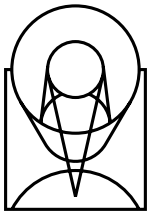
STSDAS Version 3.9
November 2008

STSDAS Site Manager's Installation Guide and Reference



SPACE
TELESCOPE
SCIENCE
INSTITUTE

Science Software Branch
OED Division
3700 San Martin Drive
Baltimore, Maryland 21218

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

Science Software Branch

Perry Greenfield
Howard Bushouse
Ivo Busko
Nadia Dencheva
David Grumm
Warren Hack
Chris Hanley
Phil Hodge
Robert Jedrzejewski
Vicki Laidler
Todd Miller
Chris Sontag
James Taylor
Mark Sienkiewicz

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

M0222

Table of Contents

CHAPTER 1:
Introduction

The Space Telescope Science Data Analysis System (STSDAS) is a set of application programs designed for the calibration and analysis of data from the Hubble Space Telescope. Applications include general image processing as well as tasks specific to the HST. STSDAS also has its own graphics package and a FITS I/O package specifically designed to read HST format image data.

The STSDAS web site is:

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

Binaries for STSDAS/TABLES version 3.9 are now available for Solaris, Redhat, MacOS X PPC and MacIntel. Binaries were compiled on Solaris 5.8, Redhat Enterprise 3 and MacOS X 10.4. Generally they will work with later versions of the operating system, but not with earlier versions.

STSDAS is fully layered upon IRAF, the Image Reduction and Analysis Facility, which is developed and supported by NOAO. If you attempt to install STSDAS without IRAF you will get nowhere—the STSDAS installation procedures use IRAF utilities.

Note: Binaries for Solaris were built using IRAF 2.12.2a. Other binaries were built using IRAF 2.14. If you intend to do a binary installation, you should install the corresponding version of IRAF

STSDAS uses the TABLES external package, also available from STScI, which is a table I/O system that supports the transfer of tabular data from one application to another. There is a table manipulation tool kit that allows one to use tables as small relational databases. Note that *you must*

already have installed TABLES in order to compile STSDAS. STSDAS links against some of the TABLES system libraries, causing unresolved references if you try installing STSDAS without already having installed TABLES.

Changes to the TABLES libraries can also affect STSDAS compilation. We suggest that you match the version numbers of TABLES and STSDAS to maximize compatibility.

This release of STSDAS has been extensively tested at STScI. We encourage off-site users to try as much of the system as possible and send us comments and criticisms.

If You Have Problems...

If you have any problems *installing or using* STSDAS or TABLES contact the Help Desk staff by sending e-mail to: help@stsci.edu, or by calling (410) 338-1082.

CHAPTER 2:

Example Installation

STSDAS and TABLES are available from the STSDAS web site:

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

Below is a concise example of building TABLES and STSDAS from source on RedHat in `/usr/local/tables` and `/usr/local/stsdas`.

The commands below should be executed at the host level:

```
% edit $iraf/unix/hlib/extern.pkg
   define the stsdas and tables packages as shown in figure 1
% cd /usr/local
% mkdir tables
% mkdir stsdas
% cd tables
% tar -xvf /tmp/tables39.tar
% mkpkg redhat
% cd ../stsdas
% tar -xvf /tmp/stsdas39.tar
% mkpkg redhat
```

Start iraf and build the packages from the cl command line:

```
% cl
cl> softools
so> tables
```

```
ta> cd tables
ta> mkpkg -p tables update >& spool
ta> mkpkg summary > tables.summ
ta> stsdas
st> cd stsdas
st> mkpkg -p tables -p stsdas update >& spool
st> mkpkg summary > stsdas.summ
st> logout
% cd /usr/local/stsdas
% python -m compileall -q .
```

```
# External (non core-system) packages. To install a new package, add the
# two statements to define the package root directory and package task,
# then add the package helpdb to the 'helpdb' list.

reset    noao         =  iraf$noao/
task     noao.pkg     =  noao$noao.cl

reset    tables       =  /usr/local/tables/
task     tables.pkg   =  tables$tables.cl

reset    stsdas       =  /usr/local/stsdas/
task     stsdas.pkg   =  stsdas$stsdas.cl
task     apropos      =  stsdas$apropos.cl

reset    helpdb       =  "lib$helpdb.mip \
                         ,noao$lib/helpdb.mip \
                         ,tables$lib/helpdb.mip \
                         ,stsdas$lib/helpdb.mip \
                         "

keep
```

Figure 1: Example extern.pkg file

CHAPTER 3:

Installing TABLES

In This Chapter...

Installation of TABLES is fairly easy and straightforward. It is done within the IRAF environment. This Chapter is intended as a cookbook to help you do a TABLES installation.

It should be noted that there already exists a task in the **lists** package of IRAF named **table** which may cause a conflict with the package name **tables**. When loading this package, the whole name should be typed out to ensure that the correct task is being run.

The Installation Process

TABLES 3.9 is a new release of TABLES. You will need both the source code tar file and the binary tar file for a binary installation, or just the source code tar file if you will be compiling your own binaries.

To install TABLES, you must:

1 • Create the top-level directory for TABLES (below).

2 • Edit the file `hlib$extern.pkg` to define pointers.

3 • Install the TABLES source code from the tar files. (See “Installing the Source Code” on page 12) The latest versions of the tar files are available on the STSDAS web site:

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

4 • Install the TABLES binaries for your architecture: **Solaris, Redhat and MacOS X only**. (See “Installing the Binaries” on page 12). The latest versions of the tar files are available on the STSDAS web site:

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

5 • Modify the help database. (See “The Help Database” on page 13)

6 • Test the system. (See “Testing the TABLES Installation” on page 13)

7 • Install additional binaries, if multi-architecture support is needed. (See “Multi-architecture Support for TABLES” on page 14)

You may also choose to compile TABLES yourself. (See “Building TABLES from Scratch” on page 14)

Selecting the Top Level Directory

TABLES is based on the structure of IRAF. We suggest installing TABLES as a separate directory structure and recommend naming the top level directory `tables.` This will enable you to more easily make updates to the respective systems and allow you to easily add other packages. This is the method used in the examples in this guide.

If for some reason this procedure cannot be followed, it is still straightforward to install TABLES. All package directories are specified relative to the top level TABLES directory. There is one IRAF environment variable, **tables**, that is used as the basis for all package definitions.

Pre-Installation Site Modifications

Installation of TABLES is done within the IRAF **cl**. IRAF must know where TABLES is located. The IRAF file `hlib$extern.pkg` contains the locations of all external packages.

TABLES must be installed using IRAF. All of the binaries in the TABLES release were created with IRAF 2.14, except Solaris binaries which were created with IRAF 2.12.2a. If you do not have IRAF installed, you must install it before proceeding with the TABLES installation. You can get IRAF from the iraf web site `http://iraf.noao.edu.`

To edit the `hlib$extern.pkg` file, change your current directory to `hlib$`:

```
cl> cd hlib$
cl> edit extern.pkg
```

Two modifications need to be made to the file `hlib$extern.pkg`:

- The TABLES package and its location must be defined.
- The path to the TABLES help database must be included.

To tell IRAF where the TABLES system will be located, add the package definition lines before the '`reset helpdb ...`' line in `hlib$extern.pkg`:

```
reset tables = /path/tables/
task tables.pkg = tables$tables.cl
```

To include TABLES in the help search path, add the string '`tables$lib/helpdb.mip`' to the list of help database locations.

An example of the modified `extern.pkg` file is shown in Figure 1 on page 8.

Installing the Source Code

The TABLES source tar files are available from:

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

The source tar file is compressed. You will need to unpack it in the `tables` directory.

```
% cd tables
% gunzip tables38.tar.gz
% tar -xvf tables38.tar
```

Installing the Binaries

After you have installed the source from the tar file then you will need to install a binary tar file. Here you will find files for each supported architecture.

Architecture	*IRAF arch name*
Sun Solaris 5.8	ssun
RHEL 3	redhat
MacOS X 10.4 PPC	macosx
Mac OSX 10.4 MacIntel	macintel

Table 1: Currently Supported Binary Distributions

The binaries are stored in a gzip-compressed tar archive (`tables39.bin.`*`arch`*`.tar.gz`)

Note: *Binaries were compiled on the operating system specified in Table 1. Generally they will work with later versions of the operating system but not with earlier versions.*

You will need to unpack the binaries in tables/bin.arch. For example on a Redhat machine, the commands are:

```
% cd tables/bin.redhat
% gunzip tables39.bin.redhat.tar.gz
% tar -xvf tables39.bin.redhat.tar
```

The Help Database

The help database is provided in a machine-independent form and need not be rebuilt. If you wish to rebuild it, you may do so by running the **mkhelpdb** task within the **softools** package of IRAF.

```
cl> softools
so> mkhelpdb helpdir=tables$lib/root.hd \
>>>helpdb=tables$lib/helpdb.mip
```

Testing the TABLES Installation

Once TABLES has been rebuilt, some of the basic tasks of TABLES may be exercised. This example uses a standard IRAF sample data file:

```
% cl
ecl> tables
tables> cd /tmp
tables> imtab dev$pix[250,*] pix_table.fits colname='flux' pname='pos'
tables> tstat pix_table.fits flux
# pix_table.fits  flux
# nrows            mean        stddev        median           min           max
  512           204.875       213.032         128.5           51.         1146.
tables> tprint pix_table.fits columns='flux' rows='250-260'
#  Table pix_table.fits[1]  Thu 16:33:48 14-Feb-2008

# row        flux
#

  250         820
```

```
  251          838
  252          856
  253          912
  254          982
  255         1038
  256         1097
  257         1118
  258         1080
  259          996
  260          947
tables>
```

Multi-architecture Support for TABLES

It is possible to support multiple architectures using a single source tree. If you wish to support, for example, both Solaris and Linux architectures with a single source tree, you would follow these steps:

- Create a top-level directory for TABLES
- Edit the `hlib$extern.pkg` file. (See “Pre-Installation Site Modifications” on page 11)
- Install the TABLES source code. (See “Installing the Source Code” on page 12)
- Install the binaries for the Solaris architecture. (See “Installing the Binaries” on page 12)
- Install the binaries for the Linux architecture. (See “Installing the Binaries” on page 12)
- Modify the help database. (See “The Help Database” on page 13)

Building TABLES from Scratch

If binaries for your computer’s architecture are not available, then you will need to compile TABLES from scratch.

Earlier versions of TABLES can be obtained from `http://sts-das.stsci.edu/old_versions.html`

The first step in a relink is to ensure that some system variables are set. Type the following command at the system level before proceeding:

```
% setenv IRAFARCH arch
```

where *arch* is your specific architecture, e.g., `redhat` for a RedHat machine. A list of IRAF supported architectures is given in Table 1.

```
% setenv iraf /path/iraf/
```

where *path* is the directory path to the top-level IRAF directory.

```
% source $iraf/unix/hlib/irafuser.csh
```

This sets up some other environment variables needed to compile under IRAF. You may set these up in your `.login` file so that they will be available.

Also, ensure that the directory that contains the local Unix commands (usually `/usr/local/bin`) is included in your PATH environment variable. If it is not, you can add it to your path by typing the following:

```
% setenv PATH /usr/local/bin:{$PATH}
```

Before attempting the total system rebuild, you should check the soft link for the `bin` directory. TABLES is shipped with a link for bin pointing to `bin.generic`. This should be changed so that it points to the appropriate `bin` for your architecture. To do this type this command from the TABLES top-level directory:

```
% mkpkg arch
```

where *arch* is your specific architecture. A list of available architectures is provided in Table 1. For example, for a RedHat machine, you would type:

```
% mkpkg redhat
```

You will get a warning message about a full “sysgen” needing to be done, but that is normal.

The Unix system relink can be done with a procedure submitted while within the IRAF **cl.** Load the **tables** and **softools** packages, set the current directory to the top level TABLES directory, and execute the **mkpkg** task:

```
cl> tables
ta> softools
so> cd tables
so> mkpkg -p tables update >& log &
```

This will run the **mkpkg** task as a background process and put all output and errors into the `tables$spool` file.

The **mkpkg** program generates a long output file describing all steps taken. To reduce this log to the pertinent information about the success of your installation, re-run the **mkpkg** task with the summary option.

```
cl> softools
so> cd tables
so> mkpkg summary >& tables.summ
```

Check tables.summ for errors.

FITS Table Support

For tables in FITS files, the `tables` library routines call subroutines in the HEASARC FITSIO package, so IRAF tasks that are linked with the tables library can transparently access FITS tables as well as ASCII or STSDAS format binary tables. Two versions of FITSIO are included in the tables distribution, one version written in Fortran and SPP, and one written in C.

The C version (CFITSIO) is currently supported by HEASARC, and it is faster than the Fortran version. The Fortran version has an SPP layer for the I/O routines, so that it is fully compatible with IRAF I/O; in particular,

IRAF networking is supported. The C version, on the other hand, uses host system I/O to read and write the FITS files; IRAF virtual file names are supported by converting to host system names, but IRAF networking is not available.

CFITSIO is used by default. If the SPP and Fortran version is required instead, the following steps should be followed.

```
so> cd tables
so> delete tables$bin/libtbtables.a # if it already exists
so> mkpkg -p tables update sppfitsio=yes >& spool &
```

The only difference from a normal build is that sppfitsio=yes is specified when running mkpkg. (**Note:** mkpkg does not check the value assigned to sppfitsio, it just checks whether sppfitsio is defined, so it has the same effect regardless of the value.) Regardless of the sppfitsio switch, the CFITSIO source files will be compiled and included in the tables library, since they are used directly by some tasks in STSDAS

CHAPTER 4:

Installing STSDAS

In This Chapter...

Installing STSDAS is fairly easy and straightforward and is done within the IRAF environment. This chapter briefly explains how to do an STSDAS installation. We recommend that you at least look at Figure 3.1 on page 32 and Figure 3.2 on page 33 before doing an installation so that you understand how the software is organized and where it expects its directories and files to be located.

Before You Begin...

Before installing STSDAS, you should be aware that:

- STSDAS is linked against libraries in the TABLES package. If you do not already have TABLES installed, you must install and compile TABLES first. TABLES is available at the same web site as STSDAS.
- The version number of TABLES *must* be the same as the version number of STSDAS. If you are upgrading STSDAS, you should upgrade TABLES first. See Chapter 3 for details on installing TABLES.

STSDAS will *not* compile without TABLES being installed first!

- The calibration routines in the **hst_calib** packages **nicmos**, **stis and acs** require a large amount of static memory to run. An average estimate of the required RAM is 500 Mb and at least twice as much swap space.
- If you are interested in reading data such as the *Guide Star Catalog* from CD-ROM, you may use the **gasp** package and system-mounted CD-ROMS.

Installation Process

STSDAS 3.9 is a new release of STSDAS. You will need to get the source code tar file and a binaries tar file if you would like to do a binary installation, or just the source code tar file if you are going to compile your own binaries.

To install STSDAS, you must:

1. Create the top-level directory for STSDAS (below).
2. Edit the file `hlib$extern.pkg` to define pointers.

3 • Install the STSDAS source code from the tar files. (See “Installing Source Code” on page 20). The latest versions of the tar files are available on the STSDAS web site:

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

4 • Install the STSDAS binaries for your architecture - **Solaris, Redhat, MacOSX and MacIntel only**. The latest versions of the tar files are available from the STSDAS web site.

(See “Installing the Binaries” on page 21)

5 • Modify the help and apropos databases. See (“The Help Database” on page 22 and “The Apropos Task” on page 23)

6 • Test the system. (See “Testing STSDAS” on page 25)

7 • Install additional binaries, if multi-architecture support is needed. (“Multi-architecture Support for STSDAS” on page 27)

You may also choose to compile STSDAS yourself.

Selecting the Top-Level Directory

STSDAS is based on the structure of IRAF. We suggest installing STSDAS as a separate directory structure and recommend naming the top-level directory `stsdas`. This will enable you to more easily make updates to the respective systems and allow you to easily add other packages. This is the method used in the examples in this guide.

If for some reason this procedure cannot be followed, it is still straightforward to install STSDAS. All package directories are specified relative to the top level `stsdas` directory. There is one IRAF environment variable, called **stsdas**, that is used as the basis for all package definitions.

Pre-Installation Site Modifications

Installation of STSDAS is done from within the IRAF **cl**. IRAF must know where you intend to put STSDAS. The IRAF file `hlib$extern.pkg` contains the locations of all external packages.

STSDAS must be installed using IRAF. All of the binaries in the STSDAS release were created with IRAF 2.14, execpt Solaris binaries which were created with IRAF 2.12.2a.

To edit the `hlib$extern.pkg` file, change your default directory to `hlib` and edit the file `extern.pkg`:

```
cl> cd hlib
cl> edit extern.pkg
```

Two modifications need to be made to the file `hlib$extern.pkg`:

- The STSDAS package and its location must be defined.
- The path to the STSDAS help database must be included.

To tell IRAF where the STSDAS system will be located, add the package definition lines before the '`reset helpdb ...`' line in `hlib$extern.pkg`:

```
reset stsdas = /path/stsdas/
task stsdas.pkg = stsdas$stsdas.cl
```

To include STSDAS in the help search path, add the string '`stsdas$lib/helpdb.mip`' to the list of help database locations.

An example of a modified `extern.pkg` file is shown in Figure 1 on page 8.

Installing Source Code

The STSDAS source tar files are available from the STSDAS web site at:

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

You will need to unpack the source file in the stsdas directory:

```
% cd stsdas
% gunzip stsdas37.tar.gz
% tar -xvf stsdas37.tar
```

Installing the Binaries

After you have installed the source from the tar file then you will need to install a binary tar file. There is a compressed tar file for each supported architecture (see Table 1 on page 12) (`stsdas39.bin.`*`arch`*`.tar.gz`).

Note: *Binaries were compiled on the operating system specified in Table 1. Generally they will work with later versions of the operating system but not with earlier versions.*

You will have to unpack the file in `stsdas/bin.arch.` For example, on a Redhat machine, the commands are:

```
% cd stsdas/bin.redhat
% gunzip stsdas39.bin.redhat.tar.gz
% tar -xvf stsdas39.bin.redhat.tar
```

Compiling the Python code

STSDAS has several tasks (for example, **MultiDrizzle**) that use the Python scripting language.

To run the Python tasks more efficiently, the Python code has to be compiled by the user “iraf”. You will need the Python interpreter to run and compile the code. Check whether you have Python installed on your system by running the command:

```
% which python
```

Note: You will need Python 2.3 or later to run PyRAF and the associated Python tasks in STSDAS. If you are not using any of the Python tasks, you don't need to compile the Python code.

To compile the STSDAS python code:

- log out of iraf
- go to the `stsdas` directory
- run the `compileall` python module

```
cl> logout
% cd <stsdas>
```

where <`stsdas`> is the directory where stsdas was unpacked.

```
% python -m compileall -q .
```

Note: Figure 3.2 on page 33 shows part of the stsdas tree structure with the links in the python directory.

The Help Database

The help database is provided in a machine independent form and need not be rebuilt. If you wish to rebuild it, you may do so by running the **mkhelpdb** task within the **softools** package of IRAF.

```
cl> softools
so> mkhelpdb helpdir=stsdas$lib/root.hd \
       >>>  helpdb=stsdas$lib/helpdb.mip
```

The Apropos Task

The **apropos** task searches an apropos database for any input string and outputs descriptions of any tasks in its database that match the search string. This task is a powerful tool for those who are not familiar with the package structure of the IRAF, NOAO, STSDAS, or TABLES packages. For convenience, we placed this task at the top level of the IRAF **cl** so that it is available without loading any packages. The script for the **apropos** task is in the top level directory of STSDAS. To include this task in your system add the following line to your `hlib$extern.pkg` below the task statement for STSDAS:

```
task   apropos       = stsdas$apropos.cl
```

So, your `hlib$extern.pkg` might look like that in Figure 1 on page 8.

The apropos database, `stsdas$lib/apropos.db`, is provided in a machine-independent form and need not be rebuilt. If you wish to rebuild it, you may do so by running the **mkapropos** task within the **toolbox.tools** package of STSDAS.

```
cl> stsdas
st> toolbox
to> tools
to> mkapropos pkglist=iraf,noao,stsdas,tables \
>>> helpdir=lib/root.hd aproposdb=stsdas$lib/apropos.db
```

Figure 2.1: Rebuilding the apropos Database

Psikern Installation

STSDAS is distributed with an IRAF graphics kernel called *psikern* that produces output in Encapsulated PostScript. **psikern** provides a direct connection between the IRAF graphics system and PostScript. The kernel can use colors, fill areas, and has imaging capabilities. Many tasks in STSDAS, in particular those in the **stplot** package, take advantage of these capabilities. Installation of psikern is optional.

To use **psikern**, you must define graphics devices that invoke the kernel. To define new or different graphics devices, the file `dev$graphcap` must be modified. The file `stsdas$pkg/graphics/stplot/psikern.template` contains the basic entries necessary and some examples of how to use the basic entries to get output to a specific printer. To add **psikern** graphics devices to the `graphcap` file, follow these steps:

1 • Make a backup copy of `dev$graphcap`, so you can recover if mistakes are made. Prepend to `dev$graphcap` the two entries in the file `stsdas$pkg/graphics/stplot/psikern.template` marked "REQUIRED ENTRIES". These entries define two psikern graphics devices, `psi_land` and `psi_port`, for output on 8-1/2 x 11 inch paper. These entries create files with the names "tmp$pskxxxx" where "xxxx" are random numbers.

2 • Prepend to `dev$graphcap`, before the required entries specified above, graphics entries for specific printers. The entries marked "EXAMPLES" in `stsdas$graphics/stplot/psikern.template`, demonstrate some example devices used at STScI..

To create an entry for a specific printer, basically all that needs to be changed is the DD parameter of an entry. For an example in Unix, to create an IRAF graphics device that will produce plots in landscape mode on the printer attached to queue `lw`, the entry you would add to the beginning of `dev$graphcap` would be:

```
lw|PostScript In Landscape mode on queue lw:\
   :DD=lw,tmp$psk,!{lpr -Plw $F; rm $F ;}:tc=psi_land:
```

3 • Once `dev$graphcap` has been modified, you can use the newly defined graphics devices as you would any other IRAF graphics device. For example, to make the device defined in the above example the default output device for plots, make the following definition in your `login.cl` file:

```
set stdplot = lw
```

Also, for any graphics task that uses the `device` parameter to set the output printer, you can specify the newly defined devices. For example:

```
cl> plot
pl> prow dev$pix 256 device=lw
```

For more information on the IRAF graphics system, type the following command:

```
cl> help gio$doc/gio.hlp file+
```

For help about **psikern**, use the following command:

```
cl> help psikern
```

Testing STSDAS

Once STSDAS has been rebuilt, some of the basic functions of STSDAS may be exercised. This example uses a standard IRAF sample data file:

```
% cl
ecl> stsdas
stsdas> iminfo dev$pix
Imagename                                   Object
dev$pix                                     m51  B  600s

Naxis1 = 512         Datatype                    Datamin = -1.
Naxis2 = 512           short                     Datamax = 19936.

Integration time = 600                      Right ascension = 13:29:24.00
Elapsed time     = 600                      Declination     = 47:15:34.00
Universal time   =  9:27:27.00              Zenith distance = 22:14:00.00
Sidereal time    = 14:53:42.00              Airmass         = 1.080156

stsdas> listarea dev$pix[250:255,250:255]
Image:  dev$pix[250:255,250:255]
 Sample       250       251       252       253       254       255
Line
    250      820.      811.      809.      841.      907.      973.
    251      838.      846.      858.      890.      936.     1017.
    252      856.      877.      907.      917.      942.     1035.
    253      912.      938.      971.      988.     1004.     1083.
    254      982.     1029.     1083.     1135.     1175.     1244.
    255     1038.     1124.     1224.     1308.     1388.     1488.
stsdas>
```

If you are using xgterm, you can test the graphic features. To plot a graph on the screen:

```
% cl
cl> stsdas
stsdas> sgraph dev$pix[*,255] marker='plus' pointmode=yes
```

To plot the same graph to a postscript file,

```
stsdas>sgraph dev$pix[*,255] marker='plus' pointmode=yes device=psi_landm;gflush
```

If these commands don't work, check that the basic IRAF features work with:

```
stsdas> prow dev$pix 255 pointmode=yes marker='plus'
stsdas> prow dev$pix 255 pointmode=yes marker='plus' device=psi_landm; gflush
```

Reading Exported Data Files

A few sample HST data files are provided in the directory `stsdas$data/fits`. These are in FITS disk format, and need to be expanded into IRAF or STSDAS disk format files before they can be accessed by STSDAS applications programs. You can use the STSDAS **fitsio** package to read these files. The expanded version of the files should be placed into the directory `stsdas$data/scidata` using the commands shown below:

```
cl> stsdas
st> fitsio
fi> cd stsdas$data/fits
fi> cl < read_fits.cl
```

If your users want to use the **synphot** package, you will need to install the STDATA files described here.

Additional sample data files and throughput tables for the HST components are needed by the **synphot** package; these files are provided separately and can be retrieved using anonymous ftp. The installation instructions for these files are in Appendix A.

Multi-architecture Support for STSDAS

It is possible to support multiple architectures using a single source tree. If you wish to support, for example, both Solaris and Redhat architectures with a single source tree, you would follow these steps:

- Create a top-level directory for STSDAS.
- Edit the `hlib$extern.pkg` file. (See “Pre-Installation Site Modifications” on page 19)
- Install the STSDAS source code. (See “Installing Source Code” on page 20)
- Install the binaries for the Solaris architecture. (See “Installing the Binaries” on page 21)
- Install the binaries for the Redhat architecture. (See “Installing the Binaries” on page 21)

Building STSDAS from Scratch

If binaries for your computer’s architecture are not available, then you will need to compile STSDAS from scratch.

Earlier versions of STSDAS can be obtained from http://stsdas.stsci.edu/old_versions.html.

The first step in a system build is to ensure that some system variables are set. The following should be typed at the system level before proceeding:

```
% setenv IRAFARCH arch
```

where arch is your specific architecture, e.g., “macosx” for a MacOS X machine (see Table 2.1).

```
% setenv iraf /path/iraf/
```

where *path* is the directory path to the top-level IRAF directory.

```
% source $iraf/unix/hlib/irafuser.csh
```

This sets up some other environment variables needed to compile under IRAF. You may set these up in your `.login` file so that they will be available.

Also, ensure that the directory that contains the local Unix commands (usually `/usr/local/bin`) is included in your PATH environment variable. If it is not, you can add it to your path by typing the following:

```
% setenv PATH /usr/local/bin:{$PATH}
```

Before attempting the total system rebuild, you should check the soft link for the `bin` directory. STSDAS is shipped with a link for `bin` pointing to `bin.generic`. This should be changed, so that it points to the appropriate `bin` for your architecture. To do this simply type the following command from the STSDAS top-level directory:

```
% cd stsdas
% mkpkg arch
```

where *arch* is your specific architecture. (A list of architectures is provided in Table 2.1). For example, for a RedHat machine, you would type:

```
% mkpkg redhat
```

You will get a warning message about a full “sysgen” needing to be done, but that is normal.

Architecture	*Command*
Sun Solaris	`mkpkg ssun`
PC platforms	
Linux Slackware 3.3	`mkpkg linux`
Linux Red Hat	`mkpkg redhat`
FreeBSD 2.2.5	`mkpkg freebsd`
Suse	`mkpkg suse`
MacOS X PPC	`mkpkg macosx`
Mac OSX MacIntel	`mkpkg macintel`

Table 2.1: Supported Architectures

The Unix system rebuild can be done with a batch procedure submitted while within the IRAF **cl.** Load the **stsdas** and **softools** packages, set the current directory to the top STSDAS directory, and execute the **mkpkg** task:

```
cl> stsdas
st> softools
so> cd stsdas
so> mkpkg -p tables -p stsdas update >& spool &
```

This will run the **mkpkg** task as a background process and put all output and errors into the `stsdas$spool` file.

The **mkpkg** program generates a long output file describing all steps taken. To reduce this log to the pertinent information about the success of you installation, re-run the **mkpkg** task with the summary option.

```
cl> softools
so> cd stsdas
so> mkpkg summary > stsdas.summ
```

Check the file `stsdas.summ` for errors.

After you have build STSDAS from scratch, you will need to compile the Python code. First go to the stsdas directory, and then run the `compileall.py` script:

```
% cd <stsdas>
% python -m compileall -q .
```

CHAPTER 5:

STSDAS Site Manager's Reference

In This Chapter...

This Chapter explains information needed to maintain and troubleshoot the STSDAS package.

User Account Privileges and Quotas

STSDAS does not require any special privileges or quotas when operated in a Unix environment. However, some of the calibration tasks (e.g. in the **acs**, **nicmos** and **stis** packages), require about 500Mb of RAM on the average and at least twice as much swap space.

STSDAS Directory Structure

STSDAS is organized in a hierarchical directory structure (see Figure 3.1) that reflects the organization seen by users of the system. Since STSDAS is a part of IRAF, we have adopted the *package* structure of IRAF to organize the application functions available to users. When STSDAS is installed, the system manager controls the name of the directory in which the structure is rooted; typically, this is `stsdas`, and we will assume this

in the discussions that follow. All directory path names in STSDAS are relative to this top level directory. All STSDAS directories have names assigned as IRAF environment variables. To go to any STSDAS application package directory, just use the **cd** command in IRAF and specify the name of the package (e.g., `cd fourier`).

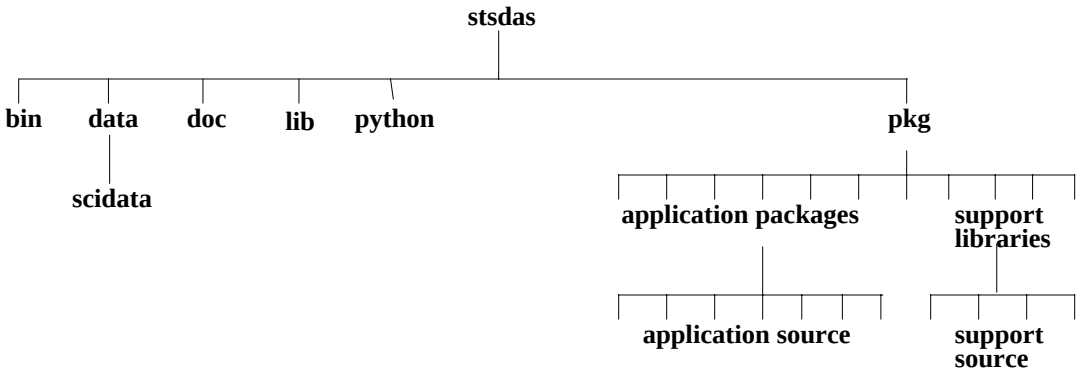


Figure 3.1: Overall STSDAS Directory Structure

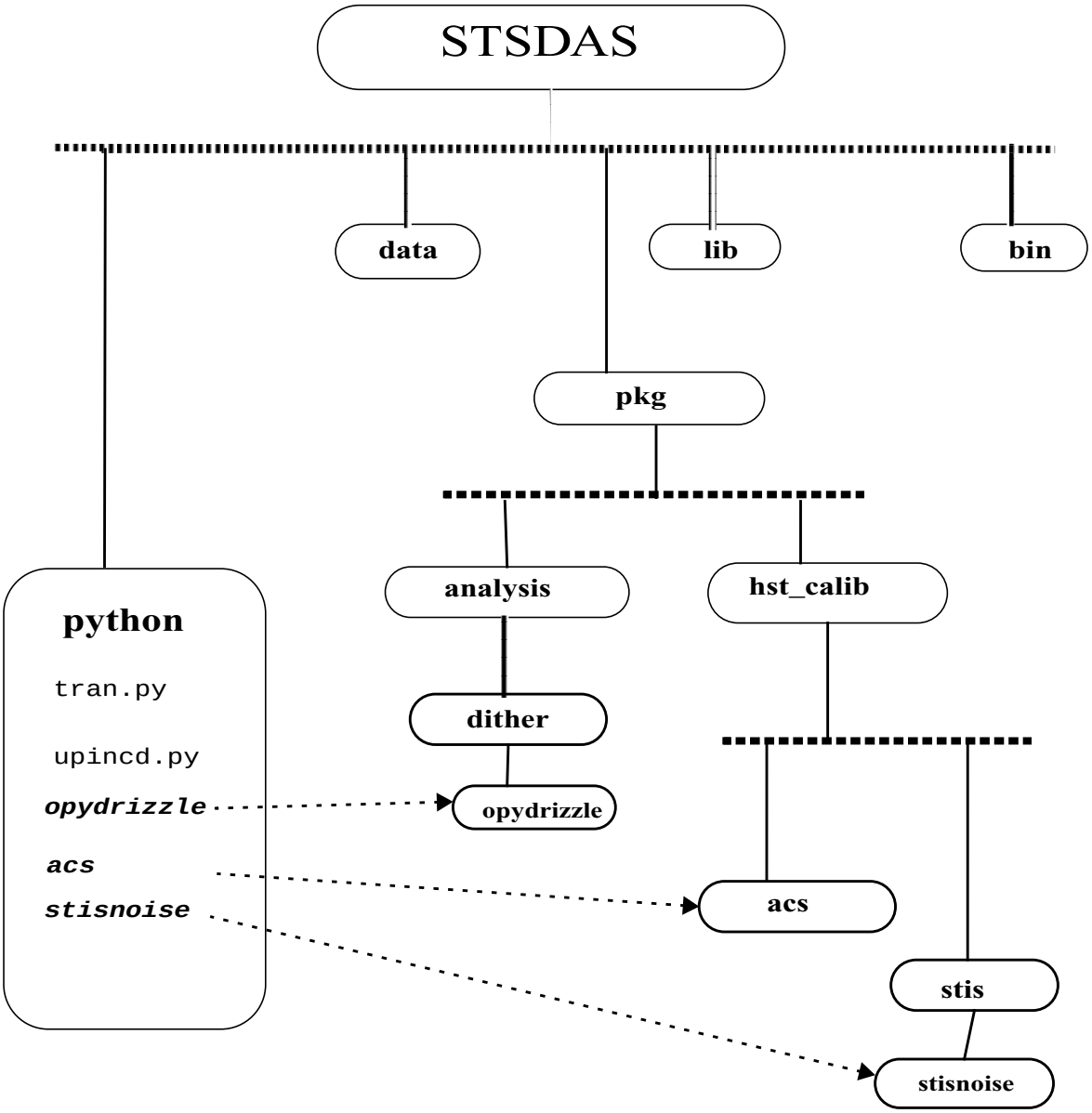


Figure 3.2: Tree structure of the Python code in STSDAS

Applications Software Directories

Each STSDAS package has a corresponding directory in the host file system that is a subdirectory of the `pkg` directory; the name of the subdirectory is the same as the name of the applications package. These

package-level directories contain all the run-time files that may be needed by tasks within that package, including parameter files, help files, and a `mkpkg` file (used for recompiling and relinking the package). For example, suppose there were an STSDAS applications package called **applpkg**.

The files listed in Table 3.1 are stored in the package level directory (`applpkg` in this example):

File	*Description*
`*.cl`	CL scripts. There is one that defines the package, and one for each of the logical tasks within the package.
`*.par`	Parameter files for the logical programs in the package. Generally, there will be one for each source-level subdirectory.
`*.hd`	The help data base index for this package (one per package).
`*.men`	The help menu file for this package (one per package).
`*.hlp`	Help text file for this package as a whole.

Table 3.1: Files in Package Level Directory of an Application

File names given here are in the syntax of IRAF's virtual filename mapping. To avoid confusion, we recommend that you always view the contents of the STSDAS system while running the IRAF CL and using the `cd` command to change directories.

Beneath most applications package directories are subdirectories that contain the source code for the various tasks within that package. Each major task resides in its own directory. Once STSDAS has been installed (recompiled and relinked), the source code can be removed to conserve disk space (See "Saving Space" on page 39 for instructions).

File Name	*Contents*
`*.c`	C source code for program
`*.f`	Fortran source code for program
`*.x`	SPP source code for program
`mkpkg`	The IRAF mkpkg file used to update the object library for program and to relink package executable image.
`program.o`	Object library for program. This file is not exported; it will not exist unless **mkpkg** has been run.

Table 3.2: Common Files in STSDAS Task Directories

Help text files for each of the logical tasks in the package are contained in the `doc` subdirectory of the package.

The executable images of the packages are located in `stsdas$bin` as `x_pkg.e`.

A complete directory tree for STSDAS is shown in Figure 3.3.

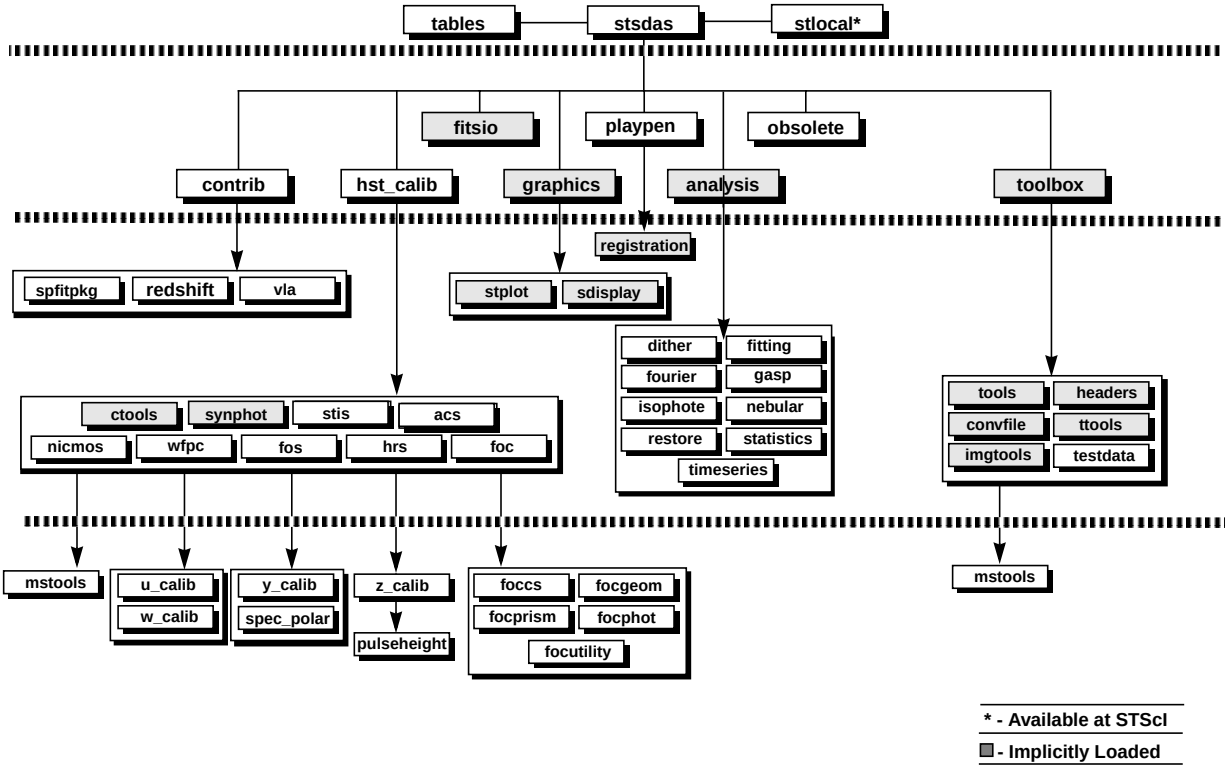


Figure 3.3: STSDAS Organization

Support Software Directories

The applications directories described above contain the run-time connections to IRAF and the source code for the STSDAS applications. There are a number of other directories needed to support STSDAS. Most important is the subdirectory structure in which all STSDAS I/O and utility software is stored; this structure is rooted in the `lib` subdirectory of STSDAS (`stsdas$lib/`). This subdirectory contains several libraries, and for each of these there is a related subdirectory in which the corresponding source code resides.

Directory	*Contents*
`applib`	Applications subroutines
`cvos`	C programming interface
`f77util`	F77VOS utilities
`hstio`	HST data file I/O C interface
`iraf77`	F77VOS interface
`stalone`	F77 stand-alone interface
`synphot`	Synthetic photometry interface

Table 3.3: STSDAS I/O and Utility Libraries

In addition, several utility libraries for tables I/O are used by STSDAS that are in the TABLES package, these are listed in Table 3.4.

Library	*Contents*
`display`	Terminal display routines
`gflib`	Front end to gilib
`gilib`	STSDAS IEEE and GEIS format subroutines
`tbtables`	STSDAS tables I/O subroutines
`uttables`	STSDAS table utilities
`stxtools`	STSDAS special applications tools

Table 3.4: TABLES I/O and Utility Libraries

STSDAS will *not* compile without TABLES being installed first!

Exported Data Directories

A few data files have been sent along with the STSDAS installation, and these are found in the `stsdas$data/` directory tree. The directory

`stsdas$data/fits` contains these files with the extension `.fits`. "Reading Exported Data Files" on page 26 describes how these data are to be read and installed into the `scidata` directory. After the STSDAS package has been loaded, this area can be referred to with IRAF environment variable `scidata`.

Sample calibration and image files for each of the major instruments on HST can be retrieved from the archive (http://archive.stsci.edu).

STSDAS and IRAF System Directory

The STSDAS directory contains files that establish the entire STSDAS package structure in the IRAF environment (Table 3.5).

File	*Purpose*
`stsdas.cl`	Primary CL script that defines all STSDAS packages
`stsdas.hd`	Primary help data base index for all STSDAS help files
`stsdas.men`	The help menu file for the STSDAS package itself
`stsdas.hlp`	Highest-level help file for STSDAS (as a whole)

Table 3.5: Files Establishing STSDAS Package Structure

In addition, the directory `stsdas$lib` contains the file `mkpkg.inc`. This file contains the macro definitions for the IRAF **mkpkg** facility and is used when **mkpkg** compiles or links STSDAS programs.

Rebuilding STSDAS Applications

STSDAS applications are structured so that they can be rebuilt piecewise or as a whole using the **mkpkg** utility provided with IRAF. Users who wish to rebuild IRAF/STSDAS applications should familiarize themselves with specifics about the use of **mkpkg** as described in the IRAF help documentation for **mkpkg**. There are **mkpkg** files in various directories at three levels within the STSDAS applications hierarchy: Each applications program directory contains a **mkpkg** file that will rebuild that particular library and relink the package executable image. These directories are fourth-level nodes, i.e., `stsdas$pkg/*/*/`.

Each STSDAS package directory has a **mkpkg** file that will rebuild all the libraries in the package and relink the package executable image. Package directories are third-level nodes, i.e., `stsdas$pkg/*/`.

- There is a **mkpkg** file in the directory `stsdas` that will rebuild the entire STSDAS system. This is a first-level node, i.e., `stsdas/`.

When rebuilding an STSDAS task or package, **mkpkg** must be told to use the appropriate environment variables and libraries. Therefore it is necessary to run the task **mkpkg** with the command:

```
cl> mkpkg -p stsdas
```

when making the entire STSDAS system from the `stsdas/` directory; or the command

```
cl> mkpkg -p stsdas update
```

when making a single package or application program.

Package executables are rebuilt on a time scale that is typically several minutes, although this can vary widely. STSDAS **mkpkg** files can either perform compilation *and* linking or just a relink (by typing `linkonly` at the end of the **mkpkg** command). Some programs contain a large number of subroutines; others contain few.

Recompiling all of TABLES and STSDAS takes about 10-15 minutes on a 2 GHz Pentium, 30 minutes on an older PowerPC macintosh, and 90-120 minutes on a 333 MHz SparcStation 20.

Saving Space

As with IRAF, a mechanism exists to remove the source files and other files not needed to actually run STSDAS. This procedure should only be used if disk space is a problem, you will not be doing any development using STSDAS, and you have an alternative method for getting revised object libraries and executables when system patches are made.

To remove all but the essential files, you need to run the **mkpkg strip** command from the top level of **stsdas**.

```
cl> cd stsdas
cl> mkpkg strip -p stsdas
```

APPENDIX A:

Synphot Data Set

In this Section...

Synphot is a package in **STSDAS** that calculates count rates, throughputs and sensitivities for HST instruments. It requires the use of reference files that describe HST component throughputs, allowed configurations, standard star and model spectra. These reference files are not distributed with STSDAS, but are available separately by following the instructions detailed in this Appendix.

Setting the Top Directory

The synphot tasks assume that all the synphot reference files are stored under a single top level directory. This directory is referred to inside STSDAS by the logical name `crrefer`. This directory may be anywhere you have sufficient space to install the reference files (approximately 400 megabytes is required for the full installation), but we recommend that it not be placed as subdirectory of the STSDAS or TABLES source code. This will make it easier to update STSDAS without needing to reinstall the Synphot data. Once the top directory is created, the environment variable `crrefer` should be set in your `hlib$extern.pkg` file. To set `crrefer$` add a command similar to the following to the file:

```
set crrefer = "/your/path/name/to/refer/"
```

The trailing slash is important, so do not omit it.

The Synphot data can be downloaded from our anonymous ftp site at:

`ftp://ftp.stsci.edu/pub/software/stsdas/refdata/synphot/`

There are four compressed tar files containing the data and this installation guide. The first tar file contains the Synphot component throughput tables, the second contains various observed and modelled spectral catalogs, the third contains the 1993 Kurucz model stellar spectra, and the fourth contains the HST calibration standard spectra.

First, place the compressed tar files in the top level directory you created in the first section. Then, uncompress and untar the tar files. On a Unix system, the following commands will accomplish this.

```
% uncompress synphot1.tar.Z
% tar -xvf synphot1.tar

% uncompress synphot2.tar.Z
% tar -xvf synphot2.tar

% uncompress synphot3.tar.Z
% tar -xvf synphot3.tar

% uncompress synphot4.tar.Z
% tar -xvf synphot4.tar
```

The tar file `synphotpsf.tar.Z` contains the psf images used with the simulators package of synphot. If you are not planning to use this package, you do not need to install it. The tar file should be copied to the `stsdas$data/scidata` directory of stsdas, uncompressed, and untarred.

Type the following commands when in stsdas:

```
cl> copy /your/path/to/synphotpsf.tar.Z scidata$
cl> cd scidata$
cl> !uncompress synphotpsf.tar.Z
cl> rtar -xvf synphotpsf.tar
```

When Disaster Strikes

If you encounter problems installing the Synphot data files, we encourage you to contact us via the STSDAS help desk

help@stsci.edu

Index

S

T

U

V