray Rejection Parameters Table

Column Name	Default Value	Contents
CRSPLIT		Number of exposures into which observation was split
MEANEXP	INDEF	Average exposure time (sec) for each image
SCALENSE	30.0	Multiplicative term (in percent) for the noise model
INITGUES	minimum	Method for computing initial-guess image (minimum, median)
SKYSUB	mode	Sky fitting algorithm (mode, none)
CRSIGMAS	6.5, 5.5, 4.5	Rejection thresholds (sigma) for consecutive iterations
CRRADIUS	2.1	Radius (in pixels) for propagating cosmic ray
CRTHRESH	0.5555	Propagation factor
BADINPDQ	39	Data quality file bits to reject
CRMASK	yes	Flag CR-rejected pixels in input files?
CCDCHIP		Chip to which this conversion applies

Cosmic-ray detection requires the specification of a threshold above which a pixel value is considered a cosmic-ray. This threshold is defined above as $\tau_n = \sigma^2 \times constant$ and uses the sigma rejection thresholds σ. These sigmas correspond to the column CRSIGMAS in the CRREJTAB reference file. SCALENSE is a multiplicative term (in percent) for the noise model and is given as *scale* in the threshold equation. This term can be useful when the pointing of the telescope has changed by a small fraction of a pixel between images. Under such circumstances, the undersampling of the image by the CCD will cause stars to be rejected as cosmic rays if a scale noise term is not included. This is a crude but effective step taken to satisfy the maxim of "first do no harm". However, for cases in which there have been no frame-to-frame offsets or the image is locally well-sampled, this will unduly bias *against* rejecting cosmic rays.

Pixels within a given radius, CRRADIUS, of a cosmic-ray will also be treated as cosmic-rays. A less stringent rejection threshold, CRTHRESH, can be used for detecting pixels adjacent to a cosmic-ray. As for CRSIGMAS, CRTHRESH is also given as a sigma value. If CRTHRESH is exceeded, pixels within a defined radius of the cosmic ray will also be flagged. All pixels determined to be affected by a cosmic-ray will have their DQ values set to 8192, as described in Table 3.4.

A much more detailed discussion of cosmic ray elimination in repeated ACS exposures may be found in ACS ISR 2002-08, Section 5.1. Of particular note is the appropriate value of SCALENSE to use; the pipeline adopts a conservative value to avoid doing harm. In recalibrating several frames for a new cosmic ray elimination, it would be advisable to determine the full range of relative x,y offsets. An appropriate value of SCALENSE is 100*(maximum offset in pixels), thus if the full offset range was 0.1 pixels an appropriate SCALENSE is 10.0. To alter this value, one may use **tedit** on the CRREJTAB file (calibration file with the extension 'crr.fits'). The number of exposures obtained in a repeated observation set can be larger than the maximum CR-SPLIT=8 allowed in Phase II proposals. For example, for program 9662 used in Examples 3 and 4 in Section 3.5.2, there were 14 individual 1.0s HRC exposures all at nominally the same pointing. A check of relative offsets shows that a shift of about 0.2 pixels occurred between the first and last exposures. To obtain a crj combined image of all 14 inputs at 1.0s and with better sensitivity to cosmic ray elimination than provided by the conservative default of SCALENSE = 30.0, the following should be done:

1. Create a new asn.fits table with one EXP-CRJ entry for each of the 14 raw 1.0s images, and an appropriately named PROD-CRJ file line.
2. Use **tedit** on the crr.fits table adding a line with a CR-SPLIT value of 14, and a SCALENSE value of 20.0.
3. Given such a large number of inputs it would also make sense here to change to 'median' as the INITGUES value. Then rerun **CALACS** on the new association table with 14 entries to obtain a CRJ extension image based on the full stack of 1.0s images.

For a detailed discussion on manually recalibrating ACS data, refer to Section 3.5.

3.4.3 ACS2D

Every observation, whether taken with the MAMA or CCD detectors, will be processed by **ACS2D**. The **ACS2D** primary functions are listed in Table 3.6 and include the dark current subtraction, flat fielding, and photometric keyword calculations. **ACS2D** contains the same data quality and error array initialization functions used in **ACSCCD**, but **ACS2D** will check to ensure that the array initialization is not performed twice on CCD data. Calibration switches in the image header control the performance of the remaining calibration functions, with MAMA-specific functions being initiated only when the relevant calibration switches are set.

Table 3.6: The functions performed in **ACS2D** (in operational order)

doNoise	Apply a simple noise model (if not done in ACSCCD)
doDQI	Initialize data quality array (if not done in ACSCCD)
doNonLin	Correct and flag non-linear data **(MAMA only)**
doDark	Subtract dark image
doFlat	Divide by flat field and multiply by gain
doShad	Perform CCD shutter shading correction (currently skipped)
doPhot	Compute photometric keyword values for header
doStat	Compute image statistics

doNoise - Error Array Initialization

- Header Switch: None
- Header Keywords Updated: None
- Reference File: None

ACS2D will first check that the image error array has not been expanded, indicating that no previous processing has been performed. In this case, the value of all pixels in the ERR array will have a value of zero. **ACS2D** will then perform the same initialization as described in "doNoise - Error Array Initialization" for **ACSCCD**. If, however, the input image has already been processed, no changes to the ERR array will be made.

doDQI - Bad Pixel Determination

- Header Switch: DQICORR
- Header Keywords Updated: None
- Reference File: BPIXTAB (*_bpx.fits)

If the DQICORR header keyword switch is set to COMPLETE (e.g. CCD data), this step will be skipped. Otherwise, the same initialization will be performed as described in "doDQI - Bad Pixel Determination" for **ACSCCD**.

doNonLin - Linearity Correction for MAMA Data

- Header Switch: LFLGCORR, GLINCORR
- Header Keywords Updated: GLOBLIM
- Reference File: MLINTAB (*_lin.fits)

This routine is capable of flagging global and local nonlinearity in ACS MAMA observations, where the term global refers to the entire ACS detector and local refers to an individual detector pixel. The MAMA Linearity Table, MLINTAB, provides the basic parameters for determining linearity. The global limit (GLOBAL_LIMIT) column from this table refers to the total count rate at which the data are affected by greater than 10% non-linearity across the detector.

CALACS will attempt to correct for non-linearity up to the global limit using the non-linearity time constant in the column TAU. The global linearity correction is computed for every pixel below the global linearity limit specified by iteratively solving the equation $n = Ne^{(-\tau N)}$ to get the true count rate N.

The LOCAL_LIMIT can actually be much higher than the global limit and is difficult to correct using a simple algorithm. Each pixel found to exceed this limit will simply be marked as non-linear in the DQ file. This DQ flag will be extended by a fixed radius from the original pixel, given in the EXPAND column and is currently set to 2 pixels.

If the LFLGCORR switch is set to PERFORM, **ACS2D** will *flag* excessive global and local nonlinearity in the DQ array. If GLINCORR is set to PERFORM, it will *correct* excessive global nonlinearity in the SCI array, if it is not too large. If the global linearity limit is exceeded, the keyword GLOBLIM in the SCI extension header will be set to "EXCEEDED". Otherwise, it will have the value "NOT-EXCEEDED".

doDark - Dark Image Subtraction

- Header Switch: DARKCORR
- Header Keywords Updated: MEANDARK
- Reference File: DARKFILE (*_drk.fits)

This task is responsible for subtracting the dark image from the input image. For CCD data, the dark image (in *electron* s^{-1}) is multiplied by the exposure time and then divided by the gain before subtracting. (MAMA data do not require a gain correction.) The dark reference file, DARKFILE, is read in line-by-line and subtracted from the input image in memory. The mean dark value is then computed from the scaled dark image and used to update the MEANDARK keyword in the SCI image header. The "dark time" is simply the exposure time; it does NOT include the idle time since the last flushing of the chip or the readout time. Any dark accumulation during readout time or post-flash idle will be included automatically into the BIASFILE.

A unique dark reference file is created (approximately) daily using four 1000 sec frames for both the WFC and HRC. The four darks are combined (with cosmic ray rejection) and bias-corrected to form a "daydark". Every two weeks, a high signal-to-noise "basedark" is created from all the frames (~56) in that period. It does not contain the newest hot pixels present in the daydarks since they are rejected with the cosmic rays during image combination. A copy of the basedark is made for each day in the two-week period, and daily hot pixels are skimmed from the respective daydarks and added to the basedark copies. This produces a reference file with high signal-to-noise, which accurately reflects (and corrects/flags) the hot pixels present for a given observation date. The "best" dark file is typically not available in the pipeline until 2-3 weeks after the date of observation, because it takes a few weeks to collect enough frames to make a basedark.

The reference file for dark subtraction, DARKFILE, is selected based on the values of the keywords DETECTOR, CCDAMP, and CCDGAIN in the image header. The dark correction is applied after the overscan regions are trimmed from the input science image. As for the BIASFILE, **CALACS** assumes that the images have not been binned, so any input image smaller than the full detector size will be interpreted as a sub-array image.

Sub-array science images will use the same reference file as a full-sized image. **CALACS** simply extracts the appropriate region from the reference file and applies it to the sub-array input image.

doFlat - Flat field image(s) Correction

- Header Switch: FLATCORR
- Header Keywords Updated: None
- Reference File: PFLTFILE (*_pfl.fits), LFLTFILE (*_lfl.fits), DFLTFILE (*_dfl.fits)

As done for STIS, the ACS **doFlat** routine corrects for pixel-to-pixel and large-scale sensitivity gradients across the detector by dividing the data by the flat field image. This contrasts with WFPC2 and NICMOS who deliver an “inverse” flat to the pipeline and then multiply the data by the flat field. When performing the flat field correction, **CALACS** multiplies by the gain so that the calibrated data will be in units of electrons.

Because of geometric distortion effects, the area of the sky seen by a given pixel is not constant, and therefore observations of a constant surface brightness object will have count rates per pixel that vary over the detector, even if every pixel has the same sensitivity. In order to produce images that appear uniform for uniform illumination, the flat fields make an implicit correction for the geometric distortion across the field that is equivalent to dividing each pixel by the optical distortion which is normalized to unity at the center of the field. A consequence of this procedure is that two stars of equal brightness do not have the same total counts after the flat fielding step. Thus, point source photometry extracted from a flat fielded image must be multiplied by the effective pixel area map, as shown in Figure 4.5. This correction is accounted for in pipeline processing by **PyDrizzle** which uses the geometric distortion solution to correct all pixels to equal areas. Thus, in the drizzled images photometry is correct for both point and extended sources.

The flat field image used to correct the data is created using up to three flat field reference files: the pixel-to-pixel file (PFLTFILE), the low-order flat (LFLTFILE), and the delta flat (DFLTFILE). The PFLTFILE is a pixel-to-pixel flat field correction file containing the small scale flat field variations. The LFLTFILE is a low-order flat which will correct for any large-scale flat field variations across each detector. This file is stored as a binned image which is expanded when being applied by **CALACS.** The DFLTFILE is a delta-flat containing any needed changes to the small-scale PFLTFILE.

If the LFLTFILE and DFLTFILE are not specified in the SCI header, only the PFLTFILE will be used for the flat field correction. If all three reference files are specified, they will be read in line-by-line and multiplied together to form a complete flat field correction image. Currently, the LFLTFILE and DFLTFILE flats are not used for ACS data. The PFLTFILE reference flat in the pipeline is actually a combination of the pixel-to-pixel flats taken during the ground calibration and the low-order flat correction derived in-flight.

All flat field reference images will be chosen based on the detector, amplifier, and filters used for the observation. Any sub-array science image will use the same reference file as a full-size image. **CALACS** will extract the appropriate region from the reference file and apply it to the sub-array input image.

doShad - Shutter Shading File Correction (skipped)

- Header Switch: SHADCORR
- Header Keywords Updated: None
- Reference File: SHADFILE (*_shd.fits)

The SHADCORR calibration switch is currently set to OMIT and is unlikely to be used in future versions of **CALACS**. Calibration data show that a shading correction is not needed for ACS data. For more information, refer to ACS ISR 03-03, “Stability and Accuracy of HRC and WFC Shutters” and to the report “WFC and HRC Shutter Shading and Accuracy” on the ACS Science and Engineering Team web site at:

`http://acs.pha.jhu.edu/instrument/calibration/results/by_item/shutter/shading_accuracy_feb2001`

The **doShad** routine applies the shutter shading correction image (SHADFILE) to the science data if SHADCORR is set to PERFORM. The input image is corrected for the differential exposure time across the detector caused by the shutter opening. This correction only has a significant effect on images with short exposure times (less than 10 seconds).

Pixels are corrected based on the exposure time using the relation:

$$\text{corrected} = \text{uncorrected} \times EXPTIME/(EXPTIME + SHADFILE) .$$

The SHADFILE is selected using the DETECTOR keyword in the input science image. At present, this reference file is an over scanned-trimmed image, binned by a factor of 8 for each dimension, resulting in a 128x128 HRC and a 256x512 WFC SHADFILE.

The shutter shading correction can either be applied during **ACS2D** processing for single or `REPEAT-OBS` exposures or during cosmic-ray rejection in **ACSREJ** for `CRSPLIT` exposures.

doPhot - Photometry Keyword Calculation

- Header Switch: PHOTCORR
- Header Keywords Updated: PHOTMODE, OBSMODE, PHOTFLAM, PHOTZPT, PHOTPLAM, PHOTBW
- Reference Files: GRAPHTAB (*_tmg.fits), COMPTAB (*_tmc.fits)

Before photometry can be performed on ACS observations, a transformation from electrons to absolute flux units must be performed. **CALACS** follows the WFPC2 methodology for calculating the photometry keywords in the calibration pipeline. The calibration reference files, GRAPHTAB and COMPTAB, point to the SYNPHOT tables containing the latest ACS component tables. These tables contain the throughput as a function of wavelength for the various ACS detector and filter combinations. This reliance on SYNPHOT allows the ACS team to maintain the latest throughput files in SYNPHOT to keep **CALACS** up to date. For further discussion of SYNPHOT, refer to Chapter 3 of the HST Data Handbook Introduction.

The keywords PHOTMODE and OBSMODE are built to reflect the configuration of the instrument during the exposure. **CALACS** then uses the OBSMODE string with SYNPHOT to compute the total throughput array for the observation, based on the aperture and filter throughputs, and the photometry keywords: PHOTFLAM (the inverse sensitivity in units of $erg\ s^{-1}\ cm^{-2}\ A^{-1}$ for 1 $electron\ s^{-1}$), PHOTZPT (the Space Telescope magnitude zero point), PHOTPLAM (the bandpass pivot wavelength), and PHOTBW (the bandpass RMS width). Users who wish to convert calibrated images (in units of *electrons*) to flux units may simply divide the image by the exposure time and then multiply by the PHOTFLAM keyword. Drizzled (_drz) images are already in units of *electrons per second* and may simply be multiplied by the PHOTFLAM value to attain flux units.

doStat - Image Statistics Determination

- Header Switch: STATFLAG
- Header Keywords Updated: NGOODPIX, GOODMIN, GOODMAX, GOODMEAN, SNRMIN, SNRMAX, SNRMEAN
- Reference File: None

This routine computes the number of pixels which are flagged as “good” in the data quality array. The minimum, mean, and maximum pixel values are then calculated for data flagged as “good” in both the science and error arrays. Similarly, the minimum, mean, and maximum signal-to-noise of “good” pixels is derived for the science array. These quantities are updated in the image header.

3.4.4 ACSSUM

- Header Switch: RPTCORR
- Header Keywords Updated: NCOMBINE, EXPTIME, EXPEND, ROOTNAME
- Reference File: None

ACSSUM sums a list of images one line at a time, using the association file to define the input. A straight pixel-to-pixel addition of the science values is applied, and the error calculated as the square root of the sum of the squares of the errors in the individual exposures. Note that for multiple exposures taken via REPEAT-OBS instead of the observationally equivalent CR-SPLIT, the final processed image is *not* cosmic-ray rejected.

The calibration switch RPTCORR is set to COMPLETE upon successful completion of the summation. In addition, the keywords NCOMBINE, EXPTIME, and EXPEND are adjusted to reflect the total of the summed images.

3.5 Manual Recalibration of ACS Data

3.5.1 Requirements for Manual Recalibration

Retrieving Software and Input Data Files

If the observer decides to recalibrate their ACS data, the following must be available on their system:

- **CALACS** software from the STSDAS **hst_calib.acs** package
- **PyDrizzle** task from the STSDAS **dither** package
- PyRAF, to run **PyDrizzle**, obtained from STScI
- Reference files obtained from STScI Archive, or personal versions
- Uncalibrated data (*_raw.fits) from OTFR
- Association table, if applicable.

Uncalibrated data and calibration reference files can be obtained from OTFR via StarView or the web-based archive request form:

`http://archive.stsci.edu/cgi-bin/hst`.

StarView is available for users to download from:

`http://starview.stsci.edu/html`.

The pages include instructions on installing and using StarView and on how to become a registered archive user.

The most recent version of STSDAS, which includes the **CALACS** package required for reprocessing, can be downloaded from:

http://stsdas.stsci.edu/.

The most recent versions of PyRAF, **PyDrizzle**, and **MultiDrizzle** can be downloaded from:

http://stsdas.stsci.edu/pydrizzle.

Setting up ‘jref’

Before any recalibration may be done, the user’s local directory containing the calibration reference files must be defined. For ACS, this directory is referred to as ‘jref’. The image header already contains the appropriate keywords defining the reference files. The user must simply define the location of the ‘jref’ directory in the Unix environment.

```
setenv jref /mydisk/myjref/
```

Once this path is set, IRAF must be restarted in the same window in which the setup was performed.

Selecting Calibration Switches

OTFR will use the latest available calibration reference files by default. In order to use non-default reference files, manual recalibration is required. The calibration reference file keywords will need to be updated manually in the uncalibrated data with the non-default filenames before running **CALACS**. The selection criteria in Table 3.7 are used to set the values for the calibration switch header keywords in uncalibrated ACS data.

Table 3.7: Calibration Switch Selection Criteria.

Switch	Description	Criteria
DQICORR	Data Quality Array Initialization	DEFAULT = **“PERFORM”** If OBSMODE = ACQ then **“OMIT”**
ATODCORR	Analog to Digital Conversion	DEFAULT = **“OMIT”**
BLEVCORR	Overscan Region Subtraction	DEFAULT = **“PERFORM”**
BIASCORR	Bias Subtraction	DEFAULT = **“PERFORM”**
FLSHCORR	Post Flash Subtraction	DEFAULT = **“OMIT”**
CRCORR	Cosmic Ray Rejection	If CRSPLIT >= 2 then **“PERFORM”** If CRSPLIT < 2 then **“OMIT”**
EXPSCORR	Calibrate individual exposures in an association	DEFAULT = **“PERFORM”**
SHADCORR	Shutter Shading Correction	DEFAULT = **“OMIT”**
DARKCORR	Dark Subtraction	DEFAULT = **“PERFORM”**
FLATCORR	Flat field Division	DEFAULT = **“PERFORM”**
PHOTCORR	Photometric Processing	DEFAULT = **“PERFORM”**
RPTCORR	Repeat Processing	DEFAULT = **“OMIT”** If NRPTEXP > 1 then **“PERFORM”**
DRIZCORR	Dither Processing	DEFAULT = **“PERFORM”**
GLINCORR	Global Non-Linearity Correction	SBC DEFAULT = **“PERFORM”**
LFLGCORR	Local and Global Non-Linearity Flagging in DQ Array	SBC DEFAULT = **“PERFORM”**

Bypassing the PHOTCORR Step

Raw ACS images are in units of *Data Numbers (DN)*, calibrated images are in units of *electrons*, and drizzled images are in units of *electrons/second*. The pipeline calibration tasks do not alter the units of the pixels in the image. Instead they calculate and write the inverse sensitivity conversion factor (PHOTFLAM) and the ST magnitude scale zero point (PHOTZPT) into header keywords in the calibrated data. Refer to the section "doPhot - Photometry Keyword Calculation" in Section 3.4.3 for more information.

When populating the photometric keywords during the PHOTCORR step, **CALACS** uses the STSDAS synthetic photometry package, **SYNPHOT**, which contains the throughput curves of all HST optical components. (For information on retrieving the **SYNPHOT** dataset, see Appendix A.3.2 of the HST Data Handbook Introduction.) The **SYNPHOT** dataset contains numerous files which may be updated on a regular basis, making it cumbersome for the user to set up. A simple alternative is to set the PHOTCORR calibration switch to OMIT in the primary header of the uncalibrated image. The user may simply copy the photometric keywords from the calibrated data into the raw image header and then run **CALACS** with the PHOTCORR switch set to OMIT.

Speed of Pipeline Processing

Reprocessing HRC or SBC data will not put a burden on most computing systems since the image sizes, both science data and reference files, are relatively small. Processing WFC observations, on the other hand, will require more computing power, including both CPU run time and disk space.

Great care has been taken to minimize the memory requirements of the pipeline software to accommodate most computing configurations. Line-by-line I/O is used during pipeline processing and is particularly useful when more than one image is operated on at a time, for example during flat fielding or co-adding images. This, unfortunately, places an extra burden on the I/O capabilities of the computing system. **CALACS** requires up to 130MB of memory to process WFC data, while **PyDrizzle** requires up to 400MB.

An Ultra-10 class workstation (300 MHz, 512MB RAM) using **CALACS** to process WFC data will take ~13 minutes for a CR-SPLIT=2 association or ~4.5 minutes for a single WFC exposure. **PyDrizzle** requires ~5 minutes to process a single WFC exposure. These times can be scaled by the processor speed of the computing system, and result in much faster run times for newer workstations.

3.5.2 CALACS Examples

We present several examples of **CALACS** reprocessing. The steps required for general **PyDrizzle** reprocessing are summarized in the "PyDrizzle Examples" on page 4-25.

Example 1: Reprocessing a Single Exposure

The following example uses HRC data from the flat field calibration program 9019 which observed the stellar cluster 47 Tucanae. The observations are from visit 07, exposure 12 and utilize the F814W filter. The exposures are CR-SPLIT into two exposures of 20 sec each. The association table for this exposure is j8bt07020_asn.fits. Typing 'tprint j8bt07020_asn.fits' reveals the rootnames of the individual exposures: j8bt07oyq and j8bt07ozq.

For the purposes of this first example, let us assume that the observations are *not* part of an association. This example will illustrate the steps required to reprocess a single exposure after changing the bias reference file from the default value to a file specified by the user.

1. In the Unix environment, we must specify the location of the 'jref' directory where the calibration reference files reside and then start IRAF. We assume here that the user has requested calibration files as part of the data archive request and stored these in their directory '/mydisk/myjref/', where '/mydisk/' specifies the directory containing the pipeline data.

(Note: This setup must be done in the same window in which IRAF will be used. Note also that setting 'jref' from within IRAF will not work even though typing 'show jref' would suggest it might.)

```
unix> setenv jref /mydisk/myjref/
unix> cl
```

2. To see what bias reference file is currently in use, we use the hedit task. Note that the period at the end of the line is required. The lines below beginning with the 'iraf>' symbol indicate commands typed into IRAF. Lines with no symbol indicate output from IRAF.

```
iraf> hedit j8bt07oyq_raw.fits[0] BIASFILE .
j8bt07oyq_raw.fits[0], BIASFILE = jref$m4r1753rj_bia.fits
```

3. Next, we edit the global image header of the raw image to reflect the name of the new bias file.

```
iraf> hedit j8bt07oyq_raw.fits[0] BIASFILE jref$mybias.fits verify-
j8bt07oyq_raw.fits[0], BIASFILE: jref$m4r1753rj_bia.fits->
jref$mybias.fits
j8bt07oyq_raw.fits[0] updated
```

4. We then set the PHOTCORR processing step to ‘OMIT’ and copy the photometric keywords from the calibrated to the raw image. (Alternately, the user may keep track of these numbers in any other preferred manner. Most users will only require knowledge of the PHOTFLAM keyword for photmetric calibration.) These steps allow users to bypass this SYNPHOT-based calibration step, due to the cumbersome requirements for setup. (See “Bypassing the PHOTCORR Step” on page 3-26 for more information.)

```
iraf> hedit j8bt07oyq_raw.fits[0] PHOTCORR omit verify-
j8bt07oyq_raw.fits[0], PHOTCORR: PERFORM -> omit
j8bt07oyq_raw.fits[0] updated
iraf> hedit j8bt07oyq_flt.fits[1] phot* .
j8bt07oyq_flt.fits[1], PHOTMODE = "ACS HRC F814W"
j8bt07oyq_flt.fits[1], PHOTFLAM = 1.3260861E-19
j8bt07oyq_flt.fits[1], PHOTZPT = -2.1100000E+01
j8bt07oyq_flt.fits[1], PHOTPLAM = 8.1209805E+03
j8bt07oyq_flt.fits[1], PHOTBW = 7.0114880E+02
iraf> hedit j8bt07oyq_raw.fits[1] PHOTFLAM 1.3260861E-19 verify-
j8bt07oyq_raw.fits[1],PHOTFLAM: 0.000000E+00 -> 1.3260861E-19
j8bt07oyq_raw.fits[1] updated
```

5. Within the directory containing the pipeline products (“/mydisk/”), we create the subdirectory “recal”, copy the RAW and ASN files to this subdirectory, and ‘cd’ to this directory.

6. Finally, we load the packages **stsdas.hst_calib.acs** and run **CALACS** on the single raw exposure.

(Note that invoking this package will result in the message: "Warning: package ‘acs’ includes Python tasks that require PyRAF". This applies only for **calacsdriz** which calls the **CALACS** and **PyDrizzle** tasks in succession.)

```
iraf> stsdas.hst_calib.acs
iraf> calacs j8bt07oyq_raw.fits
```

The product will be a single calibrated image with the FLT extension.

Example 2: Reprocessing Multiple Exposures within an Association

This example uses the same data from Example 1 and illustrates the steps required to reprocess an ACS association after changing the bias reference file from the default value to a file specified by the user. The steps required are similar to the previous example, with a few modifications. IRAF output comments which are similar to Example 1 have been omitted.

1. First, we look at the contents of the image association.

```
iraf> tprint j8bt07020_asn.fits
# MEMNAME       MEMTYPE        MEMPRSNT
j8bt07oyq       EXP-CRJ        yes
j8bt07ozq       EXP-CRJ        yes
j8bt07021       PROD-CRJ       yes
```

2. To see what bias reference file is currently in use, we use **hedit**.

```
iraf> hedit j8bt07*raw.fits[0] BIASFILE .
```

3. Next, we edit the global image header of all the raw images to reflect the name of the new bias file.

```
iraf> hedit j8bt07*raw.fits[0] BIASFILE jref$mybias.fits verify-
```

4. We then set the PHOTCORR processing step to 'OMIT' and copy the photometric keywords from the calibrated to the raw image.

```
iraf> hedit j8bt07*raw.fits[0] PHOTCORR omit verify-
iraf> hedit j8bt07*flt.fits[1] phot* .
iraf> hedit j8bt07*raw.fits[1] PHOTFLAM 1.3260861E-19 verify-
```

5. Within the directory containing the pipeline products ("/mydisk/"), we create the subdirectory "recal", copy the RAW and ASN files to this subdirectory, and 'cd' to this directory.
6. Finally, we run **CALACS** on the image association.

```
iraf> stsdas.hst_calib.acs
iraf> calacs j8bt07020_asn.fits
```

(If executing this command in the same directory in which you have run the previous example, then one of the flt.fits files will already exist and **CALACS** will not overwrite existing images. To execute in this case, first delete the existing flt.fits files or move them to a separate directory.)

The product will be two separate calibrated images with the FLT extension (j8bt07oyq_flt.fits, j8bt07oyq_flt.fits) and a single cr-combined image with the CRJ extension (j8bt07021_crj.fits).

Example 3: Removing Cosmic-rays from Repeated (RPT-OBS) Exposures

The following example uses HRC data from the shutter stability calibration program 9662 which observed the stellar cluster 47 Tucanae. The observations are from visit 01, exposure 20 and utilize the CLEAR filter. The two repeated (RPT-OBS) exposures are 1.0 sec each.

In this example, we illustrate the steps required to remove cosmic-rays from repeated observations which use the RPT-OBS syntax. By definition, the standard calibration product of multiple repeated exposures is a summed image. We can fool the software into rejecting cosmic-rays from the images, but this requires modification of the image association table.

1. First, we look at the contents of the image association.

```
iraf> tprint j8is01020_asn.fits
# MEMNAME      MEMTYPE        MEMPRSNT
j8is01j2q      EXP-RPT        yes
j8is01j3q      EXP-RPT        yes
j8is01021      PROD-RPT       yes
```

2. Update calibration switches in the global header using the **hedit** task.

```
iraf> hedit *raw.fits[0] CRCORR perform add-
iraf> hedit *raw.fits[0] RPTCORR omit add-
```

3. Update association table MEMTYPE column, by changing each instance of RPT to CR. This may be done using the text editor functionality of **tedit**.

```
iraf> tedit j8is01020_asn.fits
# MEMNAME      MEMTYPE        MEMPRSNT
j8is01j2q      EXP-CRJ        yes
j8is01j3q      EXP-CRJ        yes
j8is01021      PROD-CRJ       yes
```

4. We then set the PHOTCORR processing step to ‘OMIT’ and copy the photometric keywords from the calibrated to the raw image.

```
iraf> hedit j8is01*raw.fits[0] PHOTCORR omit verify-
iraf> hedit j8is01*flt.fits[1] phot* .
iraf> hedit j8is01*raw.fits[1] PHOTFLAM 4.5803514E-20 verify-
```

5. Within the directory containing the pipeline products ("/mydisk/"), we create the subdirectory "recal", copy the RAW and ASN files to this subdirectory, and 'cd' to this directory.
6. Finally, we run **CALACS** on the image association.

```
iraf> stsdas.hst_calib.acs
iraf> calacs j8is01020_asn.fits
```

The product will be two calibrated images with the FLT extension (j8is01j2q_flt.fits, j8is01j3q_flt.fits) and a cr-combined image with the CRJ extension (j8is01021_crj.fits) instead of the usual SFL extension.

Example 4: Combining Exposures from Multiple Associations

This example uses the same data from Example 3 plus an additional exposure from the same program: visit 01, exposure 40. These additional data also use the CLEAR filter and contain two repeated 1.0 sec exposures. The association table for this exposure is j8is01040_asn.fits. In this example, we illustrate the steps required to combine two separate sets of repeated observations.

1. First we merge the two association tables and edit them appropriately (using **tmerge** and **tedit**, respectively).

```
iraf> tmerge j8is01020_asn.fits,j8is01040_asn.fits \
>>>   outtable="merged_asn.fits" option=append
iraf> tprint merged_asn.fits
# MEMNAME      MEMTYPE        MEMPRSNT
j8is01j2q      EXP-RPT         yes
j8is01j3q      EXP-RPT         yes
j8is01021      PROD-RPT        yes      <---remove this line
j8is01j8q      EXP-RPT         yes
j8is01j9q      EXP-RPT         yes
j8is01041      PROD-RPT        yes      <---edit this line
iraf> tedit merged_asn.fits
# MEMNAME      MEMTYPE        MEMPRSNT
j8is01j2q      EXP-RPT         yes
j8is01j3q      EXP-RPT         yes
j8is01j8q      EXP-RPT         yes
j8is01j9q      EXP-RPT         yes
j8is01xx1      PROD-RPT        yes      <---new product name
```

2. We then set the PHOTCORR processing step to ‘OMIT’ and copy the photometric keywords from the calibrated to the raw image using **hedit**.

```
iraf> hedit j8is01*raw.fits[0] PHOTCORR omit verify-
iraf> hedit j8is01*flt.fits[1] phot* .
iraf> hedit j8is01*raw.fits[1] PHOTFLAM 4.5803514E-20 verify-
```

3. Within the directory containing the pipeline products (“/mydisk/”), we create the subdirectory “recal”, copy the RAW and ASN files to this subdirectory, and ‘cd’ to this directory.
4. Finally, we run **CALACS** on the new image association.

```
iraf> stsdas.hst_calib.acs
iraf> calacs merged_asn.fits
```

The product will be 4 calibrated images with the FLT extension and a single summed image with the SFL extension (j8is01xx1_sfl.fits).

Example 5: Reprocessing Images taken as part of a Dither Pattern

The following example uses WFC data from the GOODS program 9425. These observations are from visit 54, exposure 219 and target the CDF-South with the F606W filter. The images are part of a 2-point line dither pattern and are 480 sec each. In this example, we illustrate the steps required to reprocess data which is part of a dither pattern, using a non-default dark reference file. The steps are similar to Example 2, but the format of the association and the data products are unique.

1. First, we look at the contents of the image association.

```
iraf> tprint j8e654010_asn.fits
# MEMNAME      MEMTYPE        MEMPRSNT
j8e654c0q      EXP-DTH        yes
j8e654c4q      EXP-DTH        yes
j8e654011      PROD-DTH       yes
```

2. To see what dark reference file is currently in use, we use **hedit**.

```
iraf> hedit j8e654*raw.fits[0] DARKFILE .
```

3. Next, we edit the global image header of all the raw images to reflect the name of the new dark reference file.

```
iraf> hedit j8e654*raw.fits[0] DARKFILE jref$mydark.fits verify-
```

4. We then set the PHOTCORR processing step to ‘OMIT’ and copy the photometric keywords from the calibrated to the raw image.

```
iraf> hedit j8e654*raw.fits[0] PHOTCORR omit verify-
iraf> hedit j8e654*flt.fits[1] phot* .
iraf> hedit j8e654*raw.fits[1] PHOTFLAM 7.7792272E-20 verify-
```

5. Within the directory containing the pipeline products (“/mydisk/”), we create the subdirectory “recal”, copy the RAW and ASN files to this subdirectory, and ‘cd’ to this directory.
6. Finally, we run **CALACS** on the image association.

```
iraf> stsdas.hst_calib.acs
iraf> calacs j8e654010_asn.fits
```

The product of the DITH-PATTERN will be two separate calibrated images with the FLT extension. In subsequent processing (see Chapter 4 for details), **PyDrizzle** will combine these FLT files into a single DRZ image (j8e654011_drz.fits).

CHAPTER 4:

Distortion Correction and Drizzling of ACS Images

In this chapter. . .

This chapter describes the geometric distortion of the ACS and presents the available tools and methods for removing the distortion and combining dithered datasets into clean, undistorted, and photometrically faithful output products.

The first section describes how the distortion has been quantified and how this information is held in reference files which are used by the calibration pipeline. Plots of the distortion field are given for the three ACS detectors.

The remaining sections give an overview of the drizzle method and discuss the higher level **PyDrizzle** and **MultiDrizzle** software tools available for distortion correction and image combination.

4.1 ACS Geometric Distortions

ACS produces the largest single images ever taken with HST. However, because the optics were designed with a minimum number of components, ACS focal surfaces are far from normal to the principal rays. This results in an image of the sky which is distorted in a large, but predictable manner. The distortion consists of two effects, discussed previously in Section 1.1.3. The first is the elongation of the ACS apertures, causing the pixel scale to be smaller along the radial direction of the Optical Telescope Assembly (OTA) field of view than along the tangential direction. The second effect is the variation of pixel area across the detector.

The results presented in the discussion below are derived from multiple observations of dense star fields with the images displaced from each other by amounts comparable with the size of the field of view. The following discussion is based on reports from the HST Calibration Workshop Proceedings: *Calibration of Geometric Distortion in the ACS Detectors* (Meurer et al. 2002) and *Astrometry with the Advanced Camera: PSFs and Distortion in the WFC and HRC* (Anderson 2002) at:

`http://www.stsci.edu/hst/HST_overview/documents/calworkshop/workshop2002/CW2002_TableOfContents`

4.1.1 On-orbit Calibration Program

The ACS geometric distortion campaign observed the core of 47 Tucanae with the WFC and HRC (program 9028, Meurer). The exposures were designed to detect stars on the main sequence turn-off, allowing a high density of stars with relatively short exposures. The F475W filter (Sloan *g*') was used to minimize the number of saturated red giant branch stars in the field. For calibrating the distortion in the SBC, exposures of NGC6681 (program 9027, Hartig) were chosen for the relatively high density of UV emitters (hot horizontal branch stars).

For the WFC and HRC pointings, the dither pattern was designed so that the offsets between all pairs of images adequately sample all spatial scales from about 5 pixels to 3/4 the detector size. For the SBC pointings, a more regular pattern is used, with a series of 5 pixel offsets.

Data obtained with F775W for the WFC and HRC detectors were used to check the wavelength dependence of the distortion, but little or no increase in the fit rms was found. While it was expected that the same distortion solution would be applicable to all filters except the polarizers, recent work has shown that at least one other optical filter (F814W) induces a significant plate scale change (factor of ~4x10^{-5}).

In the future, separate IDC tables will be available for each filter, and the **CALACS** pipeline will select the appropriate solution automatically. For now, the solution derived for F475W is used for all filters. Preliminary investigations reveal that filter-dependent changes in the distortion pattern exist at the level of 0.1 pixel.

4.1.2 Distortion Model

The heart of the distortion model relates detector pixel position (x, y) to sky position (see ACS ISR 2000-11) using a polynomial transformation given by:

$$x_c = \sum_{i=0}^{k} \sum_{j=0}^{i} a_{i,j}(x - x_r)^j (y - y_r)^{i-j}$$

$$y_c = \sum_{i=0}^{k} \sum_{j=0}^{i} b_{i,j}(x - x_r)^j (y - y_r)^{i-j}$$

where k is the polynomial order of the fit, (x_r, y_r) is the reference pixel position taken to be the center of each detector or WFC chip, and (x_c, y_c) is the undistorted position in arcseconds. The coefficients to the fits, $a_{i,j}$ and $b_{i,j}$, are free parameters. A similar form is used for the inverse relation which provides the conversion from units of arcseconds to distorted input pixels.

For the WFC, an offset is applied to put the two CCD chips on the same coordinate system:

$$X' = x_c + \Delta x(chip)\,,\quad Y' = y_c + \Delta y(chip)\,.$$

The offsets $\Delta x(chip), \Delta y(chip)$ are 0,0 for chip 1 and correspond to the separation between chips 1 and 2 for chip 2. The chip 2 offsets are free parameters in the fit. (X', Y') correspond to tangential plane positions in arcseconds which are tied to the HST $(V2, V3)$ coordinate system. Next the positions are corrected for velocity aberration: $X = \gamma X'$, $Y = \gamma Y'$, where

$$\gamma = \frac{1 + \boldsymbol{u} \cdot \boldsymbol{v}/c}{1 - (v/c)^2}\,,$$

u is the unit vector towards the target, and **v** is the velocity vector of the telescope (heliocentric plus orbital). Neglect of the velocity aberration correction can result in misalignments on the order of a pixel for WFC images taken six months apart for targets near the ecliptic. For a further discussion of the effect of velocity aberration, see *The Effect of Velocity Aberration Correction on ACS Image Processing*: HST Calibration Workshop Proceedings (Cox & Gilliland 2002) at:

`http://www.stsci.edu/hst/HST_overview/documents/calworkshop/workshop2002/CW2002_Papers/CW02_cox`

Finally, all frames are transformed to the same coordinate grid on the sky:

$$X_{sky} = \Delta X_i + \cos(\Delta\theta_i)X - \sin(\Delta\theta_i)Y\,,$$

$$Y_{sky} = \Delta Y_i + \sin(\Delta\theta_i)X + \cos(\Delta\theta_i)Y\,,$$

where the free parameters $\Delta X_i, \Delta Y_i, \Delta\theta_i$ are the position and rotation offsets of frame i.

4.1.3 Analysis

The positions of stars observed multiple times in the dithered star fields are used to iteratively solve for the free parameters in the distortion solution: fit coefficients $a_{i,j}$, $b_{i,j}$; chip 2 offsets Δx, Δy (WFC only); frame offsets ΔX_i, ΔY_i, $\Delta\theta_i$; and tangential plane position X_{sky}, Y_{sky} of each star used in the fit.

Originally, only images taken with a single roll angle were used to define the distortion solutions. The solution using only these data is degenerate in the zeroth (absolute pointing) and linear terms (scale, skewness). Therefore the largest commanded offsets with a given guide star pair were used to set the linear terms. However, comparison of corrected coordinates to astrometric positions showed that residual skewness in the solution remained. Hence, as of November 2002, the IDC tables for the WFC and SBC are based on data from multiple roll angles, incorporating data obtained from the calibration outsourcing program 9443 (King & Anderson). The overall plate scale is set by the largest commanded offset. For the HRC, the linear scale is set by matching HRC and WFC coordinates, since the same field was used in the SMOV observations. The zeroth order terms (position of ACS apertures in the V2,V3 frame) were determined from observations of an astrometric field.

4.1.4 Results

The most obvious component of the ACS distortion is a marked skew which results in the X/Y detector axes lying at ~85 degrees to each other on the sky. In addition, the distortion in all ACS detectors is highly non-linear (see Figure 4.1-4.4). A quartic fit (*k=4*) is adequate for characterizing the distortion to an accuracy much better than 0.2 pixels over the entire field of view.

WFC

The WFC detector is tilted at 22 degrees giving an elongation of 8% along the diagonal. The non-square projected aperture shape is evident from Figure 1.1 in Chapter 1, where the x axis is approximately in the V2 direction and the y axis is in the -V3 direction.

The WFC distortion is also illustrated in Figure 4.1, a vector displacement diagram which shows the contribution of the non-linear part of a quartic fit to the data. The vectors represent the degree of distortion to be expected in the WFC beyond the directional dependence of the plate scale. For display, the vectors are magnified by a factor of 5 compared to the scale of the x and y axes. The largest displacement indicated at the top left corner of the figure is ~82 pixels or about 4 arcseconds. While a quartic solution is adequate for most purposes, binned residual maps (Figure 4.4) show that there are significant coherent residuals in the WFC solutions which have amplitudes up to ~0.1 pixels. These residuals are not currently accounted for in pipeline calibration, although efforts are underway to do so in the future.

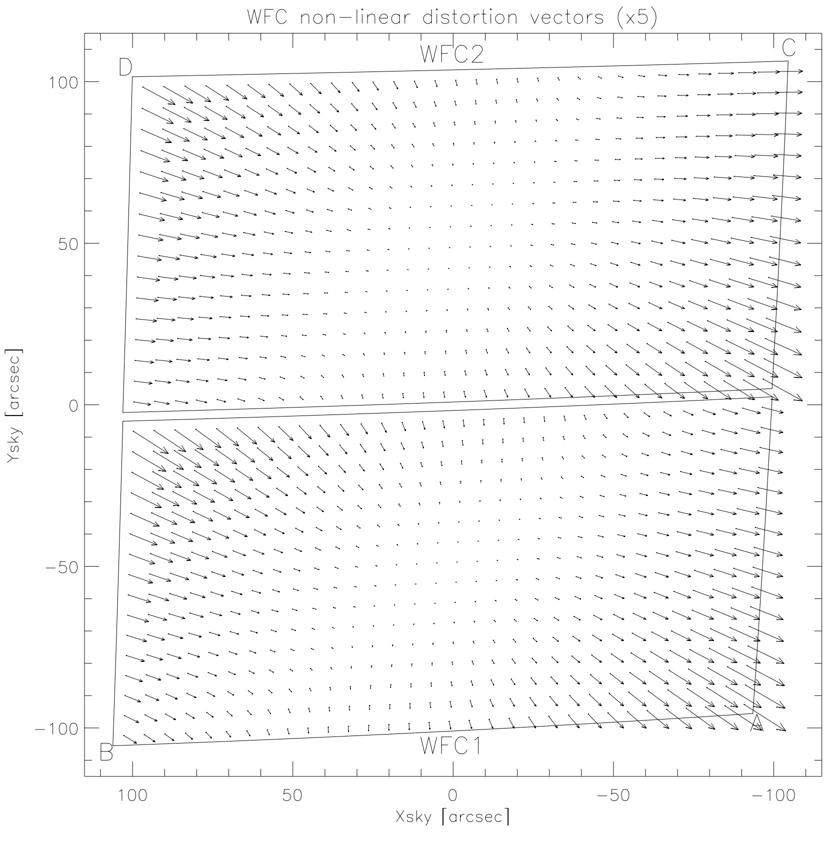
The pixel scales and array sizes are summarized in Table 4.1 for each ACS detector. The resulting variation of the projected pixel area on the sky requires corrections to the photometry of point sources in images which are still distorted. A contour plot of relative pixel area across the WFC, normalized to the central pixel, is shown in Figure 4.5. The range of area is from 0.89 to 1.08 times the central value.

When deriving the ACS flat fields, an implicit correction is made for the distortion which is equivalent to dividing each pixel by the effective pixel area in Figure 4.5. Thus, point source photometry extracted from a flat fielded image must be multiplied by the effective pixel area. This correction is accounted for in pipeline processing by **PyDrizzle** which uses the distortion solution to correct all pixels to equal areas.

Table 4.1: Pixel scale *(arcsec/pixel)* and array size (*pixels*) for each ACS detector.

Detector	Pixel Scale at Center X_{scale}, Y_{scale}	Average Pixel Scale X_{scale}, Y_{scale}	Array size (pixels)	Array size (arcsec)
WFC1	0.04927, 0.04864	0.04917, 0.04861	4096x2048	201"x100"
WFC2	0.04985, 0.05031	0.04976, 0.05029	4096x2048	204"x103"
HRC	0.02844, 0.02484	0.02843, 0.02484	1024x1024	29"x25"
SBC	0.03374, 0.03009	0.03377, 0.03010	1024x1024	35"x31"

Figure 4.1: Non-linear component of the ACS distortion for the WFC detector. Note that this figure is rotated 180 degrees with respect to the pipeline calibration products, where WFC2 is the lower half of the detector.



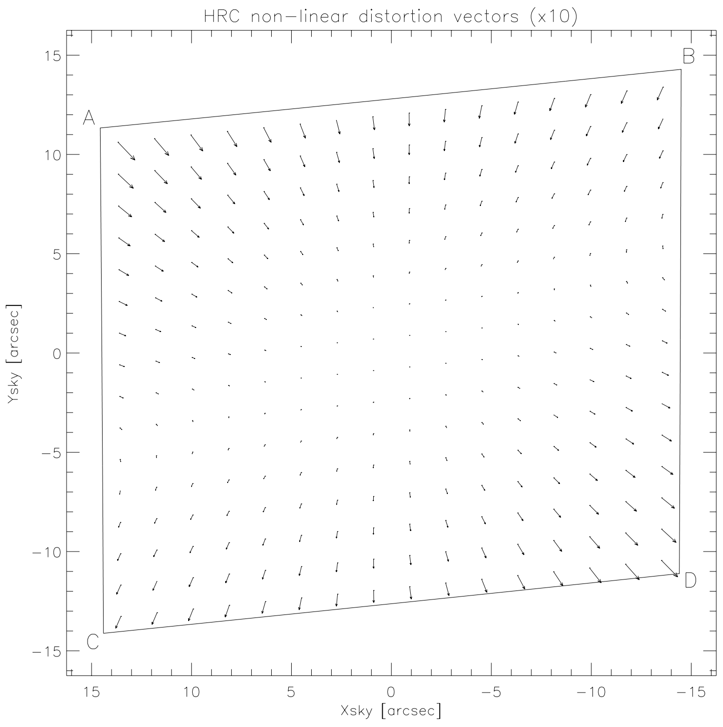
HRC

The High Resolution Channel has its edges aligned approximately along the V2 and V3 axes (see Figure 1.1). In this case, the center of the aperture lies on a line passing through the V2/V3 origin and making an angle of 22 degrees with the V3 axis. Because the aperture diagonal does not correspond to a radius of the HST field of view, the distortion has no particular symmetry with respect to the detector axes. Because the focal plane, and therefore the detector plane is 25 degrees away from the plane normal to the light path, the scales along the axes differ by ~14%. However, since the HRC is less than 30 arcsec across, the *total* variation in scale across the detector is much less than for the WFC, being only about 1%.

A vector plot of the deviation from linearity is given in Figure 4.2 in which the deviations have been magnified by a factor of 10 for illustrative purposes. The largest displacement is 4.9 pixels in the top left corner and corresponds to about 0.1 arcsec. While a quartic solution is adequate for most purposes, binned residual maps (Figure 4.4) show that there are significant coherent residuals in the HRC solutions which have amplitudes up to ~0.1 pixels. These residuals are not currently accounted for in pipeline calibration, although efforts are underway to do so in the future.

The pixel scales and array sizes are summarized in Table 4.1 for the HRC. The variation of pixel area across the detector is shown in Figure 4.5 and is required for photometric correction of point sources in distorted, uncorrected images. The maximum deviation from the central value is just over 2%.

Figure 4.2: Non-linear component of the ACS distortion for the HRC detector.



SBC

The Solar Blind Channel contains the MAMA detector. It is centered at approximately the same place as the HRC in the V2/V3 plane and is slightly larger, about 35 x 30 arcsec, versus 29 x 25 arcsec for the HRC. The average scales in the x and y directions are given in Table 4.1 and are calculated by matching observations of the same sky area by HRC and SBC.

The optical distortions are close to those displayed by the HRC. The only difference in their light paths is the presence of a plane M3 fold mirror which reflects light away from the SBC onto the HRC. Because there is one less mirror in the SBC light path, the distortion corrected image (in sky coordinates) in Figure 4.3 appears flipped with respect to the HRC image in Figure 4.2. The MAMA has a small amount of extra distortion in the detector itself, arising from irregularities in the multi-channel plate. The largest difference from a square pattern is 2 pixels, or 0.06 arcsec.

Figure 4.3: Non-linear component of the ACS distortion for the SBC detector.

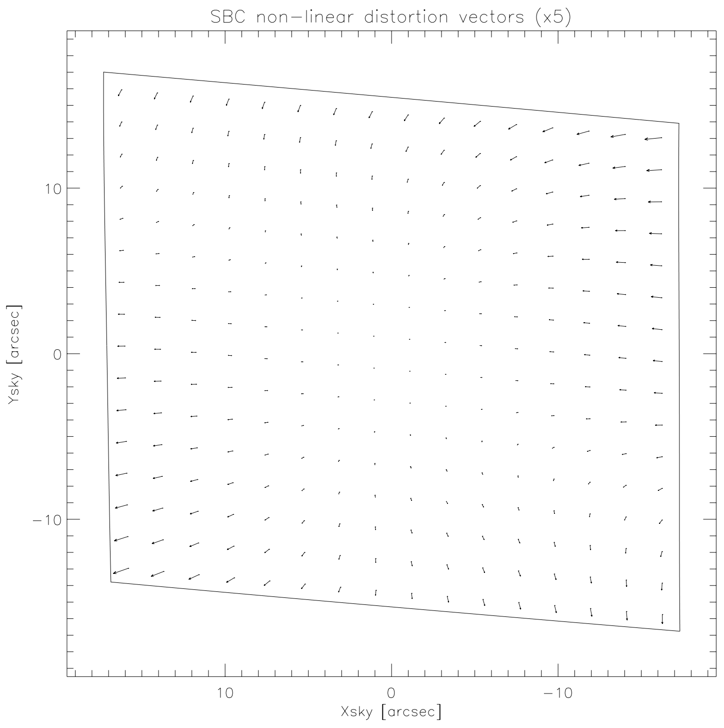
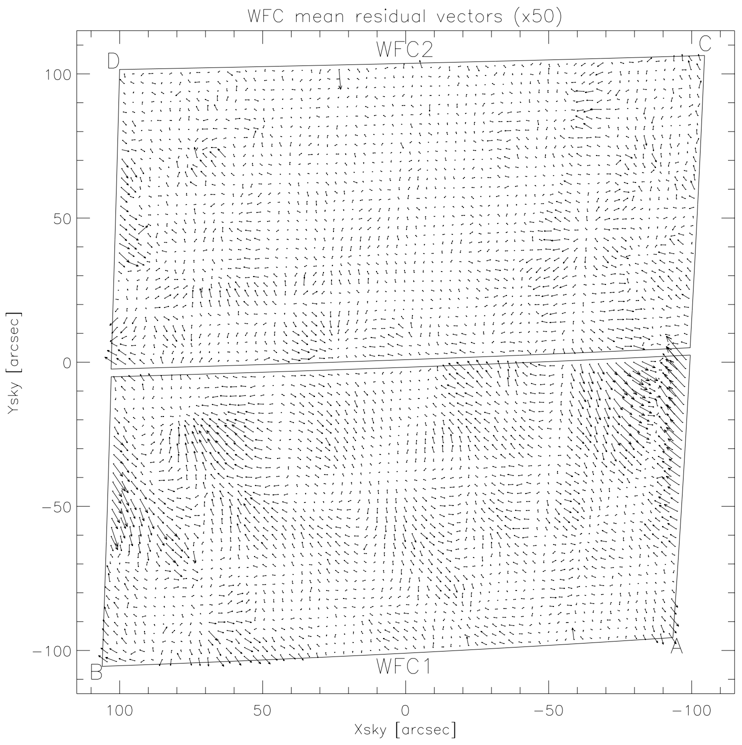


Figure 4.4: Binned residuals to quartic distortion model fits for the ACS WFC and HRC detectors. Note that the WFC figure is rotated 180 degrees with respect to the pipeline calibration products, where WFC2 is the lower half of the detector. The large residuals close to the upper edge of the HRC map, left of center, are due to the presence of the Fastie occulting finger. There is insufficient data to make the equivalent plot for the SBC.



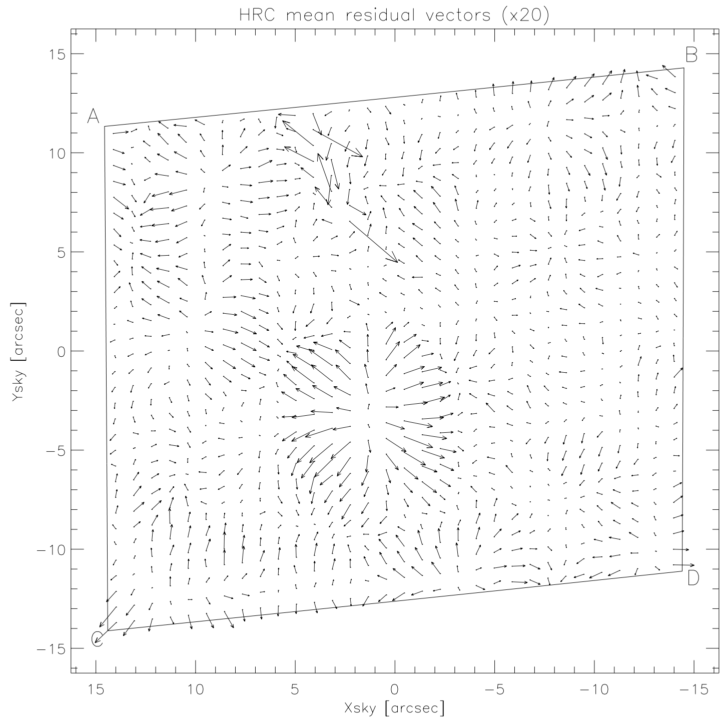
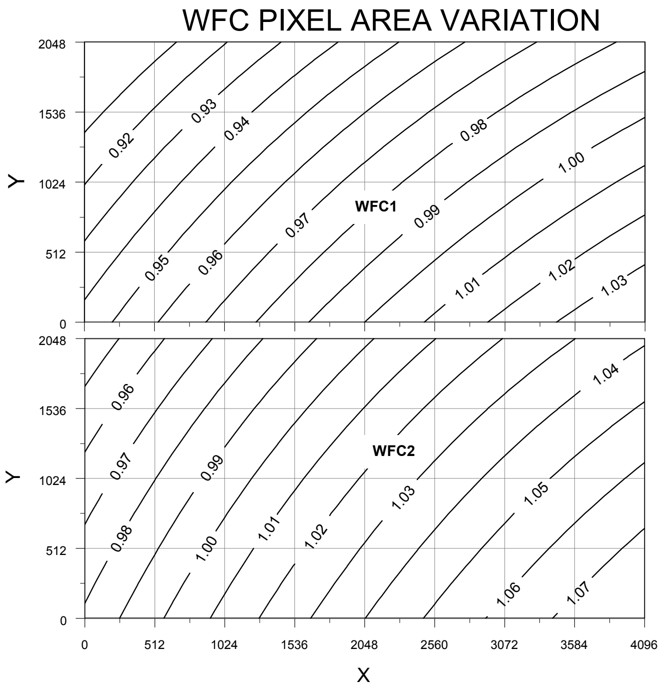
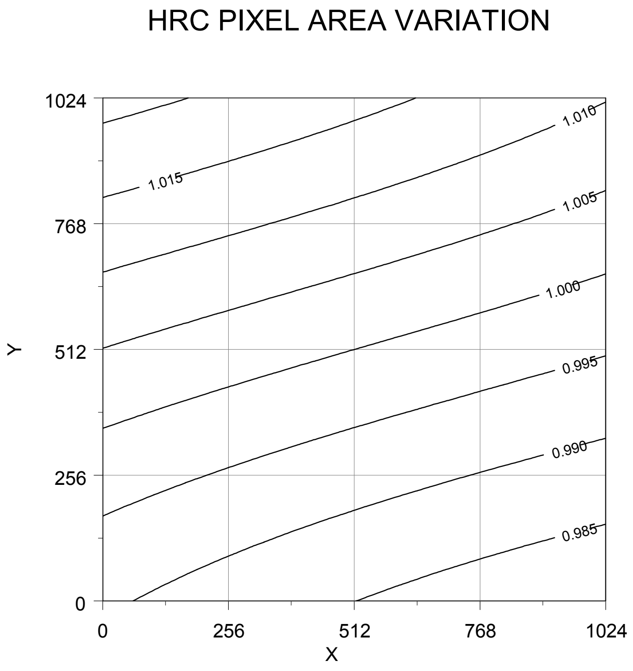


Figure 4.5: Variation of the WFC and HRC effective pixel area with position, in detector coordinates.





4.1.5 Distortion Table (IDCTAB)

The Science Instrument Aperture File (SIAF) contains position and scale information for every aperture of each of HST's science instruments. Thus, it supports accurate target acquisitions, image processing, and photometry. The conversion between distorted and undistorted positions in the SIAF is controlled by a polynomial expansion which is derived by the individual instrument teams. A simple, common reference file can therefore be created for each instrument which describes the geometric distortion of each detector. In this section, we describe the format of this common reference file which has been developed to be as general as possible so that it can be used for many different instruments or detectors.

For ACS, the coefficients to the polynomial fit are found in the IDCTAB reference file. The distortion correction is not a part of **CALACS**. A separate task, **PyDrizzle** (see Section 4.3), converts the distortion model in the IDCTAB into a form which is used by **drizzle** to produce geometrically corrected images.

Application of the Model

The steps performed in the application of the IDCTAB can be summarized in the following steps:

- First, the offset in position of each pixel relative to the reference position is computed.
- These deltas are used to compute the undistorted positions in arcseconds with the coefficients of the polynomial fit given in the IDCTAB reference file.
- The undistorted positions are divided by the desired output plate scale, where the default value is taken from the SCALE column in the IDCTAB unless specified by the user.
- Finally, both chips (WFC only) are combined into a single image by adding the appropriate offset to each pixel to get the final position.

The fit for each chip ensures that the distorted and corrected Y axes for a chip remain at the same angle with respect to the telescope V3 axis. This allows each chip to have a different rotation relative to the other, though this effect has been empirically shown to be less than 0.001 degree.

Format of the IDCTAB Reference file

The IDCTAB is stored as a FITS binary table (calibration file with '_idc.fits' extension), where the primary header contains keywords INSTRUME and DETECTOR which define the model appropriate to a specific instrument and detector. The NORDER keyword gives the order of the polynomial stored in the table. While the original laboratory data provided a cubic solution to the distortion, more accurate in-flight data now supports a fourth-order solution (NORDER=4).

The first extension of the IDCTAB contains the geometric distortion solution, where the table columns are listed in Table 4.2. The table contains distinct rows for each detector chip and filter combination specified by the DETCHIP and FILTER columns. If the DIRECTION column is ‘INVERSE’ rather than ‘FORWARD’, the coefficients specify the conversion from undistorted to distorted pixel positions. The SCALE keyword gives the default pixel scale of the drizzled image, where the default output scale for drizzled WFC images is 0.05 arcsec/pixel and for HRC/SBC images is 0.025 arcsec/pixel.

Table 4.2: Columns in the IDCTAB

Column	Description
DETCHIP	ID of chip/detector used for observation
DIRECTION	Application direction of coefficients (FORWARD,INVERSE)
FILTER1	Name of filter in filter wheel 1
FILTER2	Name of filter in filter wheel 2
XSIZE	Raw image size in X direction (pixels)
YSIZE	Raw image size in Y direction (pixels)
XREF	X position of reference pixel
YREF	Y position of reference pixel
V2REF	V2 position of reference point (arcsec)
V3REF	V3 position of reference point (arcsec)
SCALE	Scale of square corrected pixel (arcsec/pix)
CX10, CX11,...	Distortion Coefficients for X position
CY10, CY11,...	Distortion Coefficients for Y position

The distortion coefficients provide a correction for each pixel relative to the detector chip reference point specified in the XREF and YREF columns. The relationship between each chip must be factored in separately. HST relies on a V2/V3 coordinate system to define each detector’s position (in arcseconds) relative to the center of HST’s field of view. The V2REF and V3REF columns contain the reference position of each detector in the V2/V3 system. The absolute position is secondary to the relative position of this reference point, because only the offset between V2REF/V3REF positions is used for combining multiple chips into a single output image. The XREF/YREF and V2REF/V3REF positions represent the same point in different coordinate systems.

The columns CX*ij* and CY*ij* contain the coefficients of the polynomial fit for each chip and convert an input pixel position to an undistorted position. The column names contain the indices for the coefficients from the polynomial fit, where CX11 and CY11 correspond to the a_{11} and b_{11} coefficients. For subarray data, the input pixel position must be corrected for the subarray offset in order to put it in the coordinate frame of the full chip.

4.2 Combination of Dithered Data

4.2.1 Introduction to Dithering

Many ACS datasets are dithered - they have shifts between successive images which may be small or large. Dithering is a well established technique for HST imaging and has many advantages. For a general introduction to the concepts behind dithering, please consult the HST Dither Handbook (Koekemoer et al. 2002).

The primary motivations for dithering are:

- To allow the detection and flagging of image artifacts. Cosmic rays may be detected by taking multiple CR-SPLIT images at the same position, but other artifacts such as hot pixels require exposures with small dithers if they are to be detected and flagged. In addition, small-scale dithers are useful for filling in regions not covered by the detector, like the WFC chip gap or the HRC occulting finger.
- To improve the sampling of the PSF. In many ACS observation modes the pixel grid spacing is inadequate to fully sample the optical PSF falling on it. In these cases sub-pixel dithering will allow some of this lost information to be regained and the subsequent reconstruction of an improved final image.
- Larger dithers are needed to create mosaics of larger regions of sky or to cover the gap between the two chips of the ACS/WFC.

Because the non-linear component of the ACS optical distortion is large, dither shifts will vary across the field of view. This must be taken into account when planning observations. For more information on ACS dither patterns, see the ACS Patterns section of the HST Phase II proposal instructions and additional variations on the ACS “dither” webpage:
`http://www.stsci.edu/hst/acs/proposing/dither.`

4.2.2 Introduction to Drizzling

The methods for combining dithered ACS data which are described in this chapter are all based on the drizzle method (Fruchter & Hook, 2002 PASP, 114, 144) and are described in detail in the *HST Dither Handbook* (Koekemoer et al. 2002). Here we give a brief summary of the method and refer the reader to the aforementioned paper for further details and quantitative information about its properties.

Drizzle maps each pixel of the input image onto pixels in a sub-sampled output image, taking into account shifts and rotations between images and the optical distortion of the camera. However, in order to avoid re-convolving the image with the large pixel of the camera, the user may shrink the pixel before it is averaged into the output image. This shrinking factor is known as the “pixfrac” and is normally between 0.0 (corresponding to interlacing input pixels onto single output ones) and 1.0 (which corresponds to shift-and-add). The degree of sub-sampling (the size of the output pixels relative to that of the original ones) is also a user controllable parameter called “scale” which is typically in the range 1-0.5 for the ACS.

This algorithm has a number of advantages. It preserves both absolute surface and point-source photometry, hence flux can be measured using an aperture whose size is independent of position on the chip. And because the method anticipates that a given output pixel may receive no information from a given input pixel, missing data (due for instance to cosmic rays or detector defects) do not cause a substantial problem, so long as there are enough dithered images to fill in the gaps caused by these zero-weight input pixels. Finally, the linear weighting scheme is statistically optimum when inverse variance maps are used as the input weights.

The original drizzle method mapped a shrunken version of each input pixel onto the output and distributed weight according to the overlaps with the output pixel grid. For the case of multiple sub-pixel dithers of significantly undersampled images, this was close to optimum. However, for some purposes this “kernel” is not ideal and the latest implementations of drizzle allow the user to select from other options. For example, if the original method is applied to a single image, where pixfrac and scale are both normally unity, the method becomes equivalent to bilinear interpolation. This results in an output image having strongly correlated noise and also degraded resolution. For images from the ACS in the red, even the WFC channel is close to critically sampled, rather than strongly undersampled, and it would appear to be useful to consider more optimal interpolators as drizzle kernel options. Some of these are now implemented in **drizzle** and may also be used via the higher-level scripts described. The **drizzle** help file lists all the options. The “Lanczos3” kernel is particularly recommended as it minimizes apparent noise correlation in the output image and maximizes resolution. It is, however, significantly slower.

The basic drizzle method is implemented as an IRAF task within the STSDAS **dither** package which is invoked by **PyDrizzle** or **MultiDrizzle**.

4.2.3 Overview of Drizzling Software Tools

The tools for the combination of dithered HST images have evolved through the years, starting with those developed for the Hubble Deep Field in late 1995. First came low level tools which required considerable effort to use. Subsequently higher level wrapper scripts, written in the Python language, have been developed to ease such combination. Some of these capacities are now available within the pipeline. The main three levels applicable to current ACS imaging are as follows:

- The low level tools for mapping pixels from input to output images and support tasks for measuring shifts, detecting cosmic ray events, etc. are available within the dither package of STSDAS. These include **drizzle** itself and **blot** (reverse **drizzle**). The user will now probably prefer to access these tools via one of the wrappers rather than use them directly.
- **PyDrizzle** and associated Python scripts are layered on these and automate some of the processing. They convert the IDCTAB information into a form suitable for drizzle, create appropriate association tables and drizzle images into an appropriate output frame. These functions are implemented in the pipeline and are described in detail below. The user will again probably not use these tools directly. **PyDrizzle** is available within the STSDAS system and requires the PyRAF environment.
- The third level is the **MultiDrizzle** script which is again layered on top of the above tools. It aims to provide automated image combination, including the detection and flagging of cosmic rays. Although **MultiDrizzle** can be run using its default settings to do a good combination for many typical ACS dithered datasets, it also allows the user considerable control and flexibility. **MultiDrizzle** is available as a beta-test version but will be implemented in the STSDAS system in 2004. Like **PyDrizzle**, it requires the PyRAF environment.

We now briefly describe the pipeline software **PyDrizzle** and several associated tools. A detailed description of **MultiDrizzle** follows, as this is the recommended interactive tool for drizzling ACS images.

4.3 PyDrizzle

4.3.1 Application by the Pipeline

The ACS calibration pipeline currently consists of 2 processing stages, **CALACS** and **PyDrizzle**, which are run one after the other. Standard **CALACS** processing includes bias and dark subtraction, removal of the overscan regions, and flat fielding. It will produce, among other products, a calibrated image with the FLT suffix for each input file. The **CALACS** portion of the pipeline will calibrate and combine CR-SPLIT images. It will not, however, correct for geometric distortion or combine images from multiple dither positions.

Geometric distortion is removed for all images (dithered or not) in the second stage of processing via **PyDrizzle**. This software understands the ACS association tables, allowing the pipeline to combine associated dithered observations into a single image. These images will still contain cosmic-rays unless CR-SPLIT observations were specified within the dither pattern. For WFC observations, both chips are combined into a single extension. Finally, **PyDrizzle** converts the data to units of count rate (*electrons/sec*).

The IRAF **dither** package has provided users the basic tools for linear image combination via the task **drizzle**. This task can produce mosaics, correct for geometric distortion, and even remove cosmic rays, creating a single, undistorted output image with uniform photometric and astrometric properties. A significant amount of work, however, is required to determine the necessary input parameters, causing **drizzle** to be fairly user-intensive. These steps include:

- requiring the offsets in x and y from the reference image as input.
- requiring the output image size and pixel scale as input.
- only working with a single image extension at a time.
- requiring the rotation of each image extension as input.

These requirements made it impossible to use **drizzle** in a pipeline environment to automatically process images.

PyDrizzle was written to automate the use of **drizzle** in the pipeline environment. It can also simplify using **drizzle** for manual image reprocessing. Pipeline use relies on default parameter values which are taken from the IDCTAB reference file and include the default plate scale and the distortion model. This file provides information about the position of each ACS detector on the sky, allowing **PyDrizzle** to combine images from multiple detectors at once. **PyDrizzle**, like **drizzle**, works with count rates, allowing images with differing exposure times to be combined easily. Regardless of whether observations were taken as a single image or a set of dithered exposures, **PyDrizzle** produces images which are both astrometrically and photometrically accurate.

Processing comments are recorded in the trailer file for the DRZ image, including which version of **drizzle** was used, what parameters were used, and which images were combined (if dithered). The same default parameters are used for all observations in the pipeline. These parameters were chosen to avoid introducing any scale changes, shifts, or rotations relative to the original pointing, aside from those corrections incorporated in the distortion model itself. While the pipeline products will be properly corrected for distortion, the combination of dithered observations may not be ideal, and small pointing errors or defects, such as hot pixels or cosmic-rays, may still be present.

4.3.2 Association Tables

Before the **drizzle** task could be used in an automated manner, two obstacles had to be overcome: calculating the required input parameters and working with a wide range of input image formats. **PyDrizzle** relies on several features to address these problems:

- An association table will relate multiple input images to an output product.
- Python “structures” will be created in memory for each input image and for the output field.
- Output field parameters (RA/Dec of center, pixel size, field of view) will be specified using a ‘SkyField’ function.

The calibration pipeline uses association tables to define how multiple images, taken at different pointings, relate to a single output image. **PyDrizzle** relies on these association tables to automatically combine and correct ACS dithered observations or any other related set of observations.

Each input/output exposure within **PyDrizzle** knows about:

- its own world coordinate system,
- the distortion model for its detector,
- the range of output pixels covered by the image, and
- any instrument specific characteristics.

The **PyDrizzle** design provides the flexibility to support multiple instruments simultaneously, and to combine observations from several instruments into a single output field with arbitrary specification. The virtual output image is built using the World Coordinate System (WCS) information from each input image and the distortion model. This provides all the information necessary to calculate the input parameters for **drizzle**. These parameters are then passed to **drizzle** which actually performs the image combination.

4.3.3 PyDrizzle Data Products

The output from **PyDrizzle** is a single multi-extension FITS file with the suffix ‘_drz.fits’. The first extension contains the science (SCI) image which is corrected for distortion and which is dither-combined (or mosaiced), if applicable. The drizzled SCI image is in units of *electrons per second*. All pixels have equal area on the sky and equal photometric normalization across the field of view, giving an image which is both photometrically and astrometrically accurate for both point and extended sources. The dimensions of the output image will be computed on-the-fly by **PyDrizzle** and the default output plate scale will be read from the IDCTAB. These parameters, however, may be chosen by the user during reprocessing to best suit the actual data.

The second extension of the output image contains the weight (WHT) image. It gives the relative weight of the output pixels and can be considered an effective exposure time map. The data quality map created by **CALACS** is used by **PyDrizzle** in creating the weight image.

The third extension of the **PyDrizzle** output image contains the so-called context (CTX) image which encodes information about which input image contributes to a specific output pixel. This is done using a bitmask for each output pixel, where 'bit set' means that the image, in the order it was combined, contributed with non-zero weight to that output pixel. The context image starts as a single 32-bit integer image but is extended as a cube with additional 32-bit deep planes as required to handle all the input images.

4.4 Reprocessing with PyDrizzle

The goal of the ACS pipeline is to provide data calibrated to a level suitable for initial evaluation and analysis for all users. Observers frequently require a detailed understanding of the calibrations applied to their data and the ability to repeat, often with improved products, the calibration process at their home institution. There are several occasions when off-line interactive processing with **PyDrizzle** is required:

- Observations which have been manually reprocessed with **CALACS** will still require a correction for geometric distortion.
- Users may wish to modify the default **PyDrizzle** parameters, specifying an alternate output image orientation, pixel "shrinking factor" pixfrac, or pixel scale.
- When images have been taken across multiple visits, the observations are not automatically associated. Creating custom association tables will allow data from multiple visits to be combined.
- For optimal image registration, the user may wish to fine-tune the WCS information by supplying additional shifts and/or rotations.
- During pipeline processing, **PyDrizzle** makes assumptions about which DQ flags should be considered bad. These pixels will be excluded when creating the drizzled product. Users may wish to select which flagged pixels should actually be considered good and included in the drizzled product.

The **PyDrizzle** software uses the Python language and can only be executed in the PyRAF environment. PyRAF, developed at STScI, allows users to run tasks using the standard IRAF parameter-based interface or by using separate Python programs. The PyRAF interface and the most recent version of **PyDrizzle** can be downloaded from:

http://stsdas.stsci.edu/pydrizzle.

The calibrated products from **CALACS** and the corresponding input to **PyDrizzle** are described for various observing modes in Table 3.1. For example, a single image will produce an FLT file, a RPT-OBS exposure will produce an SFL file, a CR-SPLIT exposure will produce a CRJ file, and a DITH-PATTERN will produce a series of FLT, CRJ, or SFL files at *each* dither position. **PyDrizzle** may be executed using any of the calibrated data products or the association table itself. When executed on an association table, **PyDrizzle** looks only for the existing product file and geometrically corrects that image. **PyDrizzle** will create a single drizzled image with the DRZ suffix for each calibration product. In the case of dithered observations which are part of a pattern, the individual FLT files will be combined to create a single DRZ image.

PyDrizzle was not developed with the capability of detecting cosmic-rays, so unless cosmic-ray rejection is performed in the first stage by **CALACS**, the calibrated, drizzled product will still contain cosmic-rays. A new package, **MultiDrizzle**, addresses this deficiency and is discussed in detail in Section 4.5.

4.4.1 Specifying the ‘Bits’ Parameter

The data quality flags which were set during **CALACS** processing can be used as bit masks when drizzling, and the user may specify which bit values should actually be considered ‘good’ and included in image combination. This is done via the parameter ‘bits’. Any pixel flagged otherwise will be considered ‘bad’ and will be given zero weight. If a pixel is flagged as bad but has non-zero weight in any other image, the pixel will be replaced with the value from that image during image combination. Otherwise, the pixel will be replaced with the ‘fill value’. The default ‘fill value’ used by **PyDrizzle** during pipeline processing is zero. If the ‘fill value’ is set to INDEF during reprocessing with **PyDrizzle** or **MultiDrizzle**, and there are no pixels with non-zero weight, the pixel will retain its original value.

The flags for the DQ array were presented in Chapter 3 and are summarized in Table 3.4. The bits value used during pipeline processing is 8578 which is the sum of 8192 (corrected cosmic rays) + 256 (saturated data) + 128 (bad pixel in bias file) + 2 (pixel rejected in second iteration of cr-rejection). Note that these pixels were flagged during **CALACS** processing as “bad” or “suspect”, but may have been corrected in later processing steps. For example, cosmic rays are flagged 8192, but these pixels are replaced by unaffected pixels from other images in the CR-SPLIT association. The bits parameter indicates which “suspect” pixels to keep. When the total counts in a given pixel have exceeded the full well, for example, the DQ flag is 256. However, testing shows that counts still accumulate in a highly linear manner and that photometry of saturated stars is quite practical if using a gain that samples the full well depth.

The default bits value in the **PyDrizzle** and **MultiDrizzle** software is zero, but should be reset by the user prior to reprocessing. The value chosen for this parameter is completely up to the user and should be selected based on the specific calibration needs. For HRC processing, the user may want to add 8 (masked by aperture) to the 8578 value so that pixels which lie behind the occulting finger mask are not given zero weight. (In Figure 4.8 of Section 4.4.4 "PyDrizzle Examples", bit 8 was not included in the bits parameter prior to drizzling. Pixels behind the occulting finger were therefore given zero weight and replaced with the default fill value=zero.) Of course, the photometry in this region will be inaccurate due to improper flat fielding, so this bit would likely only be set for aesthetic reasons.

4.4.2 Creating Custom Association Tables

Association tables for dithered, REPEAT-OBS, or CR-SPLIT observations are generated by the **CALACS** pipeline. These tables keep track of the input exposure filenames and the output product filenames. Some types of observations, however, will not have association tables. Others will have multiple tables from different visits which need to be combined into one. In the following discussion, we present the methodology for creating custom association tables, either by merging association tables or creating them from scratch. Users also have the capability to manually edit association tables to include any known image offsets. This can be done using the TTOOLS task **tedit** in IRAF, where the user adds the columns XOFFSET, YOFFSET and/or ROTATION to the association table. Alternately, an ascii file with the known image offsets may be specified and the association automatically updated.

Merging Association Tables

Observing programs which cover a large portion of the sky will generally be made up of multiple pointings. Each of these pointings may be dithered or split for cosmic-ray rejection and will possess their own association table. In order for **PyDrizzle** to produce a single mosaic product, a composite association table must be built from the separate association tables. Users can easily create this composite by merging the individual tables using the TTOOLS task **tmerge** with "option = append".

The default product rootname will be taken from the first listed DTH-PROD in the composite association table. This rootname can be modified by the user (with **tedit**) to suit the desired naming convention. Generally, this rootname should match the rootname of the composite association table. A detailed example of merging association tables is given in step 1 of Example 4 from Section 3.5.2.

Creating Association Tables from Scratch

Some images will not have association tables. These may be single exposures of a given target which were taken at different times and orientations. They may even be exposures taken with different instruments. Observations taken using POS-TARG offsets, which are specified manually in the proposal, will not have association tables either.

A Python tool called **buildAsn** has been developed to create association tables which may then be used by **PyDrizzle**. This task can be run under PyRAF using an IRAF parameter-based “epar” interface or directly using a Python command-line interface. Since it resides alongside the **PyDrizzle** code, no extra setup is required to use this task.

A built-in help facility provides details on additional parameters which can be used with **buildAsn** when using the Python syntax. (For those needing a step-by-step introduction to starting PyRAF, see the examples in Section 4.4.4. Building an association table will be part of the **PyDrizzle** Examples.) It can be accessed using:

```
pyraf> from pydrizzle import buildasn
pyraf> buildasn.help()
```

The following illustrates how to use **buildAsn** to create an association table from all calibrated input files with the FLT suffix found in a single directory. Within PyRAF and using the IRAF syntax, we type:

```
pyraf> buildasn mymosaic suffix='flt'
```

Alternately, the same result may be obtained using the Python syntax:

```
pyraf> buildasn.buildAsnTable('mymosaic',suffix='flt')
```

In this example, the association table will have the name ‘mymosaic_asn.fits’, the product name will be ‘mymosaic_drz.fits’, and all files with the FLT suffix found in the working directory will be included in the association. To specify a subset of images within a given directory, the user may specify the ‘suffix’ parameter to be a filelist (‘@filename’), a wild-card list (‘*8cd*flt*’), or any portion of the filename (‘crj’ or ‘f555w’).

If user determined offsets are available, **buildAsn** has the capability of incorporating them into the association table. These offsets (XOFFSET, YOFFSET, and/or ROTATION) are specified by the file listed in the ‘shiftfile’ parameter. Using the IRAF syntax:

```
pyraf> buildasn mymosaic suffix='flt' shiftfile='shift.txt'
```

This option allows users to fine-tune the final image combination by providing corrections to the header WCS information. For more information on the format of user-defined shifts, see Section 4.4.3. For an example illustrating the steps required to apply a shiftfile to an existing association, see Example 3 in Section 4.4.4.

4.4.3 Applying User-defined Shifts

The ability of **PyDrizzle** to properly align dithered observations relies on the accuracy of the header WCS information. Unfortunately, there are times when those WCS values may be slightly inaccurate, resulting in a mis-alignment of the final drizzled product. For example, one might wish to combine observations taken in different visits. If the visits were separated by a substantial time interval, then the telescope probably used different guide stars. Due to limitations in the intrinsic accuracy of the Guide Star Catalog, this can lead to WCS coordinates that differ by up to 1-2" between visits. Also when all exposures that one wishes to combine were taken in a single visit, it might still be important to correct for relative shifts between images. Even in two-star fine-guiding mode, thermal effects on the telescope can cause slow drifts that can build up to ~50 mas over a few orbits (or in rarer occasions within a single orbit). This corresponds approximately to 1 WFC pixel or 2 HRC pixels. Such shifts between exposures can degrade image quality and corrupt cosmic-ray rejection when left uncorrected.

By deriving shifts based on the position of objects in the data, the user may refine the shifts computed from the WCS header information to create a precisely-aligned drizzled product. Many observers have developed their own methods of comparing images and computing offsets. The conventions described here should support the majority of users, making it a simple matter to incorporate shift computations into **MultiDrizzle** or **PyDrizzle.**

Delta Shifts

A *delta shift* is defined as the residual shift required to align sources, *after* applying the offsets implied by the header WCS. Delta shifts may be determined by separately drizzling images onto a common WCS frame with the same central RA/Dec. This is performed in the '*driz_separate*' step of **MultiDrizzle** (see section 4.5 for more information) or by setting the **PyDrizzle** parameter 'single=yes'. Object lists derived for each drizzled image may then be matched and fit to derive a single delta shift for each image.

Delta shifts are defined to be in the 'input' frame of reference when the distortion-corrected, drizzled image has the same orientation and scale as the distorted image. Shifts will be in the 'output' frame of reference if any rotation or scaling was applied while drizzling. When the shifts are given in the ‘output’ frame of reference, **PyDrizzle** requires the specification of the reference image to properly account for any orientation and scale changes.

Absolute Shifts

An *absolute shift*, on the other hand, is the total shift between two images and includes offsets implied by the header WCS. Absolute shifts may be determined by separately drizzling images to their own unique WCS frame. **PyDrizzle** will use the header information to optimally place the drizzled image in the frame.

To derive absolute shifts in the ‘input’ frame of reference, each distorted image is drizzled separately, and **PyDrizzle** uses the unique WCS information from each frame to choose the central RA/Dec, the image orientation and final image size. No additional rotation or scaling is applied while drizzling. (Alternately, the user may catalogue sources in each distorted image and then apply the distortion model to the X/Y positions. However, this requires tasks to transform coordinates from distorted to undistorted space. These tasks are still being developed for release in the STSDAS dither package.)

Absolute shifts in the ‘output’ frame of reference may be computed by separately drizzling each distorted image, with the same rotation and scaling, onto a unique WCS frame. Target lists in each drizzled image may then be matched to derive the absolute shifts.

Figure 4.6: Delta Shifts in the ‘input’ and ‘output’ frame of reference.

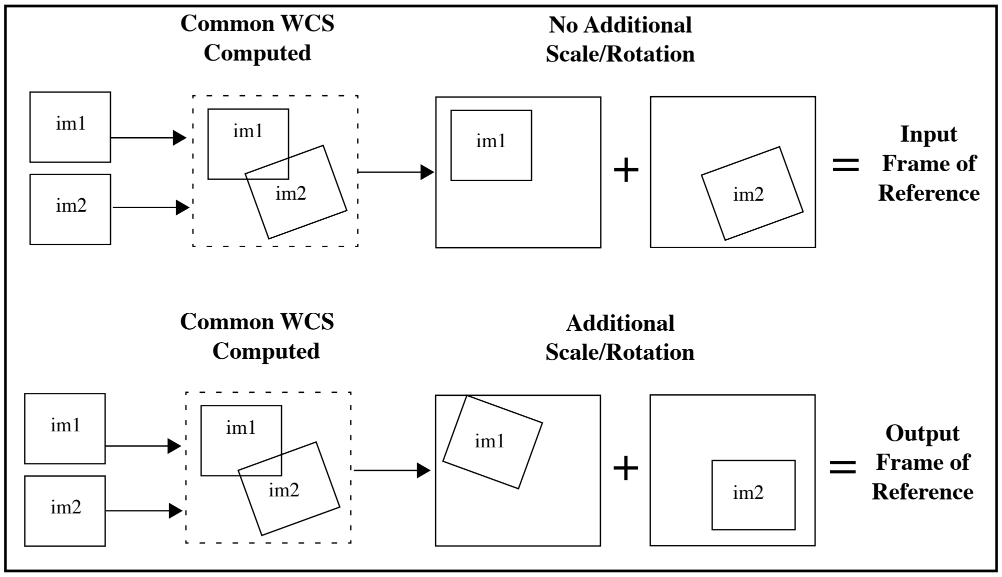
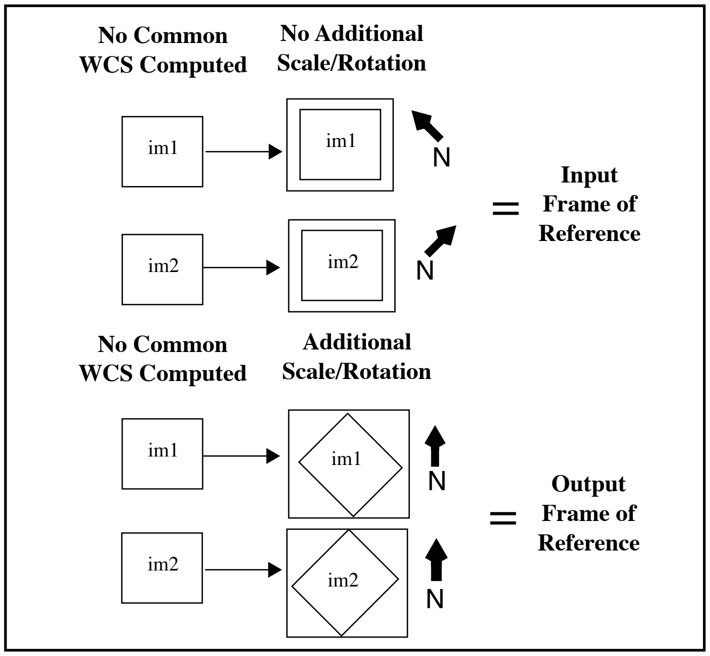


Figure 4.7: Absolute Shifts in the 'input' and 'output' frame of reference. The direction of north is indicated by the bold arrows.


No Common WCS Computed
No Additional Scale/Rotation
im1
im1
N
im2
im2
N
= Input Frame of Reference
No Common WCS Computed
Additional Scale/Rotation
im1
im1
N
im2
im2
N
= Output Frame of Reference

Shift Units

PyDrizzle is capable of interpreting shifts in units of *pixels* or *arcseconds*. A *pixel shift* is the difference in the X/Y pixel position of a source in a given image compared to the reference image. The shifts are interpreted with respect to each image's X/Y pixel reference position and require the specification of 'input' or 'output' frame of reference.

An *arcsecond shift* is the difference in the RA/Dec coordinates of a source in a given image compared to the reference image. Object coordinates must be derived *after* correcting for distortion. These shifts are *not* angular offsets on the sky, rather they are changes in the RA/Dec of the reference pixel and do not require any 'cos(Dec)' adjustment. Arcsecond shifts are a natural result of using the RA/Dec coordinates of image sources. They eliminate any ambiguity with the chosen reference frame since RA/Dec coordinates are fixed on the sky. Arcsecond shifts are interpreted as offsets in the direction of increasing RA and Dec.

Sign Conventions

The shifts are defined in terms of how the targets in the image need to move, rather than on how the telescope would need to move, to allow precise registration. **PyDrizzle** assumes that the shifts were computed as 'image - reference'. For example, when using **geomap** to cross-correlate two source lists, the input image coordinates will be in the first two columns and the reference image coordinates will be in the last two columns.

Use of Shifts in Association Tables

The *shift file* is an ASCII file containing the user-computed shifts for a set of images and is used for updating the offsets in the association table. Observers may use either the ASN table delivered by the pipeline or create/modify their own table as described in Section 4.4.2. Both **PyDrizzle** and **MultiDrizzle** use the **buildasn** task to incorporate information from the shift file into the association table. The *shift file* uses the following format:

```
# units: pixels            Values: pixels (default), arcseconds
# frame: input             Values: input (default), output
# form: absolute           Values: absolute (default), delta
# reference: <filename>    Only required when 'frame = output'
filename1  xshift1 yshift1  [rotation1]
filename2  xshift2 yshift2  [rotation2]
```

The first three lines specify the units, the frame of reference, and the form of the shifts, and only those lines with non-default values are necessary. While the '#' symbol is required, the 'Values:' comments are not. When shifts are given in the 'output' frame of reference, it is necessary to specify the name of the reference image used.

The following is an example of a shift file in units of pixels, measured in the 'input' frame of reference, with the form specified as 'delta' shifts. An x and y shift and a rotation (in degrees) is specified for each dataset. If only a simple shift is required, then the rotation column need not be specified.

```
#units: pixels
#frame: input
#form: delta
j8c0b1skq_flt  -2.4 -1.2 -0.002
j8c0b1snq_flt  -4.3 -2.3  0.001
j8c0b1sqq_flt  -6.9 -3.5  0.003
```

Using this shiftfile as input, the **buildasn** task will update the ASN table as indicated in Table 4.3.

Table 4.3: Association table updated with user-defined shifts.

MEMNAME	MEMTYPE	XOFF-SET	YOFF-SET	XDELTA	YDELTA	ROTATION
j8c0b1skq	EXP-DTH	0	0	-2.4	-1.2	-0.002
j8c0b1smq	EXP-DTH	0	0	-4.3	-2.3	0.001
j8c0b1sqq	EXP-DTH	0	0	-6.9	-3.5	0.003

The shift file can contain shifts for files other than those contained in the ASN table. In that case, only those entries found in the ASN table will be used to update the table. Thus, a single shift file can be generated for a whole set of observations which are represented by multiple association tables.

Association (ASN) tables can support either delta shifts or absolute shifts in units of pixels or arcseconds. Absolute shifts are stored in columns 'XOFFSET' and 'YOFFSET', while delta shifts are stored in columns 'XDELTA' and 'YDELTA'. The 'ROTATION' column is also available if a simple shift does not produce ideal registration and indicates how much should be added to the image's ORIENTAT to get proper alignment.

If *absolute offsets* are provided in the shift file, **PyDrizzle** will populate the OFFSET columns in the association and fill the DELTA values with zeroes. Similarly, if *delta shifts* are provided, those values will be used to populate the DELTA columns and the OFFSET columns will be zero. If shifts are given in both the OFFSET and DELTA columns, the OFFSET column will be applied and the DELTA columns will be ignored. This convention used by **buildasn** eliminates any ambiguity as to which values are applied to the data. Editing the ASN table directly will then allow the user to further update the offsets using either *delta* or *absolute* shifts.

4.4.4 PyDrizzle Examples

These examples further describe the steps required for pipeline reprocessing and serve as a continuation of the **CALACS** examples described in Section 3.5.2. We advise the reader to work through the **CALACS** examples first to develop a solid understanding of the calibrated data products and their potential limitations.

Example 1: Drizzling a Single Exposure

The following is a continuation of Example 1 from Section 3.5.2 where manual recalibration of a single raw exposure with **CALACS** produced a single image with the FLT suffix.

1. In the Unix environment, the 'jref' environment must be established prior to loading PyRAF. This directory indicates the location of the geometric distortion reference file (IDCTAB) specified in the image header. For more information, see "Setting up jref" in Section 3.5.1. If running IRAF and PyRAF simultaneously, 'jref' will need to be defined in each xterm window before loading the software. Alternately, this path may be initialized in the user's .setenv file. PyRAF

should then be started in its own xterm window and the **stsdas**, **analysis**, and **dither** packages loaded.

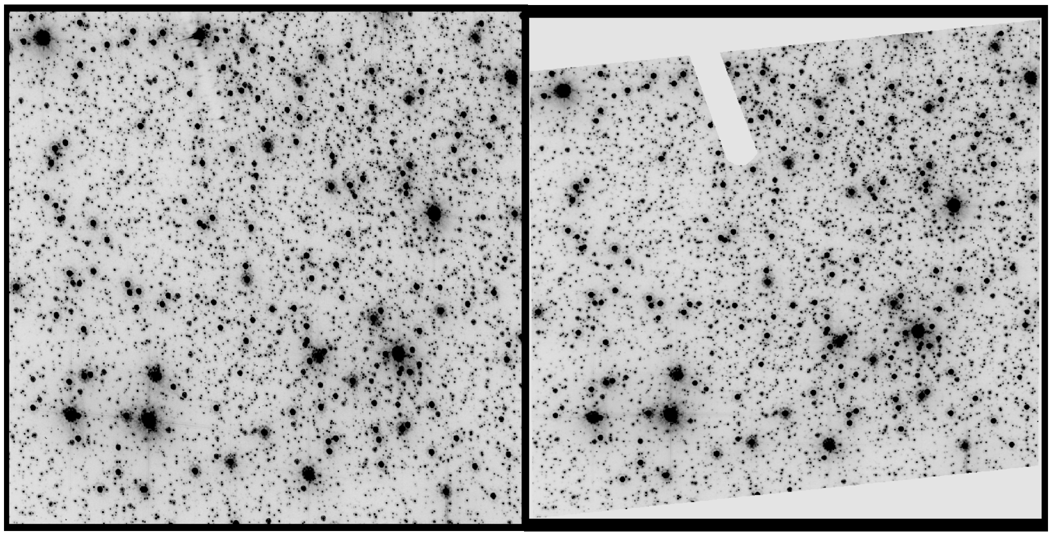
```
unix> setenv jref /mydisk/myjref/
unix> pyraf
pyraf> stsdas
pyraf> analysis
pyraf> dither
```

2. Next, we run **PyDrizzle** on the calibrated FLT image (j8bt07oyq_flt.fits). By default, the drizzled output product will be named j8bt07oyq_drz.fits. To specify an output file which is different than the default, the user may set the 'outfile' parameter to 'myfile_drz.fits', for example. Section 4.4.1 defined the 'bits' parameter; here we adopt the recommended value of 8578. The reader may also wish to add 8 to this value to see the effect of not dropping pixels behind the occulting finger. Prior to running, we recommended clearing any preset values using the 'unlearn' command.

```
pyraf> unlearn pydrizzle
pyraf> pydrizzle j8bt07oyq_flt.fits bits=8578
```

The distortion corrected HRC science image is shown in Figure 4.8. The image on the left is the flat fielded FLT product from **CALACS** and still contains distortion. The image on the right is the calibrated, distortion-corrected DRZ product from **PyDrizzle** using 'bits=8578'.

Figure 4.8: The effect of applying the distortion correction to a single HRC image. The image on the left is the flat-fielded FLT product from **CALACS**. The image on the right is the distortion-corrected DRZ product from **PyDrizzle**.



Example 2: Drizzling Images taken as part of a Dither Pattern

The following is a continuation of Example 5 from Section 3.5.2 where manual recalibration of a 2-point line dither pattern produced two FLT images (j8e654c0q_flt.fits, j8e654c4q_flt.fits).

1. Using **tprint**, we can view the contents of the image association, including the rootnames of the individual dithered images and the default rootname of the drizzled product.

```
pyraf> tprint j8e654010_asn.fits
# MEMNAME      MEMTYPE        MEMPRSNT
j8e654c0q      EXP-DTH        yes
j8e654c4q      EXP-DTH        yes
j8e654011      PROD-DTH       yes
```

2. Next we run **PyDrizzle** on the image association. The two FLT images are combined to create a single DRZ image called ‘j8e654011_drz.fits’.

```
pyraf> pydrizzle j8e654010_asn.fits bits=8578
```

Note that **PyDrizzle** is not designed to remove cosmic rays from these images. This step requires the new software **MultiDrizzle**, which is described in Section 4.5.

Example 3a: Improving the Image Registration - ‘Delta Shifts’ in the ‘Input Frame’

For observations with large dithers or for images which were taken at nominally the same pointing but during different visits, users may wish to fine-tune the image registration, improving on the world coordinate system in the image header. In this case, the images must be separately drizzled and the position of stars cross-correlated. The derived shifts may then be used to update the image association prior to drizzling the final product. More generally, if the drizzle combined product of multiple exposures have PSFs that are not round or as sharp as expected, a likely culprit is the need for improved shifts.

The following example uses WFC observations from separate visits in program 9018: visit 61, exposure 02 (j8c061020_asn.fits) on 19April 2002 and visit C1, exposure 01 (j8c0c1010_asn.fits) on 09May2002. Each observation is CR-SPLIT with an exposure time of 45 seconds in F606W. The images are at the same pointing, use the same guide stars, and are at the same ORIENT. We describe the steps required to register the calibrated CRJ products from separate visits. However, the individual FLT files could be alternatively used if improved registration of each CRJ image is desired, although use of FLT images would result in cosmic-ray retention.

1. First, we use the CRJ images from visits 61 and c1 to create a new association table with the suffix ‘v61c1’. We advise using the **tprint** task to verify the contents of the association.

```
pyraf> import buildasn
pyraf> buildasn f606w_v61c1 suffix=’crj.fits’
pyraf> tprint f606w_v61c1_asn.fits
# MEMNAME   MEMTYPE  MEMPRSNT  XOFF YOFF XDELTA YDELTA ROTATION
j8c061021   EXP-DTH    yes      0.   0.    0.     0.     0.
j8c0c1011   EXP-DTH    yes      0.   0.    0.     0.     0.
f606w_v61c1 PROD-DTH   no       0.   0.    0.     0.     0.
```

2. Next, we separately drizzle the images onto a common output WCS. This is achieved by setting the **PyDrizzle** parameter ‘single’=yes. Any differences in the aperture RA and Dec, any POS-TARG offsets, or any orientation differences will be automatically accounted for by **PyDrizzle** which uses the header WCS to transform the images. The drizzled CRJ products are appended with the suffix ‘_single_sci.fits’.

```
pyraf> pydrizzle f606w_v61c1_asn.fits bits=8578 single=yes
```

3. Using point sources in each image, we create and cross correlate coordinate lists from each visit. For example, we used **daofind** to create star lists for each DRZ product separately. Then we used **xyxymatch** to match the coordinate lists within a user specified tolerance. Finally, we use **geomap** with the matched coordinate list to compute the transformation required to map the reference coordinate system to the input coordinate system, allowing for a shift and/or rotation. We create the following “shiftfile”, a simple ascii table called ‘shift.txt’, using the derived offsets. The shifts were derived in units of *pixels* and are in the ‘input’ frame of reference since no additional rotation or scale terms were introduced while drizzling. The derived shifts using the above methods are *delta* shifts because the images were drizzled onto a common WCS.

```
#units: pixels
#frame: input
#form:  delta
j8c061021_crj    0.00    0.00  0.000
j8c0c1011_crj   24.46  -56.54  0.001
```

4. Next, we delete and rebuild the previous association table, this time specifying the derived offsets via the shiftfile. To verify that the association has been correctly populated, we use the **tprint** command.

Note that the *delta* shifts appear in the DELTA columns in the association table.

```
pyraf> delete f606w_v61c1_asn.fits
pyraf> buildasn f606w_v61c1 suffix='crj.fits'
shiftfile='shift.txt'
pyraf> tprint f606w_v61c1_asn.fits
# MEMNAME    MEMTYPE   MEMPRSNT  XOFF YOFF XDELTA YDELTA ROTATION
j8c061021    EXP-DTH    yes       0.  0.    0.      0.     0.
j8c0c1011    EXP-DTH    yes       0.  0.   24.46  -56.54   0.001
f606w_v61c1  PROD-DTH   no        0.  0.    0.      0.     0.
```

5. Finally we run **PyDrizzle** on the new image association. The two CRJ images are combined to create a single DRZ image called 'f606w_v61c1_drz.fits'.

```
pyraf> pydrizzle f606w_v61c1_asn.fits bits=8578 single=no
```

Example 3b: Improving the Image Registration - 'Absolute Shifts' in the 'Output Frame'

As a continuation of the previous example, we illustrate how to specify *absolute* shifts in the 'output' frame of reference.

1. First, the CRJ images from each visit are drizzled to separate output images. Absolute shifts are determined by separately drizzling images onto their own unique WCS frame. In this example, an additional rotation is applied to the drizzled images so that north is oriented up.

```
pyraf> pydrizzle j8c061021_crj.fits output=j8c06102X_drz.fits
bits=8578 rotate+ orient=0
pyraf> pydrizzle j8c0c1011_crj.fits output=j8c0c101X_drz.fits
bits=8578 rotate+ orient=0
```

2. As in the previous example, we create and cross correlate coordinates lists from each visit using point sources in each image. We create the following "shiftfile", a simple ascii table called 'shift1.txt', using the

derived offsets. Note that because we are using absolute shifts, we must specify the name of the reference image.

```
#units: pixels
#frame: output
#form:  absolute
#reference: j8c06102X_drz.fits
j8c061021_crj    0.00   0.00  0.000
j8c0c1011_crj  -41.64  45.51  0.001
```

3. Next, we build a new image association using the two CRJ files and the derived shift file.

```
pyraf> import buildasn
pyraf> buildasn f606w_v61c1rot suffix='crj.fits'
shiftfile='shift1.txt'
```

4. To verify that the association has been correctly populated, we use the **tprint** command. Notice that the *absolute* shifts have been correctly added to the `OFFSET` columns, not the `DELTA` columns.

```
pyraf> tprint f606w_v61c1rot_asn.fits
# MEMNAME     MEMTYPE  MEMPRSNT  XOFF  YOFF XDELTA YDELTA ROTATION
j8c061021      EXP-DTH    yes      0.    0.     0.    0.    0.
j8c0c1011      EXP-DTH    yes    -41.64 45.51   0.    0.    0.001
f606w_v61c1rot PROD-DTH   no       0.    0.     0.    0.    0.
```

5. Finally we run **PyDrizzle** on the new image association, specifying a rotation such that north is up. The two CRJ images are combined using the updated shift information to create a single DRZ image called 'f606w_v61c1rot_drz.fits'.

```
pyraf> pydrizzle f606w_v61c1rot_asn.fits bits=8578 rotate+
orient=0
```

4.5 MultiDrizzle

MultiDrizzle, developed in its original form by A. Koekemoer (Koekemoer et al. 2002, HST Calibration Workshop 2002, p.381), is a general purpose tool for drizzle combination of images. It provides a single-step interface to the complex suite of tasks in the STSDAS **dither** package. These tasks include, in order: initial image registration, creation of a cleaned median image, transformation back to the input image plane, creation of cosmic ray masks, and final drizzling. (For a general introduction, refer to the HST Dither Handbook.) The goal of **MultiDrizzle** is to provide a high-level task which has an extensive suite of user-adjustable parameters which, if left at their default values, will allow the task to perform all steps in a single operation with no user intervention. At the same time, the parameters allow the user a large amount of flexibility in controlling the relevant aspects of these steps, in case the default parameters are not sufficient for specific scientific applications.

The current version of **MultiDrizzle** uses the image header world coordinate system (WCS) to deduce the image-to-image offsets. However, user-supplied offsets may also be applied in a flexible manner. In the future, tasks to ease the determination of such shifts will be made available. **MultiDrizzle** processing with default parameters may eventually be part of pipeline processing, if a detailed assessment shows that this is feasible.

MultiDrizzle and **PyDrizzle** are both Python scripts which can only be run within the PyRAF environment. **MultiDrizzle** calls **PyDrizzle** to manage the images, to compute the **drizzle** parameters, and to control the operation of both the **drizzle** and **blot** tasks. While the latest release of STSDAS includes **PyDrizzle**, **MultiDrizzle** must be obtained separately (see below). It is recommended that the complete current STSDAS package is installed to avoid potential missing components or mismatched versions. STSDAS and PyRAF are available from:

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

Because many of these components are evolving rapidly, subsequent updates will be available from the same site.

While the software has been well tested and is fairly stable, it is still periodically being improved. We therefore refer readers to the **MultiDrizzle** web site for the most up-to-date software:

`http://stsdas.stsci.edu/pydrizzle/multidrizzle`

and to the ACS Drizzling web site for additional documentation:

`http://www.stsci.edu/hst/acs/analysis/drizzle.`

The following sections describe in detail the tasks performed by **MultiDrizzle** and give an example of its use. A help document describing the numerous **MultiDrizzle** parameters can be accessed by typing ‘help multidrizzle’ from within PyRAF. HST Dither Handbook is currently under revision to include documentation for **MultiDrizzle** and several detailed ACS drizzling examples. Version 3.0 is expected to be released by mid-2004.

4.5.1 Reprocessing with MultiDrizzle

In Section 4.4, we described several occasions when off-line interactive processing with **PyDrizzle** is required, for example, when images are taken across multiple visits, when fine-tuned image registration is required, or when the user wishes to specify which DQ flags should be considered *good*. **MultiDrizzle** will address these same issues, while at the same time providing additional capabilities such as cosmic ray rejection and sky subtraction. Because it allows access to an extensive suite of user-adjustable parameters, **MultiDrizzle** gives users more flexibility in selecting the parameters that best suit their observations. This is all made possible within a single, easy-to-use interface.

MultiDrizzle creates many large files and does a considerable amount of processing. It is hence recommended that a modern fast machine with several GBytes of free space be used if possible. In the example to follow, each of the input FLT files is ~168MB and the final combined output product is 310MB. Many large intermediate files are also created but may be cleaned up if required.

The only calibration reference file required by **MultiDrizzle** is the distortion table (IDCTAB) discussed earlier in this chapter. This table describes the ACS distortion in terms of polynomial coefficients. The appropriate reference file is recorded in the image header keyword IDCTAB. This file should be obtained from the STScI archive (via Starview) and placed in some local directory. Before starting **MultiDrizzle** the environment variable “jref” must be set appropriately to point to this directory (see Example 1 in Section 4.4.4 for more information). Once the FLT files have been collected and the ‘jref’ environment established, PyRAF should be started and the **stsdas**, **analysis**, and **dither** packages loaded.

The IDCTAB reference file in the image header must NOT be modified prior to running MultiDrizzle or PyDrizzle. To make use of an updated distortion solution, users are advised to re-retrieve their data from the archive such that the CD-matrix keywords in the image header are correctly populated.

To avoid potential problems, it is recommended that the parameters for both **MultiDrizzle** and **PyDrizzle** be reset to the default values:

```
pyraf> unlearn pydrizzle
pyraf> unlearn multidrizzle
```

4.5.2 Tasks Performed by MultiDrizzle

MultiDrizzle starts with the flat fielded, but still distorted, FLT files created by **CALACS** in the pipeline. While these are the only mandatory input, there are many additional optional inputs including lists of improved registration information. In default mode **MultiDrizzle** performs the following steps in order, although the user has full control via the parameter file, of which steps are to be performed or omitted. Each step is described in detail in the example which follows in Section 4.6.

1. StaticMask: Create a mask of bad pixels, using information from the images themselves and associated data quality files,
2. SkySub: Carry out sky subtraction on each individual input image,
3. Driz_Separate: Calculate relative image shifts based on header information in the world coordinate system keywords (optionally supplemented by user-supplied shift information) and drizzle the input images onto a series of separate output images,
4. Median: Combine these images to create a clean median image,
5. Blot: Transform, or blot, the median image back to the frame of the input images,
6. Driz_cr: Create a cosmic ray mask for each input image based on comparison between the blotted median and its derivative and the original input image,
7. Driz_Combine: Perform the final **drizzle** combination step, applying the cosmic ray masks.

A warning to users: MultiDrizzle modifies the original calibrated images, for example, when updating the image DQ array with the static pixel mask and when performing the sky subtraction. When reprocessing, we advise retaining a copy of the original files in a separate directory.

4.5.3 Specifying Shifts Between Images

By default, **MultiDrizzle** uses the world coordinate system (WCS) information in the image headers to align the images. While dithers applied to a target within a single visit of HST are usually accurately reflected by the WCS, this is not the case for multiple visits which normally require guide star re-acquisitions and may utilize different guide stars. Even for nominally back-to-back exposures that are part of a CR-SPLIT association, offsets large enough to degrade final combinations do sometimes occur. As a result, it is essential to accurately determine image-to-image shifts, and possibly rotations, before running **MultiDrizzle**.

These (delta) shifts may be determined by separately drizzling each image onto a common WCS frame. This is performed by the 'driz_separate' task in step 3. Object lists derived for each drizzled image may then be matched and fit to derive a single shift for each image. More details on this topic and a description of the shift file formats are given in Section 4.4.3.

Once the shifts have been determined, **MultiDrizzle** must be rerun from the beginning with the shift file specified in the 'shiftfile' parameter. **MultiDrizzle** will update the association table to reflect these 'delta' offsets and will now use both the header WCS information plus the shift file to appropriately align images.

4.6 MultiDrizzle Example

The MultiDrizzle software is under continuing development. The following example was written in December 2003 should be used as a guide to facilitate reprocessing. While the algorithms and the overall processing steps will remain stable, specific details of the software may change.

This example describes the combination of six ACS/WFC images of the "Tadpole Galaxy" UGC 10214 which formed part of the ACS Early Release Observations (EROs) from program 8992. These images were acquired in two visits, with a significant shift between visits, as noted by the target RA and Dec, and with small dithers within each visit. The images were taken with the F606W filter and have the exposure times listed in Table 4.4. Prior to attempting this example, we assume that users have worked through the **PyDrizzle** examples in Section 4.4.4 and are familiar with using association tables and shift files.

The ERO images are available from the HST archive for anyone wishing to repeat this example. A new capability has been developed in the

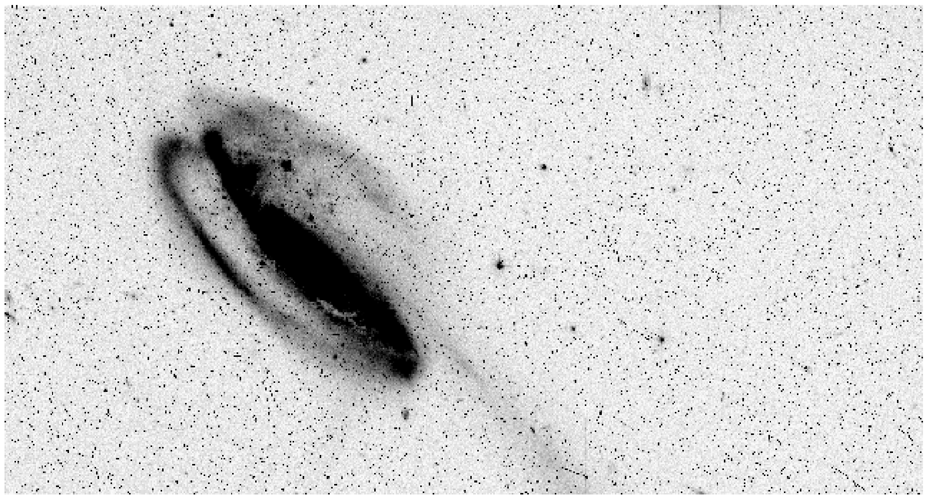
StarView archive interface, allowing the user to request FLT files only. The FLT files are the output from **CALACS** and have been corrected for bias, dark current and flat fielding. They are not corrected for distortion and still contain numerous cosmic rays. They form the input for **MultiDrizzle**.

Figure 4.9 shows chip 1 for one of these FLT files, j8cw54ovq_flt[sci,2], which still contains numerous cosmic ray events, hot pixels, and other artifacts. The rectangular, rather than 'rhombus' shaped image is a clear indicator that the geometric distortion has not yet been corrected.

Table 4.4: ERO datasets used in the accompanying **MultiDrizzle** example. The two sets of images were taken at different epochs and have a significant shift between epochs. Within a given epoch, small POS-TARG offsets are specified in accordance with the ACS-WFC-DITHER-LINE pattern which spans the WFC interchip gap.

Dataset	Date-Obs	RA (deg), Dec (deg)	POS-TARG1	POS-TARG2	Exposure Time (sec)
j8cw04abq_flt	01 Apr 2002	241.553625,+55.431094	0.000	0.000	150.0
j8cw04c2q_flt	01 Apr 2002		0.248	3.001	580.0
j8cw04c7q_flt	01 Apr 2002		0.248	3.001	840.0
j8cw54oiq_flt	09 Apr 2002	241.537833,+55.427594	0.000	0.000	150.0
j8cw54orq_flt	09 Apr 2002		0.248	3.001	580.0
j8cw54ovq_flt	09 Apr 2002		0.248	3.001	840.0

Figure 4.9: A single, calibrated FLT file, j8cw54orq_flt.fits[sci,2], which still contains numerous cosmic rays, hot pixels, and other artifacts.



4.6.1 Detailed Processing Steps & Parameters

First, the appropriate calibrated FLT files and the geometric distortion reference table (IDCTAB) should be placed on the user's local disk. Next the 'jref' directory should be defined, PyRAF started, and the **dither** package loaded. To run **MultiDrizzle**, the parameters may be edited in the standard way using the "epar" facility.

The **MultiDrizzle** software has an extensive set of parameters, but the default values should allow the task to process nearly any set of images for an initial review. The parameters are separated according to the processing step they control, making it easier to interpret them.

In this example, we describe the parameters for each step in succession, though in practice, the user would set all relevant parameters at once. While the majority of relevant parameters are discussed here, a help document describing all parameters and tasks can be accessed by typing 'help multidrizzle' from within PyRAF.

Initial Setup

Parameters	Default	Description
output		Rootname for output drizzled products
suffix	flt	Suffix of input files in current directory
filelist		List of input files
refimage		Reference image with desired output WCS
runfile	multi.run	File for logging the script commands
restart		Parameter allowing processing to resume at a given step
coeffs	header	Use header-based distortion coefficients? (header, none)
context	no	Create context image during final drizzle?
clean	no	Remove temporary files?
section		Extension or group to be drizzled
bits	0	Integer mask bit values considered good
ra		right ascension output frame center
dec		declination output frame center
build	yes	Create multi-extension output file?
shiftfile		Shift file for improving WCS registration

The only required parameter is the rootname for the 'output' drizzled product. By default, **MultiDrizzle** will look for all files in the working directory with the FLT extension. The user may modify the 'suffix' parameter to include some other extension or may specify a subset of images via the 'filelist' parameter. Using the images specified, **MultiDrizzle** calls **PyDrizzle** which uses the '**buildAsn**' task (see Section 4.4.2) to create an association table named 'output_asn.fits'. This association will be used to define the data set. The header WCS information from the entire set is used to define a common WCS output frame, and **MultiDrizzle** sets the **drizzle** parameters appropriately. If user defined shifts are available, these may be specified in the 'shiftfile' parameter, and the association table will be updated accordingly. However, a shift file is not usually available until after step 3, separately drizzling the images onto a common WCS, has been performed and objects matched.

A reference image which has the desired output WCS may be specified, and the input images will be drizzled to match the WCS of this image. Alternately, the central RA and Dec (ra, dec) of the reference pixel and the dimensions of the output frame (outnx, outny) may be specified, if desired, though reasonable values will be automatically determined from the images' WCS if these parameters are left blank. While the central RA and Dec are specified in the initial setup parameters listed above, the output image dimensions are specified in both the 'driz_separate' and 'driz_combine' parameters in steps 3 and 7, respectively.

The 'bits' parameter is defined as the integer sum of all bit values from the input images' DQ array that should be considered 'good' when building the weighting mask. Because **MultiDrizzle** was designed for use with multiple instruments, the default value is set to zero. For ACS data, the recommended default value is 8578. (For more information on selecting the appropriate bits for your data, refer to Section 4.4.1.) Information from the DQ array for each chip is used in combination with the 'bits' parameter to create temporary mask files called '*_inmask?.fits'. Pixels which were flagged in the DQ array *and* which were not specified as good via the bits parameter are assigned the value 0. All other pixels are set to 1 in the mask.

The distortion reference file is read from the image header via the IDCTAB keyword which specifies the name and location of the appropriate file. The distortion coefficients for each chip are written to temporary ascii files named '*_coeffs?.dat'. If the user wishes to retain these files, the parameter 'clean' should be set to 'no'.

In general, the default **MultiDrizzle** parameters will work well for most data. When setting up the Tadpole images, for example, we have specified *only* the following non-default parameters: output='example', 'bits=8578', 'combine_nhigh=2', and 'driz_cr_snr=4.0 3.0'. The choice of these last two parameters is explained in the sections that follow. In default mode, **MultiDrizzle** performs each of its 7 steps in order. In this example, however, we perform some of the steps and examine the intermediate products before final drizzle combination is performed. Approximately 5GB of free disk space is required for this example when intermediate products are not removed. An outline of the entire process is described below:

1. Run only steps 1 through 3 of Section 4.5.2 to create sky-subtracted, separately drizzled images which are based on a common WCS.

```
pyraf> multidrizzle output='example' bits=8578 static+ skysub+
driz_separate+ median- blot- driz_cr- driz_comb-
```

2. Measure the positions of stars in the separately drizzled images and derive a shift file which defines the residual offsets.
3. Rerun step 3 using the derived 'shiftfile' to create new separately drizzled images. Also turn on step 4 to create a well-aligned median image.

```
pyraf> multidrizzle output='example' bits=8578 shiftfile='shifts'
static- skysub- driz_separate+ median+ combine_nhigh=2 blot-
driz_cr- driz_comb-
```

4. Examine the median image to ensure that the PSF is 'round' and 'narrow' and that cosmic-rays and other artifacts are appropriately rejected.
5. Run steps 5 through 7 to transform the median image back to the reference frame of each of the original input images and to derive cosmic ray masks. Using these new masks, perform the final drizzle combination.

```
pyraf> multidrizzle output='example' bits=8578 shiftfile='shifts'
static- skysub- driz_sep- median- blot+ driz_cr+ driz_cr_snr='4.0
3.0' driz_comb+
```

Once the optimal set of parameters and the optional shift file is derived, as described in the outline above, **MultiDrizzle** may be executed in a single pass by turning all steps on and by specifying any desired non-default parameters:

```
pyraf> multidrizzle output='example' bits=8578 shiftfile='shifts'
static+ skysub+ driz_sep+ median+ combine_nhigh=2 blot+ driz_cr+
driz_cr_snr='4.0 3.0' driz_comb+
```

While **MultiDrizzle** may be executed from the command line, as shown in the above examples, it may also be run from the 'epar' facility which allows the user to see all parameters at once and to turn particular steps on and off. We recommend that beginners use the 'epar facility' to become familiar with all steps and parameters before running **MultiDrizzle** from the command line.

1. Creating the Static Mask

Parameters	Default	Description
static	yes	Create static bad-pixel mask from data?
staticfile		Name of (optional) input static bad-pixel mask
static_goodval	1.0	Value of good pixels in the input static mask

Output Files	Modified DQ arrays in the original input files

When ‘static=yes’, this step goes through each of the input images, calculates the rms value for each chip, and identifies pixels that are below the median value by more than 5 times the rms. It is aimed at identifying pixels that may have high values in the dark frame, which is subtracted during calibration, but may not necessarily have high values in the images, and thus subtraction gives them strongly negative values. Such pixels are not always flagged in the DQ file, and this step allows them to be identified. Sometimes such pixels fall on bright objects so instead of being negative, they would be positive but lower than surrounding pixels. If the images are dithered, then they should land on blank sky at least some of the time, in which case they will appear negative and will be flagged.

For the Tadpole Example, we have left the ‘StaticMask’ parameters to their default values. After this step is performed, the image DQ arrays are updated with the static mask, and new flagged pixels are set with bit 64. The ‘*_inmask?.fits’ files are subsequently updated during further processing.

2. Performing Sky Subtraction

Parameters	Default	Description
skysub	yes	Perform sky subtraction?
skytype	“single”	Type of sky subtraction (single, quadrants)
skyname	“SKYSUM”	Header keyword containing sky value (SKYSUM = ACS, BACKGRND = WFPC2)
skywidth	50.0	Interval width for sky statistics
skystat	“median”	Sky correction statistics parameter (median, mode, mean)
skylower	-50.0	Lower limit of usable data for sky (always in DN)
skyupper	200.0	Upper limit of usable data for sky (always in DN)

Output Files	Modified science array in the original input files

When ‘skysub=yes’, this task will subtract the sky from each chip (‘skytype=single’) or from each of the four individual amplifiers on the WFC chips separately (‘skytype=quadrants’). The other parameters correspond directly to those in the **sky** task in the **dither** package, and are passed to it exactly as they are specified here.

The main parameter that users must set is the histogram width, though the default value is good for most cases. The width determines the region of the pixel histogram used to determine the image statistics. It should be set to include most pixels in the sky (so substantially more than the FWHM of the sky distribution) but not so large as to include a substantial amount of power from objects or cosmic rays.

The ‘SkySub’ task will update the header keyword defined by the parameter ‘skyname’ with the derived sky value for each chip and will subtract the sky from the original FLT images. Sky subtraction is recommended for effective cosmic-ray flagging and removal, but only if sufficient blank sky is available to perform an accurate determination.

Great care must be taken when choosing to implement sky subtraction, because the original calibrated images will be modified, and because if sufficiently blank sky is not available, sky subtraction will produce erroneous results. In the case of the Tadpole images, adequate blank sky is available for each WFC chip to allow an accurate sky determination when ‘skytype’ is set to ‘single’. Choosing ‘skytype=quadrants’, on the other hand, results in an over-subtraction of the ‘A’ amplifier (upper-left quadrant) since the galaxy light dominates the background in this region of the detector.

3. Drizzling to Separate Output Images

Parameters	Default	Description
driz_separate	yes	Drizzle to separate output images?
driz_sep_outnx		Size of x-axis for separate output frame
driz_sep_outny		Size of y-axis for separate output frame
driz_sep_kernel	“turbo”	Shape of kernel function (square, point, gaussian, turbo, tophat, lanczos3)
driz_sep_scale	1.0	Linear size of output pixels (relative to input)
driz_sep_pixfrac	1.0	Linear size of drop in input pixels
driz_sep_rot	0.	Rotation of input image to be applied (degrees anti-clockwise)
driz_sep_fillval	INDEF	Value assigned to undefined output pixels

Output Files	Images= ‘*_single_sci.fits’, ‘*_single_wht.fits’

When ‘driz_separate=yes’, the input images are corrected for geometric distortion and drizzled onto separate output frames which have a common WCS. Any shifts, rotations, or scale changes are calculated from the image headers by **PyDrizzle**. The output image dimensions are calculated on-the-fly and the pixel scale is taken from the column ‘scale’ from the IDCTAB, where the default values are 0.05 *arcsec/pix* for the WFC and 0.025 *arcsec/pix* for the HRC. The drizzled images are in units of *electrons/sec*.

By default, the ‘driz_separate’ task uses the ‘turbo’ drizzle kernel and **drizzle** parameters ‘pixfrac=1’ and ‘scale=1’. For more information on setting these parameters, refer to the HST Dither Handbook. These parameters can be changed; for example, masks can be substantially improved by specifying a smaller value of scale (e.g., 0.5 or 0.66), with the primary trade-off being much larger images (their size increases as the inverse square of the value of ‘scale’) and increased computation time.

In the Tadpole example, we have left the ‘Driz_Separate’ parameters to their default values. While **PyDrizzle** produces a final drizzled image containing 3 extensions (the science, weight, and context images), the ‘driz_separate’ products are separate science and weight images named ‘*_single_sci.fits’ and ‘*_single_wht.fits’. No context image is created.

One of the singly drizzled FLT images, ‘j8cw54ovq_flt_single_sci.fits’, is shown in Figure 4.10. This image still contains numerous cosmic ray events, hot pixels, and other artifacts. The ‘rhombus’ shape is a result of correcting the geometric distortion. The corresponding weight image, ‘j8cw54ovq_flt_single_wht.fits’, is shown in Figure 4.11, where white indicates pixels with zero weight. Due to the effects of distortion and varying pixel area in the FLT images, the weight image changes gradually across the detector. Because the association table was used to define a common WCS for all images, the drizzled image is ‘padded’ with zeros outside the boundary of the original array. The weight image is set to zero in these regions, allowing these pixels to be rejected during median combination.

The separately drizzled science images may be used to improve the image registration prior to final drizzle combination. In this example, the images form two groups of three. While the WCS information for images within a single group (visit) are adequate to align them, there is a small residual offset between visits. Shifts which are determined from separately drizzling images onto a common WCS are by definition ‘delta’ shifts (see Section 4.4.3 for details), and will be applied in addition to any offsets from the WCS when a ‘shiftfile’ is provided. Because no additional scale or rotation was applied to the singly drizzled images, the shifts are in the ‘input’ frame of reference.

In this example, unsaturated stars in the short exposures (j8cw04abq and j8cw54oiq) were used to derive the offsets between the two groups of images. Shifts were measured to the nearest 0.1 pixel and are listed below in the form of a shift file. Refined shifts within a given visit may also be determined, but these are typically less than a tenth of a pixel, and accuracy at this level is not as critical for extended sources as it may be for point sources.

```
#units: pixels
#frame: input
#form: delta
j8cw04abq_flt  0.0  0.0
j8cw04c2q_flt  0.0  0.0
j8cw04c7q_flt  0.0  0.0
j8cw54oiq_flt -5.6 -0.3
j8cw54orq_flt -5.6 -0.3
j8cw54ovq_flt -5.6 -0.3
```

When the ‘driz_separate’ step is run for the second time, but with a ‘shiftfile’ specified, the association table for the data set will be automatically updated. To confirm that the new separately drizzled images are appropriately registered, the position of stars should again be examined. It is also useful to create a median image and examine the width and shape of the PSF over the entire FOV to look for any effects of mis-registration. Median combination is performed in Step 4 of **MultiDrizzle.**

Figure 4.10: The singly drizzled FLT image 'j8cw54ovq_flt_single_sci.fits' from the **MultiDrizzle** example.

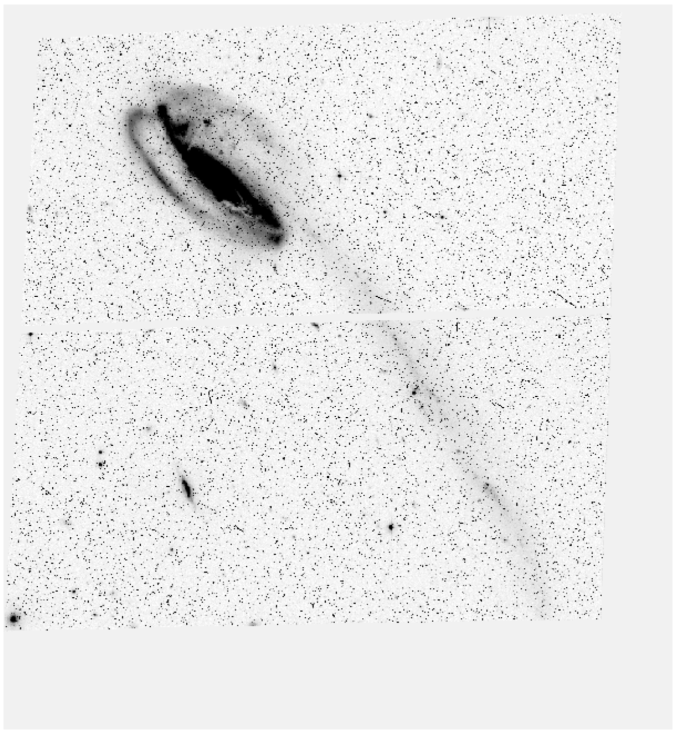
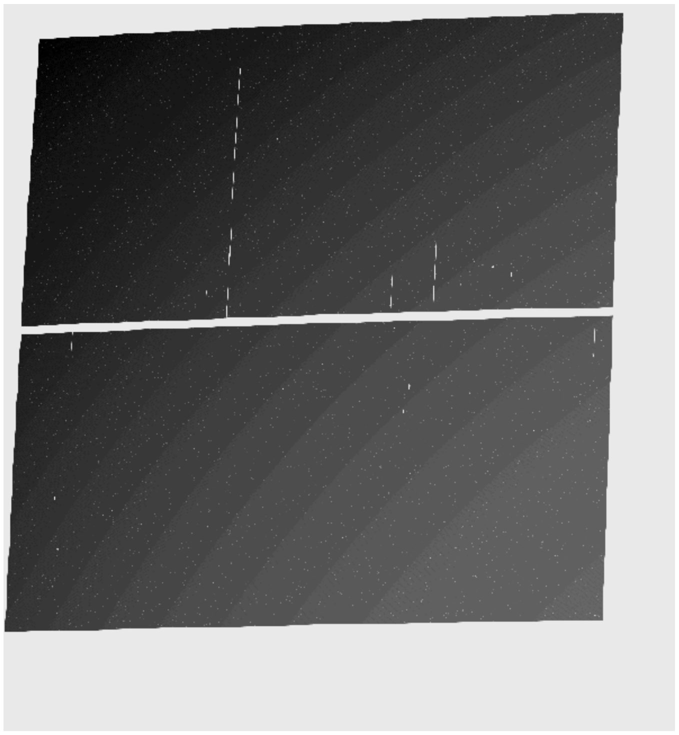


Figure 4.11: The weight image 'j8cw54ovq_flt_single_wht.fits' corresponding to the singly drizzled image in Figure 4.10 where white indicates zero weight.



4. Creating the Median Image

Parameters	Default	Description
median	yes	Create median image?
median_newmasks	yes	Create new masks when doing the median?
median_preclean	no	Remove intermediate files prior to doing median?
combine_type	"median"	Type of combine operation (average, median, sum, minmed)
combine_reject	"minmax"	Type of rejection (none, minmax, ccdclip, crreject, sigclip, avsigclip, pclip)
combine_nsigma	6 3	Significance for accepting min instead of median
combine_nlow	0	minmax: Number of low pixels to reject
combine_nhigh	1	minmax: Number of high pixels to reject
combine_lthresh	INDEF	Lower threshold for clipping input pixels
combine_hthresh	INDEF	Upper threshold for clipping input pixels
combine_grow	1.0	Radius (pixels) for neighbor rejection

Output Files	Images= 'output_med.fits', '*_single_wht_maskhead.pl'

When 'median=yes', this step creates a median image from the separate drizzled input images, allowing a variety of combination and rejection schemes. If 'combine_type' is set to 'median' or 'average', then the routine calls the IRAF task **imcombine**, passing to it the values of 'combine_reject' (the rejection algorithm chosen, usually expected to be 'minmax'), 'combine_nlow' and 'combine_nhigh' (the number of low and high pixels to reject), and 'combine_grow' (the amount by which flagged pixels can grow). All **imcombine** parameters other than those specified above are reset to their default values. While the 'median' is recommended, a slightly more sophisticated algorithm than those available in **imcombine** will be used when 'combine_type=minmed'. This algorithm is described at the end of this section.

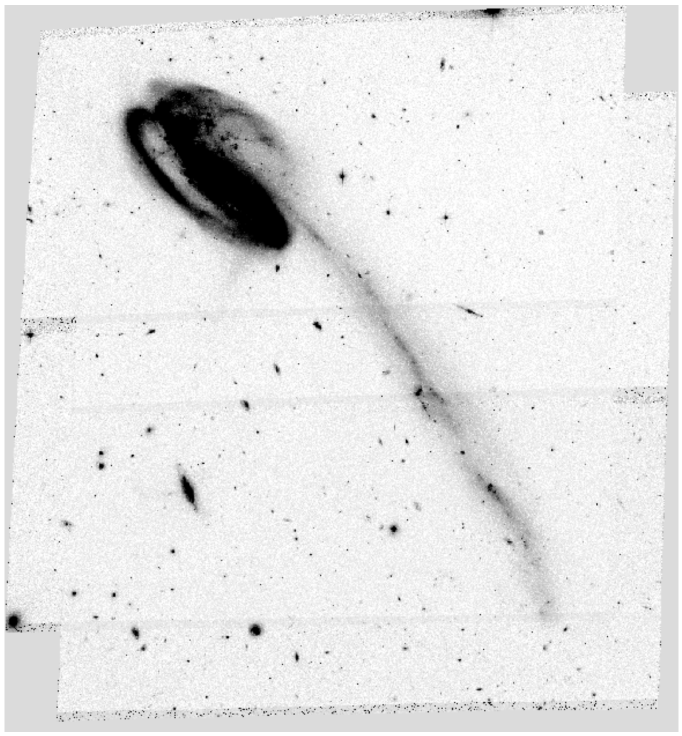
If 'median_newmasks=yes', then the singly drizzled weight maps ('*_single_wht.fits') are used to create pixel masks for each image (with values 0 and 1) which are named '*_single_wht_maskhead.pl'. The IRAF task **mask_head** prepares the singly drizzled images for use with **imcombine** by populating the header bad pixel mask keyword 'BPM' for each image. These masks will be used by **imcombine** when combining images, where the assumed mask parameters are 'masktype=goodvalue' and 'maskvalue=1', indicating that pixels assigned a value of 1 are considered good.

If 'median_newmasks=no', this task will use whatever masks are specified by the user (and which are created offline) in the 'BPM' header keyword of each image. In general, however, it is recommended that the pixel masks which are generated by default are used instead.

Selecting the best parameters for the median step can be an iterative process and should always involve examination of the clean, combined product to verify that the majority of cosmic-rays and other artifacts are successfully removed. The rejection algorithm which is ultimately chosen depends largely on the number of datasets being combined and the amount of overlap between dithered images.

In this example, we have chosen the default parameters 'combine_type=median', 'combine_reject=minmax', and 'combine_nlow=0'. Instead of the default value, we have set 'combine_nhigh=2' so that hot pixels are rejected in the outer portions of the image, which have only 3 input images contributing to the median calculation, and for which 2 of the 3 images are at the same dither position. Thus, the **imcombine** parameter 'nkeep' (the minimum number of pixels retained) is 1 in the outer portion of the median image. In the central regions of the image, six images have contributed to the median, but two high images are rejected, so 'nkeep' is 4. The six separately drizzled images are combined using the bad pixel masks and the rejection parameters specified above to create a single clean median image named 'example_med.fits'. This median image is shown in Figure 4.12.

Figure 4.12: The cleaned median image created using the 6 separately drizzled Tadpole images and their bad pixel masks.



When ‘combine_type=minmed’, a slightly more sophisticated algorithm will be used to combine images. This algorithm requires significantly more computation time and disk space (>1GB) compared to the other available algorithms. The basic concept is that each pixel in the output combined image will be either the median or the minimum of the input pixel values, depending on whether the median is above the minimum by more than *n* times sigma. An estimate of the “true” counts is obtained from the median image (after rejecting the highest-valued pixel), while the minimum is actually the minimum unmasked (“good”) pixel. This algorithm is designed to perform optimally in the case of combining only a few images (3 or 4), where triple-incidence cosmic rays often pose a serious problem for more simplified median combination strategies. The algorithm performs the following steps:

1. Create median image, rejecting the highest pixel and applying masks.
2. Use this median to estimate the true counts, and thus derive an rms.
3. If the median is above the lowest pixel value by less than the first value in ‘combine_nsigma’, then use the median value, otherwise use the lowest value.
4. If ‘combine_grow’ > 0, repeat the above 3 steps for all pixels around those that have already been chosen as the minimum, this time using a lower significance threshold specified as the second value in ‘combine_nsigma’.

The last step is very successful at flagging the lower signal-to-noise “halos” around bright cosmic rays which were flagged in the first pass.

5. Blotting Back the Median Image

Parameters	Default	Description
blot	yes	Blot the median back to the input frame?

Output Files	Images= ‘*_sci?_blt.fits’

When ‘blot=yes’, this task takes the median image and uses the **dither** package **blot** task to apply the geometric distortion and to transform (‘reverse drizzle’) it back to the reference frame of each of the original individual input images. This involves reversing the shifts and reapplying the geometric distortion that had been removed in step 3. In addition, the median image is resampled to the pixel scale of the original images and is trimmed to match the dimensions of each input image. This step is done in preparation for subsequent cosmic-ray rejection in step 6. The blotted frames are named ‘*_sci?_blt.fits’.

If desired, the user may wish to display the input images and blink them with their ‘blotted’ counterparts. The ‘blotted’ images should align perfectly with their respective input images and should be reasonably similar in appearance, except for the fact that they should be cleaned of cosmic rays and other defects.

6. Creating Cosmic Ray Masks

Parameters	Default	Description
driz_cr	yes	Perform CR rejection with deriv and driz_cr?
driz_cr_snr	“3.0 2.5"	Driz_cr.SNR parameter
driz_cr_scale	“1.2 0.7"	Driz_cr.scale parameter

Output Files	Images= ‘*_sci?_blt_deriv.fits’, ‘*sci?.fits’, ‘*_sci?_crderiv.pl’, ‘*_sci?_cor.fits’, ‘*_sci?_mask.pl’

When ‘driz_cr=yes’, this step uses the original input images, the blotted median images, and the derivative of the blotted images (which it creates using the **deriv** task) to create a cosmic ray mask for each input image (using the **driz_cr** task).

First, the **deriv** task uses the blotted median images (‘*_sci?_blt.fits’) from step 5 to calculate the absolute value of the difference between each pixel and its four surrounding neighbors. For each pixel, the largest of these four values is saved in an output image, ‘*_sci?_blt_deriv.fits’, which represents an effective gradient or spatial derivative.

These derivative images are used by the task **driz_cr** when comparing the original and blotted images. First, the original FLT images for each chip are copied to files named ‘*_sci?.fits’ which are the required input for the **driz_cr** task. These images are compared with the corresponding blotted median image ‘*_sci?_blt.fits’ and its absolute derivative ‘*_sci?_blt_deriv.fits’ to create a mask of cosmic rays (and other blemishes, like satellite trails). Where the difference is larger than can be explained by noise statistics, or the flattening effect of taking the median, or perhaps an error in the shift (the latter two effects are estimated using the image derivative), the suspect pixel is masked. Cosmic rays are flagged using the following rule:

|data_image - blotted_image| > scale*deriv_image + SNR*noise

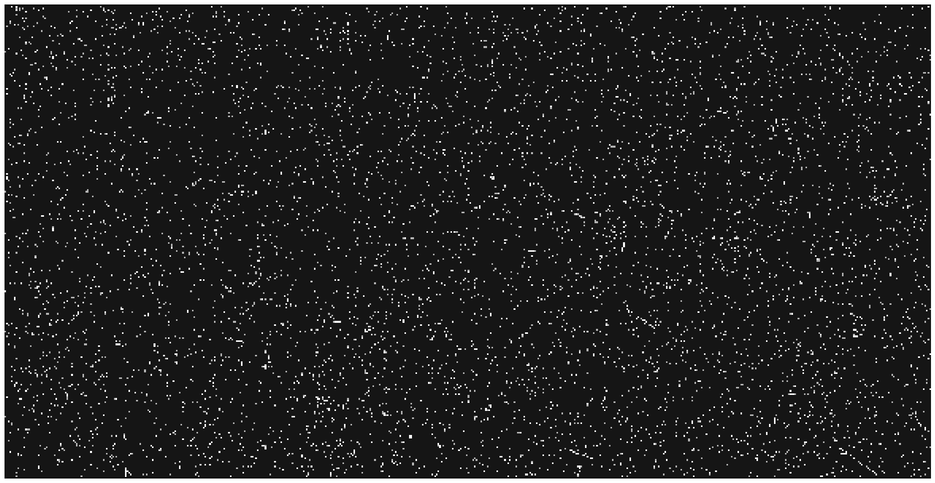
where ‘scale’ is the user supplied **driz_cr** parameter listed above and is defined as the multiplication factor applied to the derivative before determining if the difference between the data image and the blotted image is sufficiently great to require masking. ‘Noise’ is calculated using a combination of the detector read noise and the poisson noise of the blotted median image plus the sky background.

The user must specify a cut-off signal-to-noise (SNR) value for determining whether a pixel should be masked. Actually, two cut-off signal-to-noise ratios are needed, one for detecting the primary cosmic ray, and a second for masking lower-level bad pixels adjacent to those found in the first pass. After the first pass through the image, the procedure is thus repeated on pixels that are adjacent to previously masked pixels using a lower SNR threshold, since cosmic rays often extend across several pixels.

The final output is a cosmic-ray mask file named '*_sci?_crderiv.pl'. One of the resulting masks for chip 1 is shown in Figure 4.13 and should be blinked with the original image j8cw54orq_flt[sci,2] from Figure 4.9 (or the equivalent j8cw54orq_flt_sci2.fits file) to visually ascertain that all cosmic rays were flagged. If it appears that the central pixels of some stars were unnecessarily masked, the 'driz_cr_scale' parameter should be increased. If not enough cosmic rays were masked out, this parameter should be decreased. In this example, the default 'driz_cr_snr' values "3.0 2.5" were too stringent and resulted in flagging the centers of stars and the core of the Tadpole galaxy. Instead, we have increased the default SNR values to "4.0 3.0" to create ideal cosmic ray masks for this data set.

The **driz_cr** task also creates a '*_sci?_cor.fits' image, where flagged pixels are replaced with pixels from the blotted median image. The cosmic ray mask files are then multiplied by the bad pixel masks (which are a combination of the image DQ array and the static masks) to create a final mask file for each input image, '*_sci?_mask.pl', which will be used during final drizzle combination.

Figure 4.13: A single cosmic ray mask 'j8cw54orq_flt_sci2_crderiv.pl'. This mask should be blinked with the original image 'j8cw54orq_flt[sci,2]' from Figure 4.9 or the equivalent 'j8cw54orq_flt_sci2.fits' output file to assure that optimal parameters were chosen in the **driz_cr** task.



7. Performing the Final Drizzle Combination

Parameters	Default	Description
driz_combine	yes	Perform final drizzle image combination?
final_outnx		Size of FINAL output frame x-axis
final_outny		Size of FINAL output frame y-axis
final_kernel	"square"	Shape of kernel function (square, point, gaussian, turbo, tophat, lanczos3)
final_scale	1.0	Linear size of output pixels (relative to input)
final_pixfrac	1.0	Linear size of drop in input pixels
final_rot	0.	Rotation of input image to be applied (degrees anti-clockwise)
final_fillval	INDEF	Value given to undefined output pixels

Output Files	Images= output_sci.fits, output_wht.fits

When 'driz_combine=yes', this step takes the original input images, together with the final cosmic ray masks, and drizzles them all onto a single output image. The standard **drizzle** parameters kernel, scale, pixfrac, and rot can be specified by the user, if desired. By default the scale of the output image is 1.0, but the user is encouraged to experiment with other options (e.g. scale=0.5 and pixfrac=0.7 yields a sharper output PSF).

When the following initial setup parameters are set: 'build=yes' (default) and 'context=yes' (non-default), the final **MultiDrizzle** output image will be a single multi-extension FITS file named 'example_drz.fits'. The format of this file is identical to the DRZ product from **PyDrizzle** which is delivered from the archive (see Section 4.3.3) and which contains the science image in extension 1, the weight image in extension 2, and the context image in extension 3. When 'build=no', these files will be written to separate output files. When the default value 'context=no' is used, no context image is created.

The first extension of the drizzled product contains the science (SCI) image which is corrected for distortion and which is dither-combined (or mosaiced), if applicable. The drizzled SCI image derived from the Tadpole example is presented in Figure 4.14 and is in units of *electrons/sec*. All pixels have equal area on the sky and equal photometric normalization across the field of view, giving an image which is both photometrically and astrometrically accurate for both point and extended sources. The dimensions of the output image are computed on-the-fly by **MultiDrizzle** and the default output plate scale is read from the 'scale' column in the IDCTAB. These parameters, however, may be chosen by the user to best suit the actual data.

The second extension of the output image contains the weight (WHT) image. This image gives the relative weight of the output pixels and, in standard processing using the **MultiDrizzle** defaults, it can be considered an effective exposure time map. The weight image from the example is shown in Figure 4.15, where darker areas have higher weight. The chip edges and gaps are clearly visible, as are column defects and cosmic ray features. The bulk of the image is “dark gray” corresponding to the overlap of all six inputs. In this area the weight value is ~3140, equal to the sum of the exposure times of the six images which contribute. There is also a smooth variation across the image due to the variation of the pixel area on the sky caused by the distortion.

The third extension of the **MultiDrizzle** output image contains the context (CTX) image which encodes information about which input image contributes to a specific output pixel. This is done using a bitmask for each output pixel, where ‘bit set’ means that the image, in the order it was combined, contributed with non-zero weight to that output pixel. The context image starts as a single 32-bit integer image but is extended as a cube with additional 32-bit deep planes as required to handle all the input images. The context image for the Tadpole example is shown in Figure 4.16. As there are six input images, each with two chips which are treated separately, this image has 12 bit planes which may be set. The darkest area shown corresponds to the [sci,2] chip from all six inputs and hence has all the following even bits set: 2+8+32+128+512+2048=2730. Cosmic ray hits or other defective pixels contribute to the appropriate bit plane with zero weight and hence appear as lighter spots.

Figure 4.14: The science (SCI) extension of the drizzled product from the **MultiDrizzle** example. This image has been corrected for distortion and drizzled onto a single mosaic using the six images in the dither pattern.

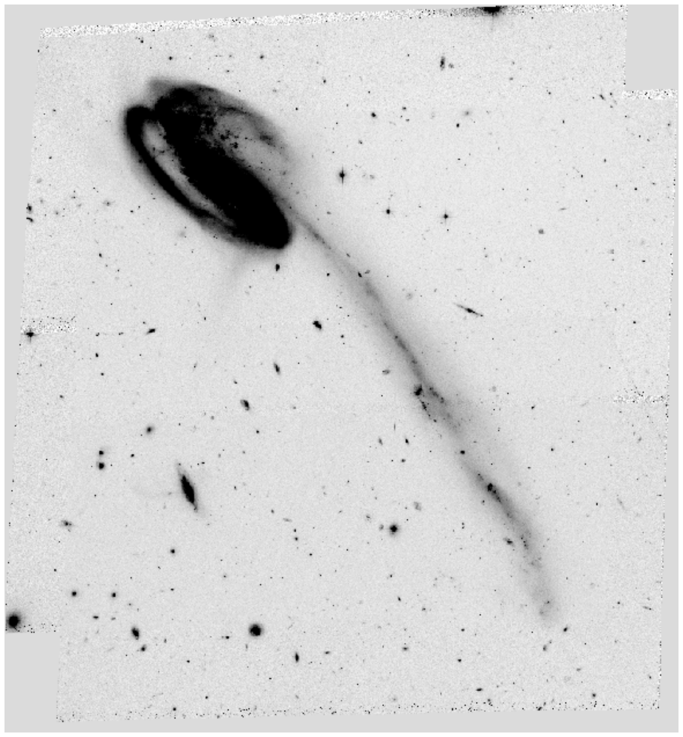


Figure 4.15: The corresponding weight (WHT) extension of the drizzled product from the example.

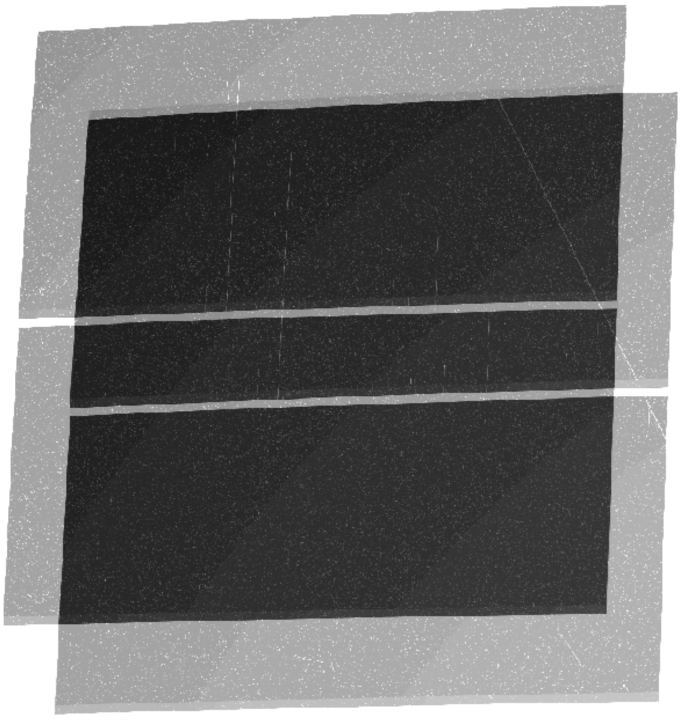
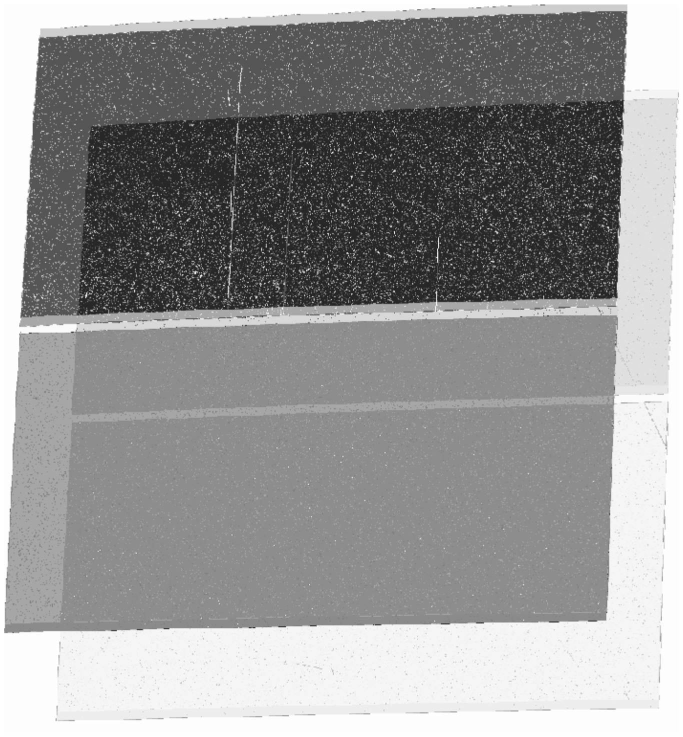


Figure 4.16: The corresponding context (CTX) extension of the drizzled product from the example.



PART III:

Appendixes

Part III:Appendixes

APPENDIX A: IRAF Primer

In this appendix . . .

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 tapes to producing plots and images. Most astronomers will already be familiar with IRAF, but we provide this tutorial for HST observers who are beginners with 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 on-line help facility.

Additional information on IRAF, in particular *A Beginner's Guide to Using IRAF* is available through the NOAO IRAF Home Page at: http://iraf.noao.edu

A.1 Initiating IRAF

This section explains:

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

We assume that your site has IRAF and STSDAS installed. If not, you must obtain and install the software. See Appendix Section A.3 for details.

A.1.1 Setting Up IRAF

Before running IRAF for the first time you need to follow these three steps:

1. Create your 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 set it up in their account's root directory (i.e., the default directory that you are in when you log in to the system). The IRAF home directory doesn't need to be in your account's root 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

Under VMS:

```
$ CREATE/DIR [.IRAF]
$ SET DEFAULT [.IRAF]
$ IRAF
$ MKIRAF
```

Can be placed in LOGIN.COM file

The **mkiraf** command initializes IRAF by creating a `login.cl` file and a subdirectory called `uparm`. After typing the **mkiraf** command, you will see 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. [1] Generic terminal types that will work for most users are:

- `vt100` for most terminals.
- `xtermjhs` for most workstations running under X-Windows.

`xgterm` for sites that have installed X11 IRAF and IRAF v2.10.3 BETA or later.

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 output 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.COM` file used by VMS or the `.login` file used by Unix. Whenever IRAF starts, it looks at 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 format images.

The `uparm` directory will contain your own copies of IRAF task parameters. This directory allows you to customize your IRAF

1. Users at STScI should consult the *STScI Site Guide for IRAF and STSDAS.*

environment by setting certain parameter values as defaults. Once you set up IRAF, you should rarely need to do it again, expect when updated version of IRAF are installed.

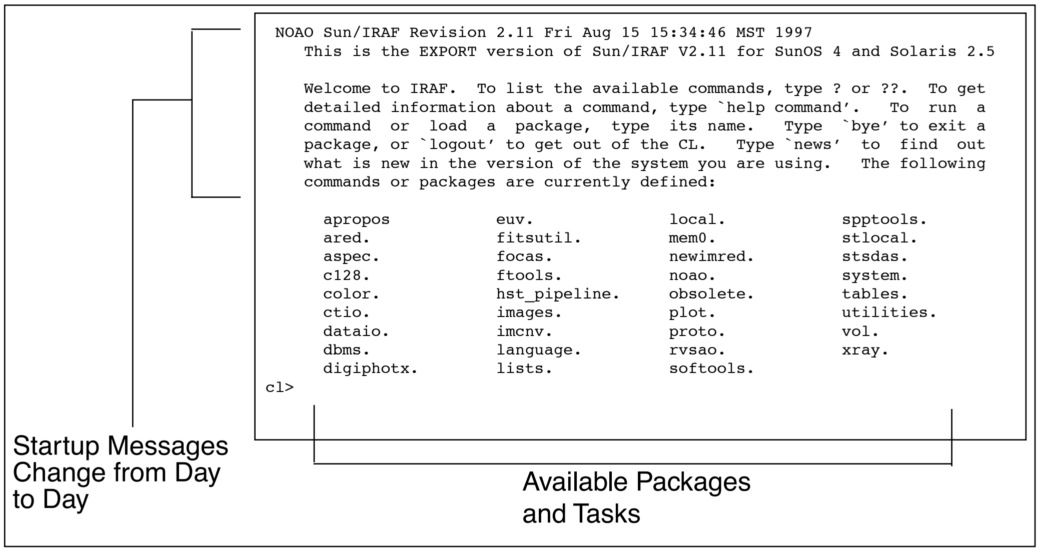
A.1.2 Starting and Stopping an IRAF Session

To start an IRAF session:

1. Move to your IRAF home directory.
2. Type `cl`.

IRAF starts by displaying several lines of introductory text and then puts a prompt at the bottom of the screen. Figure A.1 is a sample IRAF startup screen.

Figure A.1:IRAF Startup Screen



```
  NOAO Sun/IRAF Revision 2.11 Fri Aug 15 15:34:46 MST 1997
     This is the EXPORT version of Sun/IRAF V2.11 for SunOS 4 and Solaris 2.5

     Welcome to IRAF.  To list the available commands, type ? or ??.  To get
     detailed information about a command, type `help command'.   To  run  a
     command  or  load  a  package,  type  its name.   Type  `bye' to exit a
     package, or `logout' to get out of the CL.   Type `news'  to  find  out
     what is new in the version of the system you are using.   The following
     commands or packages are currently defined:

       apropos          euv.             local.           spptools.
       ared.            fitsutil.        mem0.            stlocal.
       aspec.           focas.           newimred.        stsdas.
       c128.            ftools.          noao.            system.
       color.           hst_pipeline.    obsolete.        tables.
       ctio.            images.          plot.            utilities.
       dataio.          imcnv.           proto.           vol.
       dbms.            language.        rvsao.           xray.
       digiphotx.       lists.           softools.
cl>
```

Startup Messages Change from Day to Day
Available Packages and Tasks

To quit an IRAF session:

1. Type `logout`.

A.2 IRAF Basics

This section describes basic IRAF techniques such as:

- Loading packages (below).
- Running tasks and commands.
- Getting online help.
- Viewing and setting parameters (see Appendix Section A.2.4).

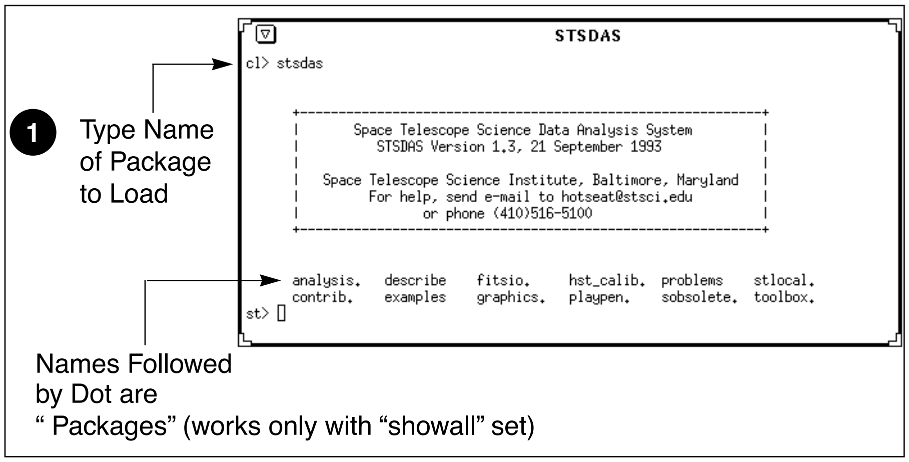
- Setting and using environment variables (see Section A.2.5).
- File management
- Troubleshooting

A.2.1 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.

Note that 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**).

Figure A.2:Loading Packages


STSDAS
1
Type Name of Package to Load
Names Followed by Dot are " Packages" (works only with "showall" set)

```
cl> stsdas

    +------------------------------------------------------------+
    |       Space Telescope Science Data Analysis System         |
    |          STSDAS Version 1.3, 21 September 1993             |
    |                                                            |
    |   Space Telescope Science Institute, Baltimore, Maryland   |
    |          For help, send e-mail to hotseat@stsci.edu        |
    |                 or phone (410)516-5100                     |
    +------------------------------------------------------------+

      analysis.  describe   fitsio.    hst_calib.  problems    stlocal.
      contrib.   examples   graphics.  playpen.    sobsolete.  toolbox.
st> 
```

Some helpful commands for managing packages are:

- `?` - Lists tasks in the most recently-loaded package.
- `??` - Lists all tasks loaded, regardless of package.
- `package` - Lists names of all loaded packages.
- `bye` - Exits the current package.

A.2.2 Running Tasks

This section explains how to run tasks, background 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*, such as the names of input files. Alternatively, you can specify the values for the required *parameters* on the command line when you run the task. For example, if you want the task `imheader` to print header information on the file `myfile.hhh`, you can type

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

IRAF does not require you to type 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 or VMS commands) 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*, and 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 textual output to the screen, you will often 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 this:

```
st> taskname &
```

A.2.3 Getting Help

This section describes:

- How to use IRAF's on-line help facility.
- How to find a task that does what you want (see "Finding Tasks" on page A-8).

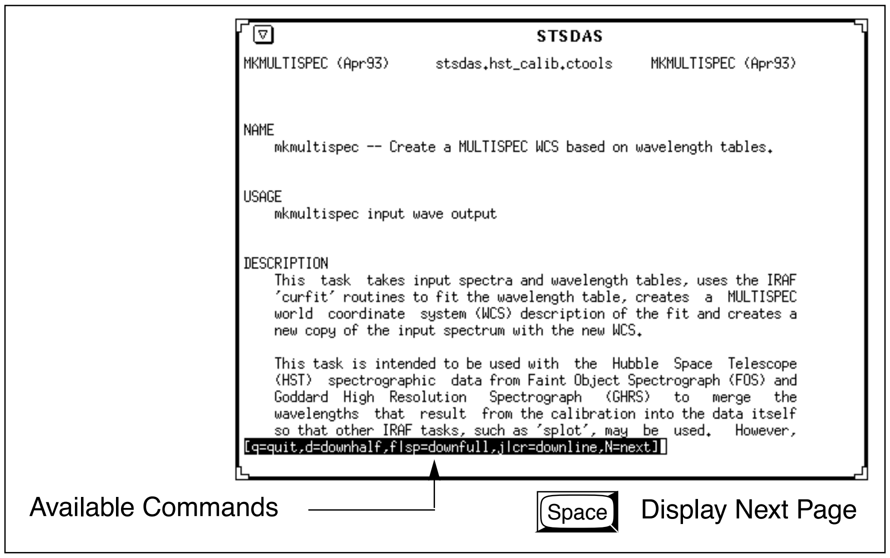
On-Line Help

You can get on-line help with any IRAF task or package by using the **help** command,[2] which takes as an argument the task or package name about which you want help. Wildcards are supported. For example, to display the on-line help for the STSDAS **mkmultispec** task, you would type:

```
fi> help mkmultispec
```

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

Figure A.3:Displaying On-line Help



```
STSDAS
MKMULTISPEC (Apr93)      stsdas.hst_calib.ctools     MKMULTISPEC (Apr93)


NAME
    mkmultispec -- Create a MULTISPEC WCS based on wavelength tables.


USAGE
    mkmultispec input wave output


DESCRIPTION
    This  task  takes input spectra and wavelength tables, uses the IRAF
    'curfit' routines to fit the wavelength table, creates  a  MULTISPEC
    world  coordinate  system (WCS) description of the fit and creates a
    new copy of the input spectrum with the new WCS.

    This task is intended to be used with  the  Hubble  Space  Telescope
    (HST)  spectrographic  data from Faint Object Spectrograph (FOS) and
    Goddard  High  Resolution  Spectrograph   (GHRS)   to   merge   the
    wavelengths  that  result  from the calibration into the data itself
    so that other IRAF tasks, such as 'splot', may  be  used.   However,
[q=quit,d=downhalf,f|sp=downfull,j|cr=downline,N=next]
```

Two STSDAS tasks that display only certain sections of the help file are also available:

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

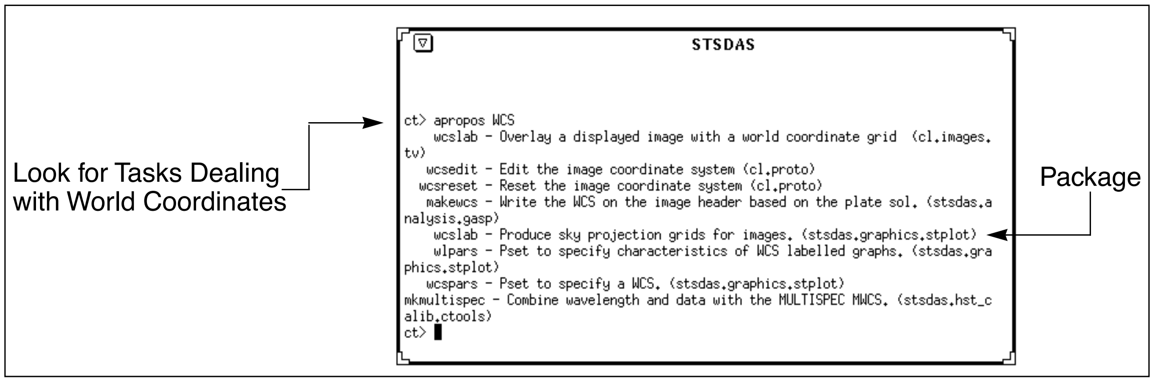
Typing `help` *package* will produce one-line descriptions of each task in the package.

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.
- Use the **apropos** task as shown in Figure A.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.
- Ask more experienced user, who can usually point you in the right direction.

Figure A.4:The **apropos** task Using apropos



```
ct> apropos WCS
    wcslab - Overlay a displayed image with a world coordinate grid  (cl.images.
tv)
   wcsedit - Edit the image coordinate system (cl.proto)
  wcsreset - Reset the image coordinate system (cl.proto)
   makewcs - Write the WCS on the image header based on the plate sol. (stsdas.a
nalysis.gasp)
    wcslab - Produce sky projection grids for images. (stsdas.graphics.stplot)
    wlpars - Pset to specify characteristics of WCS labelled graphs. (stsdas.gra
phics.stplot)
   wcspars - Pset to specify a WCS. (stsdas.graphics.stplot)
mkmultispec - Combine wavelength and data with the MULTISPEC MWCS. (stsdas.hst_c
alib.ctools)
ct> █
```

A.2.4 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.

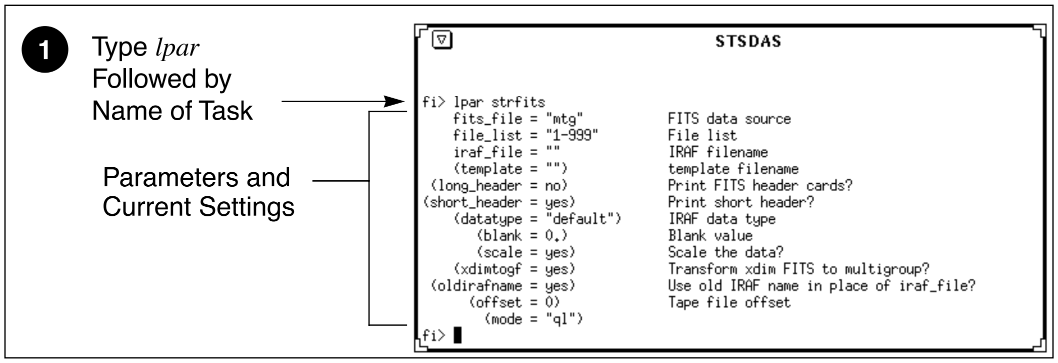
The two most useful commands for handling parameters are:

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

Viewing Parameters with lparam

The **lpar** command lists the current parameter settings for a given task (Figure A.5).

Figure A.5:Displaying Parameter Settings with lpar

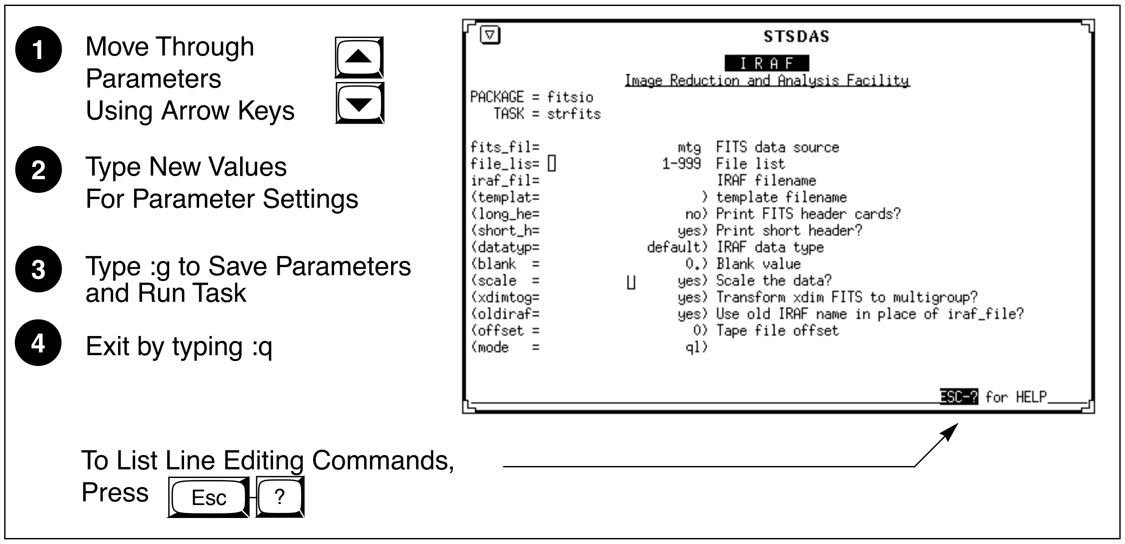

1
Type *lpar* Followed by Name of Task
Parameters and Current Settings
STSDAS

```
fi> lpar strfits
    fits_file = "mtg"          FITS data source
    file_list = "1-999"        File list
    iraf_file = ""             IRAF filename
    (template = "")            template filename
 (long_header = no)            Print FITS header cards?
(short_header = yes)           Print short header?
    (datatype = "default")     IRAF data type
       (blank = 0.)            Blank value
       (scale = yes)           Scale the data?
    (xdimtogf = yes)           Transform xdim FITS to multigroup?
 (oldirafname = yes)           Use old IRAF name in place of iraf_file?
      (offset = 0)             Tape file offset
        (mode = "ql")
fi> █
```

Setting parameters with eparam

The **epar** command is an interactive parameter set editor. It displays all of the parameters and their current settings on the screen. You can move around the screen using the arrow keys (also called *cursor* keys) and type new settings for any parameters you wish to change. Figure A.6 shows a sample of the **epar** editor at work (invoked by typing `epar strfits`).

Figure A.6:Editing Parameters with epar


1 Move Through Parameters Using Arrow Keys
2 Type New Values For Parameter Settings
3 Type :g to Save Parameters and Run Task
4 Exit by typing :q
To List Line Editing Commands, Press Esc ?

```
                              STSDAS
                              I R A F
                Image Reduction and Analysis Facility
PACKAGE = fitsio
   TASK = strfits

fits_fil=                  mtg  FITS data source
file_lis=               1-999  File list
iraf_fil=                      IRAF filename
(templat=                    ) template filename
(long_he=                  no) Print FITS header cards?
(short_h=                 yes) Print short header?
(datatyp=             default) IRAF data type
(blank  =                  0.) Blank value
(scale  =                 yes) Scale the data?
(xdimtog=                 yes) Transform xdim FITS to multigroup?
(oldiraf=                 yes) Use old IRAF name in place of iraf_file?
(offset =                   0) Tape file offset
(mode   =                  ql)

                                                  ESC-? for HELP
```

Parameter Data 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 [include reference to help]. 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 A.1 lists the different parameter types.

Table A.1: Parameter Data Types

Type	Description
`File Name`	Full name of the file. Wild card characters (* and ?) are often allowed. Some tasks allow you to use special features when specifying file names, including “@” lists, IRAF networking syntax, and image section or group syntax. (See “File Management” below).
`Integer`	Whole number. Often the task will specify minimum or maximum values (see 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 string.
`Pset`	Parameter set.

Restoring Parameter Default Values

Occasionally, IRAF (or you) will get confused by your parameter values. To alleviate this confusion, you can restore the default parameters with the **unlearn** command. You can use **unlearn** on either a task or on an entire package.

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.

A.2.5 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 through environment variables. Environment variables are set using the **set** command and are displayed using the **show** command. Table A.2 lists some of the environment variables that you might want to customize.

Table A.2: Environment Variables

Variable	Description	Example of Setting
`printer`	Default printer for text	`set printer = lp2`
`terminal`	Terminal type	`set term = xterm`
`stdplot`	Default printer for all graphics output	`set stdplot = ps2`
`stdimage`	Default terminal display setting for image output (most users will want this set to either imt512 or imt800)	`set stdimage = imt800`
`stdgraph`	Default graphics device	`set stdgraph = xterm`
`clobber`	Allow or prevent overwriting of files	`set clobber = yes`
`imtype`	Default image type for output images. "imh" is original IRAF format, "hhh" is STSDAS GEIS format.	`set imtype = "hhh"`

If you are working with GEIS files, you should set imtype to "hhh". If you are working with STIS and NICMOS data in FITS files, you can set imtype to "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.

A.2.6 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 of the HST Introduction), 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 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*,[3] consisting of pixel information. IRAF data file names end with a `.pix` suffix. STSDAS data files end with an suffix of `.hhd` or another suffix that ends with "d", such as `.c0d` or `.q0d`.

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.

When working with IRAF or STSDAS 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. Several types of 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.: `z*.c0h`
  - ? Matches any single character, e.g.: `z01x23x.c?h`

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

When using wildcards 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

- **List files**, often called *@-files*, which are ASCII file 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: `@files.txt`
- **Image section** specification. Tasks that work with image data will often let you specify that you want to work on only a small area 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 Users 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: `ra!mta`.

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.

A.2.7 Troubleshooting

There are a couple of easy things you can do to make sure that you don't 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 twice in succession.
- Occasionally, you may need to logout of the CL, restart IRAF, and try your command again.

If you still have a problem, contact the STScI Help Desk at `help@stsci.edu`

A.3 Getting IRAF and STSDAS

Both IRAF and STSDAS are provided free of charge to the astronomical community. You must have IRAF to run STSDAS. Detailed information about installing and retrieving STSDAS is found in the *STSDAS Site Manager's Installation Guide and Reference*. If you have any problems getting and installing STSDAS, TABLES, or any other packages or data described in this handbook, please contact the Help Desk by sending e-mail to: `help@stsci.edu`.

A complete description of how to install the **synphot** data files is provided in Section A.3.2.

A.3.1 Retrieving the IRAF and STSDAS Software

There are three ways to get the software:

- Use the World Wide Web.
- Use anonymous FTP.
- Request a tape.

World Wide Web

The STSDAS World Wide Web page:

`http://stsdas.stsci.edu/STSDAS.html`

provides links and instructions for downloading the appropriate files to your local system or to display the software directory, from which you can select the series of smaller files.

Anonymous FTP

- **IRAF**: `iraf.noao.edu` (140.252.1.1)
- **STSDAS**: `ftp.stsci.edu` (130.167.1.2)

There are two points to remember when using FTP to retrieve STSDAS:

- You must retrieve and install the TABLES package before STSDAS.
- You should retrieve the `README` file from the directory `/software/ stsdas/v2.0` and read it to find out which files you should retrieve.

You must have IRAF installed on your system to install TABLES and STSDAS. When you retrieve STSDAS, you must also retrieve the TABLES package, and TABLES must be installed first.

Instructions for installing STSDAS are available in the `doc` subdirectory of the directory where you find STSDAS. The complete instructions for installing STSDAS, TABLES, and all of the supporting software and reference files (including instrument reference files and the **synphot** dataset) are found in the *STSDAS Site Manager's Installation Guide and Reference*.

Registration

The software can also be registered and requested using on-line forms available through World Wide Web at the following URL:

`http://stsdas.stsci.edu/RegistForm.html`

When you request the STSDAS software, you can also ask for the appropriate version of IRAF, which will be requested for you— simply check the appropriate box on the form under "Do You Already Have IRAF Installed?" If you prefer to request the IRAF software independent of STSDAS, you can do so by sending e-mail to: `iraf@iraf.noao.edu`

A.3.2 Getting the Synphot Database

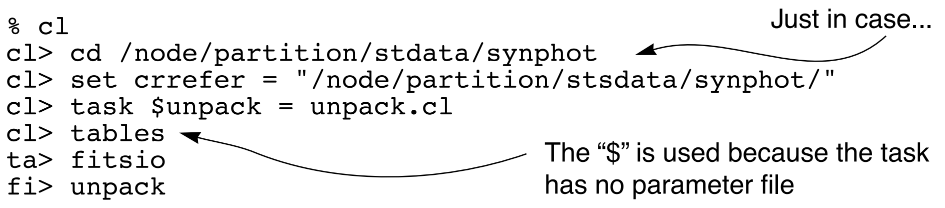
This manual sometimes refers to the **synphot** dataset, which must be available in order to run tasks in the STSDAS **synphot** package. These data files are not included with the STSDAS software and must be retrieved independently. To do this, you need to retrieve a series of compressed tar files from the STScI FTP site (ftp.stsci.edu) in the directory `software/stsdas/refdata/synphot`. After uncompressing and extracting the tar files (see below), you need to unpack the FITS files as described below.

The synthetic photometry data are read in similar way as the instrument datasets, using the script `unpack.cl` provided in the top directory. This script is run within IRAF to convert data from FITS format into the format used by the **synphot** task. This script assumes you have the logical `crrefer` set up in your `extern.pkg` file (which is in the directory `$iraf/unix/hlib` (Unix) or `$iraf/vms/hlib` (VMS)) or have it set up in your session. You do this by placing the command below in `extern.pkg` or by typing it on the command line:

```
set crrefer = "/node/partition/stdata/synphot/"
```

Figure A.7 shows how to convert the files.

Figure A.7:Unpacking Synthetic Photometry Files



```
% cl
cl> cd /node/partition/stdata/synphot
cl> set crrefer = "/node/partition/stsdata/synphot/"
cl> task $unpack = unpack.cl
cl> tables
ta> fitsio
fi> unpack
```

Note that all three synphot files must be unloaded for the script to complete successfully.

A.3.3 Extracting the Synphot Unix Tar Files

If you retrieved the **synphot** database as compressed tar files, you will need to copy them to an appropriate subdirectory and then expand and unpack the files. The tar and compress utilities that do this are are commonly available on most Unix systems, but are not standard in the VMS environment. The examples shown below reflect Unix usage. If you are on a VMS system, you should consult with your systems support staff regarding the availability and usage of these commands. To process the files on a Unix system:

1. Get the compressed tar file that you want, as described in previous sections.
2. Make an appropriate subdirectory using the `mkdir` command.
3. Pipe the compressed tar file through the `uncompress` and `tar` files to expand and unpack the file.

The following example shows how to do this. The example assumes that you are putting the files in a subdirectory under `/usr/iraf/stdata` (note that the name of your file here is assumed to be `XXX.tar.Z`).

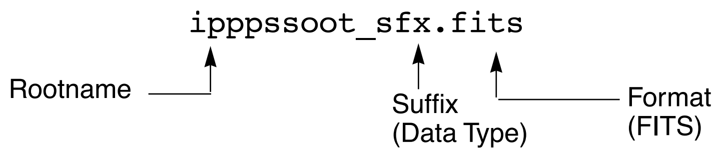
```
% pwd
/usr/iraf/stdata
% mkdir XXX
% mv XXX.tar.Z XXX/
% cd XXX
% cat XXX.tar.Z | uncompress | tar -xf -
```

APPENDIX B:

HST File Names

In this appendix. . .

This appendix describes the syntax of HST data file names, which encode a large amount of information about the files themselves. Datasets retrieved from the Archive as described in consist of multiple files in 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 three-character second part of the name (`sfx`) is called the *suffix*, and it indicates the type of data the file contains.
- ***Format***: The identifier `.fits` indicates that this file is in FITS format.

For example, an FOC data file named `x3l80101t_d0f.fits` is a FITS file belong to the dataset with rootname `x3l80101t`, and its suffix `d0f` indicates that it contains raw science data.

In order to use IRAF/STSDAS tasks to work with data from instruments other than NICMOS and STIS, you will want to convert these FITS files into GEIS format. See Section 2.2 in the HST Introduction for instructions on how to convert FITS files to GEIS files using **strfits**. Like FITS files, the names of GEIS files also derive from a file's rootname and suffix, and they look like this:

```
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 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 in the HST Introduction). For clarity, this handbook will use the term "extension" when refering to a component of a FITS file and the term "suffix" when referring to the three character identifier in a filename.

B.1 Rootnames

Rootnames of HST data files follow the naming convention defined in Table B.1, which expands on the previous convention as follows: an initial "N" indicates a NICMOS exposure, an intial "O" indicates a STIS exposure, and the rootnames of files containing association products (see below) end in a number (0-8).

Table B.1: IPPPSSOOT Root File Names

Character	Meaning
`I`	Instrument used, will be one of: ***E*** - Engineering data ***F*** - Fine Guidance Sensors ***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 combinations possible). 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 ***R*** - Real time (not recorded) ***T*** - Tape recorded ***0*** - Primary association product (number zero) ***1-8*** - NICMOS background association product

B.2 Suffixes of Files Common to all Instruments

The three-character suffix of a data file (e.g., `d0h`) identifies the type of data that a file contains. Because the meanings of these suffixes change from instrument to instrument, please refer to the appropriate instrument-specific Data Structures chapter for their definitions. Several types of file suffixes are, however, common to all instruments.

OMS Files

Observatory Monitoring System (OMS) files, having suffixes `cm*` or `ji*`, contain Observation Logs describing how the HST spacecraft behaved during a given observation. OMS headers, which you can read with the IRAF task **imheader** (see Section 2.3.3 in the HST Introduction), 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.OMS tables and images record spacecraft pointing information as a function of time. For more information on OMS files, you can consult Appendix C or the STScI Observation Logs WWW pages at:

http://www.stsci.edu/ftp/instrument_news/Observatory/obslog/OL_1.html

PDQ Files

The suffix `pdq` denotes Post Observation Summary and Data Quality Comment files—*PDQ 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 OMS files.

OCX Files

The suffix `ocx` denotes Observer Comment Files—*OCX files*—which are 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).

Trailer Files

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

Note that trailer files are formatted with 132 columns

B.3 Associations

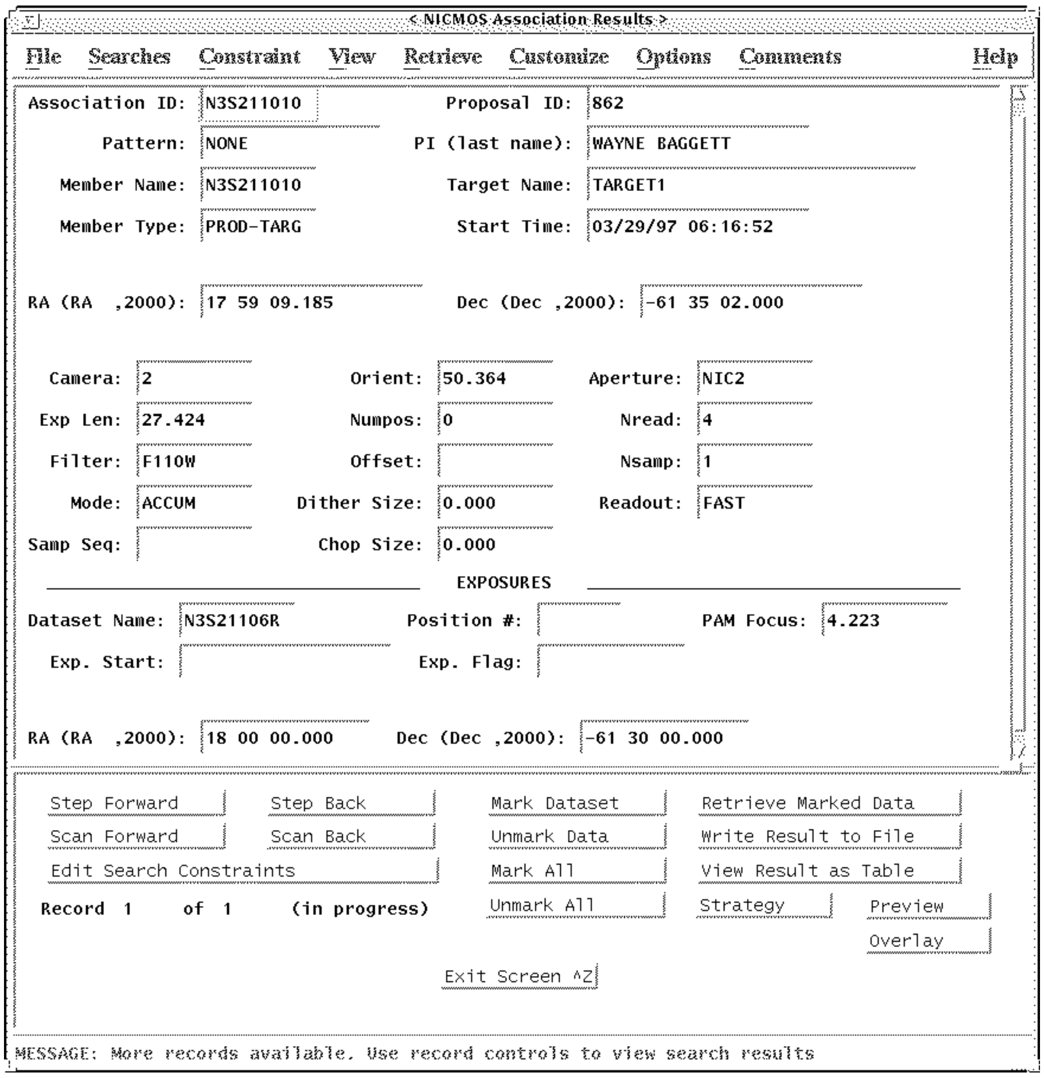
The STIS and NICMOS calibration pipelines sometimes produce single calibrated images from *associations* of many exposures. These associations allow HST pipeline processing to proceed further than it has in the past. For example, a NICMOS observer might specify a dithering pattern in a Phase II proposal. NICMOS would then take several exposures at offset positions, and the pipeline would combine them into a single mosaic. 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 single image, free of cosmic rays, that would be the association product.

When you search the Archive with StarView for observations involving associations of exposures, your search will identify the final association product. The rootnames of association products always end in zero (see Figure B.1 above.) If you request both Calibrated and Uncalibrated data from the Archive, you will receive both the association product and the exposures that went into making it. The corresponding association table, located in the file with suffix `asn` and the same rootname as the association product, lists the exposures belonging to the association. You can read this file using the STSDAS **tprint** or **tread** tasks (see Table 3.1 in the HST Introduction). 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 (see Figure B.1).

In practice, STIS and NICMOS store the exposures belonging to associations differently. The exposures belonging to a STIS association all reside in the same file, while the exposures belonging to a NICMOS association reside in separate datasets. See the relevant Data Structures chapters for more details.

Information on the exposures belonging to an association is also available through StarView (see chapter 1 of the HST Introduction). From the <Welcome> Screen, click on **[HST Instrument Searches]** to get the <HST Instruments> screen, and then click on the **[Associations]** button for the instrument of interest. You can then search for the various exposures belonging to an association by entering the rootname of the association in the Association ID field and clicking on **[Begin Search]** . An Association Results Screen will display the results of the search, which you can step though using the **[Step Forward]** button. Figure B.1 below gives an example of a NICMOS Association Results Screen. Note the differences between the association rootname and coordinates and those of the individual exposure.

Figure B.1:Association Results Screen from StarView



Association ID: N3S211010
Proposal ID: 862
Pattern: NONE
PI (last name): WAYNE BAGGETT
Member Name: N3S211010
Target Name: TARGET1
Member Type: PROD-TARG
Start Time: 03/29/97 06:16:52

RA (RA ,2000): 17 59 09.185
Dec (Dec ,2000): -61 35 02.000

Camera: 2
Orient: 50.364
Aperture: NIC2
Exp Len: 27.424
Numpos: 0
Nread: 4
Filter: F110W
Offset:
Nsamp: 1
Mode: ACCUM
Dither Size: 0.000
Readout: FAST
Samp Seq:
Chop Size: 0.000

EXPOSURES

Dataset Name: N3S21106R
Position #:
PAM Focus: 4.223
Exp. Start:
Exp. Flag:

RA (RA ,2000): 18 00 00.000
Dec (Dec ,2000): -61 30 00.000

Record 1 of 1 (in progress)

MESSAGE: More records available. Use record controls to view search results

APPENDIX C:

Observation Logs

In this appendix . . .

This Appendix describes the *Observation Log Files,* also known as OMS or *jitter* files. These 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.

C.1 Observation Log Files

Observation log files associated with each HST dataset contain pointing and specialized engineering data taken during the observation. These data files are produced by the Observatory Monitoring System (OMS), 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. This system reports the status of the instruments and observatory and flags discrepancies between planned and executed actions. OMS 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 appendix B for more on the names of HST data files). When OMS was installed in October 1994, it initially generated files with the suffixes cmh, cmj, cmi, which contained header information, high time resolution pointing data, and three-second average pointing data, respectively (see Table C.1). OMS observation logs changed to the jih/jid/jif image format after August 1995, at which time the cmi table was renamed jit to keep the naming convention consistent. In the OMS version of August 1995, cmj tables were replaced with a jitter image, which is a two-dimensional histogram of jitter excursions during the observation. The suffixes of the GEIS jitter image are jih for the header and jid for the image data. The jit table accompanies the jitter image. The header file of the image replaces the cmh file but includes the same information with the addition of some image-related keywords.

A detailed description of the observation log files can be found on-line: http://www.stsci.edu/instruments/observatory/obslog/OL_1.html.

Table C.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)

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.

C.1.1 Observation Log File Contents (October 1994 version)

Observation logs created between October 1994 and August 1995 contain:

- *rootname*`j.cmh`: This ASCII header file contains the time interval, the rootname, averages of the pointing and spacecraft jitter, the guiding mode, guide star information, and alert or failure keywords. Figure C.1 shows a representative observation log header file.
- *rootname*`j.cmj`: This table presents the data at the highest time resolution for the telemetry mode in use. It contains the reconstructed pointing, guide star coordinates, derived jitter at the instrument aperture, and guiding-related flags. The intent is: (1) to provide high-time resolution jitter data for deconvolution or for assessing small aperture pointing stability, and (2) to display the slew and tracking anomaly flags with the highest resolution. Table C.2 lists the table column heading, units and a brief definition.
- *rootname*`j.cmi`: This table contains data that were averaged over three-second intervals. It includes the same information as the `.cmj` table and also includes orbital data (e.g., latitude, longitude, limb angle, magnetic field values, etc.) and instrument-specific items. It is best suited for a quick-look assessment of pointing stability and for studying trends in telescope or instrument performance with orbital environment. Table C.3 lists the table column heading, units and a brief definition.
- rootnamej_cmi`/j/h.fits`: The above three GEIS files are actually archived as FITS files. They may be worked with as such, or run through the STSDAS task **strfits**, to convert them.
- Observation Log File Contents (August 1995 version)
- The contents of observation log files created between August 1995 and February 1997 are as follows:

- *rootname*`j.jih`: This GEIS header file, the analog to the `cmh` file, contains the time interval, the rootname, averages of the pointing and spacecraft jitter, the guiding mode, guide star information, and alert or failure keywords. Figure C.1 shows a representative observation log header file.
- *rootname*`j.jid`: This GEIS image—a significant enhancement of the old `cmj` file—presents a two-dimensional histogram of the pointing fluctuations during the observation. You can display it to visualize the spacecraft stability during you observation, and is information for deconvolutions and PSF analyses.
- *rootname*`j.jit`: This table, the analog to the `cmi` table, contains data that were averaged over three-second intervals. Its content is identical (see Table C.3).
- *rootname*`j_jif.fits`: FITS file that is actually the de-archived product. This file can be converted to the `jih/jid` GEIS file via the **strftis** routine.
- *rootname*`j_jit.fits`: The de-archived FITS file corresponding to the `jit` IRAF table. It can be converted via **strfits**.

C.1.2 Observation Log File Contents (February 1997 version)

The contents of observation log files created since February 1997 are as follows:

- *rootname*`j_jif.fits`: The de-archived FITS file. Unlike the previous OMS epoch, this FITS file does not bundle a GEIS file and cannot be converted with **strfits**. This was done to more closely correlate the observation log files with the STIS and NICMOS FITS files with extensions and associations. OMS will normally put all associated observation logs into a single file, to correspond to the associated science exposures. However, if even one science exposure is orphaned (not associated) then an individual observation log FITS file will be produced for every exposure in that association. For a description of STIS and NICMOS association files, see appendix B. All of the information contained in the old `cmh/jih` ASCII header is now available as keywords in the FITS files.
- *rootname*`j_jit.fits`: The FITS file containing the table information. The comments for the `_jif` file apply here as well.

Table C.2: Contents of `.cmj` Table

Parameter	Units	Description
`seconds`	seconds	Time since window start
`V2 dom`	arcseconds	Dominant FGS V2 coordinate
`V3 dom`	arcseconds	Dominant FGS V3 coordinate
`V2 roll`	arcseconds	Roll FGS V2 coordinate
`V3 roll`	arcseconds	Roll FGS V3 coordinate
`SI V2`	arcseconds	Jitter at aperture reference
`SI V3`	arcseconds	Jitter at aperture reference
`RA`	degrees	Right ascension of aperture reference
`DEC`	degrees	Declination of aperture reference
`Roll`	degrees	Angle between North and +V3
`DayNight`	0,1 flag	Day (0) or night (1)
`Recenter`	0,1 flag	Recentering status
`TakeData`	0,1 flag	Vehicle guiding status
`SlewFlag`	0,1 flag	Vehicle slewing status

Figure C.1:A Representative `.jih` or `.cmh` Header

```
dSIMPLE  =                    F / data conforms to FITS standard             !
BITPIX  =                    32 / bits per data value                        !
DATATYPE= 'INTEGER*4          ' / datatype of the group array                !
NAXIS   =                     2 / number of data axes                        !
NAXIS1  =                    64 / length of the 1st data axis                !
NAXIS2  =                    64 / length of the 2nd data axis                !
GROUPS  =                     T / image is in group format                   !
GCOUNT  =                     1 / number of groups                           !
PCOUNT  =                     0 / number of parameters                       !
PSIZE   =                     0 / bits in the parameter block                !
OMS_VER = '16.2C              ' / OMS version used to process this observation
PROCTIME= '1994.133:06:24:18.35' / date-time OMS processed observation
                                / date-times format (yyyy.ddd:hh:mm:ss.ss)

                                / IMAGE PARAMETERS
CRVAL1  =                   0.0 / right ascension of zero-jitter pixel (deg)
CRVAL2  =                   0.0 / declination of zero-jitter pixel (deg)
CRPIX1  =                    32 / x-coordinate of zero-jitter pixel
CRPIX2  =                    32 / y-coordinate of zero-jitter pixel
CTYPE1  = 'RA---TAN           ' / first coordinate type
CTYPE2  = 'DEC--TAN           ' / second coordinate type
CD1_1   =                   0.0 / partial of ra w.r.t. x (deg/pixel)
CD1_2   =                   0.0 / partial of ra w.r.t. y (deg/pixel)
CD2_1   =                   0.0 / partial of dec w.r.t. x (deg/pixel)
CD2_2   =                   0.0 / partial of dec w.r.t. y (deg/pixel)
COORDSYS= 'WFPC2              ' / image coordinate system
XPIXINC =                   2.0 / plate scale along x (mas per pixel)
YPIXINC =                   2.0 / plate scale along y (mas per pixel)
PARITY  =                    -1 / parity between V2V3 frame and image frame
BETA1   =                134.72 / angle from +V3 to image +x (toward +V2)
BETA2   =                224.72 / angle from +V3 to image +y (toward +V2)

                                / OBSERVATION DATA
PROPOSID=                 05233 / PEP proposal identifier
PROGRMID= '288                ' / program id (base 36)
OBSET_ID= '02                 ' / observation set id
OBSERVTN= '03                 ' / observation number (base 36)
TARGNAME= 'NGC3379-PO         ' / proposer's target name
STARTIME= '1994.133:06:24:18.35' / predicted observation window start time
ENDTIME = '1994.133:06:39:18.35' / predicted observation window end time
SOGSID  = 'U2880203           ' / SOGS observation name                      !

                                / SCIENTIFIC INSTRUMENT DATA
CONFIG  = 'WFPC2              ' / proposed instrument configuration
PRIMARY = 'SINGLE             ' / single, parallel-primary, parallel-secondary
OPERATE = '1994.133:06:22:46.91' / predicted time instr. entered operate mode
TLMFORM = 'PN                 ' / telemetry format
APERTURE= 'UWFALL             ' / aperture name
APER_V2 =                 1.565 / V2 aperture position in vehicle frame (arcsec)
APER_V3 =                 7.534 / V3 aperture position in vehicle frame (arcsec)

                                / SPACECRAFT DATA
ALTITUDE=                593.23 / average altitude during observation (km)
LOS_SUN =                106.08 / minimum line of sight to Sun (deg)
LOS_MOON=                 77.11 / minimum line of sight to Moon (deg)
SHADOENT= '1994.133:05:11:29.00' / predicted Earth shadow last entry
SHADOEXT= '1994.133:05:42:45.00' / predicted Earth shadow last exit
LOS_SCV =                 12.46 / minimum line of sight to S/C veloc. (deg)
LOS_LIMB=                  58.0 / average line of sight to Earth limb (deg)


                                / BACKGROUND LIGHT
ZODMOD  =                  22.3 / zodiacal light - model (V mag/arcsec2)
EARTHMOD=                  20.2 / peak Earth stray light - model (V mag/arcsec2)
MOONMOD =                  35.5 / moon stray light - model (V mag/arcsec2)
GALACTIC=                  -1.0 / diffuse galactic light - model (V mag/arcsec2)

                                / POINTING CONTROL DATA
GUIDECMD= 'FINE LOCK          ' / commanded guiding mode
GUIDEACT= 'FINE LOCK          ' / actual guiding mode at end of GS acquisition
GSD_ID  = '0084900235         ' / dominant guide star id
GSD_RA  =             161.70720 / dominant guide star RA (deg)
GSD_DEC =              12.45407 / dominant guide star DEC (deg)
```

Figure C.2:Representative `.jih` or `.cmh` Header

```
GSD_MAG =               12.867 / dominant guide star magnitude
GSR_ID  = '0085201189        ' / roll guide star id
GSR_RA  =            161.93314 / roll guide star RA (deg)
GSR_DEC =             12.78141 / roll guide star DEC (deg)
GSR_MAG =               12.977 / roll guide star magnitude
GSACQ   = '1994.133:06:31:02.92' / actual time of GS acquisition completion
PREDGSEP=             1420.775 / predicted guide star separation (arcsec)
ACTGSSEP=             1421.135 / actual guide star separation (arcsec)
GSSEPRMS=                  3.8 / RMS of guide star separation (milli-arcsec)
NLOSSES =                    0 / number of loss of lock events
LOCKLOSS=                  0.0 / total loss of lock time (sec)
NRECENT =                    0 / number of recentering events
RECENTR =                  0.0 / total recentering time (sec)

                               / LINE OF SIGHT JITTER SUMMARY
V2_RMS  =                  4.5 / V2 axis RMS (milli-arcsec)
V2_P2P  =                 51.6 / V2 axis peak to peak (milli-arcsec)
V3_RMS  =                 20.9 / V3 axis RMS (milli-arcsec)
V3_P2P  =                267.3 / V3 axis peak to peak (milli-arcsec)
RA_AVG  =            161.85226 / average RA (deg)
DEC_AVG =             12.58265 / average dec (deg)
ROLL_AVG=            293.01558 / average roll (deg)

                               / PROBLEM FLAGS, WARNINGS and STATUS MESSAGES
                               / (present only if problem exists)
ACQ2FAIL= '                 T' / target acquisition failure
GSFAIL  = 'DEGRADED          ' / guide star acquisition failure  (*1)
TAPEDROP= '                 T' / possible loss of science data
TLM_PROB= '                  ' / problem with the engineering telemetry
TM_GAP  =               404.60 / duration of missing telemetry (sec)
SLEWING = '                 T' / slewing occurred during this observation
TAKEDATA= '                 F' / take data flag NOT on throughout observation
SIPROBnn= '                  ' / problem with specified science instrument (*2)

END
-----------------------------------------------------------------------------

notes
*1 - GSFAIL appears only once in a single header file.
     The following table lists all current possible values for the GSFAIL
     keyword:
     -------------------------------------------------------------
     GSFAIL    |DEGRADED          | / guide star acquisition failure
               |IN PROGR          | / guide star acquisition failure
               |SSLEXP            | / guide star acquisition failure
               |SSLEXS            | / guide star acquisition failure
               |NOLOCK            | / guide star acquisition failure
               |                  |
               |SREXCS?           | / guide star acquisition failure
               |SREXCS1           | / guide star acquisition failure
               |SREXCS2           | / guide star acquisition failure
               |SREXCS3           | / guide star acquisition failure
               |                  |
               |SREXCP?           | / guide star acquisition failure
               |SREXCP1           | / guide star acquisition failure
               |SREXCP2           | / guide star acquisition failure
               |SREXCP3           | / guide star acquisition failure
               |                  |
               |UNKNOWN           | / guide star acquisition failure
               |VEHSAFE           | / guide star acquisition failure
     -------------------------------------------------------------

*2 - The SIPROBnn keywords appear in the header file with nn = 01 - 99.
     The following table lists all current possible values for the SIPROBnn
     keyword:
     -------------------------------------------------------------------------
     SIPROBnn  |DCF_NUM unchanged| / This observation may not have been taken
               |FOS Safing!       | / This observation affected when FOS Safed!
               |HRS Safing!       | / This observation affected when HRS Safed!
               |WFII Safing!      | / This observation affected when WFII Safed!
               |FOC Safing!       | / This observation affected when FOC Safed!
               |Shut              | / FOS aperture door is not Open!
               |FAILED            | / FGS astrometry target acquisition failed
     -------------------------------------------------------------------------
```

Table C.3: Contents of `.jit` or `.cmi` Table, Three-Second Averaging

Parameter	Units	Description
`seconds`	seconds	Time since window start
`V2 dom`	arcseconds	Dominant FGS V2 coordinate
`V3 dom`	arcseconds	Dominant FGS V3 coordinate
`V2 roll`	arcseconds	Roll FGS V2 coordinate
`V3 roll`	arcseconds	Roll FGS V3 coordinate
`SI V2 AVG`	arcseconds	Mean jitter in 3 seconds
`SI V2 RMS`	arcseconds	rms jitter in 3 seconds
`SI V2 P2P`	arcseconds	Peak jitter in 3 seconds
`SI V3 AVG`	arcseconds	Mean jitter in 3 seconds
`SI V3 RMS`	arcseconds	rms jitter in 3 seconds
`SI V3 P2P`	arcseconds	Peak jitter in 3 seconds
`RA`	degrees	Right ascension of aperture reference
`DEC`	degrees	Declination of aperture reference
`Roll`	degrees	Angle between North and +V3
`LimbAng`	degrees	Angle between earth limb and target
`TermAng`	degrees	Angle between terminator and target
`LOS_Zenith`	degrees	Angle between HST zenith and target
`Latitude`	degrees	HST subpoint latitude
`Longitude`	degrees	HST subpoint longitude
`Mag V1,V2,V3`	degrees	Magnetic field along V1, V2, V3
`EarthMod`	V Mag/arcsec^2	Model earth background light
`SI_Specific`	–	Special science instrument data
`DayNight`	0,1 flag	Day (0) or night (1)
`Recenter`	0,1 flag	Recentering status
`TakeData`	0,1 flag	Vehicle guiding status
`SlewFlag`	0,1 flag	Vehicle slewing status

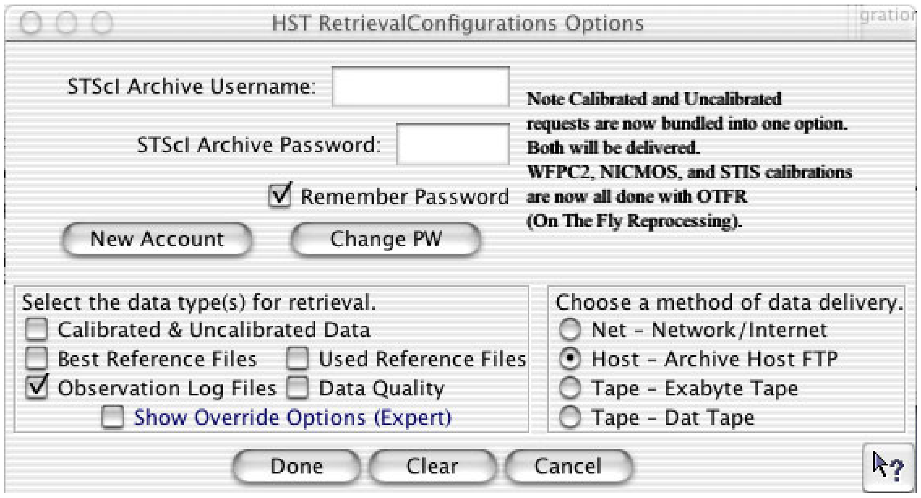
C.2 Retrieving Observation Logs

You can retrieve observation log files for data taken after October 20, 1994 from the HST Archive using StarView as described in chapter 1 of the HST Introduction. Unlike science data, which generally has a one-year proprietary period, observation log files become public as soon as they are archived.

The easiest way to get OMS files through StarView is to identify the observation of interest and proceed with your request as described in chapter 1 of the HST Introduction, until you reach the "HST Retrieval Configurations Options" screen, reproduced in Figure C.3. You can then check the Observation Log Files box, along with any other desired boxes, and continue with your request. StarView will then deliver the associated observation log files.

For observations logged between October 1994 to August 1995, you will be delivered the `cmi`, `cmj`, and `cmh` files in FITS form (e.g., `_cmi.fits`). Observations archived from August 1995 to February 1997 will return `_jif.fits` and `_jit.fits` files. These, and the earlier FITS files can be worked with as such, or converted to their GEIS counterparts via the STSDAS **strfits** task. However, as of February 1997, the `_jif.fits` and `_jit.fits` files are standard FITS files with extensions and cannot be converted to GEIS.

Figure C.3:Choosing Observation Log Files in StarView


HST RetrievalConfigurations Options

STScI Archive Username:

STScI Archive Password:

[✓] Remember Password

Note Calibrated and Uncalibrated requests are now bundled into one option. Both will be delivered.
WFPC2, NICMOS, and STIS calibrations are now all done with OTFR (On The Fly Reprocessing).

Select the data type(s) for retrieval.
- [ ] Calibrated & Uncalibrated Data
- [ ] Best Reference Files
- [ ] Used Reference Files
- [✓] Observation Log Files
- [ ] Data Quality
- [ ] Show Override Options (Expert)

Choose a method of data delivery.
- ○ Net - Network/Internet
- ● Host - Archive Host FTP
- ○ Tape - Exabyte Tape
- ○ Tape - Dat Tape

C.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 of the HST Introduction.

C.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 won't 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 new 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 cmh keywords is an example of two star guiding.

```
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
```

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 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
```

C.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 cmh 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 following observations 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 observational dataset spans multiple orbits, the guide star will be re-acquired, 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.

C.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.

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 axis) can be quite large for moving targets. Also, a special anomaly keyword (SLEWING) will be appended to the OMS header stating movement of the telescope during the observation. This is expected for observing moving targets. The following list of `.cmh` 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
```

C.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 axis. The following list of `.cmh` header keywords 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)
```

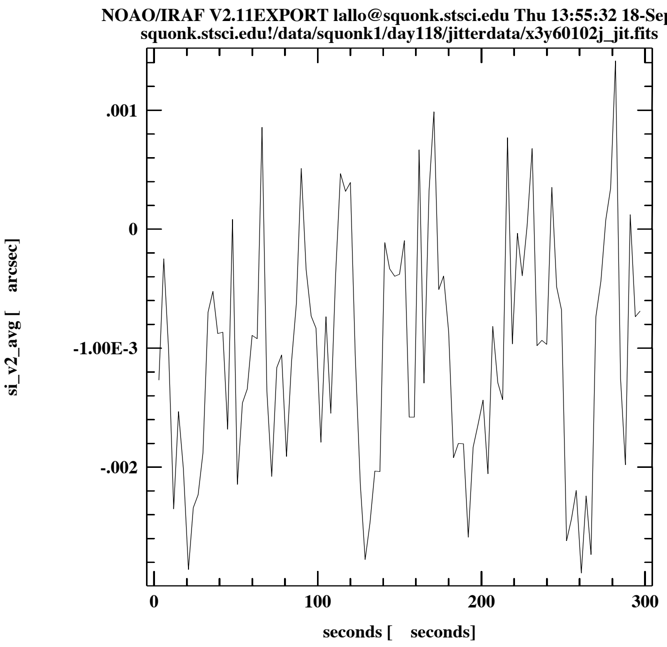
Recentering events occur when the spacecraft software decides that shaking is too severe to maintain lock. The FGS will release guide star control and within a few seconds reacquire the guide stars. It is assumed the guide stars are still within the FGS field of view. During the recentering time, INDEF 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 start time will occur after the guide stars are acquired (or re-acquired), and the observation stop time may occur before the “Loss of Lock” or “Recentering” event that occurred at the end of an OMS window.

The **sgraph** commend in the **stsdas.graphics.stplot** package will plot time vs. jitter along the direction of HST’s V2 axis (see Figure C.4):

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

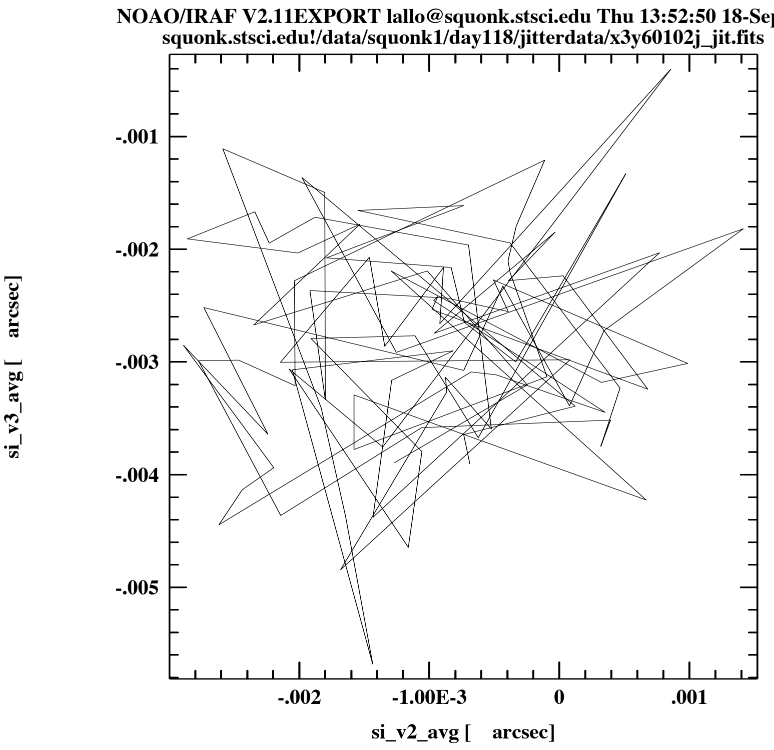
Figure C.4:Plotting Jitter Along V3 Axis



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 C.5):

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

Figure C.5: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 arcseconds) 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 post-servicing mission data:

Figure C.6: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. Documentation is available via the WWW at: http://www.stsci.edu/instruments/observatory/obslog/OL_1.html.

Index