BUILDING THE CONTROL SYSTEM TO OPERATE THE CRYOGENIC NEAR INFRARED SPECTROPOLARIMETER INSTRUMENT FOR THE DANIEL K. INOUYE SOLAR TELESCOPE

R. J. Williams*, A. J. Borrowman, A. Greer, A. Yoshimura,
Observatory Sciences Ltd, St Ives, UK
A. Fehlmann, B. D. Goodrich, J. R. Hubbard, DKIST/NSO, Boulder, USA
I. F. Scholl, University of Hawaii, Pukalani, USA

Abstract

The Cryogenic Near Infrared Spectropolarimeter (Cryo-NIRSP) will be one of the first-light instruments on the Daniel K. Inouye Solar Telescope (DKIST) currently under construction in Hawaii. Cyro-NIRSP is a near- and thermal-IR imager and spectrograph operating in a cryogenic environment. It will be used to study the faint solar coronal magnetic field across a large field of view. Such a complex and precise instrument demands equal requirements from the control system, which must manage the setup, timings, synchronization, real time motion, and overall monitoring of the many sub-components (e.g. cameras, polarimeter, mirrors). The control system is built within the pre-defined DKIST software framework, which provides consistency across all instruments. This paper will discuss how such a control system has been achieved for the Cryo-NIRSP instrument, detailing some of the challenges that were overcome relating to the synchronization of specific components and the complex inter-dependencies between configurables. It will also describe the data processing and visualization software development for the end-to-end functioning of the instrument.

INTRODUCTION

The DKIST (currently under construction) will be the world's largest ground-based solar telescope, with a 4-meter aperture. It will study the sun in visible to near-IR wavelengths and has adaptive optics to provide high spatial resolution of the sun's features. The telescope will have 5 first-light instruments, one of which will be the Cyro-NIRSP currently being fabricated by the Institute of Astronomy at the University of Hawaii. The main purpose of the Cryo-NIRSP is to study the solar coronal magnetic field over a large field of view (FoV) at near- to thermal- IR wavelengths and measure the full polarization state (Stokes I,Q,U,V) of spectral lines originating from the sun. The thermal IR capabilities mean that it will be able to study the solar disk in the CO lines and it will also be able to study both the near-limb and more distant corona, providing data on features such as prominences, flares, and other events in the low corona.

In parallel to the construction of the Cryo-NIRSP instrument itself has been the development of the control and data-processing software, both essential for the running of the instrument. The Cryo-NIRSP software system is responsible for: monitoring and controlling all mechanisms, interacting with the instrument cameras, interacting with the modulator system, providing an interactive control for the instrument operation, and providing instrument-specific data-processing plugins. This paper describes how the control software has been developed, including a brief outline of the instrument's many components and complexities, and details the implemented software solutions to manage these. The final section will address the data processing side, which is vital for correctly calibrating the instrument and ultimately meeting the end science goals of the instrument.

* rjw@observatorysciences.co.uk

The Instrument

Cryo-NIRSP is both a near- to thermal-IR imager and a spectropolarimeter, with a spectral resolution of 30000 to 100000. Its critical optics are cooled to cryogenic temperatures giving it the capability of achieving photon-noise-limited observations. Warm pre-optics relay the image from the DKIST coude optics onto mirrors that scan a field of view for 2D spectropolarimetric imaging. Some of the key cooled optics include mirrors, wiregrid polarisers, a grating, filter wheels, and a slit wheel (accommodating both single- and mutli-slit spectroscopy). See Fig. 1 for a schematic of the instrument [1]. The Cryo-NIRSP can facilitate many different observing modes, and all the light can be sent exclusively to the context imager or exclusively to the spectropolarimeter or it can be used for simultaneous imaging and spectroscopy where the light is split off to both.

The camera system consists of the camera hardware itself and the supporting Camera System Software (CSS) [2]. The Cryo-NIRSP uses two IR cameras, one that records the spectra provided by the spectropolarimeter (SP) and another that images the field of view of the spectropolarimeter (i.e. the context imager - CI). Each camera consists of a 2048 x 2048 IR detector array with an accompanying array controller that transfers data collected from the detector to the machine hosting the CSS. The camera hardware sends raw frames to the CSS. Sets of Fully Processed Arrays (FPAs) are then sent to the Data Handling System (DHS). The CSS also contains a Time Reference and Distribution System (TRADS) for computing trigger times based on a reference time and rate. The cameras use nondestructive read ramps (NDRs) and support 3 different readout modes: slow up the ramp, fast up the ramp, and line by line. The up-the-ramp mode samples the signal at regular intervals throughout the exposure and so the signal is seen to ramp up. The line-by-line mode reads out each line of the array one by one.

Movement of the optics and other mechanisms is implemented through the use of the Delta Tau Power PMAC which is used as the standard motion controller hardware for DKIST instruments. The Power PMAC supports both basic control of the motor axes connected to it as well as being able to exercise specific motion sequence programs giving multi-axis synchronization.

The instrument also consists of a polarizing modulator which is used to measure the Stokes angles. This can be operated in a stepped mode where it steps through a series of angles (also known as states), in a continuous mode where it spins at a given rate, or in off mode where it remains at a constant position (state). The modulator is configured to change the polarization state at a constant rate for 16 states per full 360 degrees rotation. This means it will cycle through all 8 states every 180 degrees capturing a complete set of Stokes parameters.

The timing and synchronization requirements are satisfied through the use of TRADS which provides real-time motion control with tight synchronization. The modulator spinning rate, camera frame rate, and all mechanical device motion are synchronized via the TRADS. A TRADS-generated timing signal can be fed as an external time base to the Delta Tau Power PMAC to synchronize mechanisms with the polarizing modular and cameras.

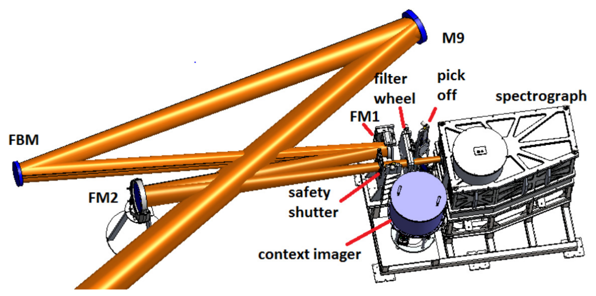


Figure 1: Schematic of the Cryo-NIRSP instrument. Light is directed from the DKIST M9 mirror to the 2D scanning mirrors. These then focus it through the filter wheel and pickoff which splits it off to the CI, SP, or both.

Software Requirements

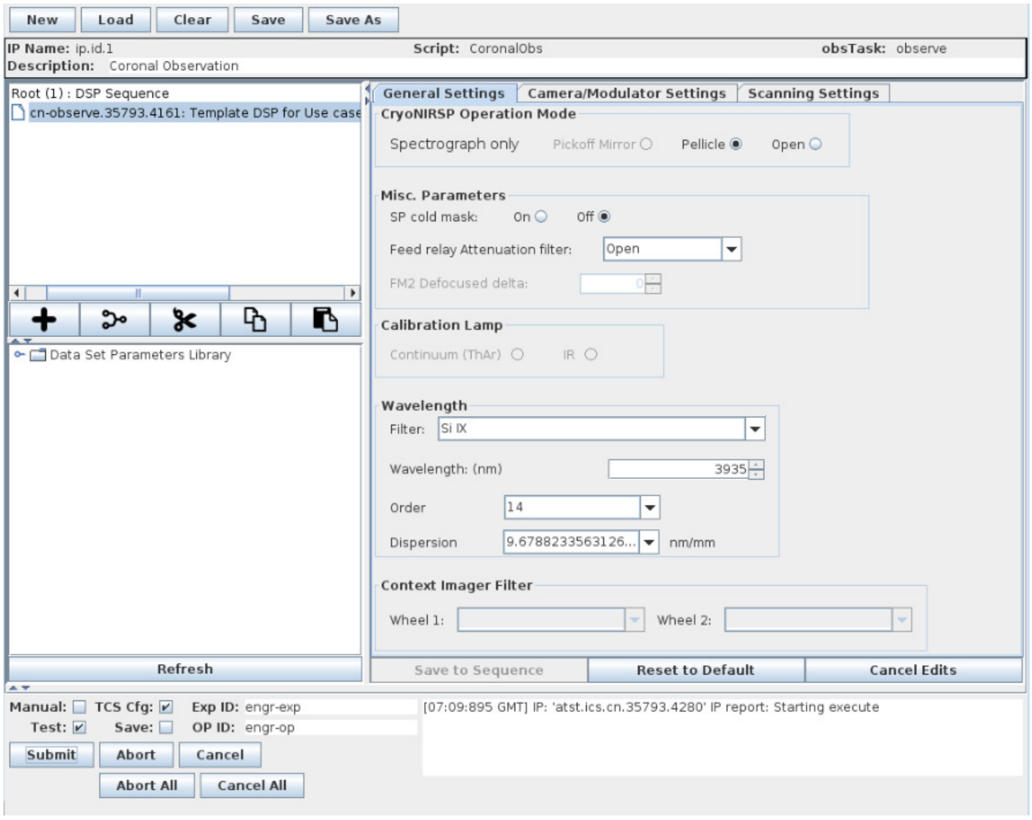
The Cryo-NIRSP supports two primary tasks: calibrations and science observations. Calibration tasks are required to setup the instrument before an observe task is run. An observe task refers to the actual taking of scientific data. There are several calibration tasks that must be run before an observe task, including taking dark frames to check the detector performance, spatial gain calibrations to measure the spatial photometric gain variation, wavelength calibration required for spectrograph observations, focussing of the spectrograph and context imager, and alignment of the mirrors for spatial registration.

The control software has been built within the DKIST software framework described above. DKIST uses the Common Services model to provide a standard infrastructure for the large, distributed software system named the Common Services Framework (CSF) [3] [4]. All major services are provided through this standard interface making it easier to support the individual development teams working on each instrument. It supports a Container/Component Model (CCM) which provides simplified application deployment and execution in a distributed environment. In this model, the deployment and lifecycle of an application are separate from the functional aspects. Containers represent the management applications that are responsible for starting and stopping the functional applications (components). A Controller adds configuration management support to a component. The CSF provides deployment support (i.e. management of applications in a distributed system through the CCM), communications support (notification, logging, connection, and alarm services), persistence support, application support, and additional tools. The CSF is available in both Java and C++, and scripting is also supported using Jython.

The Cryo-NIRSP control software provides a Graphical User Interface (GUI) that shows all configurable parameters used and gathers the information from the user to run the desired observing task. The task information (i.e. parameters) are saved as a Data Set Parameter object (DSP). One or more DSPs can be contained in what is called an instrument Program (IP), which gives all the information required to run and track the observing task. It is this IP that is submitted from the GUI to the Instrument Controller (IC) which, via the Instrument Manager (IM), instigates the Instrument Sequencer (IS) to run the IP by calling a Jython script that calls out to the Java classes of the instrument control system.

At the start of an observing sequence, the IS processes a configuration sent by the Cryo-NIRSP IM. The IS will manage the forwarding of the appropriate configurations to other sub-controllers, namely the Mechanism Controller (MC), Polarising Modulator Controller (PMC), Time Base Controller (TBC), and the Detector Controller (DC), each of which will execute actions according to the configuration that is submitted to them. The MC is a controller for different types of mechanisms that use the Power PMAC connection. The PMC defines what the modulator must be able to do. The DC is the controller responsible for monitoring and controlling the Virtual Cameras (VCs), which include the SP Camera Controller and the CI Camera Controller. The TBC provides the start trigger indicating when to begin execution to all mechanisms, the PMC, and the cameras.

The IS uses a Jython script to run the two main phases involved in the IP processing, the setup and the execution. The setup phase is first carried out where the mechanisms (e.g. filters, slit, mirrors) are moved to the correct positions for the observing task. Instructions are also sent to the cameras (in the form of configurations) allowing them to perform their setup. The next phase is the execute phase, which is when the modulator is configured, the real time motion of any



IP Name: ip.id.1 Script: CoronalObs obsTask: observe
Description: Coronal Observation

Root (1) : DSP Sequence
cn-observe.35793.4161: Template DSP for Use case

Data Set Parameters Library

General Settings | Camera/Modulator Settings | Scanning Settings

CryoNIRSP Operation Mode
Spectrograph only Pickoff Mirror ○ [✓ Pellicle ●] Open ○

Misc. Parameters
SP cold mask: On ○ [✓ Off ●]
Feed relay Attenuation filter: Open
FM2 Defocused delta: 0

Calibration Lamp
Continuum (ThAr) ○ IR ○

Wavelength
Filter: Si IX
Wavelength: (nm) 3935
Order 14
Dispersion 9.6788233563126... nm/mm

Context Imager Filter
Wheel 1: Wheel 2:

Manual: ☐ TCS Cfg: ☑ Exp ID: engr-exp
Test: ☑ Save: ☐ OP ID: engr-op

[07:09:895 GMT] IP: 'atst.ics.cn.35793.4280' IP report: Starting execute

Figure 2: An example of the IP GUI. The top left shows the DSP currently loaded into the sequence. Bottom left displays template DSPs that can be loaded in from the library. The right panel show the *General Settings* panel displaying the inputs contained in the DSP. The other input panels can be seen on the top tabs: *Camera/Modulator Settings* and *Scanning Settings*. The very bottom section contains the *Submit* and *Abort* buttons along with the IP message panel to display the IP run status.

mechanisms moving during the observation execution (i.e. scanning) are configured, and the cameras are provided with a start time of when to begin the acquisition. Mechanisms, cameras, and modulator have to be perfectly synchronized to ensure no mechanism is moving while the camera is exposing, which requires precise timing calculations performed by the software. It also sends a start time to the TBC that interfaces with a time card to provide the trigger pulses. The next two sections will give more details of these processes by going through the general procedure of submitting and running an IP from start to finish.

USER INTERFACE

A GUI is provided to allow users to input their requirements for a calibration or observe task. The user is first required to select a task, which could be one of the many calibration tasks (e.g. dark, gain, focus) or could be an observe task (for scientific data acquisition). The subsequence display options and restrictions depend heavily on the task, hence this is the first choice to be made. From here the user must then decide what mode of operation they wish to use by selecting whether the observation is the SP only, the CI only, or both simultaneously. The user can then start inputting how they want their observing task to be carried out and hence create their own DSP or sequence of DSPs within the GUI. The control GUI is shown in Fig. 2. For the calibration tasks, there are also a set of templates that can be quickly loaded into the sequence where the parameters have already been defined to specify the configurations that must be run in order to calibrate the instrument. Multiple DSPs can be created and added to the DSP sequence, which will be contained in the IP when the user submits them from the GUI.

The GUI responds to a user's input by disabling, enabling, or changing the options of a field if certain configurations are no longer available given the current inputs. An example of this is the minimum exposure time allowed, as this value differs for the different camera modes. The GUI will automatically be updated if the camera mode is changed and the exposure time is no longer valid. Equally, different calibration tasks allow for different filters to be selected and this is also reflected in the GUI. The GUI also invokes limits on the input values. One of the main aims of the GUI is to allow a user to create DSPs which the GUI ensures will contain viable parameters that will run when submitted to the IS and will not fail the verification process.

CONTROL SOFTWARE

The DSP is the fundamental element needed by the IS to run the IP. The Jython script called by the IS has two

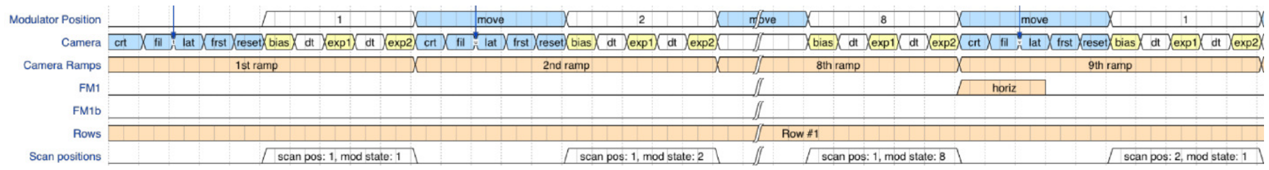


Figure 3: A timing diagram for an IP running the modulator in stepped mode and scanning the FoV. The camera setup times (blue in the *Camera* row) occur during the modulator move time (blue in the *Modulator* row). When the modulator is in a settled state (e.g. *1*), the camera exposes taking 3 frames including a bias. Each ramp is defined as the camera set up time and exposure. After completing all 8 modulator positions, the FM1 mirror performs a horizontal move (which takes less time than that already allowed for the camera setup so no addition time is required.). The bottom row indicates the scan position, so each scan position will contain 8 ramps.

phases; a setup phase and an execution phase. These are implemented by the Java class *IPWorker* (for the following this is referred to as the Worker).

Setup Phase

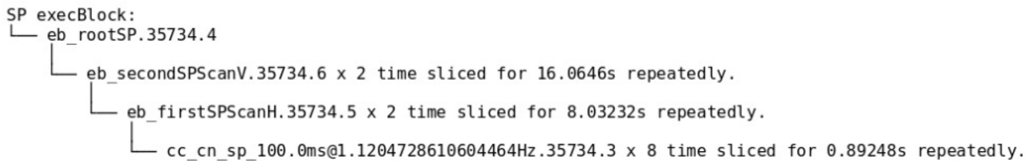
During the setup phase the IS interprets the DSP input parameters and verifies that the parameters are valid and also consistent. It will then begin to setup the mechanisms and cameras for the observing task. The setup phase iterates through all DSPs in the IP, performing a sequence of tasks for each and mapping to an object that can be accessed later during the execute phase.

The position of most mechanisms is defined by the user, for example the user will define the filter that they want or the slit size and the software will simply instruct each of these mechanisms to move to that position by submitting a configuration to the MC. In an observe task, other mechanisms, such as the focus mirrors, will have their positions defined by the results from the focus calibration task. All mechanisms are managed by the MC. The positions are submitted to this controller which then forwards the instructions on to the relevant sub-controller for each piece of hardware. This is a blocking process which means that no other operations will be performed by the Worker until the mechanisms are in the correct positions and have reported this back to the MC.

Next the Worker sets up the cameras by creating a Data Acquisition Tree (DAT), which is submitted to the CSS. DATs are described by a set of execution blocks (EBs) and camera configurations (CCs), which together describe the order in which the CCs are executed, the number of times they are executed, and the relative timings between. Depending on the observing mode either only the SP or CI is configured or if both are being run together in simultaneous mode then both must be configured. The key elements to configure for the cameras are the exposure time, the camera readout-mode, the associated exposure rate, and the number of nondestructive readouts.

The timing synchronization is primarily calculated and contained within the DAT and specifically within the EB. An EB consists of tiers, each of which contain the number of times to execute each tier and the time that a single execution of each tier should take (called a time slice). These timing calculations are fundamental in also calculating the modulator rate and the real-time motion configurations for the mechanisms, as one of the key requirements is that all mechanisms/modulator movement must be fully synchronized with the camera's exposure and wait times so that nothing is moving while the camera exposes. Figure 3 shows the timing diagram requirement for this setup and demonstrates how moves and exposures fit together. An example EB submitted to the CSS to reflect this is shown in Fig. 4. The initial parameter to consider is the exposure time, which is defined in the DSP and indicates how long each camera exposure should be. The next consideration is the time required for the camera to setup/reset between each ramp (see Fig. 3 for an illustration of what a ramp includes). In addition, depending on the modulator mode, an additional time must be considered in between each ramp for the modulator to move to its next position. However, in most cases the camera setup times are longer than the modulator move times and so the modulator can move while the camera performs its setup/resets. The final timing to consider applies if a mechanism is to be moved during the execution. In this case the camera will carry out N ramps (for all modulator states) and then the mechanism will move to its next position before the camera takes another N ramps and so on. The EB includes any extra time required for the mechanism(s) to move, which may be zero if the mechanisms can perform the movements within the time already allowed for the camera setup. During an observe task, both the horizontal and vertical scanning mirrors may move. This may require a mirror to reset to its original position which necessitates extra time and hence another tier to the DAT can be used to allow for this.

During the DAT construction, the timing calculations for the modulator are also carried out, as they can influence the final DAT values. Fundamentally this means calculating the rate that the modulator should step at and the wait time in between each move. In stepped mode, the modulator will step through 360 degrees at a given rate and at each position will wait for an amount of time. The step size is defined by the number of modulation states, which is 8 per 180 degrees for the Cryo-NIRSP. The modulator must be stationary (i.e. waiting) while the camera is exposing and then be performing its step when the camera is not. The modulator is the driving component here, as it is instructed at the beginning of the run to start stepping at a constant rate



```
SP execBlock:
└── eb_rootSP.35734.4
    │
    └── eb_secondSPScanV.35734.6 x 2 time sliced for 16.0646s repeatedly.
        │
        └── eb_firstSPScanH.35734.5 x 2 time sliced for 8.03232s repeatedly.
            │
            └── cc_cn_sp_100.0ms@1.1204728610604464Hz.35734.3 x 8 time sliced for 0.89248s repeatedly.
```

Figure 4: An example of the EB created for an exposure time of 100ms using the stepped mode of the modulator. The bottom tier contains the CC with the exposure time and rate. Each tier describes the number of times each node should be executed and how long a single execution of it should take. See main text in the Setup Phase section for details on this example.

and once the configuration defining this has been submitted then there is no other interaction with it. The camera and modulator must be synchronized, and while the camera can skip a modulator state, it must not be exposing part way through a move to the next. The implication of this is that all of the EB time slices must be some multiple of the modulator time (1/rate). For example, if there is an additional time required for a mechanism to move, then this exact amount of time cannot simply be added to the DAT tier as it would mean that the camera would become out of sync with the modulator. Instead a whole modulator timestep must be added, which may be longer than the required move time of the mechanism.

An example EB is shown in Fig. 4. In this example the exposure time is set to 0.1s. 0.79248s is required for the camera to setup/reset, which is more than the modulator move time of 0.110s and hence the total time for one ramp is given as the sum of the exposure time and setup time, 0.89248s, on the bottom tier of the EB. The modulator is used in stepped mode and so this tier is executed 8 times (one for each modulator state). The FoV is also being scanned in this example in a 2x2 pattern. The middle tier time slice incorporates time for this move. It states that the bottom tier should take 8 x 0.89248s plus some value which is the time allowed at the end of the 8 ramps for the first mirror to move. The mirror takes 0.7s to move, but because the modulator is in use, this added value must be a multiple of the time the modulator remains in a state (in this case it is 1/rate = 0.89248s), therefore 0.89248s is added. The first mirror will move to position 1 and then position 2 while the second mirror remains at position 1. Then in the next move mirror 2 will move to its second position and the first mirror will reset to position 1. The times for these moves are less than the time already allowed in other tiers and hence the top tier is simply 2 x 8.03232s.

The Worker has one more set of timing calculations to perform and that is to configure how the real-time motion of the scanning mechanism should be carried out. The configuration defining when and where the mechanism should move is comprised of a set of triplets, one set for each step. A triplet defines the position to move to, the amount of time the move should take (accounting for the the mechanism's speed limitation), and the amount of time to wait/remain at that position before executing the next triplet. The remaining time is composed of the amount of time that the camera needs to expose for and any padding time to allow for other mechanism movements, camera set up times, or modulator synchronization. The timing values are based on those calculated in the DAT/EB.

Execution Phase

Following setup, the execution phase begins. This will occur for each DSP in the IP in order. If there is more than one DSP and it is second or later in the sequence then firstly mechanisms are moved to their initial positions (this is already done in the setup phase for the first DSP in the sequence). Other than this, the execution phase mostly involves submitting the final configurations to the mechanisms/cameras telling them when to execute their respective configurations.

Firstly, the configuration of scanning triplets (calculated in the setup phase) are submitted to the MC (and then on to the sub-controller) as a custom motion program to be executed by the Delta Tau Power PMAC when a trigger is received. The trigger will be sent from TRADS at the same time that the camera execution starts. At this point the Power PMAC will implement the custom program and carry out the real time motion.

The global start time of the execution is calculated and submitted to the TBC. It is also submitted individually to the cameras and the modulator. The start time is calculated to be 15 seconds into the future which allows time for the cameras and modulator to accept and implement the configurations before the start time. The modulator configuration also contains the rate, wait time, number of states, and the starting modulator state. This means that the modulation state of the first ramp is known and so the state of all other ramps can be calculated from these variables. The final submit is to the cameras which are given the time to begin running the DAT previously loaded. Metadata information on the observing task is also submitted and will be attached to each frame that the CSS sends out. This is important for both real time data processing (described in the Data Handling System section) and data processing off of the summit.

The progress of the cameras and real time motion program of the mechanisms is monitored (see section below) during the execution and through to the completion. The IS will return a message to the GUI indicating the successful completion of the IP.

Monitoring

The IS provides constant monitoring of its controller throughout both the setup and execution phases. The GUI status panel is updated initially with information about the IP ID and DSP ID for reference. Details on the camera setup, including the camera mode and rate, are also reported once the Worker has received and processed the DSP. During the running/execution of the IP, the number of scan positions and the number of frames captured are also reported regularly to indicate progress.

After the actual execution of the DAT and real time motion begins, a separate thread is created and used to constantly monitor the status of the camera(s) and mechanisms that are running. It checks whether they have completed and, if so, whether they completed with success or error. If an error occurs with one of these components, then the Worker will call an abort on all the other running components and send a failed status to the GUI. The thread will only run for the amount of time that the DAT should take to complete, and if the completion notice is not received from the components within this time then it will call an abort and fail the IP.

Error Catching

The main error that can occur is if one of the subcomponents is unresponsive or experiences an error. To ensure that such errors are caught and that the IP is safely aborted, all configurations that are submitted are done so with a timeout so that if the success status is not received in a given time, an abort will be called and the IP execution halted.

THE DATA HANDLING SYSTEM

The DKIST framework also provides tools and components to manage the handling of the data that is transported from the CSS to the data handling plugins. The DHS is comprised of multiple components including a Raw Probe which receives the buffers from the CSS, the Data Processing node which subscribes to the Raw Probe to receive the buffers and can perform calculations to insert extra metadata information (e.g. the modulation state) which are then published and picked up by the Data Storage node which stores the processed data. Finally, there is a detailed display (DD) component which processes the data to either display quality assurance (QA) measures in the DS9 application [5] or, in the case of calibration tasks, produce the outputs required from these tasks to setup the instrument. Each camera (SP and CI) has its own data processing line, as shown in Fig. 5.

The Detailed Display

If a task is labelled as a calibration task in the metadata, then the DD is responsible for processing the buffers received from the CSS and obtaining calibration parameters or calibration frames that are required for setup and further tasks. It collects all of the buffers from the CSS and runs them through specific Python plugins to process them. For example, a focus task will scan through different focal values to determine what the best focus position is. The DD collects

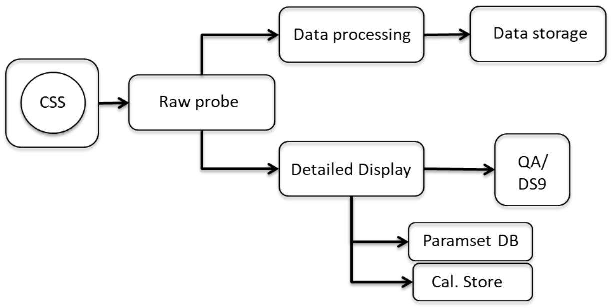


Figure 5: Each camera (SP and CI) has its own processing line. The CSS publishes buffers which the Raw Probe subscribes to. The Data Processing component and the Detailed Display subscribe to the published buffers from the Raw Probe. The Data Storage picks up the buffers from the Data Processing component and stores them. Meanwhile the Detailed Display publishes buffers to DS9 and stores any required results to the Parameter Set Database or Calibration Store.

all of the buffers and all the focus mirror positions and passes them to the focus plugin. A Python script then analyses the buffers to determine which position represents the best focus. This value is then returned to the DD which saves it in a Parameter Set database, labelled by the list of instrument components that affect the focus, for future access.

Other calibration tasks have a different output in the form of calibrated master data frames which are saved to the calibration store. For example, the Dark task involves collecting all the buffers and running them through the Dark Python plugin to produce a master dark. This will be used on other calibration and observe data to correct for detector imperfections.

Not only will the DD produce the calibration results, but it also publishes the raw and/or reduced data frames to a DS9 display window. DS9 is a tool for displaying images and FITS files and can be run from the command line. It has been built into the DKIST DHS. This is the QA aspect which allows the user to examine the data coming out of the instrument.

For the observation tasks, the DD processes the incoming buffers and reduces the data to produce images used for QA. Two Python plugins are used to achieve this. The first collects a full ramp and then applies dark, gain, and linearity corrections. The output reduced data is displayed on a panel in the DS9 window. Meanwhile, buffers will still be coming in and ramps will continue to be collected and reduced. When the plugin has a full set of modulation states (i.e. 8 sets of ramps which have been reduced), it performs a demodulation that produces the Stokes images I, Q, U, and V. Each of these is also output to a display panel within the DS9 window so that the full set can be examined. An example output of the DS9 window using simulated data is shown in Fig. 6.

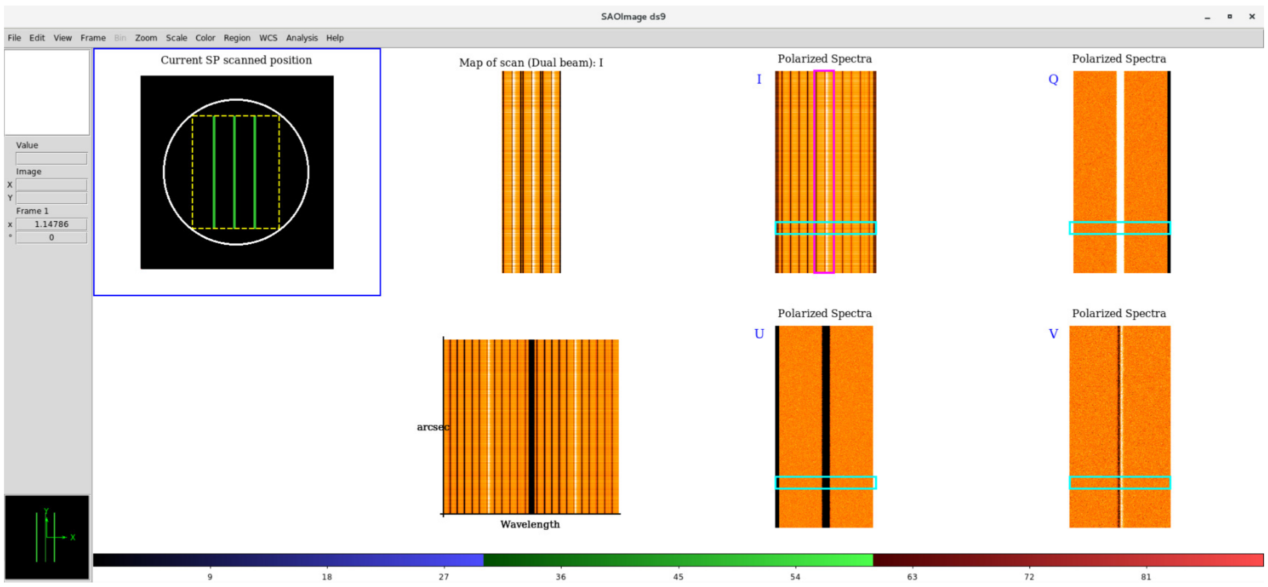

SAOImage ds9
File Edit View Frame Bin Zoom Scale Color Region WCS Analysis Help
Value
Image
Frame 1
1.14786
0
Current SP scanned position
Map of scan (Dual beam): I
Polarized Spectra
I
Q
U
V
arcsec
Wavelength
9
18
27
36
45
54
63
72
81

Figure 6: An example DS9 display from an observe task using the Spectrograph with simulated input data. Top row: from left to right shows the current scan position (here green means complete), a composite image collecting the region outlined in pink from the I image for each scan positions, the demodulated stokes I image, the demodulated stokes Q image. Bottom: left to right shows one of the reduced ramps (dark, gain and linearity corrected) before demodulation, the demodulated stokes U image, and the demodulated stokes V image.

Monitoring

The monitoring of the DD processing plugins is particularly important for the calibration tasks. This is because the tasks must produce a result (e.g. a value or calibrated frame) otherwise they are deemed a failure (even if the camera and mechanisms complete successfully). For this reason, the IS also monitors the status of the DD during and after the execution phase. Again a separate thread is used to listen for events sent from the DD. If a failed status is received while the DAT is running then it will abort the cameras and mechanisms in the observation and report the error to the GUI. The processing plugins can take some time to run, especially when the dataset is large, and so it is possible that the DAT has already completed. In this case the GUI status panel will indicate to the user that it is waiting for the DHS to complete the processing and the thread will continue to check the status.

CONCLUSION

The Cryo-NIRSP control and data processing system has been developed in line with the DKIST CSF to facilitate the complex design and operation of the instrument, which will allow it to meet its science goals. The software that has been developed is very versatile and can handle the different observing modes, observing task requirements (calibration and observe), camera readout modes, scanning operations and modulator configurations; these provide the Cryo-NIRSP with its unique observing flexibility and hence vast scientific capabilities. The data processing software has also been developed in conjunction with the control software to feedback processed data and display QA images. The combination of both provides a full end-to-end solution and, on completion, will allow the Cryo-NIRSP to function to its optimum ability.

ACKNOWLEDGEMENTS

We are grateful for the general support of the Principle Investigator, Jeff Kuhn.

REFERENCES

[1] J. R. Kuhn, I. F. School, D. L. Mickey, ‘Solar Dark Matter and Dark Energy: How can CryoNIRSP Help?’ in Proc. of Astronomical Society of the Pacific Conference Series 463, Magnetic Fields from the Photosphere to the Corona, p.207, Dec, 2012.

[2] C. Berst, ‘Camera Systems Software Functional Interface Specification’ , NSO, Tucson, USA, SPEC-0098, July. 2017.

[3] S. Guzzo et al., ‘Common Services Framework Reference Guide’, NSO, Tucson, USA, SPEC-0022-2, Oct. 2016.

[4] J.R. Hubbard and E.M. Johansson, ‘A standard framework for developing instrument controllers for the ATST’ in Proc. of the SPIE 8451, Software and Cyberinfrastructure for Astronomy II, article no. 845100, Sept. 2012, doi:10.1117/12.926283

[5] DS9, `http://ds9.si.edu/site/Home.html`