

# STIS Time-Tag Analysis Guide

---

Ilana Dashevsky, Kailash Sahu, Ed Smith  
September 6, 2000

---

## ABSTRACT

*The Time-Tag observing mode has been used extensively since the installation of STIS on HST in February 1997. However, working knowledge has been gained only recently. The objective of this report is to document these experiences for the increasing number of HST users who are using the Time-Tag observing mode. Presented are the basics of reducing and calibrating Time-Tag observations. Also, the data management is discussed since the observations are volume and processing-time intensive. Finally, examples of the data processing and analysis procedures are given, using observations of the Crab Pulsar.*

---

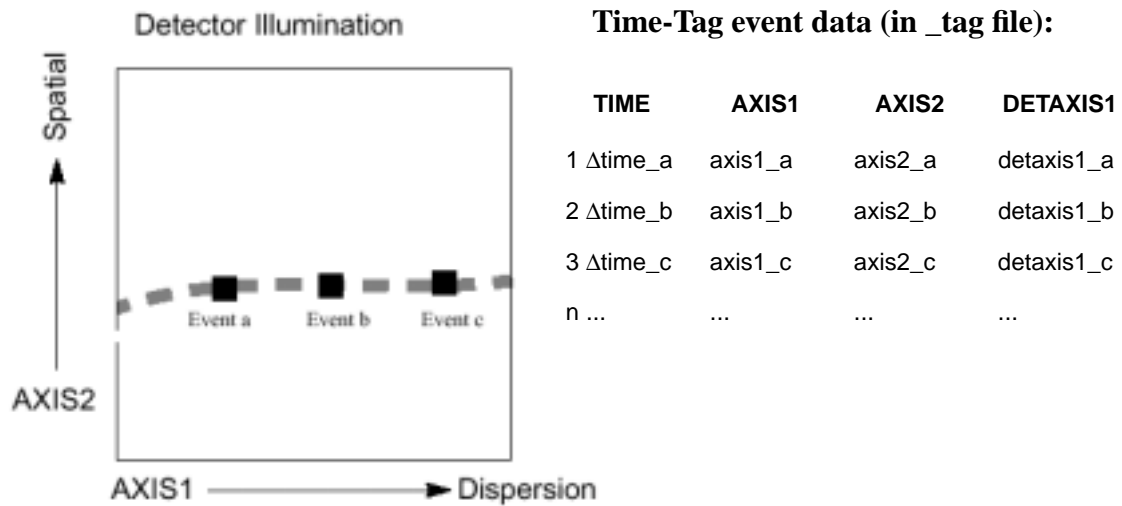
## 1. Introduction

The STIS MAMA (Multi-Anode Multichannel Array) detectors allow for high resolution temporal data when used in the Time-Tag observing mode. In this mode, photons are recorded at a time resolution of 125 microseconds. Each photon event is tagged with a corresponding event time. The recorded times are relative to the start time of the exposure and are in units of spacecraft time. Figure 1 illustrates the Time-Tag observing mode. An observing limitation that Time-Tag users should be aware of is that the MAMA detectors are not able to count reliably at rates greater than  $30000 \text{ count second}^{-1}$  (i.e., non-linearity and problems with event tagging sets in). More observing limitations for the Time-Tag mode are listed in Chapter 11 of the STIS Instrument Handbook.

The basics on working with STIS data observed in the Time-Tag mode are presented in Section 2, including some considerations for timing analysis. Calibration issues are discussed in Section 3. Time-Tag data tend to be volume and processing-time intensive. For this reason we have included a discussion and some test cases of data volume and processing in Section 4. Section 5 explains the retrieval process for ephemeris files used for

heliocentric and barycentric time correction and provides an example on how to apply the correction. In Section 6 we work through an example to convert Time-Tag data into an image using IRAF. We've also included a timing analysis example using IRAF tasks in Section 7. The references in Section 9 list the documentation published by STScI on using the STIS Time-Tag observing mode, as of July 2000. It is assumed that the reader is familiar with retrieving HST data, working with HST file formats, and with STSDAS/IRAF tasks. If not, we highly recommend the perusal of the first three chapters in the HST Data Handbook.

**Figure 1:** An illustration of the STIS Time-Tag observing mode (based on Fig. 11.5 from the STIS Instrument Handbook).



The Time-Tag data products from the HST Archive are the same as for other STIS data with the exception of the non calibrated data product, which has a `_tag` suffix added to the root name. The `_tag` file is in FITS format with multiple file extensions. It consists of a primary header ([0] extension), one or more `EVENTS` tables, depending on the number of buffer dumps ([1] to [n] extensions), and a single `GTI` or Good-Time-Interval table ([gti] extension). Information about the observation and data is presented as keywords in two headers: a *primary header*, which for the purposes of this report is always in the [0] extension, and a *science header*, which is in the `_tag` file [1] to [n] extension, for n `EVENTS` extensions. The primary header provides general information for the observation and the science header is specific to the Time-Tag data or, in the case of the Accum mode, to each science sub exposure (i.e., [SCI,1] to [SCI,n] extensions, where n is the number of sub exposures).

The `EVENTS` table has four columns (see Figure 1): for each event there is (1) `TIME`, relative spacecraft time (see Section “2.1 Time-Tag Exposure Timing”), (2) `AXIS1`, Doppler-corrected pixel coordinate in the dispersion direction, (3) `AXIS2`, pixel coordinate in the spatial direction, (4) `DETAXIS1`, pixel coordinate in the dispersion direction not Dop-

pler corrected. The GTI table lists the relative start and end times for uninterrupted events. It consists of two columns: (1) *START*, start time of the good interval and (2) *STOP*, end time of the good interval. The time values recorded in the *EVENTS* and *GTI* tables are relative to the exposure start time.

## 2. Working with Time-Tag Data

In this section we provide details on several issues and questions that were raised by Time-Tag users, including a description of Time-Tag timing, useful IRAF tasks, heliocentric and barycentric time correction, converting Time-Tag data to an image, working with echelle Time-Tag data, and timing analysis.

### 2.1 Time-Tag Exposure Timing

To determine the start time, end time, and duration of a Time-Tag observation use the Time-Tag header keywords *TEXPSTRT*, *TEXPEND*, *TEXPTIME*, which may be read with the IRAF task *hselect* from the *images.imutil* package:

```
cl> hselect rootname_tag.fits[0] $I,texpstrt,texpnd,texptime yes
```

where *rootname\_tag.fits* is an imaginary Time-Tag file, *\$I* lists the filename and *yes* unconditionally prints the values of the keywords.

The keywords *TEXPSTRT*, *TEXPEND*, and *TEXPTIME* are the *total* start time, end time, and duration, found in the primary header ([0] extension). The values for the science header (e.g., [1] extension) keywords *EXPSTART*, *EXPEND*, *EXPTIME* will be different for multiple sub exposures, after processing the data using *inttag*.

The values in the *\_tag* file *TIME* column are in relative spacecraft time units. The conversion to seconds is transparent for those using IRAF tasks, such as in the *tables* or *stis* packages, as well as, for most FITS reader utilities. However, users writing their own software will have to apply a linear conversion for each event time. If you are not sure whether you need to apply the conversion, check the time of several events. If they are integer values then the time is in spacecraft time units. The conversion from units of spacecraft time to seconds for each event or *TIME(row)* is:

$$\text{TIME}(\text{row}) * \text{TSCAL1} + \text{TZERO1}$$

The *TSCAL1* and *TZERO1* values are keywords in the Time-Tag table science header. These may be read using the task *thedit* in the *tables.ttools* package:

```
cl> thedit rootname_tag.fits tzerol .
```

```
cl> thedit rootname_tag.fits tscall .
```

Make sure to use ‘.’ with *thedit* so that the keyword values are not overwritten.

The *TSCAL1* parameter is the conversion factor for the spacecraft time unit:

$$\text{TSCAL1} = 0.000125 \text{ second/unit}$$

The TZERO1 value (in seconds) is the time when the first event was recorded. It is also the TIME column zero point. TZERO1 is relative to TEXPSTRT, given in MJD (Modified Julian Date), which is converted from the spacecraft time.

Time-Tag mode has two timing delays that while understood are not immediately obvious. Following the exposure start time being recorded by the on-board computer (i.e., TEXPSTRT) there is a period of approximately 25 milliseconds during which no events can be processed due to other initialization activities in the software. Once this period has elapsed the events will be processed as they arrive. This means that the data will always have a lag between the recorded start time and the first event of 25 milliseconds plus the arrival delay of the first event, which depends on the source count rate. The second is a 200 milliseconds delay between the exposure formally ending and the last event being processed. This delay is also due to the software. The timer that signals the end of the exposure is separate from the event processing resulting in an inherent delay due to communication between the two systems.

**Figure 2:** Simplified Time-Tag mode flow diagram.

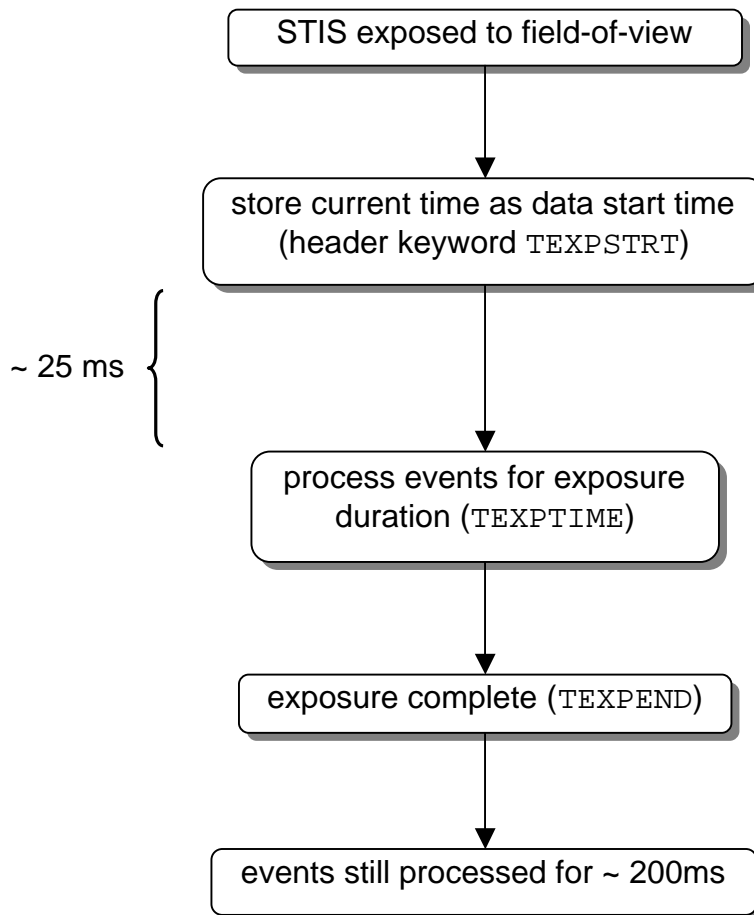


Figure 2 shows the simplified processes involved in the operation of the STIS Time-Tag mode. There is no concern about shutter shading effects, such as with the old generation HST instruments. The shutter is completely open before the start of the exposure. At the end of the exposure, when the event stream is no longer recorded, the shutter stays open until the Time-Tag related Flight Software operations are complete.

## 2.2 Useful IRAF Tasks

The STSDAS/IRAF tasks useful for working with STIS Time-Tag data are listed in Table 1. These tasks are described in the sections below and in the examples. The IRAF tasks for timing analysis are listed in Table 4.

**Table 1.** Useful IRAF tasks for analyzing and manipulating data taken in the Time-Tag observing mode. Refer to IRAF help files for more details.

| IRAF Task           | STSDAS Package   | Purpose   |
|---------------------|------------------|---|
| THEDIT and THSELECT | tables.ttools    | Read the values for the science header keywords.      |
| HEDIT and HSELECT   | images.imutil    | Read the values for the primary header keywords.      |
| HSTEPHEM            | hst_calib.ctools | Convert the ephemeris file into an STSDAS table.      |
| ODELAYTIME          | hst_calib.stis   | Apply the light delay correction.                     |
| INTTAG              | hst_calib.stis   | Convert Time-Tag events into an Accum image.          |
| CALSTIS             | hst_calib.stis   | Calibrate the Accum image and extract a 1-D spectrum. |
| SGRAPH              | graphics.stplot  | Display the 1-D spectrum.                             |
| ECHPLOT             | hst_calib.stis   | Generate plots of STIS extracted spectral orders.     |

## 2.3 Heliocentric and Barycentric Time Correction

Time-Tag observations using STIS have a temporal resolution of 125 microseconds. The effects on the observed times of both the orbital motions of the Earth and the HST may need to be corrected relative to the solar-system barycenter. A task in the IRAF package `stsdas.hst_calib.stis` called `odelaytime` corrects the column of time events for light delay from three sources: (1) general relativistic effects (up to 2 milliseconds), (2) displacement of the telescope from the center of the Earth (up to 20 milliseconds), (3) and displacement of the Earth from the solar-system barycenter (up to 500 seconds). Doppler correction to the wavelength (`AXIS1` position) of each event have already been applied in the `_tag` file.

In order to apply the `odelaytime` timing corrections, the task requires information about the spacecraft position and velocity during your observations. That information is

stored in the engineering telemetry that may be downloaded via the HST Archive's Starview tool. You will need to retrieve the HST ephemeris file from Starview, convert it to a FITS table with the task `hstephem`, and use `odelaytime` to perform the correction. The input to `odelaytime` is the Time-Tag data file and the ephemeris file. The parameters are listed in Table 2. Refer to Section 5 for detailed steps.

The **odelaytime** task will be updated in the next release (around Feb. 2001) so that the column `TIME` will be modified in place (i.e., the `out_col` parameter will not be necessary) and the `GTI` values will be updated to reflect the corrected time values. The keywords `TEXPSTRT` and `TEXPEND` in the primary header, and `EXPSTART` and `EXPEND` in each `EVENTS` table header will be corrected for the light delay. Also, the updated version of `odelaytime` will write a keyword (`DELAYCOR= ' COMPLETE '`) to the data primary header, so you cannot run the task twice on the same file.

**Table 2.** List of parameters for `ODELAYTIME` (task will be modified in the next release).

| <b>ODELAYTIME Parameter and Defaults</b>                  | <b>Description</b>   |
|---|--|
| <code>input = ""</code>                                   | Input table template name.   |
| <code>earth_ephem = "stsdas\$data/fits/de200.fits"</code> | Table name of earth's state vectors (do not change).   |
| <code>obs_ephem = ""</code>                               | Table name of observer's state vectors at the time of the observation (download file from Archive).  |
| <code>distance = 0</code>                                 | Distance of the target (i.e., parallax).   |
| <code>dist_unit = "arcsec"</code>                         | Unit of distance.  |
| <code>(in_col = "TIME")</code>                            | Input time column name.  |
| <code>(out_col = "HTIME")</code>                          | Output (corrected) time column name. Suggested <code>out_col</code> value is <code>TIME</code> . This parameter will be removed in the next release. |

#### ***2.4 Converting Time-Tag Data to an Image***

To manipulate the event data in the `_tag` file, use the IRAF task `inttag`, available in the `stsdas.hst_calib.stis` package. Refer to the STIS Instrument Handbook for a description of the Time-Tag mode.

After you retrieve your Time-Tag data you may customize the reduction process with the IRAF task `inttag` in the `stis` package, which is used to convert a Time-Tag file into any number of sub exposures, such as for accumulated photon data, with a user input time interval. `inttag` parameters are listed in Table 3. The parameters in brackets are optional and list the default setting. The input file is the Time-Tag data, appended with the `_tag` suffix. The output parameter cannot be left blank. The recommended nomenclature is to use the `_raw` suffix in the output name, since the output is in a similar format

as the MAMA Accum mode `_raw` data. The default (`INDEF`) start time will be the first value in the `GTI` table. You may wish to test the time increment so that each sub exposure has reasonable signal-to-noise. For more details refer to the IRAF help pages for the `inttag` task.

**Table 3.** List of important parameters for the task `inttag`.

| INTTAG Parameters and Defaults   | Description  |
|----------------------------------|--|
| <code>input = ""</code>          | Time-Tag data, e.g., <code>o45701c0q_tag.fits</code> .   |
| <code>output = ""</code>         | Name of output file, e.g., <code>o45701c0q_raw.fits</code> .   |
| <code>(starttime = INDEF)</code> | Starting time (in seconds). <code>INDEF</code> refers to 1st <code>GTI START</code> value.   |
| <code>(increment = INDEF)</code> | Time increment (in seconds). Exposure will span to last <code>GTI STOP</code> value if the default value <code>INDEF</code> is used. |
| <code>(rcount = 1)</code>        | Number of time intervals or output sub exposures.  |
| <code>(highres = no)</code>      | Choose high or low resolution output. See Section “3.1 MAMA Spatial Resolution Modes”.   |
| <code>(allevents = no)</code>    | Use <code>no</code> for multiple sub exposures. Use <code>yes</code> for a single sub exposure.                                      |

### 2.5 Echelle Time-Tag Data

The STIS echelle Time-Tag data files require longer processing and calibration time, since they consist of many orders, especially if you are using multiple sub exposures. After converting the Time-Tag data into an image and calibrating it, two IRAF tasks that you may find useful for working with echelle Time-Tag data are `txttable` and `echplot`. Refer to the HST Data Handbook or the IRAF help files for more information on these tasks.

The output flux-corrected, extracted spectra are in binary tables with arrays including flux, wavelength, flux error, and spectral extraction parameters. The task `txttable` may be used to extract wavelength-dependent data for a single order from the `_x1d` file.

Example:

```
cl> txttable "rootname_x1d.fits[3][r:sporder=116][c:wavelength,flux]" \
table.tab
```

where `[3]` refers to the third sub exposure, the desired order is 116, and the wavelength and flux columns written to the output table `table.tab`. All the arrays for the specified order will be written to the output table if the column names are not specified. You may also write an output table for each order:

```
cl> txttable "rootname_x1d.fits[sci,1][c:wavelength, flux, error]"
```

this will create `N` tables, where `N` is the total number of orders, and the tables will be called `rootname_x1d_r0001.fits ... rootname_x1d_r000N.fits`.

A very useful task to display echelle spectra is `echplot`. There are 4 styles of plotting: (1) order-by-order display (`plotstyle=single`), (2) display over the entire wavelength range of the echelle (`plotstyle=multiple`), (3) display the background, gross, and net counts, flux, flux error, and data quality bits for each order (`plotstyle=diagnostic`), (4) display 4 orders simultaneously (`plotstyle=panel`). Output may also be directed to a postscript file or IGI script.

## 2.6 Timing Analysis Tasks

There are several IRAF packages that can be used for the timing analysis. The `stis` package maintained at STScI has the `inttag` task which may be used to create sub exposures for a fixed time interval, as discussed in Section “2.4 Converting Time-Tag Data to an Image”. Other tasks that may be useful for timing analysis are listed in Table 4.

The timing analysis tasks in IRAF that are part of the PROS/XRAY Data Analysis software are not supported by STScI. Documentation may be accessed using the IRAF help files or online at [hea-www.harvard.edu/PROS/pros.html](http://hea-www.harvard.edu/PROS/pros.html). There are at least two issues that you may need to resolve before proceeding with your analysis: (1) merging Time-Tag event tables and (2) converting Time-Tag FITS data into QPOE format.

### *Merging Time-Tag Event Tables*

Many Time-Tag data sets will consist of multiple EVENTS extensions in a single FITS file, depending on the number of buffer dumps. There are plans to change this in the ground system, for now it may be convenient to merge them into a single EVENTS table. However, merging the EVENTS extensions can take a long processing-time, depending on the total number of events. The task `tmerge` will run faster in the `tables` package (better than Version 2.2). On the other hand, it may save time since you will not have to repeat processing steps for each EVENTS extension, also, some software or tasks may require a single EVENTS extension. Refer to the task `stis2qp` in Appendix B for an example of a script that merges multiple EVENTS extensions into a single EVENTS extension.

To merge the EVENTS tables in IRAF, use the task `tables.ttools.tmerge`. Example, for an imaginary Time-Tag file, called `rootname_tag.fits`, containing 3 EVENTS extensions:

```
cl> tmerge "rootname_tag.fits[1], rootname_tag.fits[2],\  
rootname_tag.fits[3]" "rootname_all.fits" option="append"
```

where `rootname_all.fits` is the output file and the type of merging is `append`. Depending on the software that you are using you may need to include the GTI table, which will not be part of the merged product. To copy the GTI table use the task `tables.ttools.tcopy`:

```
cl> tcopy "rootname_tag.fits[gti]" "rootname_all.fits[2]"
```



where `rootname_tag.fits[gti]` is the input table and `rootname_all.fits[2]` is the file name and extension of the merged output table.

### *Converting STIS Time-Tag Data into QPOE Format*

To use STIS Time-Tag data with PROS tasks requires that the FITS file be converted into QPOE format. At least three conditions must be met for the conversion to be successful with standard STIS Time-Tag data:

1. the TIME column values must be *double* precision,
2. the keyword XS-SORT=TIME must be in the FITS science header, and
3. multiple EVENTS tables should be merged into a single EVENTS table.

The task `fits2qp` is available in the `xray.xdataio` package, however, it is not always successful in converting STIS data. In the Appendices, we've included an IRAF script to convert STIS Time-Tag FITS data into QPOE format called `stis2qp`, which will also merge multiple EVENTS extensions. An advantage to using the merging procedure in the task `stis2qp` is that the TIME values are formatted using double precision. There are two versions of `stis2qp` (refer to Appendix A and B) optimized for different versions of the `tables` package.

**Table 4.** IRAF tasks to prepare Time-Tag data for timing analysis and timing tasks.

| IRAF Task | IRAF Package               | Description  |
|-----------|----------------------------|--|
| TMERGE    | <code>tables.ttools</code> | Merge (option=append) the Time-Tag EVENTS tables into a single FITS file.  |
| TCOPY     | <code>tables.ttools</code> | Copy a table or a binary table from a FITS file with more than one extension.                                    |
| PDM       | <code>noao.astutil</code>  | Find periods in light curve data by Phase Dispersion Minimization.   |
| FITS2QP   | <code>xray.xdataio</code>  | Convert Time-Tag data into the PROS package QPOE format for use with time series analysis tasks (sse Section 7). |
| PERIOD    | <code>xray.xtiming</code>  | Find or verify the period from a data set.   |
| FLDPLOT   | <code>xray.xtiming</code>  | Plot the light curve with the period folded into it.   |
| CHIPLOT   | <code>xray.xtiming</code>  | Chi-square plot for various periods.   |
| LTCPLOT   | <code>xray.xtiming</code>  | Plot simple light curve from a data set.   |

### **3. Time-Tag Data Calibration**

After processing with `inttag`, the `_raw` data output may be calibrated using `calstis`. Refer to the HST Data Handbook Chapter 21 for a description of `calstis`. The

same rules apply as with STIS data taken in other modes, described in the HST Data Handbook and the STIS Instrument Handbook. However, two calibration steps that require some explanation for Time-Tag mode are the Doppler correction and the spatial resolution correction.

### ***3.1 MAMA Spatial Resolution Modes***

MAMA data has two spatial resolution modes, low resolution and high resolution. These modes are a product of the way the anode array encodes the data. In the low resolution mode the detector has 1024x1024 pixels and each pixel measures 25x25 microns. In the high resolution mode the data are sub-sampled and the detector effectively has 2048x2048 pixels, each low resolution pixel having been divided by four 12.5x12.5 micron pixels. MAMA data, including Time-Tag data, are taken in the high resolution mode. There is a spatial resolution gain in going to the 2048x2048 format, which is approximately 10% (Sahu *et al.* 1996). However, since the flat-field images used for calibration are more variable in the 2048x2048 pixel format and the gain in spatial resolution is usually not significant it is recommended that the low resolution be used. The calibrated products retrieved from the HST Archive are converted to the low resolution mode.

#### *High Resolution Mode*

If the `highres` parameter is set to `yes` then the `LORSCORR` calibration switch in the primary data header (FITS file extension: [0]) should be set to `OMIT` and the output calibrated data will have 2048x2048 pixel format. The `LORSCORR` calibration switch set to `OMIT` tells the calibration software to leave the reference files, such as darks and flats, in the original format of 2048x2048 pixels.

#### *Low Resolution Mode*

If the `inttag` parameter `highres` is set to `no` then the output science data will have low resolution, 1024x1024 pixel format. It doesn't matter how the `LORSCORR` calibration switch is set, since, the software will bin the reference files to the 1024x1024 pixel format.

### ***3.2 Doppler Correction***

For the Time-Tag mode the data is in a form of a table (see Figure 1) with columns for time (`TIME`), X- (`AXIS1`) and Y- (`AXIS2`) pixel positions, and a fourth column called `DETAXIS1`, which is the X-pixel position without the Doppler correction. The `AXIS1` column gives the Doppler-corrected data, similar to the MAMA science data taken in the Accum mode using first order M or echelle gratings. Refer to the STIS ISR 98-05 (Sahu *et al.* 1998) for more details about the Doppler correction for STIS data.

### *Doppler Correction for Calibration Reference Files*

The header keyword DOPPCORR is used to apply a Doppler correction to the reference files such as the data quality, flat-field, and dark files used during calibration of the science data. The parameters for the Doppler correction are in the science header. These are DOPPMAG (Doppler shift magnitude), DOPPZERO (Doppler shift zero phase time), and ORBITPER (orbital period used on board for Doppler correction). The function that is used to calculate the Doppler shift is given in the HST Data Handbook Chapter 21. To check the related keywords from the science data in IRAF use the task `thselect` in the `tables.ttools` package:

```
cl> thselect rootname_tag.fits[1] doppon,doppzero,doppmag,dopper yes
```

### *Checking Whether Doppler Correction is Needed*

Normally, the header keyword DOPPCORR will be properly set by the ground system. The Doppler correction may be omitted if the Doppler shift (given by the DOPPMAG science header keyword) is less than a pixel. If it is greater than a pixel then the calibration keyword DOPPCORR in the science header should be set to PERFORM to correct the reference files. To verify that the Doppler correction was done (for Archive or re-calibration products) check the HISTORY in the primary header, do not refer to the keyword calibration switch, which may be misleading.

## **4. Data Management**

Time-Tag data processing can take up a lot of disk space and processing-time. Each event position (AXIS1, AXIS2, DETAXIS1) is recorded as a 16 bit integer (2 bytes) per data value and each TIME value is recorded as a 32 bit integer (4 bytes). The GTI table contents are recorded in 64 bit double precision (8 bytes) per data values.

### **4.1 Data Volume**

The standard delivered products and corresponding disk space volume estimate is given in Table 5. Also, a disk space volume estimate for the calibrated Time-Tag products is given in Table 6. The disk volume estimates are based on about 60 Time-Tag data sets from observations during 1999.

**Table 5.** Non calibrated Time-Tag products downloaded from the HST Archive.

| Suffix | Type  | Contents                                   | Data Volume Estimate    |
|--------|-------|--|-------------------------|
| _raw   | image | Raw science (2048x2048 pixels).            | 16.8 Mbytes             |
| _tag   | table | Time-Tag EVENTS and GTI tables.            | (no. events x 10) bytes |
| _spt   | image | Support file (planning & telemetry info.). | < 1 Mbytes              |
| _wav   | image | Wavelength calibration or wavecal file.    | < 5 Mbytes              |
| _wsp   | image | Wavelength calibration support file.       | < 0.1 Mbytes            |
| _asn   | table | Association file.                          | < 0.02 Mbytes           |
| _trl   | table | Trailer file.                              | < 1 Mbytes              |
| _lrc   | image | Local rate check image.                    | < 0.1 Mbytes            |
| _lsp   | text  | Local rate check support file.             | < 0.01 Mbytes           |
| _pdq   | table | Data quality file.                         | < 0.02 Mbytes           |

Table 5 gives the Time-Tag non calibrated products retrieved from the HST Archive and the corresponding data volume estimates. Jitter files (\_jit and \_jif suffix) are optional products and are not included in this table. The primary data header was not included in the volume estimate, it is usually less than 0.05 Mbytes per file.

To check the number of events in a \_tag file (for an imaginary input file called `rootname_tag.fits`) use the IRAF task `catfits`:

```
cl> catfits "rootname_tag.fits" long=yes | match NAXIS2
```

The number of events is equal to the number of rows, given by the keyword `NAXIS2`, in an extension.

#### ***4.2 Processing-Time Example***

The majority of the processing-time is used by the STIS calibration software, `sts-das.hst_calib.calstis`. Table 7 gives the processing-times (approximated to the nearest minute) for two UltraSPARC Workstations, an Ultra 1 and Ultra 10, manufactured by Sun Microsystems. The Ultra 1 has a 167 MHz UltraSPARC processor and the Ultra 10 has a 300MHz UltraSPARC-IIi processor. More statistics are given in Section "4.3 Processing-Time Comparison". The calibrated Time-Tag data volume estimates are given in Table 6.

**Table 6.** Time-Tag data calibrated with the current STIS package version. The primary header is not included in the data volume estimate. Its size varies depending on the process and may add up to 0.1 Mbytes per file to the total volume.

| File Suffix     |        | IRAF Task   | File Type | Data Volume Estimate  |
|-----------------|--------|---|-----------|---|
| Input           | Output |   |           |   |
| _tag            | _tag   | ODELAYTIME  | table     | (no. events x 18) bytes   |
| _tag            | _raw   | INTTAG (highres=yes)                                | image     | (33.6 x no. sub exposures) Mb   |
| _raw            | _flt   | CALSTIS or BASIC2D<br>(low resolution)              | image     | (10.5 x no. sub exposures) Mb   |
| _raw or<br>_flt | _x1d   | CALSTIS or X1D<br>(extracted spectrum)              | table     | (0.06 x no. sub exposures) Mb for<br>1st order modes<br>(35557 x no. orders x no. sub expo-<br>sures) Bytes for echelle modes   |
| _raw or<br>_flt | _x2d   | CALSTIS or X2D<br>(rectified spectrum)              | image     | (14.5 x no. sub exposure in _flt) Mb<br>for 1st order modes<br>(0.5 x no. orders x no. sub expo-<br>sures) Mb for echelle modes |
| _raw or<br>_flt | _sx2   | CALSTIS<br>(summed image for multiple<br>exposures) | image     | 14.5 Mb for 1st order modes<br>(0.5 x no. orders) Mb for echelle<br>modes   |

**Table 7.** Time-Tag data processing for two UNIX machines. The data consists of a total 40 minute exposure that was converted to an Accum image with 1 minute interval subexposures.

| Process                 | Disk Space Used  | Ultra 1 Time | Ultra 1 Rate                            | Ultra 10 Time | Ultra 10 Rate                           |
|-------------------------|--|--------------|---|---------------|---|
| Not processed           | <i>tag</i> input file: 29 Mb<br>(total 2891881 events)   |              |   |               |   |
| ODELAYTIME              | <i>tag</i> output file: 52 Mb  | 8 minutes    | 6024 events/sec                         | 6 minutes     | 8033 events/sec                         |
| INTTAG                  | <i>raw</i> output file: 300 Mb<br>(1171 events/sub exposure)                                       | 11 minutes   | 16 sec/sub exposure<br>or 73 events/sec | 3 minutes     | 4 sec/sub exposure<br>or 292 events/sec |
| CALSTIS<br>(Calibrated) | <i>flt</i> output file: 400 Mb<br><i>x1d</i> output file: 2.2 Mb<br><i>x2d</i> output file: 600 Mb | 79 minutes   | 118 sec/sub exposure                    | 26 minutes    | 39 sec/sub exposure                     |
| <b>Total</b>            | ~1.4 GB  | 98 minutes   |   | 35 minutes    |   |

### 4.3 Processing-Time Comparison

The total time for `inttag` and `calstis` processing for two unix machines is considered in Table 8 using different sub exposure durations. For equivalent sub exposures, an

Ultra 1 will take about 40% longer than an Ultra 10 to process 4 long sub exposures and 64% longer to process 40 short sub exposures. Each sub exposure will have overhead time such as for writing header information. `odelaytime`, `inttag`, and `calstis` processing are included in the totals. Refer to Table 7 for a break down of processing-time in the case of 40 sub exposures in 1 minute intervals.

The test data consists of 2891881 events accumulated in 2400 seconds (the Crab Pulsar). As shown in Table 7, a total of 40 minutes is processed in `inttag` for 1 minute intervals, which averages to approximately 1171 events per sub exposure. Prior to `inttag` and `odelaytime` the processing of the HST ephemeris file is very fast (a few seconds) and takes less than a Mbyte of space. The output `_raw` file has 40 sub exposures. The calibration process requires the `_raw` file and a `_wav` or wavelength calibration file as input. It outputs the `_flt` (flat-field corrected), `_x1d` (spectral extraction), and `_x2d` (spectral rectified image) files. Omitting the spectral rectification or `X2DCORR` step will decrease the processing-time (e.g., by about 4 minutes on a Ultra 10). The `calstis` calibration steps may also be performed separately and the input files deleted following processing. This is especially attractive if disk space is limited.

**Table 8.** Comparison of processing-times for an Ultra 1 and Ultra 10, using the same data as used for Table 7. The percent difference gives the increase in processing-time for an Ultra 1.

| Time-Tag Data Processed with<br>ODELAYTIME, INTTAG and<br>CALSTIS | Ultra 1<br>Total Processing-<br>Time | Ultra 10<br>Total Processing-<br>Time | Ultra 1 Processing-<br>Time Longer than<br>Ultra 10 by ... |
|---|--------------------------------------|---------------------------------------|--|
| 40 sub exposures<br>in 1 minute intervals<br>(1171 events/minute) | 98 minutes                           | 35 minutes                            | 63 minutes   |
| 4 sub exposures<br>in 10 minute intervals<br>(1181 events/minute) | 20 minutes                           | 12 minutes                            | 8 minutes  |

## 5. Heliocentric Time Correction Example

For Section 5, Section 6, and Section 7 use the Crab Pulsar data (proposal ID 7108). It is non-proprietary and may be downloaded by any user with an HST Archive account. Information on registering to receive HST data and using StarView is given online at [starview.stsci.edu](http://starview.stsci.edu). Analysis and results of the Crab Pulsar data taken by STIS were published by Gull *et al.* (1998). Also, some added details on the Time-Tag mode are given by Lindler *et al.* (1997). You may either request all the observations by entering the proposal ID or by searching for the data root names `o45701c0q` and `o45701c2q` (if you need to save disc space the data set `o45701c0q` will suffice). You should retrieve the *best* reference files and update the reference file keywords in the primary data header. Also, it is not necessary

to retrieve the calibrated products since we will be calibrating the data in Section 6. The reference files requires 121 Mbytes, the data set o45701c0q requires 37 Mbytes, and the data set o45701c2q requires 39 Mbytes of disk space.

### 5.1 Retrieve HST Ephemeris Files

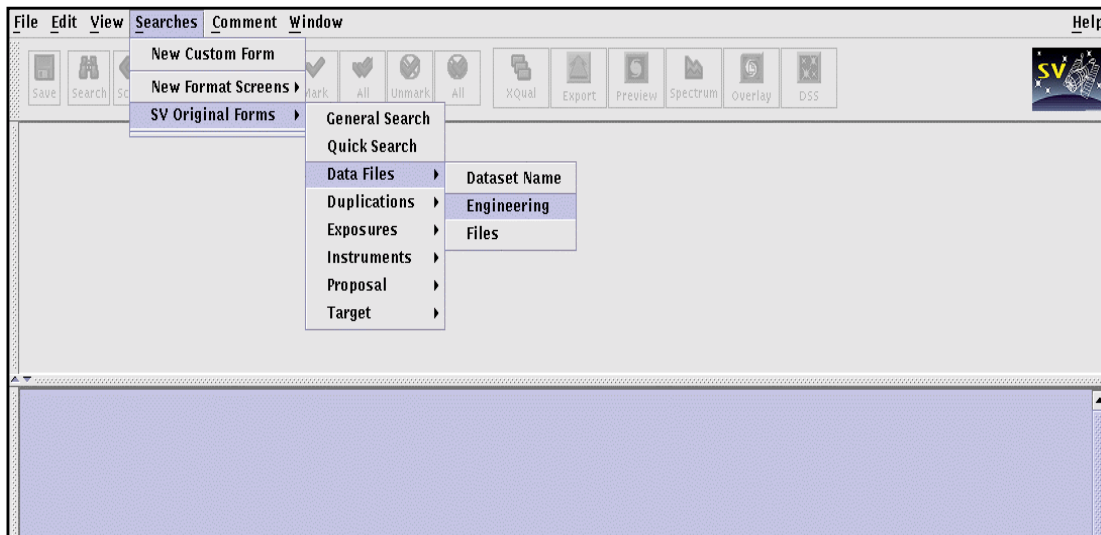
In this step we retrieve an ephemeris file for the date of the observations using StarView version 6.0. From within IRAF check the observation date and time of the Crab Pulsar Time-Tag data:

```
cl> hselect *tag.fits[0] $I,tdateobs,itimeobs yes
o45701c0q_tag.fits[0] 07/08/97 16:54:53
```

From the StarView startup screen choose the **Searches** menu, choose **New Format Screen**, choose **Data Files**, and select **Engineering**, shown in Figure 3.

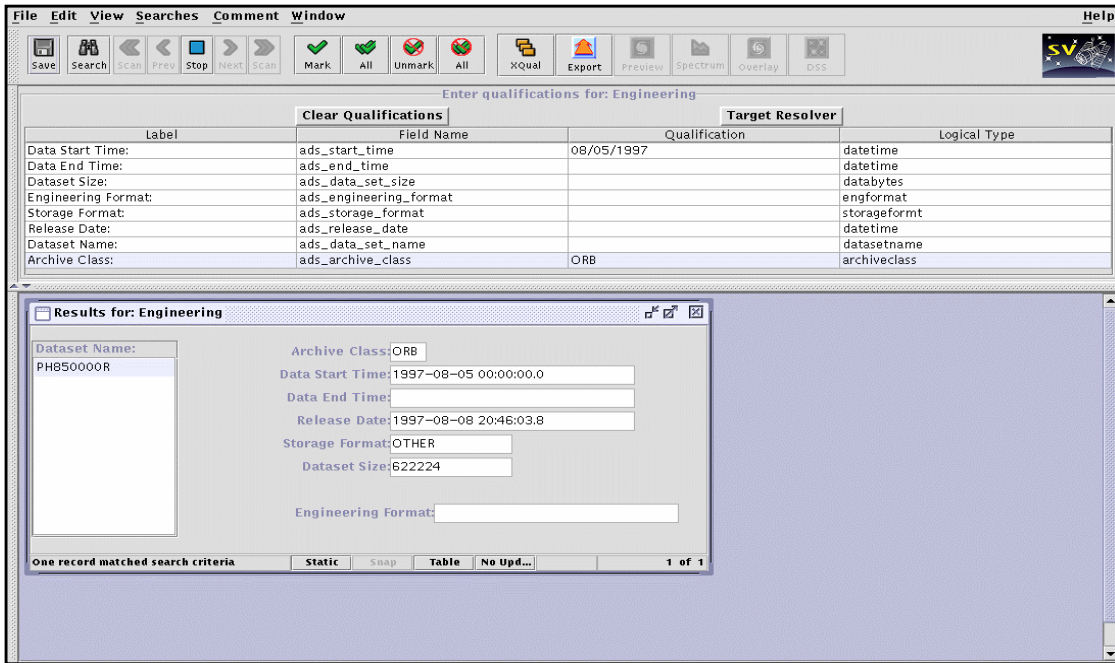
When the **Engineering** screen appears, fill in the **Data Start Time: 08/05/1997** and **Archive Class: ORB** fields in the *Qualification* column (third) in the table found below the button tool bar, as shown in Figure 4. Then click on the *Search* button.

**Figure 3:** StarView version 6.0 **Searches** menu options.



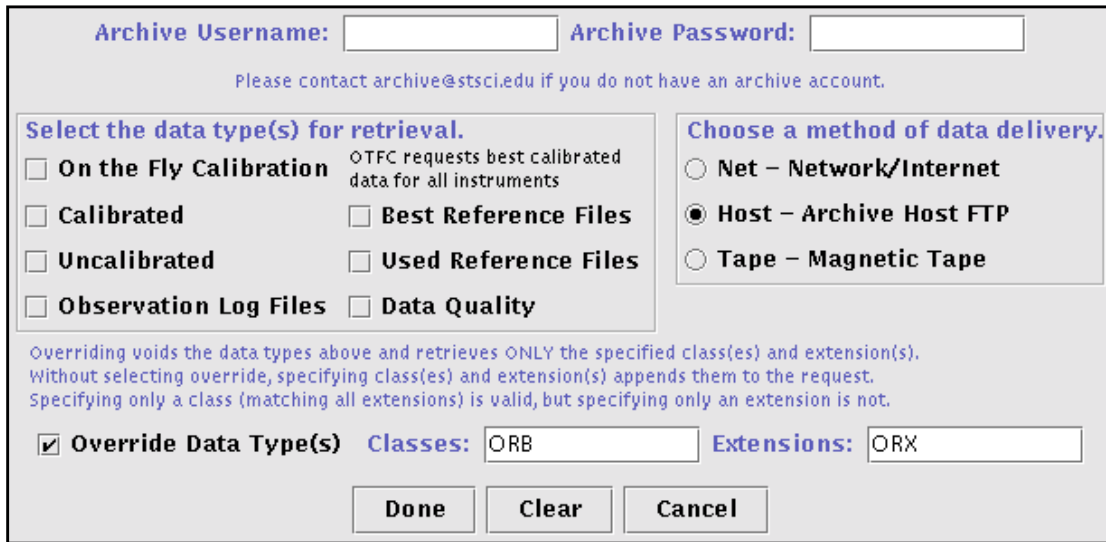
The date format used is mm/dd/yyyy (i.e., month/day/year), Universal Time. Notice that the date in the primary header is given in the format dd/mm/yyyy. In more recent FITS headers the dates will be yyyy-mm-dd, possibly with a time appended (e.g., 2000-08-23T11:14:27.3). You may have to try several dates within 2 or more days of your observations, since the ephemeris files are not generated daily. It also saves time to get more than one potential ephemeris file that covers a range of dates, if you are uncertain whether it includes the date and time of your observation.

**Figure 4:** StarView version 6.0 **Engineering Data** screen with search results.



Once the search is complete, select and *Mark* (from the button tool bar) the file PH850000R. This will bring up the **StarView Retrieval** window. Click *Submit*.

**Figure 5:** StarView version 6.0 **Retrieval Configurations Options** screen.



Next the **Retrieval Configurations Options** screen will appear, shown in Figure 5. Select **Override Data Type(s)**, fill in **Classes: ORB** and **Extensions: ORX**. Also, fill in your Archive user name and password then select *Done*.



## 5.2 Convert the Ephemeris File to a STSDAS Table

You will need to convert the ephemeris file (`ph850000r.orx`) to a STSDAS table (`ph850000r.tab`), using the `stdas.hst_calib.ctools.hstephem` task:

```
cl> hstephem ph850000r.orx ph850000r.tab
```

Check the table statistics using `tables.ttools.tstat` to assure yourself that the observation time interval is included in the ephemeris table:

```
cl> tstat ph850000r.tab mjd
# ph850000r.tab mjd
# nrows      mean      stddev      median      min      max
4321         50666.5    0.866326   50666.5     50665.   50668.
```

where the input column is called `mjd`. The time of the observation (approximately 50667.7 MJD) falls in the range defined by the `min` or minimum time (in MJD) and the `max` or maximum time (in MJD).

## 5.3 Heliocentric and Barycentric Time Correction

Currently, `inttag` only uses the values in a column called `TIME`, so it is convenient to overwrite the existing `TIME` column. To correct the time for heliocentric and barycentric light delay use the task `stdas.hst_calib.stis.odelaytime`. The name of the output column is specified as the `out_col` parameter (which will be removed in the next release), and the input ephemeris table is specified by the `obs_ephem` parameter:

```
cl> odelay o45701c0q_tag.fits obs_ephem=ph850000r.tab out_col="TIME"
```

The Crab Pulsar's annual parallax is small enough that the `distance` parameter may be left as zero. In this example, the input file is `o45701c0q_tag.fits` and the input ephemeris table is `ph850000r.tab`.

## 6. Calibration Example

We will use the Crab Pulsar data to convert the Time-Tag event stream into an image. Also, we will calibrate the data, which requires the *best* reference files. You will need to update the reference file keywords in the primary header. The procedure for this is described in the HST Data Handbook. The Crab Pulsar profile, shown in Figure 6, may look different if other reference files or keywords are used.

### 6.1 Determine the Start, Duration, and End Time

To determine the start (in MJD), end (in MJD), and duration (in seconds) of your observation print the value of the header keywords `TEXPSTRT`, `TEXPEND`, and `TEXP-TIME`, using the task `images.imutil.hselect`:

```
cl> hselect o45701c0q_tag.fits[0] texpstrt,texpend,texptime yes
```

The output is:

5.066770478669E+04

5.066773256447E+04

2400.000000

## 6.2 Convert the Data into an Image

To convert the Time-Tag data into an image, such as one would get using the Accum mode, use the task `inttag` in the `stsdas.hst_calib.stis` package. Before running `inttag` make sure that the calibration switches are set properly in the `_tag` file, so that they are inherited by the `inttag` output file (see Section “6.3 Calibrate the Data”).

```
cl> inttag o45701c0q_tag.fits o45701c0q_raw.fits highres=yes \
allevents=yes
```

The input file is `o45701c0q_tag.fits` and the output file name is `o45701c0q_raw.fits`.

For multiple sub exposures you will want to use the `increment` and `rcount` parameters. The `increment` parameter is given in seconds and refers to the time interval of each sub exposure. The `rcount` parameter refers to the number of sub exposures. You will also want to set the `allevents` parameter to `no`.

## 6.3 Calibrate the Data

You will need to make sure that you have all the necessary calibration reference files and that they are specified in the primary header. Refer to the HST Data Handbook for assistance.

The Crab Pulsar observations were not configured for supported science (i.e., the primary header keyword `CFSTATUS` is set to `AVAILABLE` instead of `SUPPORTED`) since they were taken to demonstrate the performance of the Time-Tag mode on STIS. Hence, some of the calibration switches will have to be set manually. We recommend that you set these switches in the `_tag` file, then the switches will be inherited by the `inttag` output.

Doppler correction (`DOPPCORR`) of the reference files is not required since the Doppler correction is less than a pixel. We will need to perform the low resolution conversion since the `inttag` parameter `highres` was set to `yes`. The steps `SGEOCORR` (not available), `WAVECORR` (not available for the Crab Pulsar data), `RPTCORR` (only applicable with multiple sub exposures), and `X2DCORR` (not necessary for this example) may be omitted. (The wavelength calibration file is automatically taken for science data unless specified otherwise by the observer, so, `WAVECORR` is usually set to `PERFORM`.) All the other calibration switches should be set to `PERFORM`. Example:

```
cl> imhead o45701c0q_tag.fits[0] l+ | match CORR
DOPPCORR= 'OMIT' / Doppler correction needed
LORSCORR= 'PERFORM' / Convert MAMA data to Lo-Res before proceeding
DQICORR = 'PERFORM' / data quality initialization
GLINCORR= 'PERFORM' / correct for global detector non-linearity
LFLGCORR= 'PERFORM' / flag pixels for local and global nonlinearity
```

```

DARKCORR= 'PERFORM ' / Subtract dark image
FLATCORR= 'PERFORM ' / flat field data
WAVECORR= 'OMIT ' / use wavecal to adjust wavelength zeropoint
DISPCORR= 'PERFORM ' / apply 2-dimensional dispersion solutions
SGEOCORR= 'OMIT ' / correct for small scale geometric distortions
HELCORR = 'PERFORM ' / convert to heliocentric wavelengths
FLUXCORR= 'PERFORM ' / convert to absolute flux units
X2DCORR = 'OMIT ' / rectify 2-D spectral image
X1DCORR = 'PERFORM ' / Perform 1-D spectral extraction
BACKCORR= 'PERFORM ' / subtract background (sky and interorder)
RPTCORR = 'OMIT ' / add individual repeat observations

```

**Run CALSTIS:**

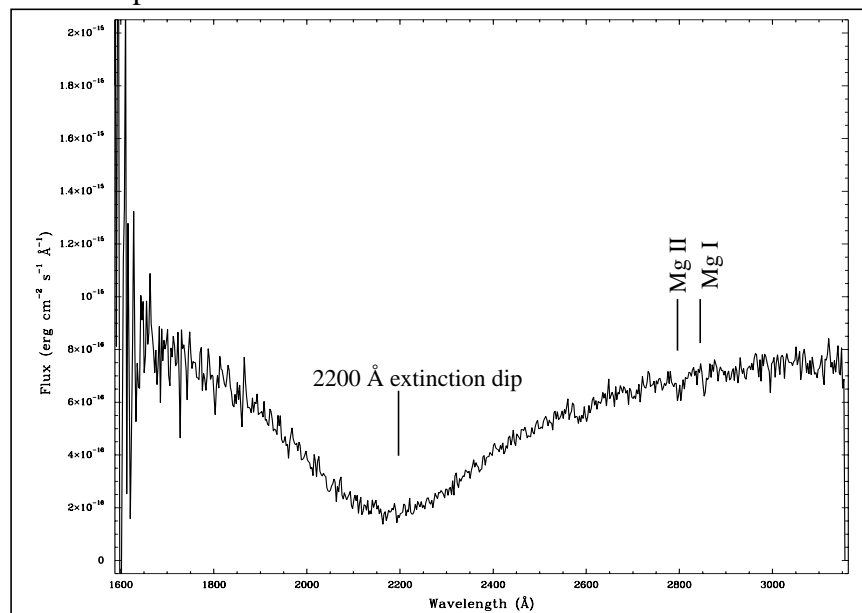
```
cl> calstis o45701c0q_raw.fits
```

In this case, we get a single sub exposure. For multiple sub exposures, the time values in the science data header (EXPSTART, EXPEND, EXPTIME) will be updated appropriately for each sub exposure. Also, with echelle data you will get multiple orders in each subexposure. To preview a spectrum you may use `stsdas.graphics.stplot.sgraph`. For `sgraph` you will need to specify the extension of the sub exposure and the data arrays to plot, example to plot the corrected flux versus wavelength:

```
cl> sgraph "o45701c0q_x1d.fits[1][r:row=1] wavelength flux"
```

For multiple sub exposures replace [1] with [n] where n is the number of the subexposure. For echelle data, to display an order you may use [r:sporder=n] where n is the order you wish to display, in place of [r:row=1]. With echelle data it is more convenient to use the `echplot` task in the `stsdas.hst_calib.stis` package (Section “2.5 Echelle Time-Tag Data”).

**Figure 6:** Summed spectrum of the Crab Pulsar.



### 6.4 Sum and Bin the Data

Repeat the above steps for the second data set, o45701c2q. We will bin the data with respect to the wavelength scale and sum the two data sets. First, generate a wavelength table using the task `stsdas.hst_calib.synphot.genwave`:

```
cl> genwave wavelength.tab minwave=1568. maxwave=3184. dwave=2.5
```

where the output table name is `wavelength.tab` the minimum and maximum wavelengths are 1568 Å and 3184 Å, and the wavelength interval is 2.5 Å per pixel. Now run the task `stsdas.hst_calib.stis.splice` using the new wavelength table:

```
cl> splice "o45701c0q_x1d.fits[1], o45701c2q_x1d.fits[1]" sum.fits \
wavetab=wavelength.tab
```

Since we are summing over the entire 3 orbits we input both the calibrated data sets. The output is called `sum.fits`. Also, we input the wavelength table. The summed spectrum is shown in Figure 6.

You may also use `splice` to bin a spectrum. `splice` allows you to choose optional weighting factors. The task `genwave` will generate a wavelength table with either an input wavelength or velocity interval.

## 7. Timing Analysis Example

There are many timing or time-series analysis tasks in the PROS/IRAF `xray` package. These were not written with STIS MAMA data in mind, however, they are still applicable. In this section we will give a very brief introduction to the timing analysis tasks using the Crab Pulsar data (for instructions on downloading the data see Section 1). The rest is left up to the diligent reader. Refer to Section “2.6 Timing Analysis Tasks” for more details.

### 7.1 Convert the Time-Tag Data to QPOE Format

In IRAF, load the `xray.xdataio` and `xray.xtiming` packages. Copy the appropriate script from Appendix A or B into a file called `stis2qp.cl` and define that script as a task:

```
cl> task stis2qp = "stis2qp.cl"
```

Convert the STIS Time-Tag data into QPOE format, using the script `stis2qp`:

```
cl> stis2qp o45701c0q_tag.fits crab
```

The input file is `o45701c0q_tag.fits` and output is called `crab.qp`.

To read the output QPOE data, for the first 20 rows, use the `xray.ximages.qplist` task:

```
cl> qplist crab firstevent=1 lastevent=20
```

`qplist` may display 4 digits after the decimal point instead of 6, which is used by STIS data. This is a formatting feature of `qplist` and does not reflect the accuracy of the data.

## 7.2 Plot a Folded Light Curve

Tasks such as `fft` and `period` in the `xray.xtiming` package may be used to find or verify the periodicity of your data. If you know the period you may plot a folded light curve using the task `xray.xtiming.fold`:

```
cl> fold crab.qp none crab fold_period=0.033473313 bins= 150 \
save_two=no
```

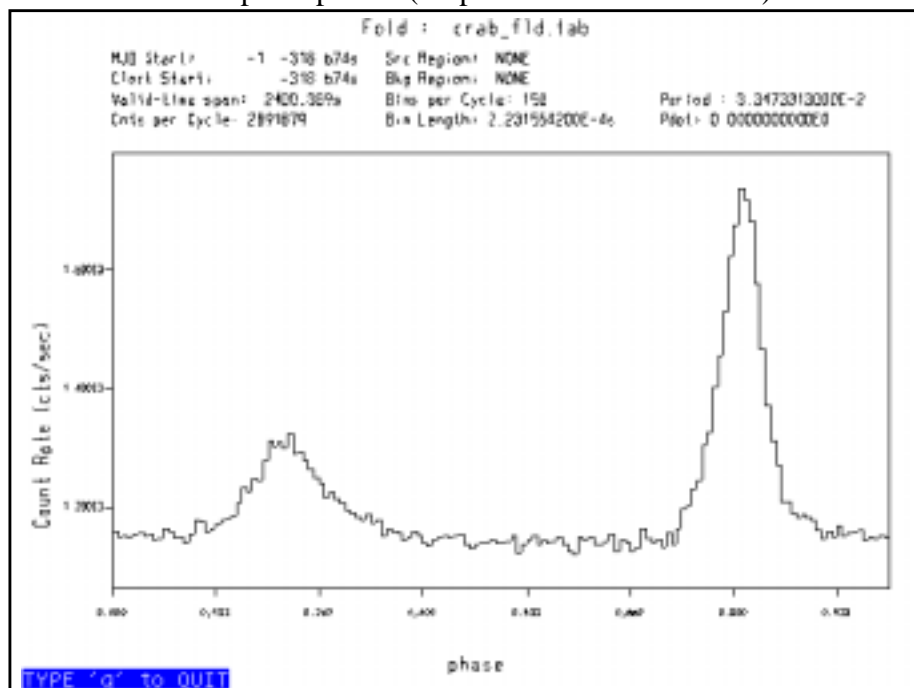
The input is the time-sorted file called `crab.qp`. We are not using a ‘background timing file’ so that parameter is set to `none`. The output file is called `crab_fld.tab`. The known period, `fold_period`, is 33.473313 milliseconds, the number of bins determined by trial and error is 150, and one phase is sufficient, thus, `save_two=no`. To plot the output, shown in Figure 7, use the task `fldplot` in the same package:

```
cl> fldplot crab_fld.tab
```

To list the output from `fold` you will need to use the task `xray.xtiming.timprint`. To read the first 20 rows, omitting the header information (`prp=no`):

```
cl> timprint crab_fld.tab 1-20 prp=no
```

**Figure 7:** The Crab Pulsar pulse profile (output from the task `fold`).



Also, you can convert the QPOE table back into a FITS file using `xray.xdataio.xwfits`:

```
cl> xwfits crab_fld.tab crab_fld.fits yes
```

where the output FITS file is a binary table (given by the boolean parameter `yes`) called `crab_fld.fits`.

## 8. Acknowledgements

Our sincere gratitude to Jerry Kriss (Spectrographs Group, STScI), Chris Long (Engineering Team, STScI), and Phil Hodge (Science Software Group, STScI) for their invaluable help in writing this report.

## 9. References

- Baum, S., Zarate, N., & Hodge, P. 1996, STIS ISR 96-13 (Baltimore: STScI)
- Gull, T. R., et al. 1998, *ApJ*, **495**, L51
- Lindler, D. J., Gull, T. R., Kraemer, S. B., & Hulbert, S. J. 1997, in *Hubble Space Telescope Calibration Workshop Proceedings* (Baltimore: STScI)
- Sahu, K., Hodge, P., & Kraemer, S. 1998, STIS ISR 98-05 (Baltimore: STScI)
- Sahu, K., et al. 1996, STIS ISR 95-11 (Baltimore: STScI)
- Leitherer, C., et al. 2000, STIS Instrument Handbook Version 4 (Baltimore: STScI)
- Voit, M., et al. 1997, HST Data Handbook Version 3.1, Volume 2 (Baltimore: STScI)

## 10. Appendix A

IRAF script, called `stis2qp.cl`, to convert STIS Time-Tag data into a QPOE file (see Section 7), uses tasks from the `tables` and `xray` packages. This version of `stis2qp.cl` is optimized for the `tables` package Version 2.2 (released August 2000). For input files with one `EVENTS` extension the output will have an additional column called `RTIME`, which consists of the original time values. The column called `TIME` in the output will be used for the timing of the events, which consists of the reformatted time values. The output will also have the original position axes. To check the version of the `tables` package type `lpar tables` in IRAF.

```
# NAME:          stis2qp
# PURPOSE:      Merge multiple EVENTS extensions and convert STIS Time-Tag FITS
#               files into QPOE format.
# EXAMPLE:      Load the packages tables.ttools and xray.xdataio,
#               copy the script to a file, define the script as a task and run
#               it:  cl> task stis2qp = "stis2qp.cl"
#               cl> stis2qp rootname_tag.fits qpfileout
# DESCRIPTION:  The task determines the number of EVENTS extensions in the
#               input FITS file. Multiple EVENTS extensions are merged into a
#               single EVENTS extension. The output event time values are double
#               precision. Also, the files are edited to include XSORT=TIME in
#               the table header (i.e., the STIS data is time-sorted).
# CREATED:      By Jerry Kriss for FUSE data.
# MODIFIED:     By Ilana Dashevsky and Phil Hodge for STIS Time-Tag data.
# VERSION A:    August 11, 2000
```

## Instrument Science Report STIS 2000-02

```
# REFERENCE: Instrument Science Report STIS 2000-02

procedure stis2qp (fitsin, qpout)

file fitsin  {"", prompt="Name of STIS Time-Tag FITS file."}
file qpout   {"", prompt="Root name of output QPOE file."}
bool fsave   {yes, prompt="Save merged/edited FITS file?"}
bool verbose {yes, prompt="Monitor progress of task?"}

begin
  # Local variables
  string fitsfile
  string qpfile
  string outtable
  string cdfile
  string dummy
  string inlist
  int    nexten

  # Get input file names
  fitsfile = fitsin
  qpfile   = qpout

  # Load necessary packages
  if (! defpac ("ttools")) {
    print("Please load the 'ttools' package.")
    bye
  }
  if (! defpac ("xdataio")) {
    print("Please load the 'xray' and 'xdataio' packages.")
    bye
  }
  # Check whether output QPOE file exist
  if (access(qpfile // ".qp")) {
    print ("The file ",qpfile," already exists.")
    bye
  }
  # Create output files with a unique name
  outtable = mktemp ("stis2qp") // ".fits")

  # Determine the number of EVENTS extensions
  hselect (fitsfile/"[0]", "NEXTEND", yes) | scanf ("%d", nexten)
  nexten = nexten - 1
  if (verbose) print ("Processing ",nexten,"EVENTS extension(s) ...")

  if (nexten > 1) {
```

## Instrument Science Report STIS 2000-02

```
# Create a new table that will use double precision.
# And merge the events into 1 extension.

# Create temporary file for table column parameters
cdfile = mktemp ("tmp$merge_tt") // ".cd"
# Create temporary file for a dummy table
dummy = mktemp ("tmp$merge_tt") // ".tab"
# Copy table column parameters to temporary file
tlcol (fitsfile // "[EVENTS,1]", nlist=4, > cdfile)
# Create an empty table with the column definitions that we want
tcreate (dummy, cdfile, "", uparfile="", nskip=0, nlines=0, \
  nrows=0, hist=no, extrapar=0, tbltype="default", \
  extracol=0)
# Create a comma-separated list of table names
inlist = dummy
for (i = 1; i <= nexten; i = i + 1) {
  inlist = inlist // "," // fitsfile // "[" + i // "]"
}
# Copy primary header to the output FITS file
imcopy (fitsfile // "[0]", outtable, verbose=no)

# Merge all the input EVENTS tables
tmerge (inlist, outtable, "append", allcols=no, \
  tbltype="default", allrows=0, extracol=0)

# Copy the GTI extension
if (verbose) print ("Copy the GTI extension ...")
tcopy (fitsfile // "[GTI]", outtable, verbose=no)

# Delete temporary files
delete (cdfile, verify=no)
delete (dummy, verify=no)
}
else {
  # For a single EVENTS extension file ...
  # Copy Time-Tag file to a temporary file
  if (verbose) print ("Copying data to a temporary file ...")
  copy(fitsfile, outtable, verbose=no)

  # Re-format the TIME column values for double precision
  if (verbose) print ("Re-formatting TIME column values ...")
  tcalc (outtable, outcol="DTIME", equals="TIME", \
    datatype="double", colunits="seconds", colfmt="")

  # Re-name DTIME column to TIME
  tchcol (outtable, "TIME", "RTIME", newfmt="", \
```



## Instrument Science Report STIS 2000-02

```
        newunits="", verbose=no)
    tchcol (outtable, "DTIME", "TIME", newfmt="", \
        newunits="", verbose=no)
}
# Fix STIS FITS file for QPOE compatibility
if (verbose) print ("Fixing STIS header for QPOE compatibility ...")
parkey("TIME", outtable, keyword="XS-SORT", add=yes)

# Do the FITS to QPOE conversion
if (verbose) print ("Converting FITS to QPOE ...")
fits2qp (outtable, qpfile, naxes=1, axlen1=0, axlen2=0, \
    mpe_ascii_fi=no, clobber=no, oldqpoename=no, display=1, \
    fits_cards="xdataio$fits.cards", qpo_e_cards="xdataio$qpo.e.cards", \
    ext_cards="xdataio$ext.cards", wcs_cards="xdataio$wcs.cards", \
    old_events="EVENTS", std_events="STDEVT", rej_events="REJEVT", \
    which_events="old", oldgti_name="GTI", allgti_name="ALLGTI", \
    stdgti_name="STDGTI", which_gti="old", scale=yes, key_x="AXIS1", \
    key_y="AXIS2", qp_internals=yes, qp_pagesize=2048, \
    qp_bucketlen=4096, qp_blockfact=1, qp_mkind=no, qp_key="", \
    qp_debug=0)
# Delete temporary files
if (fsave) {
    print ("The output FITS file is: ",outtable)
}
else {
    delete (outtable, verify=no)
}
end
```

## 11. Appendix B

IRAF script, called `stis2qp.cl`, to convert STIS Time-Tag data into a QPOE file (see Section 7), uses tasks from the `tables` and `xray` packages. This version of `stis2qp.cl` is optimized for the `tables` package version **better** than Version 2.2 or released after August 2000. The output will have a double precision TIME column used for the event timing and the original position axes. To check the version of the `tables` package type `lpar tables` in IRAF.

```
# NAME:      stis2qp
# PURPOSE:   Merge multiple EVENTS extensions and convert STIS Time-Tag FITS
#            files into QPOE format.
# EXAMPLE:   Load the packages tables.ttools and xray.xdataio, copy the
#            script to a file, define the script as a task and run it:
#            cl> task stis2qp = "stis2qp.cl"
#            cl> stis2qp rootname_tag.fits qpfileout
# DESCRIPTION: The task determines the number of EVENTS extensions in the
#            input FITS file. Multiple EVENTS extensions are merged into a
```

## Instrument Science Report STIS 2000-02

```
#           single EVENTS extension. The output event time values are double
#           precision. Also, the files are edited to include XSORT=TIME in
#           the table header (i.e., the STIS data is time-sorted).
# CREATED:   By Jerry Kriss for FUSE data.
# MODIFIED:  By Ilana Dashevsky and Phil Hodge for STIS Time-Tag data.
# VERSION B: August 19, 2000
# REFERENCE: Instrument Science Report STIS 2000-02

procedure stis2qp (fitsin, qpout)

file fitsin  {"", prompt="Name of STIS Time-Tag FITS file."}
file qpout   {"", prompt="Root name of output QPOE file."}
bool fsave   {yes, prompt="Save merged/edited FITS file?"}
bool verbose {yes, prompt="Monitor progress of task?"}

begin
  # Local variables
  string fitsfile
  string qpfile
  string outtable
  string cdfile
  string dummy
  string inlist
  int    nexten

  # Get input file names
  fitsfile = fitsin
  qpfile   = qpout

  # Load necessary packages
  if (! defpac ("ttools")) {
    print("Please load the 'ttools' package.")
    bye
  }
  if (! defpac ("xdataio")) {
    print("Please load the 'xray' and 'xdataio' packages.")
    bye
  }
  # Check whether output QPOE file exist
  if (access(qpfile // ".qp")) {
    print ("The file ",qpfile," already exists.")
    bye
  }
  # Create output files with a unique name
  outtable = mktemp ("stis2qp") // ".fits")
```

## Instrument Science Report STIS 2000-02

```
# Determine the number of EVENTS extensions
hselect (fitsfile/"[0]", "NEXTEND", yes) | scanf ("%d", nexten)
nexten = nexten - 1
if (verbose) print ("Processing ",nexten,"EVENTS extension(s) ...")

# Create a new table that will use double precision.
# For multiple EVENTS extensions, merge the events into one extension.

# Create temporary file for table column parameters
cdfile = mktemp ("tmp$merge_tt") // ".cd"
# Create temporary file for a dummy table
dummy = mktemp ("tmp$merge_tt") // ".tab"
# Copy table column parameters to temporary file
tlcol (fitsfile // "[EVENTS,1]", nlist=4, > cdfile)
# Create an empty table with the column definitions that we want
tcreate (dummy, cdfile, "", uparfile="", nskip=0, nlines=0, \
nrows=0, hist=no, extrapar=0, tbltype="default", \
extracol=0)
# Create a comma-separated list of table names
inlist = dummy
for (i = 1; i <= nexten; i = i + 1) {
inlist = inlist // "," // fitsfile // "[" + i // "]"
}
# Copy primary header to the output FITS file
imcopy (fitsfile // "[0]", outtable, verbose=no)

# Merge all the input EVENTS tables
tmerge (inlist, outtable, "append",allcols=no, tbltype="default", \
allrows=0, extracol=0)

# Copy the GTI extension
if (verbose) print ("Copying the GTI extension ...")
tcopy (fitsfile // "[GTI]", outtable, verbose=no)

# Delete temporary files
delete (cdfile, verify=no)
delete (dummy, verify=no)

# Fix STIS FITS file for QPOE compatibility
if (verbose) print ("Fixing STIS header for QPOE compatibility ...")
parkey("TIME", outtable, keyword="XS-SORT", add=yes)

# Do the FITS to QPOE conversion
if (verbose) print ("Converting FITS to QPOE ...")
fits2qp (outtable, qpfile, naxes=1, axlen1=0, axlen2=0, \
mpe_ascii_fi=no, clobber=no, oldqpoename=no, display=1, \
```

## Instrument Science Report STIS 2000-02

```
fits_cards="xdataio$fits.cards", qpoe_cards="xdataio$qpoe.cards", \  
ext_cards="xdataio$ext.cards", wcs_cards="xdataio$wcs.cards", \  
old_events="EVENTS", std_events="STDEVT", rej_events="REJEVT", \  
which_events="old", oldgti_name="GTI", allgti_name="ALLGTI", \  
stdgti_name="STDGTI", which_gti="old", scale=yes, key_x="AXIS1",\  
key_y="AXIS2",qp_internals=yes, qp_pagesize=2048, \  
qp_bucketlen=4096,qp_blockfact=1, qp_mkindex=no, qp_key="", \  
qp_debug=0)  
# Delete temporary files  
if (fsave) {  
    print ("The output FITS file is: ",outtable)  
}  
else {  
    delete (outtable, verify=no)  
}  
end
```