Data Structures in STIS and NICMOS

A. Farris
Space Telescope Science Institute, Baltimore, MD 21218
Email: farris@stsci.edu

March 22, 1996

Abstract

The new instruments to be installed during the 1997 servicing mission for the Hubble Space Telescope present challenges of managing complex associations among a variety of data. The data structures and the design of the I/O software interface to support these instruments are presented. The significance of this software architecture to the design of calibration algorithms is that it allows the implementation of algorithms, based on accessing data in a logical form, abstracting from the details of disk formats. The architecture also illustrates the principle of encapsulation, concealing the details of implementation from the developers of algorithms. The consequences of this approach are discussed.

Note: This paper has been submitted for publication as part of the proceedings of the Fifth Annual Conference on Astronomical Data Analysis Software and Systems (ADASS V) held in Tucson, Arizona, on October 22-25, 1995.

1 Fundamental Design and Implementation Considerations

STIS and NICMOS (New, 1995), instruments to be installed aboard the Hubble Space Telescope during the 1997 servicing mission, are extremely flexible and can be used in many different modes. Consequently, the data produced by these instruments contains complex internal relationships as well as external associations with other data groups. This paper will discuss the basic structure of the data from STIS and NICMOS and the structure of the software that provides access to the data. This software is being used in the STScI calibration pipelines for these instruments.

The calibration software is being written in strict conformity to ANSI C, (which includes the standard ANSI C library), with no special vendor extensions allowed. The pipeline software is modular in design with minimal and managed dependencies on the operating system in which the software executes. The pipeline utilizes the IRAF software environment (Tody, 1986). However, any IRAF routines employed use the OpenIRAF software environment, which allows IRAF library routines to be called from an ANSI C program. The design perspective toward IRAF is the same as that toward an operating system, viz., managed and minimal dependencies. The use of IRAF routines is confined primarily to basic I/O routines or special STSDAS (Hanisch, 1989) packages, such as synphot.

ANSI C data structures represent both uncalibrated and calibrated data from each instrument, forming a data model of each instrument. The pipeline software consists of calibration steps that operate on these data structures. A major goal is to make the calibration algorithms themselves only dependent on these data structures. All I/O is encapsulated in basic routines that form a mapping between these data structures and external storage media.

Exceptions to this goal are calibration steps for which I/O becomes a significant part of the algorithm itself, e. g. when there is insufficient memory to contain all the needed data. In this case, I/O needs will be satisfied by calls to the interface module, leaving the algorithms themselves free of any direct, low-level I/O. This issue will likely be significant to STIS.

2 General Structure of Science Data

While a detailed discussion of STIS and NICMOS instrument data is beyond the scope of this paper, we can present the general structure of the associated science data. The science data files utilize FITS image extensions (Ponz, 1994). Data from both instruments may be represented as sequences of FITS Header-Data-Units (HDUs) containing two-dimensional arrays. These HDUs are grouped as an array of science data, together with a data quality array and an error array, representing, respectively, an array of boolean conditions that identify various possible anomalous conditions that may be associated with each pixel and the error associated with each pixel. In addition, NICMOS has two additional arrays, an array identifying the number of samples associated with each pixel and an array identifying the integration time associated with each pixel. The structure of the FITS file for both instruments is shown below.

STIS
Primary Header Data Unit
    general keywords
    no data
first Science HDU
first Error HDU
first Data Quality HDU
second Science HDU
second Error HDU
second Data Quality HDU
etc.

NICMOS
Primary Header Data Unit
    general keywords
    no data
first Science HDU
first Error HDU
first Data Quality HDU
first Samples HDU
first Integration Times HDU
second Science HDU
second Error HDU
second Data Quality HDU
second Samples HDU
second Integration Times HDU
etc.

3 ANSI C Structures Representing the Fundamental Data Items

A number of basic ANSI C data structures have been defined to correspond to the FITS HDUs mentioned above. The most fundamental structure consists of a two-dimensional array of an arbitrary size and various types (short, float, etc.), together with an array of header lines to represent collections of keywords associated with the data. This basic structure is then specialized to represent specific types of header plus two-dimensional array, viz., Science, Error, Data Quality, Samples, and Integration Times HDUs.

In addition, the basic structures above are grouped into single and multiple groups of HDUs. A single STIS group consists of a global header together with a Science, Error, and Data Quality HDU. A single NICMOS group consists of a global header together with a Science, Error, Data Quality, Samples and Integration Times HDU. A multiple STIS group consists of a global header together with N groups of Science, Error, and Data Quality HDUs. Likewise, a multiple NICMOS group consists of a global header together with N groups of Science, Error, Data Quality, Samples and Integration Times HDUs.

4 I/O Functions Related to ANSI C Data Structures

Supporting these data structures is an I/O interface module called HSTIO. High-level functions in the HSTIO interface include: 1) functions to initialize all data structures, to allocate and free memory needed by the data structures, 2) macros to access elements of the two-dimensional data arrays, 3) functions to get and set values associated with header keywords, 4) functions to read and write entire Science, Error, and Data Quality HDUs, as well as NICMOS Samples and Integration Times HDUs, 5) functions to read and write single and multiple groups of {Science, Error, Data Quality} or, for NICMOS, groups of {Science, Error, Data Quality, Samples, Integration Times}. These functions recognize local conventions used in conjunction with the various HDUs, and generate arrays of the proper dimensionality and value.

The HSTIO interface also includes a number of lower-level functions that can be used if there is insufficient memory to store entire HDUs. These include: 1) opening and closing a particular image in a FITS file, 2) reading and writing the FITS header associated with an image, 3) reading and writing the data array associated with an image, and 4) reading and writing arbitrary lines or arbitrary two-dimensional sections of a data array associated with an image.

The structure of the software is shown in Figure 1. All of the high-level functions are implemented in terms of the lower-level functions. These lower-level functions are implemented using IRAF's image I/O, enhanced by the addition of the FITS kernel with support for image extensions.

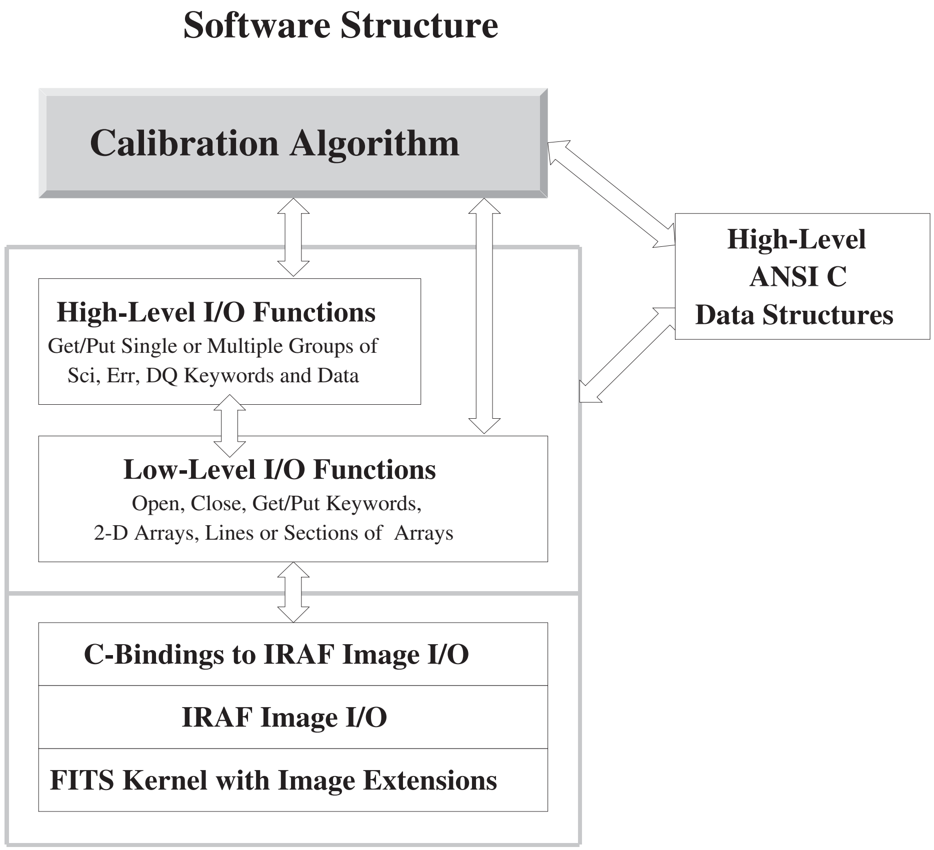

Figure 1: HSTIO Interface

5 Consequences of this Design

The design of the HSTIO interface is based on two principles: adequate modeling and encapsulation. The high-level data structures form a complete representation of basic instrument data and all associated keywords, as well as meaningful groups of these data and keywords, thus forming an adequate model of data from the instruments. The I/O functions include all functions needed to support these high-level data structures, including translating between the external disk representation and the internal data structures. Implementation details are carefully concealed within these I/O functions, thus employing the principle of encapsulation.

There are two important consequences of this design: flexibility in usage and flexibility in implementation. First, by encapsulating all I/O operations within the HSTIO interface, the pipeline calibrations can be written in the form of pure algorithms, with no operating system dependencies. Furthermore, since these algorithms are being implemented in a widely available standard language (ANSI C), they are easily used in other environments or even translated into other languages. Second, encapsulation also allows alternative implementations of the I/O functions without impacting the calibration algorithms. The low-level I/O functions are currently implemented using IRAF, but they do not have to be. A preliminary version already exists implementing the low-level functions using FITS C++ classes with no use of IRAF whatever. This is completely transparent to the calibration algorithms using these functions and data structures.

References

[1] Hanisch, R., 1989, "STSDAS: The Space Telescope Science Data Analysis System", in ***Data Analysis in Astronomy III***, V. Di Gesu, et. al., Plenum Press, 129

[2] ***New Instruments for Second Servicing Mission***, 1995, Servicing Mission Office, Space Telecsope Science Institute, 3700 San Martin Drive, Baltimore, Maryland 21218

[3] Ponz, J., Thompson, R., and Muñoz, J., 1994, "The FITS image extension", in ***Astron. and Astrophys. Suppl. Ser., 105***, 53

[4] Tody, D., 1986, "The IRAF Data Reduction and Analysis System", in ***Instrumentation in Astronomy VI***, SPIE 627, Part 2