NIRSPEC

UCLA Astrophysics Program **U.C. Berkeley** **W.M.Keck Observatory**
4444444444444444444444444444444444444444444444444444444444444444
Tim Liu **February 1, 1996**

NIRSPEC Software Design Note 2.03
Software System Design

1 Introduction

The NIRSPEC's host computer is a Sun Sparcstation running Solaris, which will communicate with a network of transputers that act as the front end system via a SCSI port. The host computer will provide the functions of user interface, communicating with the transputer system, image display and quick look, real-time data reduction, data storage, etc, while the transputer system will perform the tasks of array clocking, data acquisition, stepper motor control. This document outlines the design of the software system and describes its various components and interfaces between them. Some of the prototyping results will be presented at the end. Figure 1 shows the structure of the NIRSPEC software design as discussed in this document.

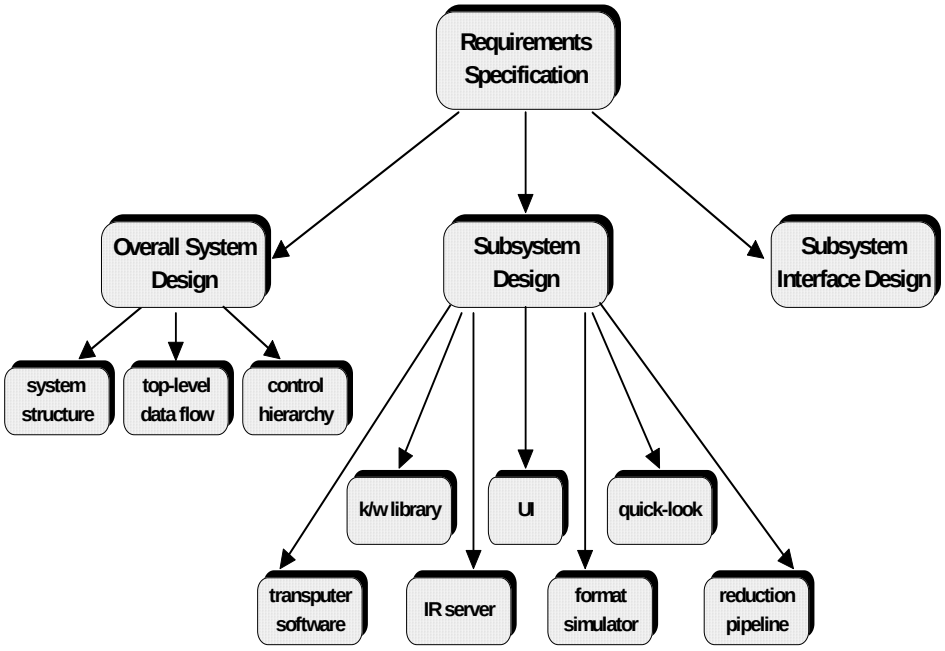

Figure 1 Software design structure

2 Requirements Specification

The NIRSPEC software requirements specify that the software system must provide the following functionalities and features:

- an X-window based graphical user interface which is user-friendly, intuitive and robust
- a command line interface with script capability which is very useful for carrying out repetitive observing sequence and engineering tests
- KTL-keyword control mechanism for application-hardware interface which is required by SCC (Keck Software Coordination Committee)
- host computer-to-transputer communication with link data transfer rate ~ 5 sec for a full 1024x1024 32-bit frame
- transputer software for low-level functions like array clocking, data acquisition, motion control and temperature read-back
- engineering diagnostic routines to exercise sub-assemblies
- a spectral format simulator for easy instrument configuration
- data I/O and storage functions
- IDL-based quick look facility with integrated image display, spectral plot and data manipulation functions
- an automated real-time data reduction pipeline
- DCS interface
- remote observing
- a simulation mode which runs without the hardware
- on-line help

The software design follows the following guidelines:

- simplicity
- modularity
- reliability

3 Overall System Design

Figure 2 is a simplified software structure chart which outlines the major subsystems. They include:

- user interface
- keyword library
- IR server
- quick-look
- data reduction pipeline
- DCS interface

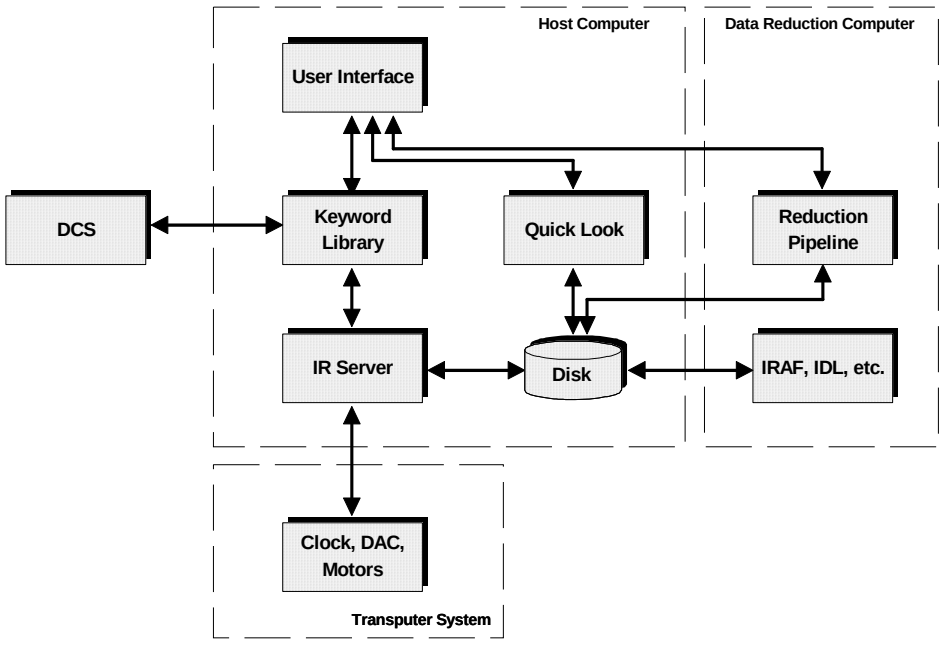

Figure 2 Software system structure

As the figure shows, the real-time data reduction pipeline will actually run on a different computer from the host. That is because the reduction pipeline is highly CPU-intensive and would affect other critical processes if it runs on the host machine. In addition, some on-line data analysis packages such as IRAF, IDL, etc. will also reside in the reduction computer.

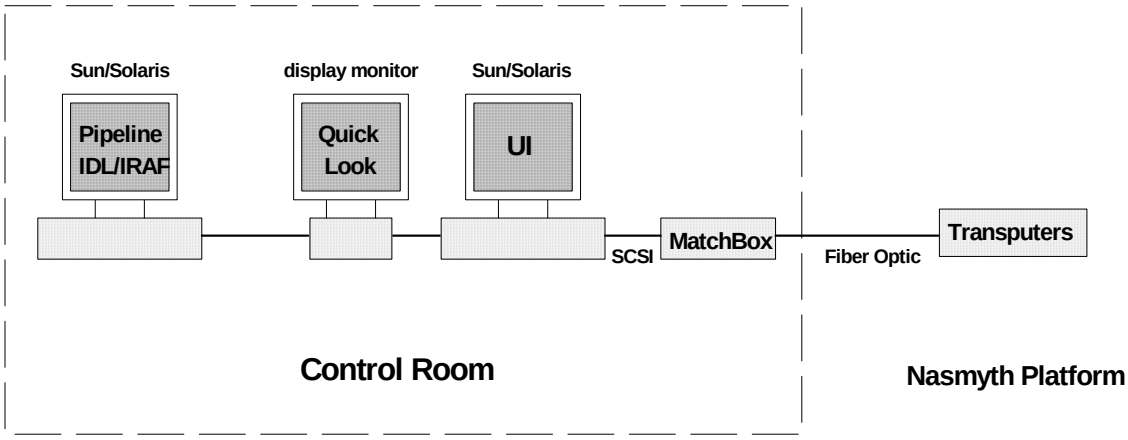

Figure 3 Physical layout

Figure 3 shows the physical layout of the NIRSPEC software system that will run at the Keck telescope. A separate display monitor from the same host machine will be used for quick-look because the graphical user interface will need most of the host display screen. The current software prototyping and development are being done from a Sun Sparcstation 20 with 32 MB RAM. However, a faster CPU and larger RAM size machine will be purchased for the host computer when the system is delivered.

Figure 4 shows the top-level logical data flow in the system. Note that NIRSPEC includes a slit-view camera which can be considered the second channel of the instrument. The slit-view camera will require software functions similar to the spectrometer such as user control, clocking, data acquisition, data I/O an display. Because the slit-view camera has the same data path as the spectrometer , no separate descriptions are presented for it in this document.

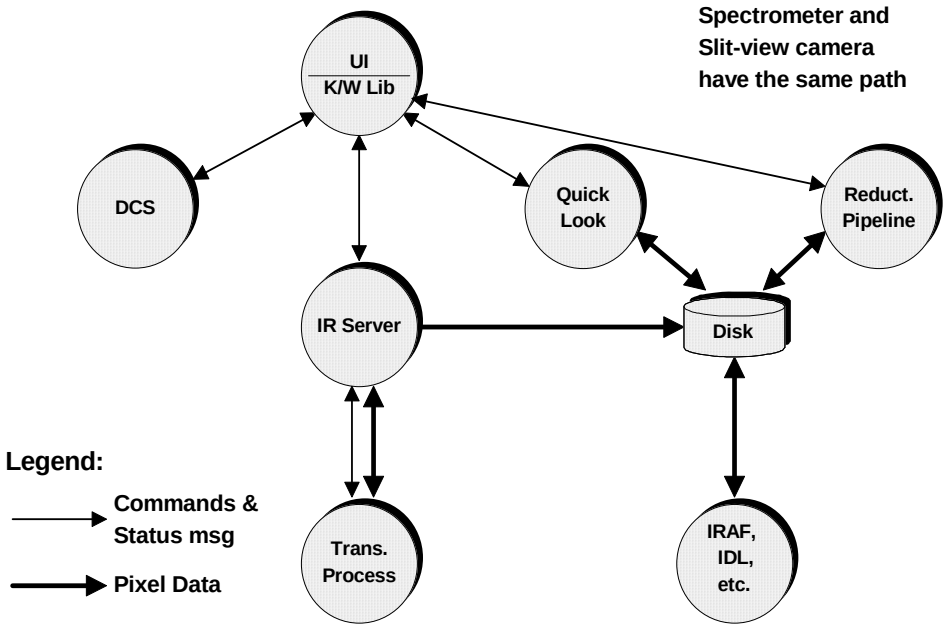

Figure 4 Top level logical data flow

The actual pixel data flow in the host software is presented in Figure 5. Raw pixel data from the transputer data acquisition system are saved in the disk in FITS format after being displayed (data can bypass the display). The raw 2-D spectral data then go through the reduction pipeline. Both raw and reduced data can be accessed from the quick-look and on-line reduction tools.

As illustrated in Figure 6, the subsystems are coordinated via different control mechanism. UI communicates with IR server, a high-level application control process, via KTL keywords. A keyword library layer interfaces between them. The keyword library uses a message system like RPC

for inter-task communications. Under this architecture, GUI and CLI become KTL clients that interact with subsystems with the same APIs if implemented in the same mechanism like DCS.

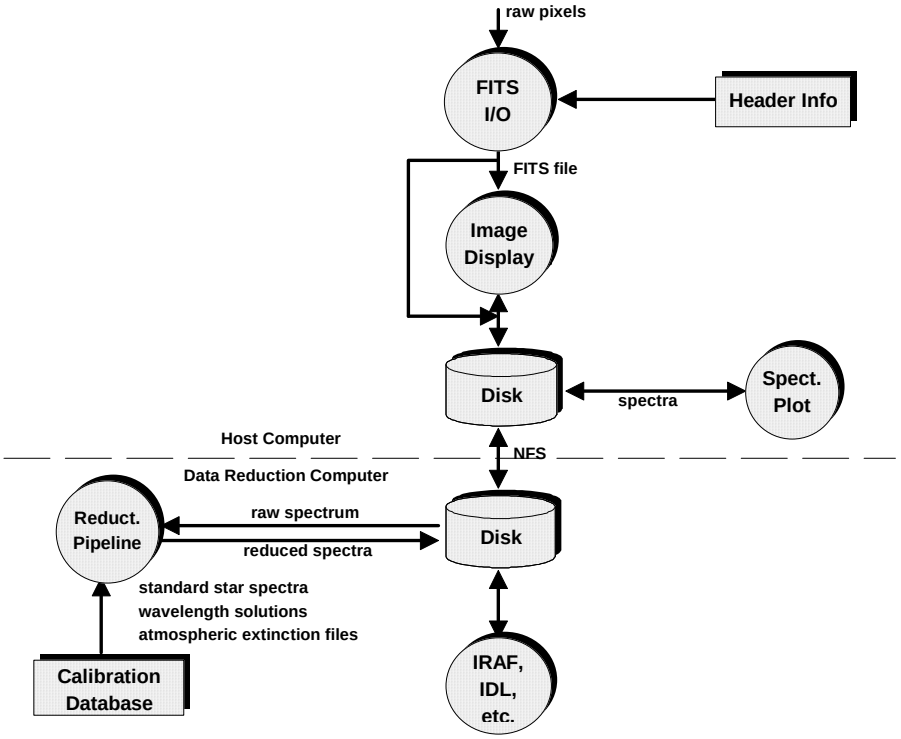

Figure 5 Pixel data flow in host software

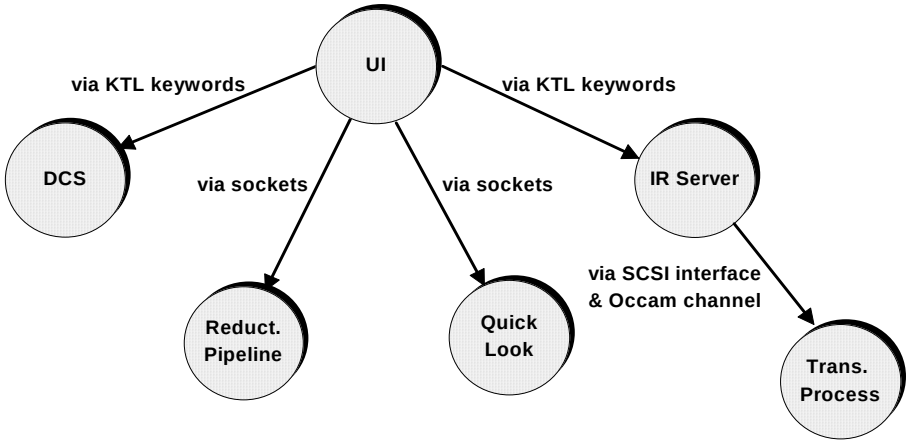

Figure 6 Software control hierarchy and mechanism

The IR server (a RPC server if RPC is used for the keyword library communication system) talks with the underlying low-level transputer process through SCSI link interface and also Occam channels among different transputer processes.

For interactions between UI and quick-look and data reduction pipeline, Unix sockets will be used for passing messages like frame arrival signals, setup parameters, etc.

4 Subsystem Design

This section describes the designs of each sub-system without giving much details. The full design of the subsystems can be found in separate design notes.

4.1 User Interface (UI)

The user will control the instrument through a software user interface which provides both window-style graphical input and command-line input. The main function of the command-line-driven interface is to provide the script capability. The following basic features are required for the UI:

- instrument and observing setup
- instrument status and exposure status display
- user input check and software interlock
- engineering interface for diagnostics

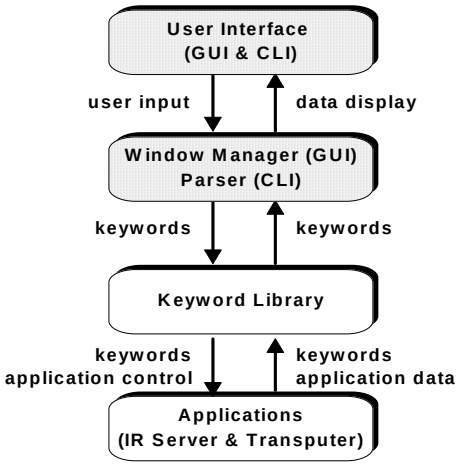

Figure 7 User interface control flow

The UI is interfaced to the instrument via keyword library, i.e. it interacts with the control system by reading and writing keywords. Figure 7 shows the control flow between the interacting components.

Graphical User Interface (GUI):

The GUI will be implemented as an X-Window application with the click-on-icon/pop-up-menu style. It is required to be intuitive, user-friendly and robust. The GUI has several main windows and they are:

- an instrument control window to access to hardware subsystems
- an observing setup window for observing sequence setup and exposure status display
- a separate window for slit-view camera for parameter setup and status display

The GUI has been prototyped with DataViews, a commercial software tool which is capable of building complex graphical interfaces. DataViews has been used for several user interfaces on Keck I like xlws. The pictures of both entire screen and the three main windows mentioned above are included in the end of the document. The whole screen picture also includes image display windows from quick-look program.

The main features of the instrument control window are outlined below:

- an instrument cartoon showing hardware components
- each user-controllable hardware unit has:
  - a graphical object with dynamic feature
  - a status display for current configuration
  - a control button with popup menu or screen
- Pull-down menus:
  - setup: read/save configuration , etc.
  - engineer: routines to access to sub-assemblies; diagnostics and analysis functions; password is required for access
  - help: on-line help facility
- Quit button:
  - shut-down system and quit
  - only place to exit

The observing setup window includes the following main features:

- Exposure setup:
  - entries: object name, itime, coadds, sampmode and comment
  - accessible during next integration for next exposure setup
- Exposure status display:
  - showing integration progress from a graphic bar

- showing remaining exposure time and percentage done
- Exposure button:
  - start: GO and TEST (TEST not save data automatically)
  - abort: ABORT
- Pull-down menus:
  - setup: general setup, etc.
  - FILE: save test frame, file overwrite flag, etc.
  - options: script commands, etc.

Command Line Interface (CLI):

A command line interface with script function is very useful for carrying out repetitive observing sequences such as making a mosaic or doing repetitive engineering tests. Common observing routines like taking dark frames can easily be constructed using the script tool and be executed more efficiently. The design requirements for CLI are:

- has script facility
- based on KTL keywords
- simple interface to application

The second requirement means that the command parser is on top of the keyword library so that it can validate keywords typed in. The command line interface will be built from Tcl, a public-domain embeddable command language. Tcl has C shell-like syntax. It provides a rich set of built-in commands like variables, conditionals, looping, procedures and scripts. The implementation strategy is to make each KTL keyword a valid Tcl command using its function library. Figure 8 shows the structure of the CLI when implemented with Tcl.

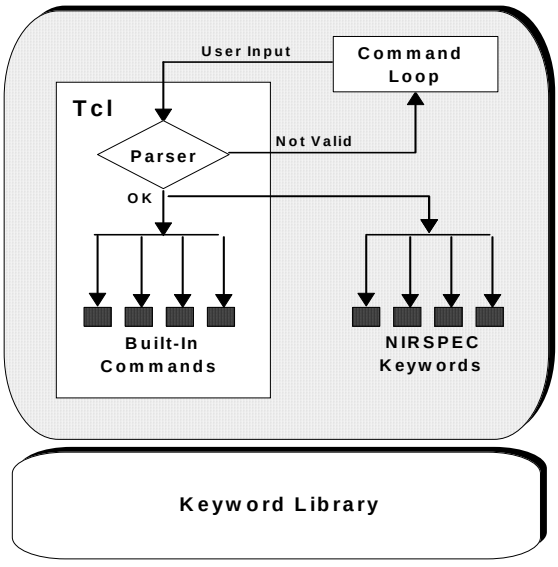

Figure 8 Command line interface structure

4.2 Keyword Library

Similar to existing Keck instruments, the NIRSPEC software will provide a keyword library which interfaces between the KTL application code (user interface) and the underlying hardware (transputer system). This will provide a consistent programming interface with other Keck software for writing KTL applications. The implementation of a keyword library is required by SCC. All NIRSPEC functions will be defined by keywords. Figure 9 shows the keyword interface layer.

At present, there are no SCC mandate on the underlying message system for the keyword library. Both MUSIC and RPCs have been used for the Keck I instruments. There are also recent considerations of using EPICS/CA for Keck II instruments. It is agreed among SCC members, however, that a keyword library doesn't have to use a single communication system and it can use a mixture of the different schemes. For example, one can use RPC for single-shot keyword read and write and use MUSIC for broadcast. For the NIRSPEC keyword library prototyping we have used RPC for the inter-process communication mechanism.

A preliminary list of keywords has been defined for NIRSPEC. They are grouped into the following category:

- observing setup keywords
- exposure control keywords
- motor control keywords
- clock timing keywords
- house-keeping keywords
- data I/O keywords

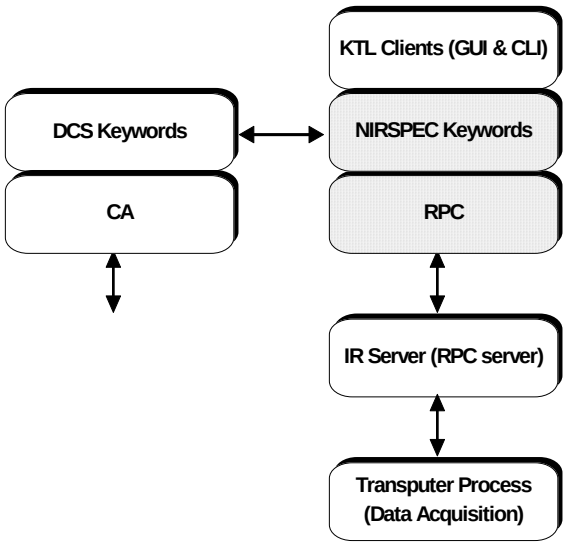

Figure 9 Keyword interface layer

4.3 IR Server

IR server is a high-level process that communicates with the low-level transputer processes. It acts as a translation layer between the KTL keyword interface and the transputer message protocol which is also keyword oriented. The functions it performs are:

- provides the mapping between NIRSPEC keywords and transputer keywords
- handles command and status message flow to and from transputer
- acquires frame data sent by transputer data acquisition

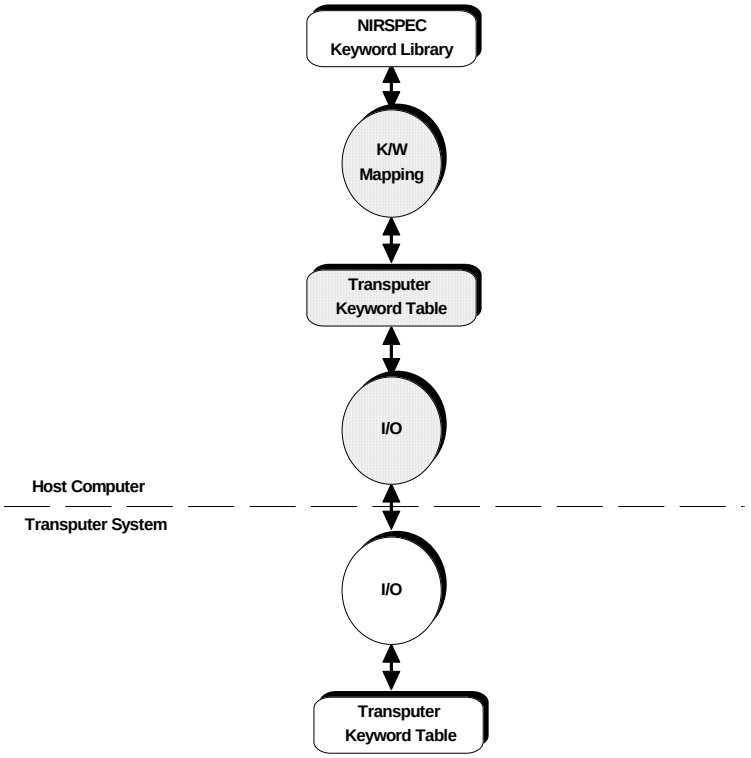

Figure 10 IR server structure

As illustrated in Figure 10, when a KTL keyword talks to the hardware, the IR server determines its corresponding transputer keyword and then send to the transputer system via the host computer-to-transputer link interface.

4.4 Host-to-Transputer Interface

The physical connection between the host Sparcstation and the transputer system is a SCSI link provided by Matchbox, a Transtech product. The custom software being developed includes communication protocol and high-level interfacing routines. The defined protocol has the following characteristics:

! uses a single protocol for both short message and long data stream
! compatible with Occam channel protocol which make message I/O easier at both ends
! message packet structure is byte-oriented with fixed length of message header and variable length of message body
! packet size is 4096 bytes, the maximum allowed, for fastest transfer rate

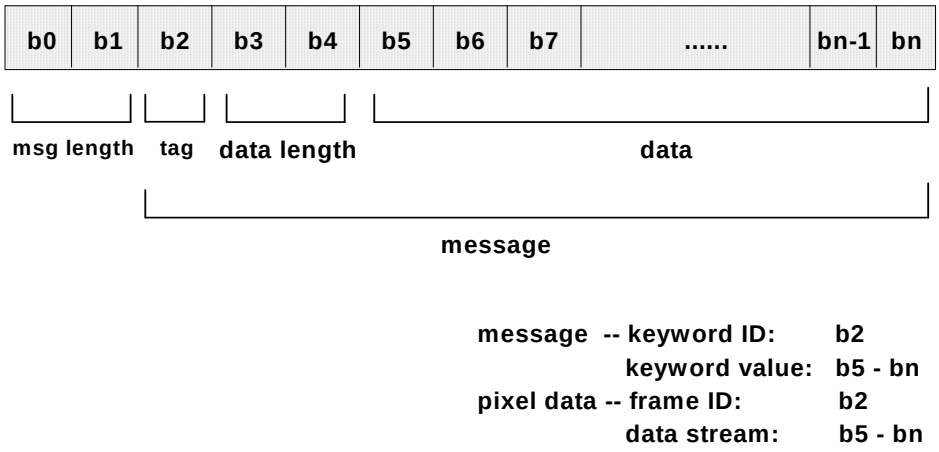

Figure 11 Message packet structure

Figure 11 shows the message packet structure with command message and data encoding scheme. The details of the protocol design will be described in a separate document. High-level communication library has been developed using vendor-supplied low-level routines.

4.5 Transputer Software

The frond-end consists of a network of transputers that performs the functions of array clocking, data acquisition, motor control and house-keeping. Figure 12 is a data flow diagram of the transputer software. The root transputer acts like a communication hub that handles data flow between the host and the transputer subsystems such as the timing generator boards and data acquisition boards.

The transputer system is programmed in the high-level language Occam. We have extensive experience in this area, as obtained from the development of the Gemini two-channel imaging camera. The NIRSPEC Occam code will have three parts in terms of their functionalities:

! Clock generator generates sub-waveforms for repeating hardware
! Data acquisition coadds contents of FIFO after polling or interrupt
! Motor control generates bit stream through memory mapped output port

Prototype Occam code has been developed and tested on the newly arrived data acquisition boards. The prototype performs two functions:

- two data acuqisition transputers (one for the spectrometer and the other for the slitview camera) generate data packets for both channels
- root transputer passes command and status message to DAQs and sends data back to the host

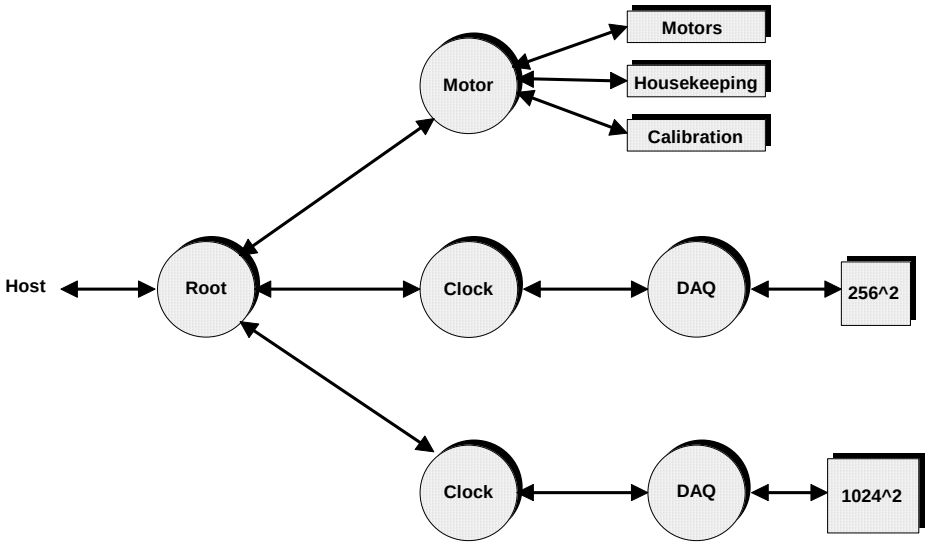

Figure 12 Transputer software data flow

The transputer software prototype has been fully integrated with the host software prototype, thus providing a test platform for transputer hardware with complete user control, data acquisition, data I/O and storage, and image display and quick-look analysis functions.

4.6 Data Storage

Image data files are written in FITS format and stored on hard disk in the host computer. The FITS header will contain complete information on observing setup and instrument configuration. Data Processing information from the reduction pipeline will also recorded in the header. One can start an exposure while a previous one is being archived on disk. To provide enough space for raw and reduced data from a typical 3-5 night run, a large disk storage (~5 GB) is required. We will implement a single command in the host software to write disk data onto Exabyte tape. We will also provide an option to record data to tape automatically after each exposure to save data writing time at end of night.

4.7 DCS Interface

NIRSPEC will use KTL keywords to interact with the telescope through the DCS keyword library as shown in Figure 13. The functions performed by the NIRSPEC-DCS interface include:

- commands DCS system and listens for response
- responds to DCS requests for instrument status
- receives DCS messages such as time message
- broadcasts instrument status

GUI will provide a display for DCS parameters.

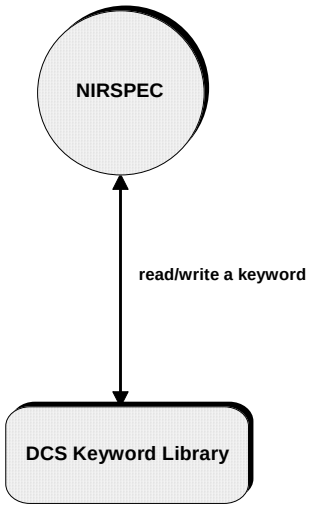

Figure 13 DCS interface

The host software will be able to control the DCS for data acquisition in different observing modes: stare, nodding and chopping. Software interlocks will be implemented in the DCS interface routines to prevent accidental user errors like moving telescope during integration.

4.8 Quick-Look

For a quick assessment of data quality, a quick-look tool is needed. The current Keck instruments uses a variety of standard image display/reduction tools for quick-look purposes, such as SAOimage, figdisp, ximtool, etc. Unfortunately, these packages either mainly serve as image display tools and don't have all the functions necessary for quick-look or provide data manipulation functions too comprehensive to be easily used by a user without much knowledge of the packages. To integrate image display with data manipulation facilities specific to our needs and easy to use, we will develop our own quick-look routines. However, this won't be started from scratch. The quick-look routines will be written using IDL which has a set of Motif-style graphical user-interface widgets and various math routines. We only need to write interface code, thus the programming effort is greatly reduced. We will get an evaluation kit of the new IDL product soon.

The quick-look requires two 2-D image display windows for images from both spectrometer and slit-viewing camera and a 1-D spectral plot window for reduced spectra from the data reduction pipeline, as shown in the entire screen picture included in the end of this document.

The quick-look functions provided by the image display include:

- basic features:
  - zoom and pan
  - scale: linear and logarithmic
  - palettes: grayscale and color
  - contrast/brightness change: automatic or manual
- computation: arithmetic, statistics, centroid, distance, and photometry
- plots: radial profile, line-cut, contour, histogram, and surface plot
- data I/O: image read/write and hardcopy

The spectral plot window will provide the following features:

- plot options:
  - plot multiple traces
  - automatic or manual xy scaling and zoom
  - various x-axis units: pixel, wavelength, frequency, and velocity
  - overplot
- fitting:
  - continuum: polynomial and spline
  - line profile: gaussian and polynomial
- computation:
  - line: center, integrated flux, FWHM, sigma, and equivalent width
  - region: arithmetic, smooth, and statistics
- data I/O: image read/write and hardcopy

In addition to the quick-look facilities, it is necessary to provide an on-line sophisticated data reduction package so that observers can examine their data more carefully than using the quick-look functions. Popular data reduction programs like IRAF and IDL will be supported. The data analysis tools will run on a different workstation from the host computer. The data retrieval/archival can be done through NFS over Ethernet or some automatic data transfer link.

4.9 Data Reduction Pipeline

The host software will provide real-time data processing. Depending upon which observing mode is used (nodding or stare), raw data from the spectrometer will go through the appropriate data reduction pipeline as shown in Figure 14 which lists various processing steps and the final reduced data are produced.

The pipeline needs to be configured before the reduction starts. This is done by visually examining a dark/sky subtracted and flat-fielded 2-D spectrum, manually extracting the rectified spectrum, and thus finding the optimal reduction algorithms and setup. Once the pipeline is set up, incoming raw data will be reduced into 1-D spectra automatically. The GUI will provide a menu that displays the processing status.

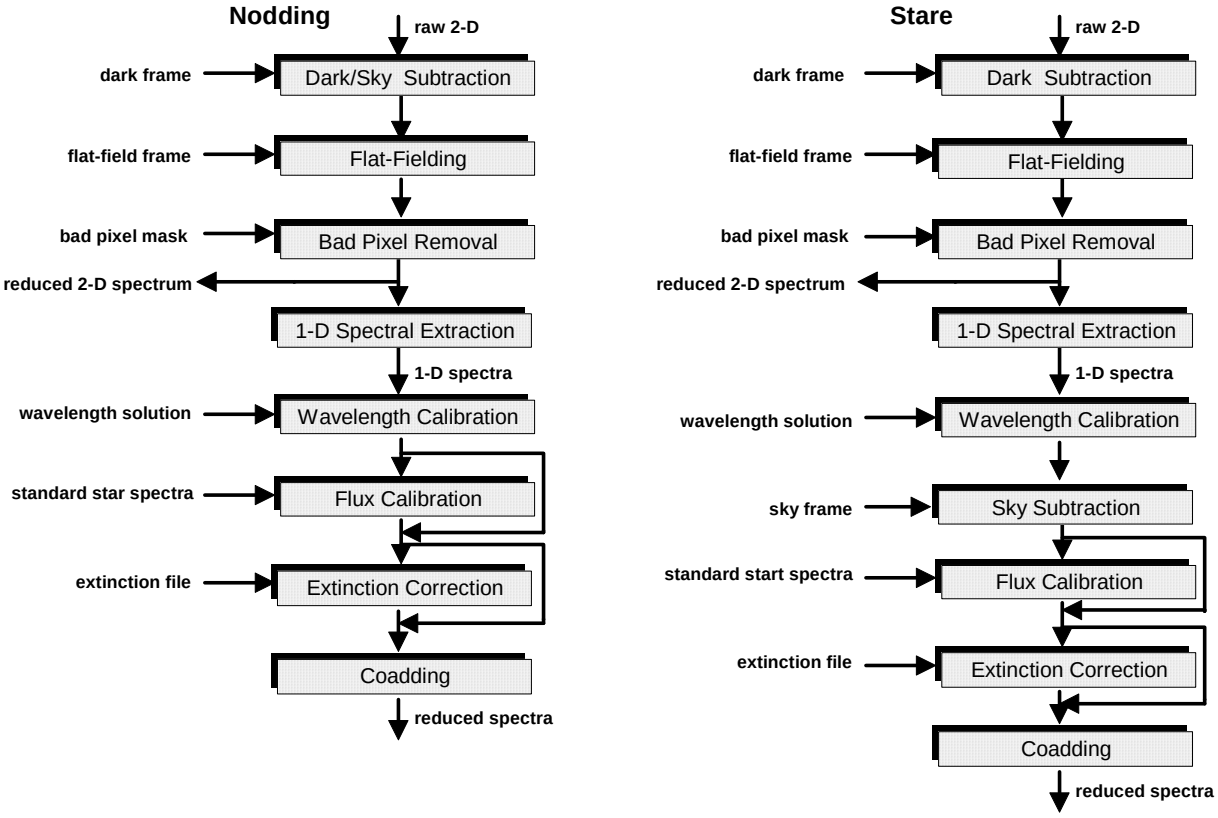

Figure 14 Data reduction pipeline

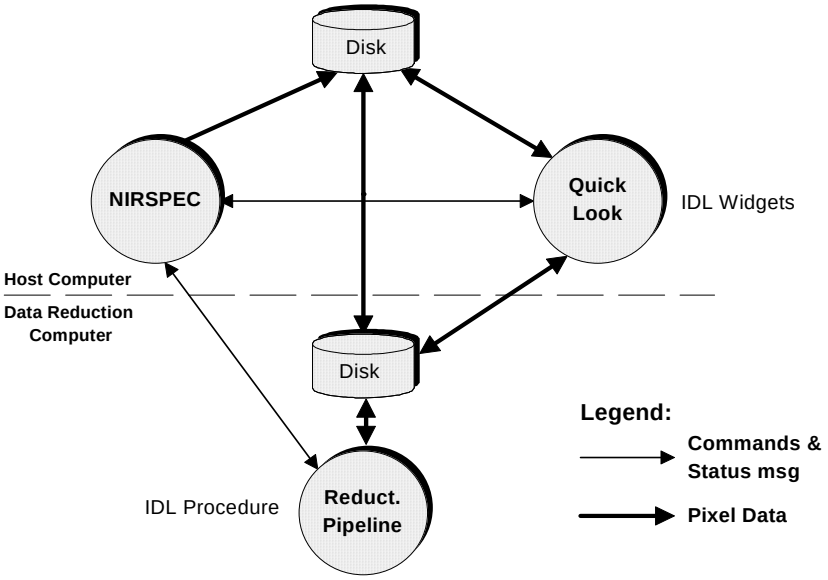

Figure 15 Data reduction pipeline data flow

As Figure 14 shows, in addition to 1-D extraction, the pipeline outputs reduced 2-D spectra as well. Sometimes it will be easier to spot a signal in a 2-D spectrum than in a 1-D one. The reduced data will be archived on disk in FITS format and they can be examined using quick-look. The FITS header will provide complete data processing information.

The pipeline will be written with IDL which has a rich set of math library routines. Figure 15 illustrates the top-level data flow in the pipeline software. The pipeline will run on a separate computer without affecting the performance of the other processes on the host.

4.10 Other Functions

The software system will have a simulation mode which runs without the transputer system. This will allow the host software development independent of the hardware and transputer software development. The simulation mode will be useful as well for observing practice before an actual run.

5 Programming Environment

5.1 Directory Structure

We will follow SCC standards on directory structure, build procedure and release mechanism. A master /kroot directory tree has been created on the host machine and is owned by the user "nirspec". The /kroot tree contains the sub-systems /kroot/kss/nirspec and /kroot/kui/xnirspec.

The host software development will be performed in programmers's private /kroot directories. The master tree will only contain the released versions.

5.2 Language

The host software will be coded in C. All the host software will be compliant with ANSI C and POSIX. We used Gnu C during our ptototyping, but we have now switched to Sun compiler .

5.3 Coding Standards

We will adopt "Indian Hill" coding standards on programming style, function and variable naming convention, in-code documentation, etc, with some SCC amendments like Keck-specific naming conventions. A source code template will be generated. We'll consider the use of manual generation tools like WFLman for automatic documentation extraction.

SCCS will be used as the source code management tool. It may be replaced later by CVS (Concurrent Versions System) when more details become known.

6 Software Documentation

Figure 16 outlines the documentation process at each phase. The software documents that will be provided upon the delivery of the NIRSPEC software system include: Programmer's Manual (or System Manual) and User's Manual. The Programmer's Manual will have the following contents:

- software structure overview
- hierarchical list of source code, make files, and libraries and their directory locations
- brief description of each source file and function routine
- how to modify the system (e.g. modify or add a function)
- how to rebuild the system from new source code
- system maintenance

The User's Manual will contain:

- instrument description
- starting/shut-down procedures
- a tutorial of GUI, command line and scripts
- list of description of script commands
- instrument configuration and description of spectral format simulator
- observing parameter setup and description of exposure estimator
- make calibration exposures
- a tutorial on quick-look
- data reduction pipeline
- data archive

! software trouble-shooting guide

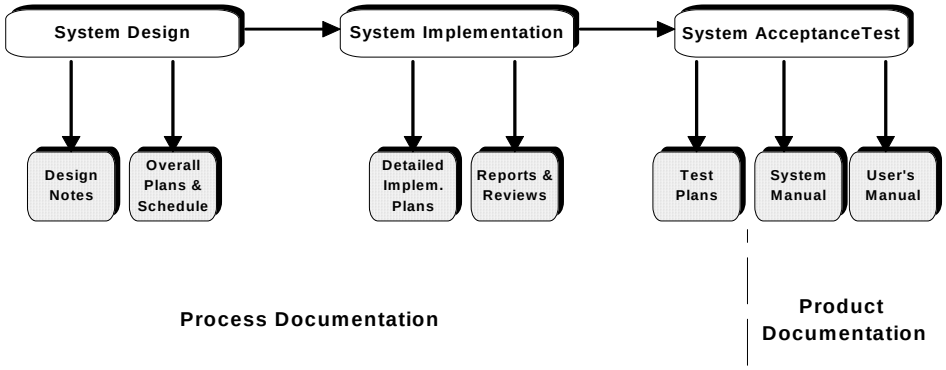

Figure 16 Software documentation process

7 System Prototyping

We have followed a software development process model given in Figure 18. As the figure shows, system prototyping is a very important part of the design phase. Our goals are:

! produce a prototype that integrates user input, application control, data collection and display
! demonstrate the feasibility of key designs
! help understand component interactions and interfacing
! find and solve major implementation issues at early stage as a risk reduction strategy
! serves as a basis for full-scale implementation

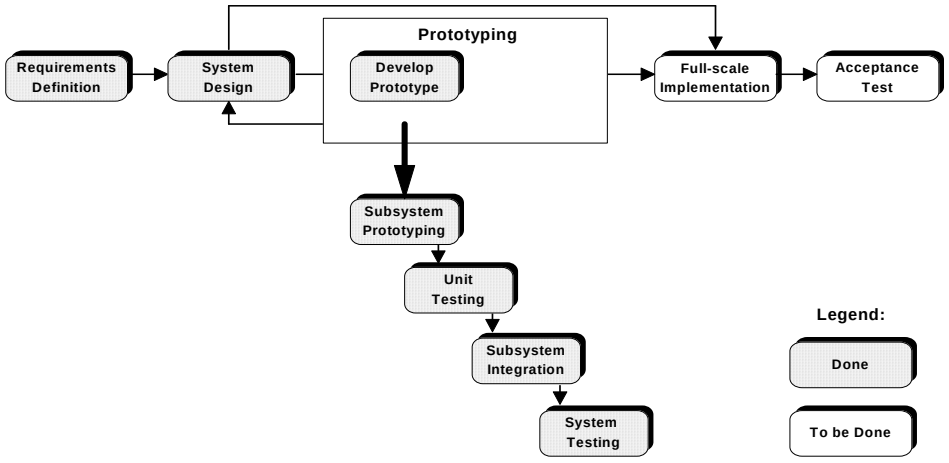

Figure 17 Software development process model

The prototyping steps include:

! identify subsystems to be prototyped
! subsystem prototyping
! subsystem integration and system testing
! prototype evaluation
! design iteration

The subsystems that have been prototyped include:

! GUI
! keyword library
! IR server
! host-transputer interface
! transputer software
! quick-look
! spectral format simulator

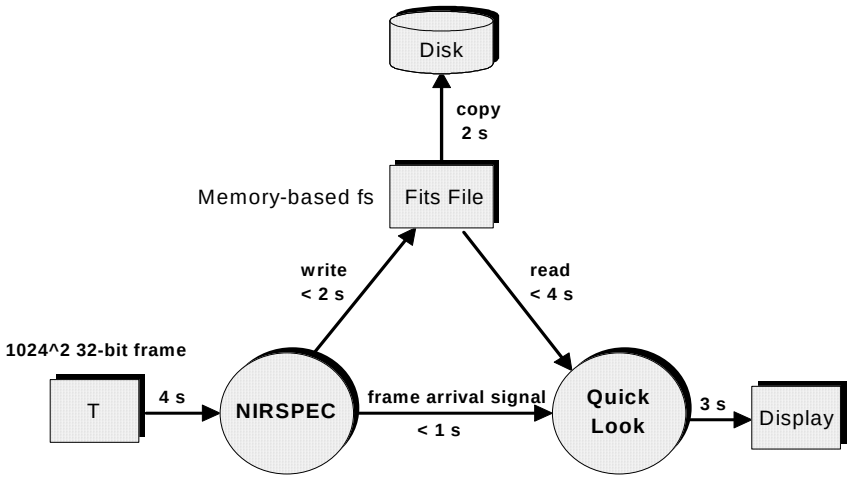

Figure 18 Data rates given by prototype

The prototyped subsystems have been integrated, which produces a NIRSPEC software prototype that has met our prototyping goals. The prototype is fully working. We have derived data rates based on the prototype as given in Figure 18. The GUI prototype provides an ideal tool for developers to interact with end-users. The prototype software also provides a test platform for electronics hardware with complete user control, data acquisition, data I/O and storage, and image display and quick analysis functions. Full-scale implementation of the software will be based on the prototype.

The current status of the software development is summarized as below:

! requirements are well understood
! system design is completed

- major subsystem are prototyped
- full-scale implementation will begin after CDR