Final Technical Report

Telescience Testbed Program
A Study of Software for SIRTF Instrument Control

Grant NAG 2-661

30 June 1992

Erick T. Young, Principal Investigator
Steward Observatory
University of Arizona

A Study of Software for SIRTF Instrument Control

1. Summary

As a continued element in the Telescience Testbed Program (TTP), the University of Arizona Steward Observatory and the Electrical and Computer Engineering Department (ECE) jointly developed a testbed to evaluate the Operations and Science Instrument System (OASIS) software package for remote control of an instrument for the Space Infrared Telescope Facility (SIRTF). SIRTF is a cryogenically-cooled telescope with three focal plane instruments that will be the infrared element of NASA's Great Observatory series. The anticipated launch date for SIRTF is currently 2001. Because of the complexity of the SIRTF mission, it was not expected that the OASIS package would be suitable for instrument control in the flight situation, however, we considered its possible use as a common interface during the early development and ground test phases of the project. The OASIS package, developed at the University of Colorado for control of the Solar Mesosphere Explorer (SME) satellite, serves as an interface between the operator and the remote instrument which is connected via a network. OASIS provides a rudimentary windowing system as well as support for standard spacecraft communications protocols.

The SIRTF instruments share many common operational, human interfacing and data processing requirements. All three SIRTF instruments use infrared detector arrays that generate two-dimensional image data, all three have filter wheel mechanisms and internal calibration sources, and all three instruments will have similar housekeeping telemetry (temperatures, voltages, currents, etc.) Our goal for this experiment was to evaluate several software alternatives in order to establish a common human/computer interface for developing instrument control and analysis software for all SIRTF instruments. By establishing a common framework early in the definition phase of SIRTF, the three instrument teams could realize a considerable savings in effort and resources. An additional aspiration was to identify a systems architecture that would separate the programming for the instrument hardware from the programming for the human interface. The key to this second goal is the establishment of a well defined communication convention between the various software components.

This study was implemented in two stages. The first consisted of an experiment evaluating the suitability of the OASIS package for instrument control of an infrared detector array from the Multiband Imaging Photometer for SIRTF (MIPS). The testbed for this activity simulated the operation of one of the MIPS instrument detectors by utilizing a program written in the C language. An IBM PC was the instrument-controlling computer. Instructions to the IBM PC were communicated remotely using an Ethernet connection from a DEC MicroVAX II workstation running OASIS. During this first stage, some of the limitations of the OASIS package became clear. In particular, the inability to support display image from the remote task was identified as a major

weakness. Consequently, the second stage involved the investigation of the availability of other software packages that might be used for instrument control.

The experiment performed all of the functions required of the MIPS simulation program. Remote control of the instrument was demonstrated but found to be inappropriate for SIRTF at this time for the following reasons: (1) programming interface is too difficult; (2) significant computer resources were required to run OASIS; (3) the communications interface too complicated; (4) response time was slow; (5) quick-look of image data was not possible.

2. Introduction

The Space Infrared Telescope Facility (SIRTF) is planned to be launched in the early 2000's. SIRTF is a 0.9-meter cooled telescope with a lifetime of >3 years. It will serve as a national facility for infrared investigations in all areas of astronomy and astrophysics. Three instruments have been selected for SIRTF: the Infrared Array Camera (IRAC); the Multiband Imaging Photometer for SIRTF (MIPS); and the Infrared Spectrometer (IRS). These instruments are being developed by separate teams centered at the Smithsonian Astrophysical Observatory, the University of Arizona, and Cornell University, respectively. During 1992, significant changes have been made in the SIRTF concept, with the goals being to greatly simplify the mission and to reduce costs. Since the redefinition work is still ongoing, this study was done using the earlier mission concepts. Those characteristics are summarized in Table 1

The Infrared Array Camera will provide wide field and diffraction-limited imaging for wavelengths between 1.8 and 30 μm. IRAC will utilize infrared arrays with formats as large as 256x256 pixels. The Multiband Imaging Photometer for SIRTF will extend the imaging capability to wavelengths as long as 120 μm with arrays as large as 32x32 elements. Additionally, MIPS will have small detector arrays that will provide a photometric capability out to 1200 μm wavelength. The Infrared Spectrometer will provide low and medium resolution ($\lambda/\Delta\lambda \sim 100 - 2000$) spectroscopy between 3 and 200 μm. Like the other SIRTF instruments, the IRS will use large arrays of infrared detectors. .

Although the three instruments are diverse, they share many common aspects in term of modes of operation, interfacing with the facility, and data processing. In particular, all three instruments utilize infrared array detectors of various types, have filter and optical mechanisms, and have similar housekeeping requirements. Moreover, the conceptual processing steps to go from raw instrument data to useful quick-look information are quite similar. Despite these many common aspects, the teams have independently developed software for the control of their test detector systems. It has become clear, that the teams could realize a considerable savings in effort and resources if they used a common framework for developing the instrument control software.

Table 1. SIRTF Instrument Characteristics

Wavelength (μm)	Array Format	Function
	IRAC	
1.8-5.3	256x256	High Resolution Imaging
5.3-27	128x128	High Resolution Imaging
	MIPS	
30-55	16x32	Photometry
50-120	32x32	Imaging, Polarimetry
120-200	2x8	Photometry
200-500	2x2	Photometry, Polarimetry
500-1200	1	Photometry, Polarimetry
	IRS	Spectroscopy Resolution:
2.5-4	256x256	R=75-150
4-36	128x128	R=75-200, R=1500-2500
36-50	2x8	R=75-200, R=1500-2500
50-115	Stacked 1x32	R=75-200, R=1500-2500
115-200	2x16	R=750-1250

Figure 1 shows a possible simplified model for the structure of the SIRTF data system. It is important to remember that the SIRTF data system is currently under conceptual design, and the model presented is only one of a number of possible configurations. The actual system configuration will not be finalized until detailed system level trade-off studies have been done. The model in Figure 1 represents a generic type where a significant amount of intelligence is present in the instrument computer. The version shown in Figure 1 has the advantage of having separable components that allows a modular development effort. In particular, the interface between the instrument and spacecraft computers is well defined. This investigation assumed this model.

In this proposed data system model, each of the instruments has a computer that is responsible for both hardware control of the instrument functions and communication with the spacecraft computer. The spacecraft computer handles the operation of the overall spacecraft systems such as attitude control, power distribution, etc., and it also serves as the communications interface between the instruments and the telemetry system. All signals between the instruments and the spacecraft are communicated via a well defined packet protocol. Hence, under this model, the instruments would require at least enough computing power and memory to handle functions such as data buffering, data compression, and data packetization.

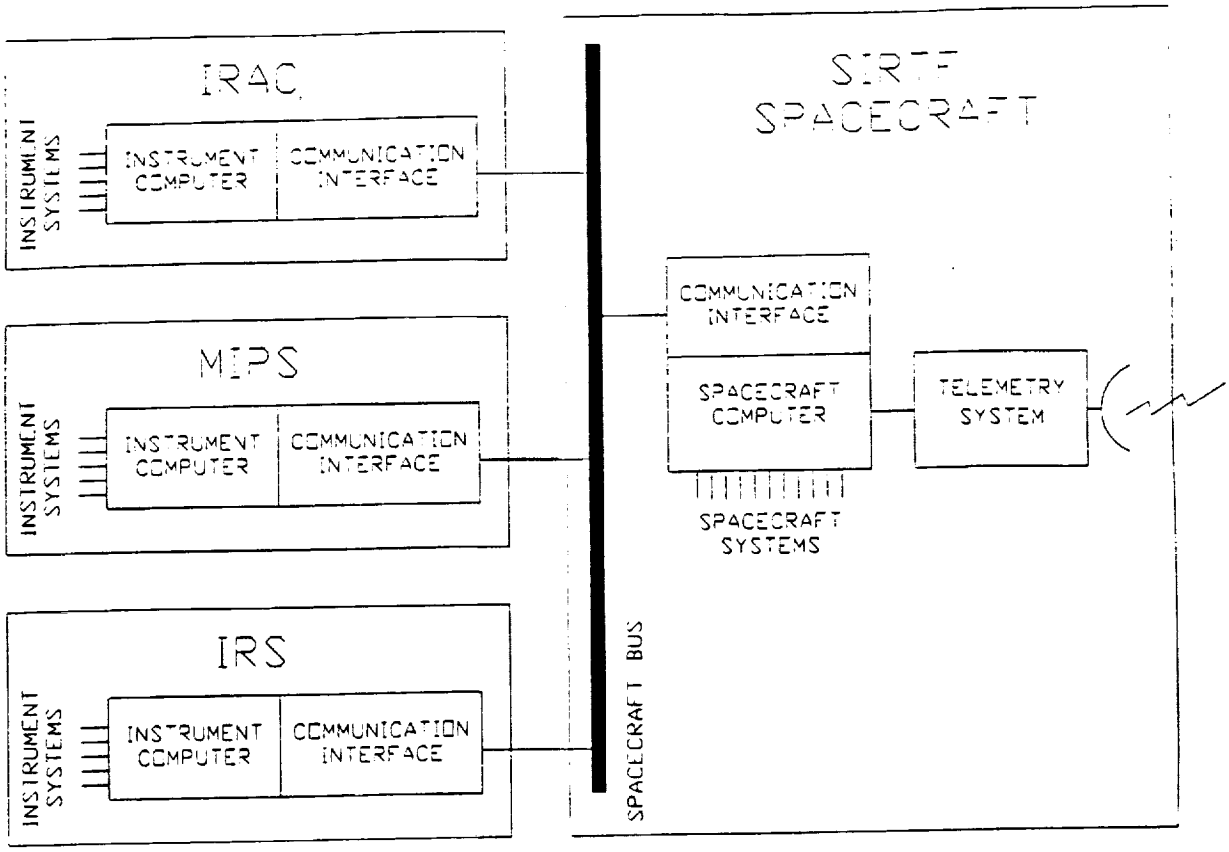


Figure 1. Strawman SIRTF Data System Model

2.1 Goals and Objectives

As envisioned in the mission operations plan, control of the satellite and the instruments will be the responsibility of an Operations Center. A key step in the development of the mission will be the transfer of knowledge from the individual SIRTF teams to the Operations Center staff. Through the use of a common software framework, the process of translating instrument requirements into Operations Center software will be greatly simplified.

Enhanced reliability of the software is an additional potential benefit. The same software could follow the instruments from the SIRTF development computers, to the ground support computers. The enhanced reliability comes from using algorithms and code that have been generated on the same software platform and tested with all of the SIRTF instruments. The degree of duplication of effort is minimized.

The primary goal of the investigation was to assess the suitability of the OASIS software package for control of SIRTF instruments during the development and ground test phases. It was not expected that a generic package would have the performance necessary for the actual SIRTF mission. Moreover, the anticipated operating scenario for the mission involves pre-planned "canned" operation sequences and observations with little opportunity for real time control.

With the data system architecture shown in Figure 1, the spacecraft computer could be replaced with a ground test computer prior to integration. Since the MicroVAX to the spacecraft or test computer is via a well defined protocol, changes in the instrument hardware do not impact the interface. A goal of this investigation was to demonstrate this architecture.

The OASIS control program is designed to be a general purpose instrument-operator interface. To simplify the development of an OASIS application, the operating functions (actions, screen displays, communications, etc.) are defined via applications databases that are interpreted by OASIS. This flexibility is also potentially a liability since an interpreter is significantly slower than a compiled custom application. One of the objectives of this investigation was to evaluate both the ease of database coding and the speed the program was able to carry out representative tasks.

Another objective of this investigation involved the identification of areas in OASIS MicroVAX that were inadequate for the SIRTF application. Prior to our work, it was recognized that the lack of image display capabilities in OASIS would be a serious drawback in working with the image-oriented SIRTF data. Because of this deficiency, we also did a preliminary investigation of two other software packages that include image display and image processing capabilities. The two packages we considered were PV-Wave (developed by Precision Visuals) and extensions to the Image Reduction and Analysis Facility (IRAF) developed by the National Optical Astronomy Observatory (NOAO).

2.2 Scope

The scope of activity for this study focused on a detailed experiment involving OASIS running on a MicroVax workstation communicating with an IBM Personal Computer (PC) instrument computer via a DECnet link. At the beginning of the investigation, the Unix version of the OASIS package was not yet available, although that was the desired configuration for study. The evaluation was done on the Vax VMS version of OASIS with the expectation of migration to the Sun Unix version when it

became available. The VMS environment limited the choices for communications protocol between the workstation and the IBM PC. We utilized the DECnet protocol for this link since existing communications drivers were available for both ends. In a separate effort, TCP/IP enhancements to OASIS have been developed (Wibowo 1990). The Unix version of the experiment was subsequently compared with the VMS version.

At this point in the development of the SIRTF mission, complete simulators of all the functions for either the MIPS or the SIRTF facility are some years away. As an example, Figure 2 shows a functional block diagram for the MIPS instrument. The instrument is envisioned to have five focal plane arrays, six rotating mechanisms, a number of reference sources, and numerous thermometers. A full scale simulation of MIPS was well beyond the scope of this investigation. Consequently, we limited the investigation to a subset of the possible operations and data that will result from the mission. Specifically, the MIPS instrument was simulated by a program that generated sampled data from a single infrared detector array and also provided the expected delays for operations such as power up sequences, filter changes, resets, etc.

Since it was recognized that image display capabilities were important in the SIRTF context, we examined other programs that had a strong emphasis on image processing and display. We limited out work to IRAF and PV/Wave since they are well supported on Sun workstations. IRAF, in particular, has become a very widely used program in the astronomical community.

2.3 Rationale

The rationale for this investigation was the potential savings in effort if a common instrument interface could be identified for the three SIRTF instruments during the development and ground test phases. Since the three instruments have many functional similarities, we considered the use of a general purpose program that could be adapted to the specific requirements of a given instrument. Moreover, the normally tedious effort associated with coding the user interface should be minimized with an effective general purpose program.

A second rationale for this line of investigation was the increase in flexibility of the overall system if a common user interface and common communications protocol were used during the development stages of the mission. In particular, the simplification of the instrument interface to a well-defined communications standard (both hardware and software) allows changes to be made in the instrument hardware or in the user control workstation with minimal impact on that interface.

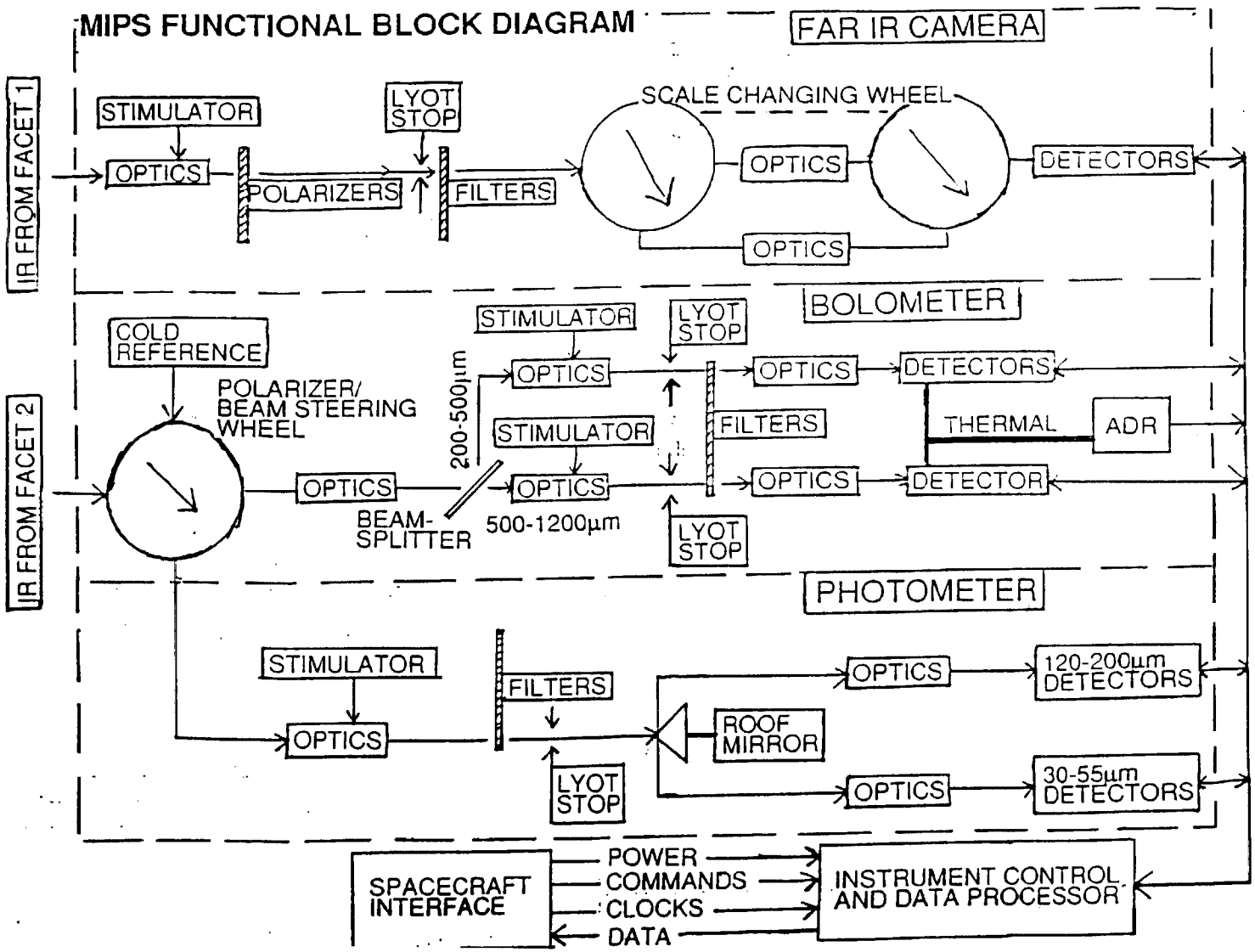


Figure 2. MIPS Functional Block Diagram

3.1 Architecture

The architecture of the SIRTF-OASIS experiment is illustrated in Figure 3. The test configuration consists of a MIPS detector test simulator written in the C language. The simulator represents the microprocessor based Instrument Controller (IC) which will control the MIPS instruments on board SIRTF. The IBM PC compatible computer used in the experiment served two functions, (1) to run the test simulator and (2) to represent the spacecraft's flight command and data subsystem and the telecommunications subsystem (Spacecraft). The Spacecraft is the interface between the Ground Support Equipment (GSE) and the SIRTF instruments. The ethernet link represents the communications link to earth. The DEC workstation running OASIS represents the GSE.

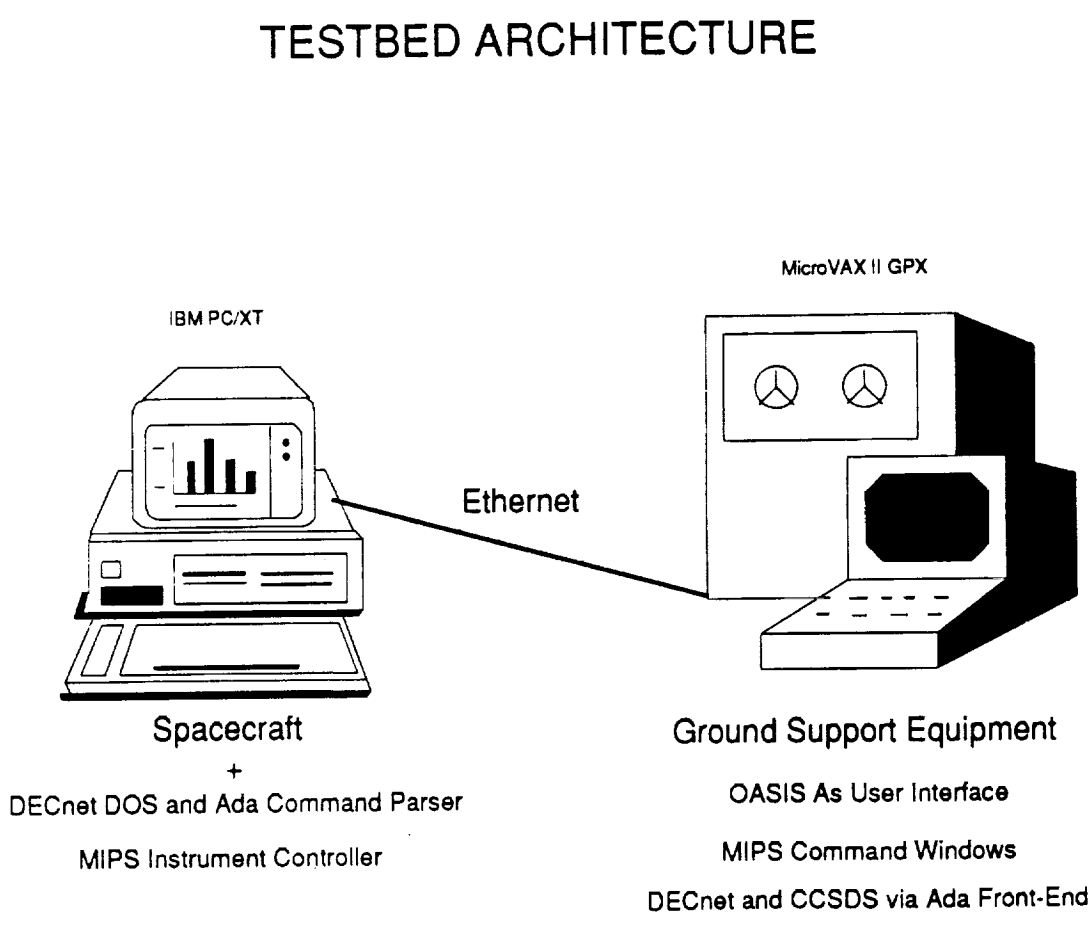
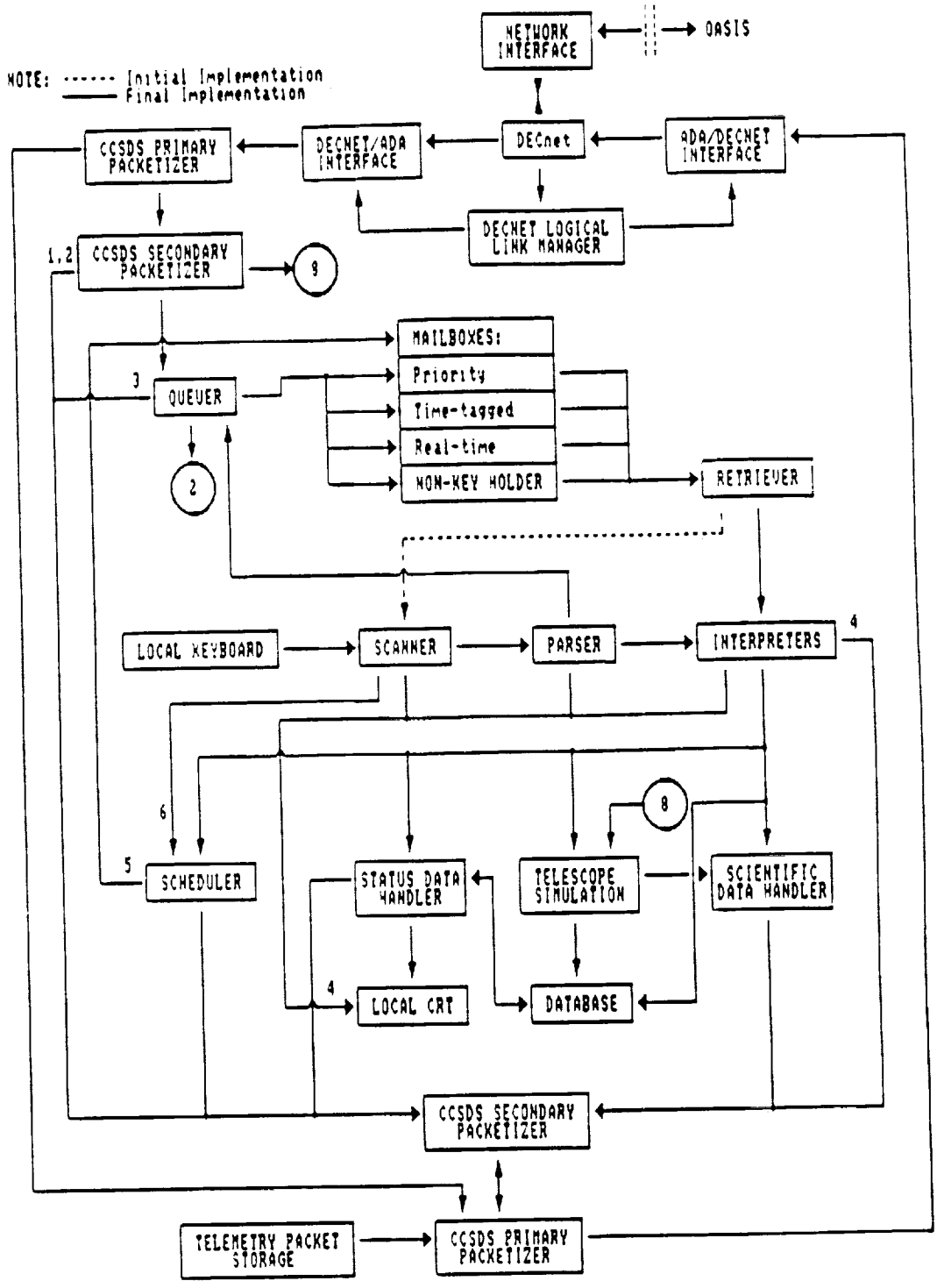


Figure 3. Testbed Experiment Architecture

3.2 Hardware, Software and Networks

An IBM PC/XT compatible computer was used to represent both the MIPS IC and Spacecraft. It was connected to the University of Arizona Ethernet network using a 3Com 3c503 Etherlink II Ethernet card. A DEC MicroVAX GPX Workstation running the VMS operating system served as the GSE. The intent of the initial experiment was to use industry standards like UNIX and the TCP/IP communication protocol. At the time of the experiment, OASIS was not available for a UNIX platform, therefore the VMS version of OASIS was used on the DEC MicroVax machine. Since the Electrical and Computer Engineering Telescience Laboratory (ECE TSL) had previous experience with two testbed demonstrations using OASIS (Schooley and Cellier, 1988) it was decided to modify existing software designed at the ECE TSL (Bienz and Hunter, 1988).

The overall block diagram for the software on the instrument control computer (the IBM PC) is shown in Figure 4 (taken from Wibowo 1990) and is based on software written by Pan and Lew (1988) for the remote fluid handling telescience project. The Command Processing software on both machines was written in Ada. The Ada compiler used for the MicroVAX was DEC Ada, and for the PC it was Meridian Ada. The communication protocol between the PC and the MicroVAX was DECnet for the lower layers. Consultative Committee on Space Data Systems (CCSDS) recommendations for telecommands and packet telemetry, were implemented for the upper layer. VAX DECnet was used on the MicroVAX, while DECnet-DOS was used on the PC. CCSDS protocol recommendations were implemented through software interfaces written in Ada on both machines. This was necessary to allow DECnet and CCSDS protocols to pass CCSDS packets between them (Bienz and Hunter, 1988). More information about the communication software design, see (Bienz and Hunter, 1988), for a more detail description of the Command Processing software used in the SIRTF experiment see, (Wibowo, 1990). OASIS was used to develop the user interface on the MicroVAX workstation.



NOTE: 1. Tele-command packet acknowledge to OASIS, wrong OASIS version number,
2. Telemetry and scientific data acknowledge from OASIS
3. MAILBOX n full signal, n = 1,2,...,
4. Command error messages
5. Key-holder MAILBOX reset
6. Key request
7. Packet sequence count error
8. Direct interrupt

Figure 4. Instrument Control Computer Communications Software

At the user interface workstation, a number of displays were developed to provide the operator with information on the status of the detector array operation. The information displayed was divided into four categories: static parameters, active parameters, display parameters, and action parameters. The static parameters were those quantities that were only infrequently changed and were displayed primarily as indicators of system health. The static parameters included various voltages, sample rates, etc. The active parameters are quantities that could be expected to frequently change during the operation of the instrument. Active parameters included filter wheel position, observation time, and system mode. The only display parameter was used to set which channel was displayed in the quick-look output. Finally, the action parameters were set the system state for commands. Tables 2-5 list the commands associated with the various parameters used in this investigation.

Table 2. Active Parameters Update Commands

Command	Type	Parameter Format	Allowed Range
SET FILTER	integer	1d	1-7
SET SYSMODE	character	1s	C: Calibration N: Normal L: Loop
SET LOOPMODE	character	1s	G: Global S: Scan Q: Quit
SET ITIME	float	5.1f	0.0-1000.0
SET RA	integer	2d:2d:2d	hour: 0-24 min: 0-60 sec: 0-60
SET DEC	integer	2d:2d:2d	deg: -90 - 90 min: 0-60 sec: 0-60

Table 3. Quick-Look Parameter Update Command

Command	Type	Parameter Format	Allowed Range
SET CHANNEL	integer	2d	1-32

Table 4. Static Parameter Update Commands

Command	Type	Parameter Format	Allowed Range
SET FILNAME	Integer	2d	
SET VOLTAGE_1	Float	7.3f	0.0-5.0
SET VOLTAGE_1	Float	7.3f	0.0-5.0
SET VOLTAGE_1	Float	7.3f	0.0-5.0
SET SRATE	Float	5.1f	0.1-30.0
SET BIAS_VOLT	Float	7.3f	0.0-0.5
SET DELAY_TIME	Float	5.1f	0.0-30.0
SET PWIDTH	Integer	5d	1-1000
SET HILEV	Float	3.1f	0.0-5.0
SET LOLEV	Float	3.1f	0.0-5.0

Table 5. Action Parameter Update Commands

Command	Type	Parameter Format	Allowed Range
SET INISYS	integer	1d	0-1
SET QLOOK	integer	1d	0-1
SET START	integer	1d	0-1
SET ABORT	integer	1d	0-1
SET SHUTDOWN	integer	1d	0-1

3.2 Issues Investigated

The main and foremost issue addressed was the ease of implementing and using OASIS as the software development platform for the SIRTF instruments teams. How much computer resources are required to implement the OASIS package and it's associative software? How easy is it to program the OASIS database? How easy is it to make quick modifications? Minimal computer resources, ease of use and a flexible programming environment are key issues.

In terms of operational issues, basic control of the instruments such as monitoring temperatures, turning filter wheels, etc., it was already clear that OASIS could do these functions. It was not clear, however, how optimally OASIS could perform these functions in a test environment typical of those performed by the various SIRTF instrument teams. A second operation issue relates to the speed of the user interface.

Response time to operators commands must be kept at a minimum to prevent operators from submitting repeated requests.

Two technical issues involve both engineering and scientific analysis. An engineering issue addresses the need for a graphic quick-look evaluation of raw data in near realtime. The ability to conduct scientific graphic quick-look analysis (not realtime) also needs to be addressed. Can we merge the capabilities of OASIS with analysis programs such as the Image Reduction and Analysis Facility (IRAF) that would provide astronomical quick-look capability?

3.4 Method of Investigation

The original experiment was to take place at University of Arizona, Steward Observatory using the UNIX version of OASIS to do real testing of the MIPS detector arrays. It was planned that an Electrical and Computer Engineering (ECE) graduate student with OASIS programming experience would convert our detector test code into OASIS database format. Because of the delay of UNIX OASIS, arrangements were made with Co-Investigator, Dr. Larry Schooley, to conduct the actual experiment at the ECE Telescience Laboratory (ECE TSL). The ECE TSL already had VMS OASIS installed on a DEC MicroVAX GPX workstation. The DECnet implementation of the communications software was previously developed the TSL for the remote fluid handling experiment for the Telescience Testbed Pilot Program (Bienz and Hunter 1988). A c-program was written to simulate a single array of the MIPS instrument and supplied to ECE. Identification of required instrument functions and GSE computer displays were developed by the MIPS systems programmer Irene Barg. These functional requirements were then converted into OASIS database by ECE graduate student Yadung Pang. An additional programmer was required to write the communication interface between the simulation program and IBM PC DECnet DOS networking package. This communications software was written by ECE graduate student Henky Wibowo.

4. Experiment Results

It took two programmers working half time approximately one month to code the OASIS database and the communications software. Since the OASIS program was already in place at ECE, the actual coding of the operational functions in OASIS database form took approximately one week. The rest of the time was involved writing the communication software. The modifications to the communication software were on the PC. The parser on the PC was modified to accept the commands characteristic of the MIPS array. The other major coding activity involved the development of the MIPS detector array simulator program.

The computer display is shown schematically in Figure 5, and consists of a set of windows identifying the MIPS Instrument Controller (IC) functions which include:

- the static parameters icon
- active parameters
- a quick look window
- current coordinates (RA and DEC)
- a group of 'action' buttons

The static icon pops up and displays the parameters that normally remain constant for the specific instrument or are changed only infrequently. These values can be changed at the beginning of an observation. They include, voltages, sampling rate, detector bias, pulse widths and pulse amplitudes.

The active parameters are those quantities that are frequently changed. The active parameters are always displayed and updated by telemetry from the Spacecraft every 10 seconds. Parameters include current temperatures, filter wheel position, detector bias, integration time, time left in integration, and RA and DEC.

The quick look graphic window plots a time series line graph of the data collected (in near realtime). The MIPS test simulator program simulated the data collected from a 32x32 detector array. Since the OASIS package did not have any support for image display, quicklook data were presented as a time series. The observer can plot one channel or a range of channels. These plots provide the ground support engineers with valuable information concerning the status of the instrument. For example, if the dewar housing the detector were warming up, the values could start to drift. This drift would appear on the time series plot, even thought the temperature values displayed appeared to be within the acceptable range.

The action buttons icons pop up sub-windows and are the primary user interface items. All action buttons initiate events such as system initialization, changing the mode of the detector, adjust active or static parameters, initiate quick look, begin an observation, abort an observation and finally shut down the connection.

The experiment performed all of the operational functions required of the MIPS simulation program. The ability to abort a task remotely was not demonstrated. The PI felt that response times to commands was sluggish. Controlled speed measurements are documented in greater detail in the Master's thesis of Wibowo (1990). These results are summarized below. Three functions were measured: window display speed, telecommand packet transmission speed, and telemetry packet decomposition speed. Ten test runs were conducted for each measurement.

OASIS USER INTERFACE

MIPS Testbed Displays

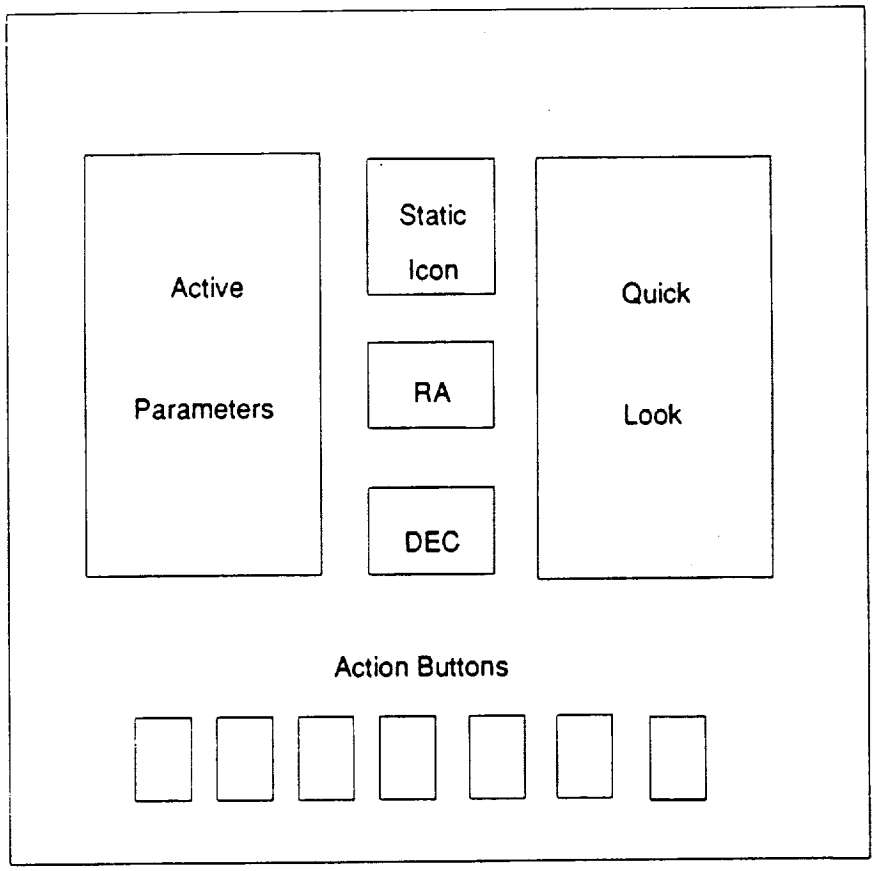


Figure 5. Workstation OASIS Display

5. FUNCTIONAL AND PARAMETRIC EVALUATION

The ease of coding of the OASIS database was judged to be good. Approximately two weeks were spend developing the database and refining the displays. Since OASIS assembles the windows elements in an interpreted manner, changes were easy to make. In that sense, OASIS is useful as an interface development tool. The flexibility exacts a penalty in performance, however, as indicated by the times needed to respond to action requests.

The window display speed measured the experiment's response to an ACTION button being pressed. From the time the button was pressed, it took an average of 115.4 seconds to bring up the required display.

The telecommand packet transmission time is a measure of the OASIS command translator and the communications link. The command SET INISYS initializes the SIRTF parameters and was used to evaluate this time. The average telecommand packet transmission time was 17.4 seconds.

The average telemetry packet decomposition time measures the time between receipt of the first packet by the receiver process and the acceptance of the final packet. For this evaluation a total of six packets were received. The measured decomposition time averaged 16.4 seconds. Added together, the packet transmission time and the packet decomposition time represent the total time spent sending a command from the remote commanding computer to interpreting it on the local controlling computer. Once the connections were made and commands interpreted, the actual data telemetry took place at the maximum speed of the ethernet connection. All these times proved to be unacceptably long for a laboratory or GSE environment.

Wibowo (1990) also reports results for the same software ported to the Sun/UNIX version of OASIS using a TCP/IP protocol. The response of the system was found to be significantly better than the MicroVAX implementation. The measured times were 39.6, 1.5, and 9.6 seconds, respectively. Although the total command response time has now been cut by a factor of 3, some additional improvement is highly desirable.

The issue of quick-look capabilities was also addressed in this investigation. Discussions with a number of astronomers underlined the need for image display to fully understand the performance of the scientific instruments. In particular, infrared astronomy has recently undergone a technological revolution with the advent of large format infrared arrays. The proper operation of these arrays requires the ability to display the data from all the pixels in a comprehensible manner. Since OASIS lacked this capability, we investigated two other approaches to the image display problem. These approaches are discussed in Section 7.

6. LESSONS LEARNED

6.1 Technical Requirements

The main technical lesson learned was the difficulty in tailoring a "general purpose" program like OASIS to a task if the performance requirements are challenging. Specifically, two areas were judged especially weak in OASIS. First, as an interpreted, database-driven program, the response time was considered far too slow for useful interaction with an infrared instrument. Since the primary use would have been during the laboratory and ground test environments, quick response is especially important. The user population has become used to good response time in windowed systems (as most a very few seconds to open a complicated window), and the very long response times associated with OASIS are clearly unacceptable. The Sun Unix implementation of OASIS is significantly faster than the MicroVAX VMS version, but the times are still a factor of ten too long.

The advantage of OASIS as an interpreted language is in the ease of developing applications. The actual coding of the OASIS database was quick, and changes were easily incorporated. The development of an OASIS compiler could do much to improve the performance of the program.

The second area where the SIRTF needs were not well served by OASIS was in the area of data display. Most astronomical data are now in the form of images, and some form of rudimentary image display would be particularly useful.

6.2 Programmatic Requirements

Significant delays were encountered in the startup of this Telescience Testbed Program activity. Most of the delays were directly attributable to the changing of the lead center for the SIRTF project from NASA Ames Research Center to the Jet Propulsion Laboratory. During this changeover, there was confusion over which center should be responsible for the monitoring of this grant. This activity was ultimately funded through Ames, but schedule for the SIRTF telescience investigation was not well synchronized with the rest of the telescience program.

Since the funding came from the individual science disciplines (specifically astrophysics in this case), coordination of various elements of the program was difficult. In future testbedding activities, funding through a single program office would facilitate one of the goals of this type of program -- interdisciplinary interaction.

6.3 How Did Telescience Help?

The main area that telescience helped was in the identification of a systems architecture that best supported remote operations. By separating the functional aspects of the data system and reducing the instrument-to-spacecraft control link to a well-

defined communications interface, many of the potential confusions resulting from misunderstood interface requirements are eliminated.

The telescience investigation also clarified the actual requirements for a user interface and the acceptable level of performance. In particular, users are especially sensitive to the responsiveness of a computer system. In this investigation, we found that delays of greater than a few seconds for the generation of windows or for the acknowledgement of requests were generally unacceptable. Additionally, image display is essential in most astronomy instrument interfaces. Since the scientific data are, for the most part, in image form, a display capability is needed if any control program is to be generally useful.

7. ISSUES IDENTIFIED/FURTHER STUDIES REQUIRED

During the course of this testbed we identified two other packages capable of providing a software development platform for SIRTF. The two packages reviewed were Steward Observatory's/IRAF data acquisition package called CCDACQ, and PV~WAVE, Precision Visuals' workstation analysis and visualization package. An overview of each package is presented below. To present a more accurate comparison between OASIS, IRAF/CCDACQ and PV~WAVE, the same MIPS experiment should be conducted with these two additional packages.

Steward Observatory's CCDACQ is a astronomical data acquisition program written by Skip Schaller, Manager of Steward's Computer Group. It is a set of routines that operate within the Image Reduction and Analysis Facility (IRAF) environment. IRAF has become a de-facto standard in the US astronomical community, and it includes most of the capabilities needed for the display and analysis of astronomical image data. CCDACQ is an extension to IRAF that provides instrument control functions to the analysis package. This program was originally written for use with the Steward Observatory Charge Coupled Device (CCD) camera, but the software is general enough for use with other astronomical imaging instruments.

CCDACQ is written in IRAF's Subset Preprocessor language (SPP) with low level functions written in the c language. CCDACQ is a set of IRAF tasks that perform various telescope, instrument and detector functions from a remote workstation. CCDACQ is currently used at Steward Observatory's 90 inch telescope at Kitt Peak, in lab testing of various optical CCD's, and is currently being incorporated into the operations programs of three telescopes operated by National Optical Astronomy Observatory at Kitt Peak.

The basic hardware architecture for implementing remote operations using CCDACQ is similar to that used by our OASIS testbed and is shown in Figure 6. In this example, the remote workstation is running the UNIX operating system and IRAF. The real-time system contains the hardware interface needed to control the instrument. The real-time system could be a VME chassis or an IBM PC housing intelligent controllers used to communicate with the instrument(s). The instrument could be a single CCD array, or an entire system of telescope, additional instruments and the detector(s). The remote workstation is connected to the real-time system using an ethernet connection (however, this could be a serial connection). The physical connection between the real-time system and the instrument can be serial or parallel.

The CCDACQ, starts three processes that act as servers for the detector, instrument and telescope. Communication between the UNIX workstation and the real-time system is accomplished through TCP/IP network protocols accessed by way of Berkeley UNIX socket library functions. The interface between the real-time system and the instrument is accomplished through the use of specialized programmable plug-in

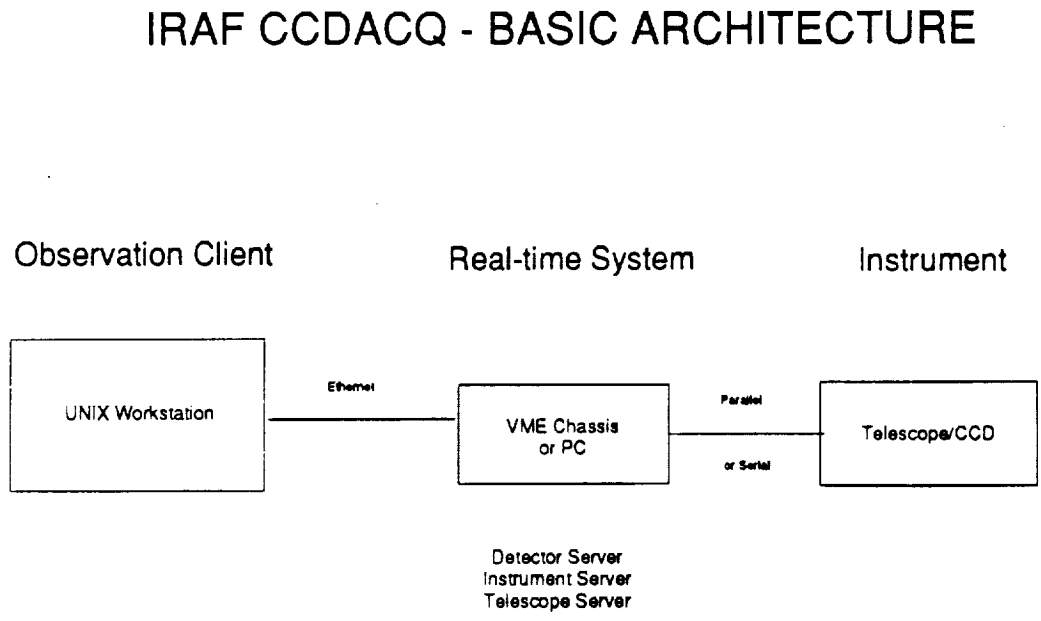


Figure 6. CCDACQ Block Diagram

boards that are used for data acquisition and control. An IBM PC or some specialized microprocessor communicates with the plug-in board through locally developed code, generally written in the C language. This local processor code controls the instrument and provides the communication interface to the remote server software.

The CCDACQ process is twofold, first a set of parameters for each of the server processes (detector, instrument and telescope, observation) must be set. These parameter set tasks only edit the parameters. They do not initiate any physical action until an actual "observe" or "detector" task is run. The observer can make a series of test observations to check all of the instruments and to receive status information on each. Once the observation has been initiated, other action tasks allow the observer to pause an exposure, then resume, stop exposure and read out data, or make a series of observations with current parameters. IRAF/CCDACQ basic functions are outlined in Figure 7.

IRAF has already been identified as the image reduction package of choice by the SIRTF teams. With CCDACQ, instrument control as well as image reduction can be done in one environment. Future plans are to convert current MIPS detector test code into portable C code incorporating the communications functions required to interface with the remote UNIX workstation. Socket-based IPC (interprocess communication) was chosen for the transport level programming interface because at the time of development, it was the preferred standard. However, other OSI-compatible transport mechanisms based upon STREAMS and accessed by way of a Transport Library Interface (TLI) will need consideration.

The second package reviewed was Precision Visuals' Workstation Analysis and Visualization Environment (PV~WAVE). PV~WAVE is an interactive data display and analysis software package currently used by the Short Wavelength Spectrometer (SWS) Team for the European Infrared Space Observatory. PV~WAVE has it's own structured application development language, and a set of procedures and functions that can be linked with existing C or FORTRAN code. PV~WAVE is currently installed on a Sun SPARCstation 1 and used by MIPS scientists in graphic analysis and modeling of MIPS detectors. In the implementation of the ground support software, 62 detector channels are displayed in real time on MicoVAX workstation. The PV~WAVE application is characterized by very high performance graphics display.

PV~WAVE could function in the capacity as IRAF/CCDACQ described above. The same communications software describe above could be linked with PV~WAVE functions for real-time data acquisition and quick look analysis. Although PV~WAVE is capable of performing many of the same data analysis and graphic functions found in IRAF, the major difference is it's target user. IRAF is a package written specifically for astronomical image reduction and analysis. It incorporates many standards used by the astronomical community, like using the FITS format for data exchange. PV~WAVE users can be any scientific or technical user. Additional programing may be required to perform specific astronomical tasks in the PV~WAVE environment.

IRAF CCDACQ PROCESS

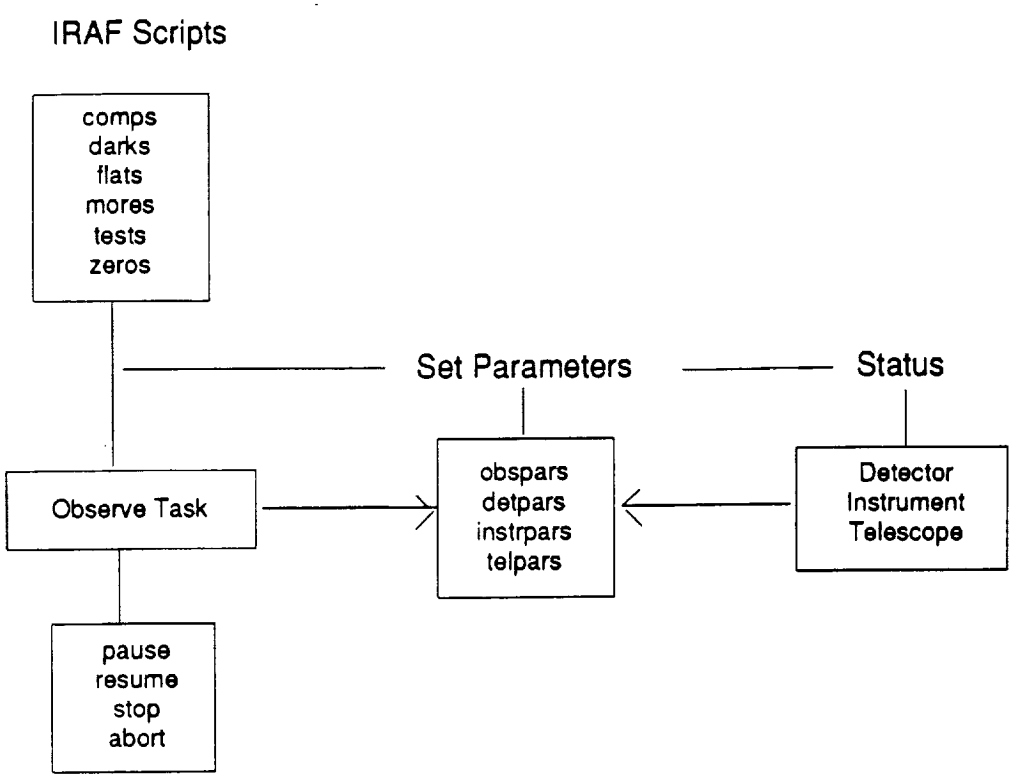


Figure 7. IRAF CCDACQ Processes

APPENDIX A

Testbed Participants

University of Arizona

Steward Observatory

Dr. Erick T. Young, Principal Investigator

Irene Barg, MIPS Systems Programmer

Electrical and Computer Engineering

Dr. Larry Schooley, Co-Investigator

Yadung Pang, Graduate Student

Henky Wibowo, Graduate Student

APPENDIX B

Glossary

CCSDS	Consultative Committee for Space Data Systems
DEC	Declination Astronomical Coordinate
DECnet	Proprietary Communications Protocol from Digital Equipment Corp.
ECE	Electrical and Computer Engineering
GSE	Ground Support Equipment
IC	Instrument Controller
IRAC	Infrared Array Camera
IRAF	Image Reduction and Analysis Facility
IRS	Infrared Spectrometer
LCC	Local Control Computer
MIPS	Multiband Imaging Photometer
OASIS	Operations And Science Instrument System
PI	Principal Investigator
RA	Right Ascension Astronomical Coordinate
RCC	Remote Control Computer
SIRTF	Space Infrared Telescope Facility
TCP/IP	Transmission Control Protocol/Internet Protocol
TSL	Telescience Laboratory
VMS	Proprietary operating system from Digital Equipment Corp.

APPENDIX C

Bibliography

Bienz, Richard and Hunter, Jerry, "Communication Software Design for Telescience Demonstrations", Telescience Technical Report TLS-019/88, Electrical and Computer Engineering Department, University of Arizona, Tucson AZ., 1988.

Schooley, L.C., and F.E. Cellier, "Telescience Testbed Pilot Program Final Report", Technical Report TSL-021/88, Electrical and Computer Engineering Department, University of Arizona, Tucson AZ., 1988.

Wibowo, Henky, "Communications Software for Telescience", Masters Thesis Electrical and Computer Engineering Department, University of Arizona, Tucson, AZ., 1990.