REPORT DOCUMENTATION PAGE

AFRL-SR-AR-TR-03-

0264

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	DATES COVERED (From - To)
11-06-2003	Final	Sep 2000 - Mar 2003

4. TITLE AND SUBTITLE
Advanced All-sky Imaging Photometer for Ionospheric Research

5a. CONTRACT NUMBER
F49620-00-C-0030

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Ning, Peter

Eather, Robert, H

Lance, Cyril

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Scion Associates
439 Fillmore Street
Port Townsend, WA 98368

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
USAF, AFRL
AF Office of Scientific Research
4015 Wilson Boulevard, Room 713
Arlington, VA 22203-1954
'rogram Manager: Major Paul J. Bellaire, Jr.

10. SPONSOR/MONITOR'S ACRONYM(S)
AFOSR/NM

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Distribution Statement A.
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

20030731 053

14. ABSTRACT

The All-sky Imaging Photometer (ASIP) has been an invaluable tool in studying ionospheric processes by characterizing the evolution of plasma structures. ASIP measurements complement the repertoire of existing ground-based and satellite-based sensors needed to study our complex ionosphere by providing estimates of height, gradient scales, and morphological signatures to these multi-scale plasma structures. They can take the form of polar cap sub-visual arcs and polar cap patches at high latitudes which often requires high temporal resolution (seconds) or they can be slow time scale (1-2 minutes) effects such as equatorial F-region bubbles found at low latitudes. A dual-mode (image intensifier mode or bare-CCD mode) ASIP has been developed for ionospheric researchers to investigate these phenomena and, unlike its predecessors, provides them the option to optimize between temporal resolution and image quality. Imager sensitivity down to tens of Rayleighs has been achieved by applying the latest innovations in image intensifier, CCD, and narrow-band filter technologies.

15. SUBJECT TERMS
all-sky imaging photometer, optical ionospheric data, image processing software, optical calibration

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Peter Ning
U	U	U	SAR	~~170~~ 164	19b. TELEPHONE NUMBER (include area code) 408-530-9751

Advanced All-sky Imaging Photometer for Ionospheric Research

Peter Ning
Robert H. Eather
Cyril Lance

Scion Associates
439 Fillmore Street
Port Townsend, WA 98368

11 June 2003

Final Report
15 September, 2000 – 14 March, 2003

AIR FORCE RESEARCH LABORATORY
Air Force Office Of Scientific Research
AIR FORCE MATERIEL COMMAND
4015 Wilson Boulevard, Room 713
Arlington, VA 22203-1954

Advanced All-sky Imaging Photometer for Ionospheric Research

Table of Contents

Figures

Tables

Chapter 1: Overview

1.1 Introduction

The All-sky Imaging Photometer (ASIP) has been an invaluable tool in studying ionospheric processes by characterizing the evolution of plasma structures. ASIP measurements complement the repertoire of existing ground-based and satellite-based sensors needed to study our complex ionosphere by providing estimates of height, gradient scales, and morphological signatures to these multi-scale plasma structures. They can take the form of polar cap sub-visual arcs and polar cap patches at high latitudes which often requires high temporal resolution (seconds) or they can be slow time scale (1-2 minutes) effects such as equatorial F-region bubbles found at low latitudes. A dual-mode (image intensifier mode or bare-CCD mode) ASIP has been developed for ionospheric researchers to investigate these phenomena and, unlike its predecessors, provides them the option to optimize between temporal resolution and image quality. Imager sensitivity down to tens of Rayleighs has been achieved by applying the latest innovations in image intensifier, CCD, and narrow-band filter technologies.

1.2 What Was Accomplished

This ASIP was built using the new Gen III image intensifier tube, a 16-bit 1300x1340 CCD, and 16Å notch filters. The instrument was installed and operated at Ny Alesund, Svalbard for use by AFRL scientists and collaborating researchers during two key campaign periods in December 2001 and in February 2002. Real-time ASIP data was provided over the internet to assist with the interpretation of data from other instruments, including the EISCAT radar, by identifying and locating the existence of polar cap patches and arcs as well as the boundaries of the auroral oval. AFRL scientists at Ascension Island also used this technology in March 2002 to monitor F-region bubbles and its impact on DoD communication and navigation systems.

1.3 Why It Is Important

The ASIP is unique as it provides time continuous two-dimensional view of plasma processes in the overhead ionosphere. As a basic research tool, scientists can more accurately analyze and specify the behavior of ionospheric processes from optical signatures both in the high temporal and high spatial (image clarity) domains and more often than not, fills the measurement "voids"

attributed by other sensors. ASIP data can be used to validate or, with certain cases predict, the onset of scintillation effects to both UHF and L-Band transmissions. As an optical diagnostic tool, ASIP was designed to keep the Air Force in the forefront of ionospheric science as well as providing a better understanding of the ionospheric impact to operational DoD systems.

1.4 Camera Side View & Data Example

1.4.1 Image Intensifer Mode

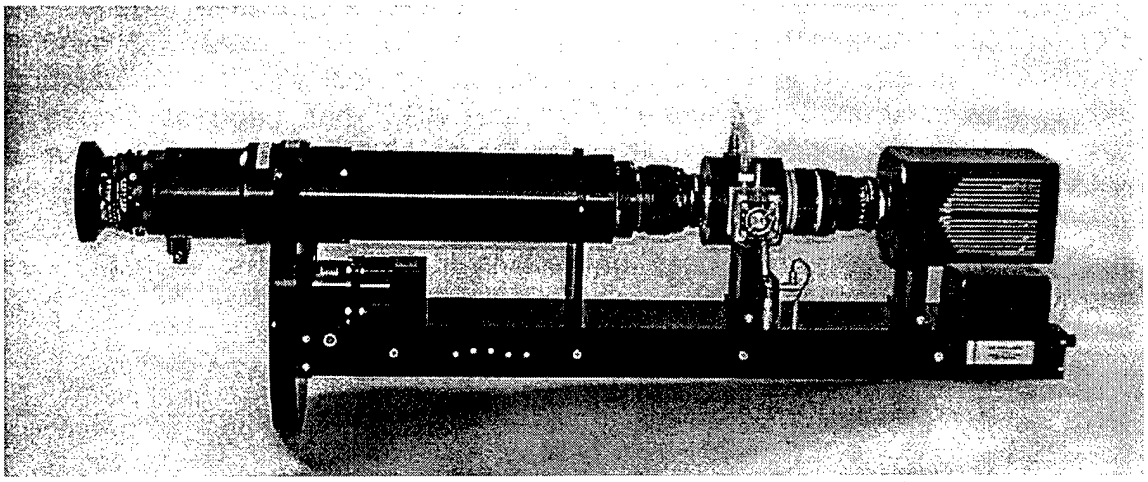


Figure 1-1 ASIP Side View - Intensifer Mode

Image intensifier mode ASIP provide researchers with an instantaneous sky map of ionospheric dynamics. This 0.5 second exposure in Figure 1-2 shows a polar cap patch migrating towards the auroral oval to the south. Plasma structures such as these affect transionospheric radio transmissions.

Figure 1-2 Intensifer Mode Data Example – Auroral Patch

1.4.2 Bare CCD Mode

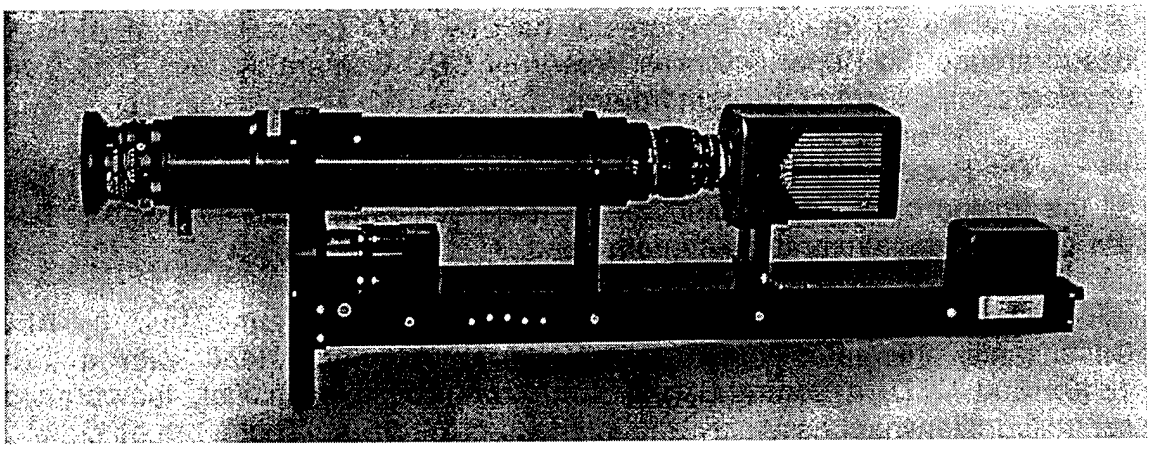


Figure 1-3 ASIP Side View - Bare CCD Mode

Bare-CCD mode ASIP (without image intensifier) help identify regions where line-of-sight radio disruptions from communication and navigation satellites can occur. This 2 minute exposure in Figure 1-4 shows detailed bifurcated depletion edges (dark band) formed by equatorial spread-F bubbles.

Figure 1-4 Bare CCD Mode Data Example - Spread-F Bubbles

1.5 Core Features of the ASIP System

The All-sky Imaging Photometer built by KEO Consultants for AFRL is based around the Roper Scientific VersArray CCD Camera system. The basic physical components of the camera are:

The Roper Scientific VersArray ST133 CCD Controller Unit. This contains all the analog control and computer interface electronics and connects to both the computer system and the camera head.

The Removable-Intensified Camera system. This unit, built by KEO Consultants, integrates the Roper VersArray Camera Head with a EEV 1340x1300 CCD, a 25mm Gen-III *removable* image intensifier, temperature-controlled filterwheel assembly and interface electronics. These are all coupled optically and include an all-sky lens in the front of the camera. The electronics to control the intensifier, shutter and filter assemblies are interfaced to the computer system via one RS-232 cable at the rear panel of the camera chassis.

This imager has the capability to acquire data using the Gen-III intensifier tube with fast, 14-bit conversion *OR* just using the bare CCD using slow, 16-bit conversion.

Computer support for this camera uses RedHat Linux 7.1 and the Roper LINUX PVCAM drivers version 2.5.8. In addition, LINUX support for the SMARTMOTOR control interface (RS-232 serial) is also supplied.

Documentation is provided for these systems. The user should familiarize themselves with the Roper Scientific CCD camera system documentation. In addition, the Animatics documentation will describe the filterwheel servo control system and the RS-232 interface that controls the intensifier, shutter and filterwheel systems.

1.6 Brief Description of the ASIP System

The Auroral Imager consists of an all-sky lens that is imaged telecentrically onto a six position 4" filterwheel containing narrow-band interference filters (typically about 20 Angstroms wide). Because the interference filters are temperature dependent (drifts are typically on the order of 1 Angstrom / 5 degrees C), a stand-alone temperature controller is provided with the instrument (ATHENA X16) to keep the temperature of the filters at approximately room temperature, or about 25C. The temperature can be controlled remotely via a standard RS-232 serial interface.

The filter image is then re-imaged either onto the front of a Gen-III type image intensifier or directly onto the EEV 1340x1300 scientific-grade CCD chip. Gain control of the image intensifier allows controlling the gain. The image on the back of the image intensifier is then re-imaged onto the CCD of the VersArray CCD camera head. The camera head is thermo-electrically cooled to provide a decreased dark-noise accumulation and allow longer integration times. The CCD camera head is connected to the ST133 camera electronics unit which supplies power, timing signals and the data interface both to the camera head and back to the HOST computer. The ST133 has capability of dual-speed conversion. When using the image intensifier, the fast conversion should be used, and when using the bare CCD, the slow conversion (16 bit) should be used. The VersArray camera head has an internal shutter directly in front of the CCD window which is used for the exposure time.

A shutter at the front of the instrument is used as a "capping" shutter (sunlight directly hitting the interference filters can damage them). A light detector at the front of the instrument is used as an automatic over-ride to avoid over-exposure and possible damage to the image intensifier. A standard RS-232 interface is used to control the filterwheel, intensifier and shutter operations.

Chapter 2: Camera Optics

2.1 Bare CCD Operation

From the front lens of the camera to the rear, the optical components for bare CCD operation are as follows:

Front primary lens (Mamiya 37mm/F4.5 with 180° field-of-view). This lens should be focused at ∞ and the F-stop should be set wide-open at F4.5, and the in-built filter wheel set at SL-1B.

Telecentric optics. (These lenses (100 mm Dia., 2 x plano-convex) are between the shutter and the filterwheel, and ensure that the image at the filter is telecentric. This is necessary for narrow-band interference filters as the transmission wavelength is dependent on the incident angle. At F4.5, the half-cone angle of light passing through the filter is 6.3 degrees.

Re-imaging optics: Consists of two achromats, with diameters 100 mm (behind filter wheel) and 72 mm (on Canon camera lens), and a camera lens. The telecentric image at the filter (~ 92 mm dia.) must be re-imaged onto the VersArray camera CCD detector (image dia. ~ 24 mm). The Canon camera lens (85 mm/F1.2) mounted to the CCD camera must have its F-stop set all the way open (F1.2).

Note: The reduction of the 92 mm/F4.5 image to 24 mm requires the camera lens to have a F number of F1.2 or less:

$$(92/24)^2 = 14.7 = 3.9 \text{ F stops, } = \text{F4.5 to F1.2}$$

A field-curvature corrector lens is mounted as the entrance window to the VersArray camera.

2.2 Intensifier Operation

An Image Intensifier Assembly may be inserted into the system if desired. This consists of an ITT 25mm Gen III (Ga-As) Intensifier (ultra-blue cathode) mounted in a custom TE Cooler Chamber, with re-imaging optics to the CCD camera.

The same Canon 85mm/F1.2 lens is used on the front of the Image Intensifier Assembly, and forms a ~24mm image on the 25mm Image Intensifier cathode.

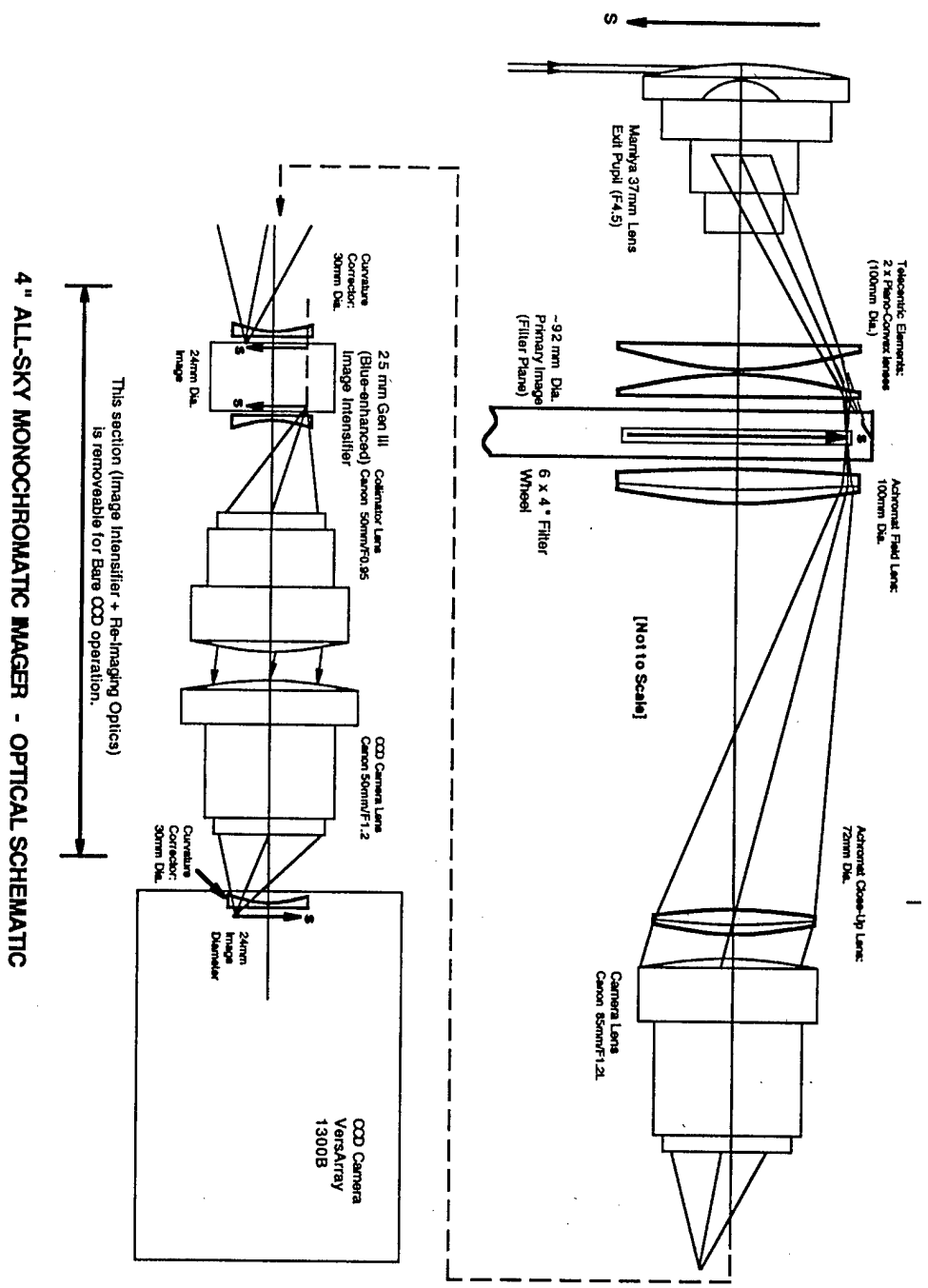


Figure 2-1 Optical Schematic

The Custom Cooler cools just the cathode of the intensifier (by ~20^{o}C), and so reduces thermal noise by an order of magnitude. The output phosphor is not cooled, so as not to compromise the phosphor lag-time characteristics.

The image on the back of the image intensifier must be re-imaged (1:1) onto the CCD in the VersArray camera head. The collimator lens that takes the image from the intensifier output phosphor is a Canon 50mm/F0.95 set wide open at F0.95 and at infinity focus, and the camera lens on the VersArray camera head is a Canon 50mm/f1.2, which should have it's *f*-stop set all the way open at F1.2. The Canon lens is used for focus.

As there is a field curvature corrector lens mounted permanently in the VersArray camera, a compensating lens is mounted in the Cooler Chamber near the Image Intensifier output phosphor, so that the re-imaging optics is corrected.

An optical schematic for this configuration is shown in Figure 2-1.

The camera can be focused in the lab by setting the Mamiya lens focus at ∞ for far objects (>10 feet) or to the actual distance for nearer objects, and then adjusting the Canon 85mm/F1.2 lens until best focus is achieved, either directly onto the CCD, or onto the Image Intensifier. When the intensifier is used, the Canon 50mm/F1.2 lens on the VersArray camera is used to then focus from the rear of the intensifier. It may be necessary to iterate back and forth between focus at the intensifier and the CCD to obtain best focus. Both lenses should be able to adjust the image 'through' focus, so that one can definitely determine the optimum focus settings.

KEO has determined through experience that the best way to critically focus the camera is by looking at a star field. Once the camera is set up to look at the night sky, remove one of the filters and take short white-light exposures to get good star images. One can also do this by taking longer exposures using the narrow-band filters, but the process will take longer.

2.3 Mounting the Intensifier Assembly

A proportionally scaled mechanical drawing is show in Figure 2-2.

The U-Channel Upright holding the VersArray camera may be taken off via two 1/4-28 screws. The Canon 85mm/F1.2 lens is removed from the VersArray Camera and mounted to the front of the Image Intensifier assembly. The Image Intensifier assembly then mounts using the same hole pattern. The Canon 590mm/F1.2 lens is then mounted on the VersArray Camera, and the camera remounted on the U-Channel at the rear of the Image Intensifier assembly, using the other set of 14/28 mounting holes.

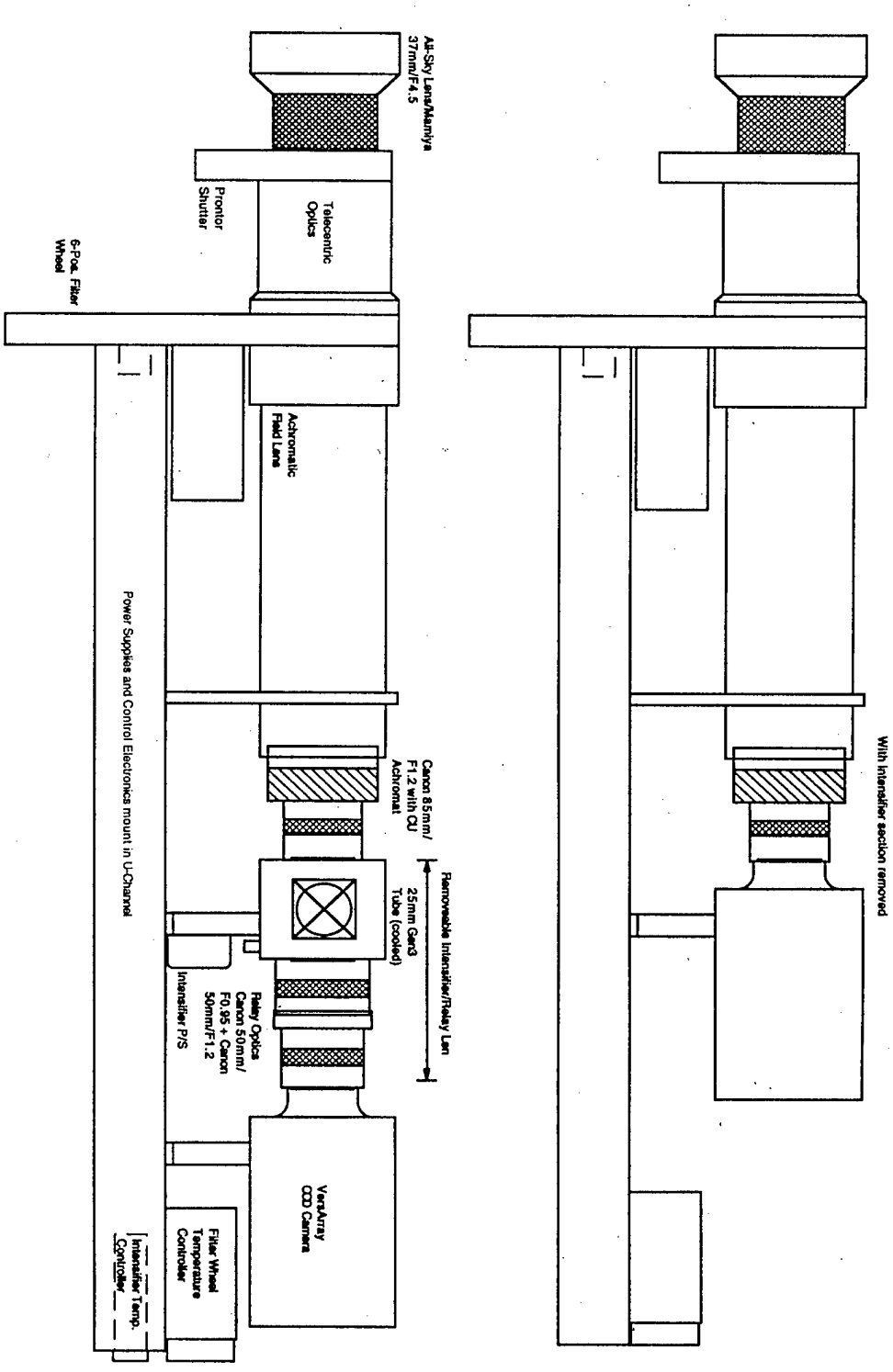


Figure 2-2 Scaled Mechanical Drawing

2.4 Removing the Primary Lens

The primary lens may be removed from its housing by turning the large clamping ring so the red dots align. The telecentric optics and shutter assembly can also be unscrewed from the filter wheel.

2.5 Removing and Adding New Filters

To remove and add new filters to the filterwheel, simply remove the front optics from the system by unscrewing the optics assembly attached to the filterwheel housing. The telecentric optics, shutter and fisheye lens will all unscrew from the filterwheel housing in one piece. The filters are accessible through the front cover and there are three retaining screws to hold them in place. A piece of scotch tape applied lightly to the filter can help pull them out of the filter housing.

2.6 Field Curvature and Focusing

All multi-element systems require some compromise when designing with off-the-shelf optical components. The telecentric and reimaging optics results in some field curvature at the final image, so that the optimum focus for stars at the center of the image is slightly different than the optimum focus for stars near the edge of the image. To correct for this, a curvature-corrector lens is mounted within the Roper CCD Camera, at a distance of about 2mm in front of the CCD.

KEO recommends optimizing the focus using stars at about 45° elevation so that the zenith and the horizon are as little out of focus as possible.

Note too that the optics is color corrected for the visible region (450-700 nm), and there may be a slight change in focus if working in the near infrared, necessitating a compromise focus be set between the visible and the near-ir.

2.7 Instrument Mounting

Mounting holes are provided on each side of the U-channel base (1/4-20 thread). The instrument should be mounted so that the front of the primary lens is near to the center of curvature of any plexiglass or glass domes used. (If the primary lens is too close to the hemispherical surface, distortions and focus problems may result.)

2.8 A Brief Discussion of Useful Calibration Procedures

As one becomes familiar with using this instrument, there will be several calibrations that you will want to perform to further understand the instrument's characteristics and to be able to use the data quantitatively. A list of recommended calibrations is presented here.

Temperature Controller Calibration: (Refer to the ATHENA Temperature Controller Manual). Once the instrument is installed, the ATHENA controller can be programmed to display in either °F or °C. An autotuning feature will determine the best PID loop parameters to accurately control the filterwheel temperature. A set point around room temperature should be selected and noted in you lab notes. *Keo Consultants* has autotuned your filterwheel system, but this might need to be redone when the final field installation is completed.

Once you have the camera set up and working in the lab, it will be a good exercise to calibrate the intensifier gain settings. Using a constant light source, take a well exposed image at maximum gain (3), and get an average of it's illuminated area and an average of that area with the light source turned off. Set the intensifier gain and adjust the appropriate potentiometer (VR3) on the intensifier control board until the value of the illuminated area minus dark image is half of that at maximum gain. Do the same for gain-=1 and set the gain=0 for minimum gain and calibrate that with respect to maximum gain. Save these values for future calculations.

Calibrate the light sensor sensitivity setting (VR4 on the control board) to switch the intensifier power on at an ambient light level roughly equivalent to civil twilight (or a solar depression angle of 6°).

Use the calibrated light source that you use to calibrate your other photometric instruments to take long exposures at the different filter wavelengths to get a spectral calibration and overall quantitative calibration of the instruments. You will need to use the filter curves and the spectral response of your light source to correctly calculate these values.

Get vignetting curves for the optics in the camera. Using a constant light source, take exposures with the source at equidistant positions from the front camera element, but at different elevation positions. Compare the output from these different exposures at the same exposure time to get a vignetting curve for the optics. This will be important to "flatten" out the measurements of structures that span large parts of the field of view.

Calibrate the center and radius of the image with respect to CCD pixels to get a spatial calibration of the instrument. The best way KEO has found to do this, is to take a bright image that clearly illuminates the edge of image. Image

processing algorithms can also help to bring out this edge. Measure about 10 points around the edge of this image in CCD pixels, and then use different sets of three points from this data set to calculate an average radius and center.

Take a few bias images with the CCD fully cooled down, and get an average statistic for the bias at different gains. It is a good idea to check this value for consistency over time to check that the camera electronics have remained calibrated.

It is a good idea to take some long dark noise exposures at the two different camera gains and calculate what the dark noise contribution is.

Although all these calibrations will take a lot of time and care, KEO recommends doing them to get a full understanding of the capabilities of your instrument and to prepare you data for good quantitative analysis. These calibrations will pay off in the long run when it comes time to analyze the data you will take with this instrument.

Chapter 3: Hardware

3.1 Overview

This section will discuss the hardware systems supplied by KEO Consultants to control the All-Sky Intensified Camera system built for ARFL.

3.1.1 Camera System

The camera system is built around the VersArray air-cooled camera head. This interfaces to the ST133 camera controller via one interface cable. You should refer to the Roper Scientific documentation for more information on this system.

3.1.2 Computer Interface

The Roper Scientific ST133 camera controller is interfaced to the computer via a PCI standard interface card supplied by Roper. This card controls the camera system and reads the digitized CCD pixel data from the controller directly into the computer's RAM. Again, for more information on this hardware, please refer to the Roper Scientific documentation.

3.1.3 Initial Hook-up

- The camera system has a power cord that connects to the back of the camera chassis. Plug this into 110VAC.
- Connect the 30' DB9 cable (M) to the back of the DB9 connector (F) on the back of the camera chassis.
- Connect the 30' DB9 cable (F) to the DB9 connector labelled CAM (M) on the DB9 Y-cable supplied with the instrument.
- Connect the DB9 connector on the DB9 Y-cable labelled FW (F) to the RS-232 port (e.g. COMM1) that you are going to use to control the camera systems.
- Connect the DB9 connector on the DB9 Y-cable labelled TEMP (F) to the RS-232 port (e.g. COMM2) that you are going to use to control the temperature controller (ATHENA XT16).
- Connect the Roper Scientific Camera up following the instructions supplied with the imager.

3.1.4 Filterwheel, Intensifier, and Shutter Interface

These are the control systems that KEO has built for your camera systems. Schematics and wiring documentation are included in Chapter 7:Hardware References of this manual but a functional description will be included here.

Control for these instrument functions are provided via an RS-232 port. Either port can be used, but the software must correctly initialize the port connected. The RS-232 interface connects with a Servomotor system connected to the filterwheel on the camera -- the SMARTMOTOR built by Animatics in California. The SMARTMOTOR system has an embedded processor with built-in EEPROM memory and is used to control the filterwheel system as well as the intensifier and shutter control systems.

KEO has built special hardware to interface between these two components. The advantages of this design are user flexibility to expand and re-program the system to meet new needs and the convenience of only having one small RS-232 interface cable to control the whole instrument. Because of the standard RS-232 interface, any terminal program, or programmed RS-232 communications can control the instrument.

The ANIMATICS temperature controller also uses an RS-232 interface to communicate with a remote HOST. Both of these RS-232 signals are brought out of the imager through one cable via a DB9 connector. KEO has supplied a RS-232 junction cable to separate out these two RS-232 interface systems at the HOST. To use both systems, you will need two available serial RS-232 ports at your HOST computer.

For more information on the SMARTMOTOR and it's programming language, please refer to the Animatics SMARTMOTOR manual. In addition, the SMARTMOTOR system comes with it's own development system that provides an easy way to view and modify the programs stored in it's EEPROM memory.

3.2 Filterwheel

The filterwheel is controlled by a servomotor system supplied by ANIMATICS called the SMARTMOTOR. It is a controller, servo-amplifier, encoder and motor all in one that fits the standard form factor of a 23T stepper motor. The SMARTMOTOR is controlled by one 1.25A 24VDC power supply. An RS-232 interface is used by the SMARTMOTOR to communicate with a host computer and other SMARTMOTORS.

The filterwheel has a Hall-Effect Sensor built into the housing and a rare earth magnet built into the actual wheel that is used as the HOME sensor. Upon

start of the filterwheel control program, the filterwheel moves slowly to the HOME sensor to calibrate the encoder position. From there on, an incremental encoder is used to accurately determine the filterwheel position. At any time, a new HOME calibration can be made if deemed necessary. The HOME detector is interfaced to the SMARTMOTOR through it's I/O connector and uses Index pin A. The schematic for the position sensor can be found in Chapter 7: below titled SS440 Digital Position Sensor Data Sheet.

In addition, the filterwheel is temperature controlled. An ATHENA XT16 temperature controller is provided at the rear of the instrument to control the temperature of the filters. Because the wavelength of the filters shifts with temperature it is important to maintain the filters at approximately room temperature, or somewhere around 25°C. Two AC heater pads are mounted symmetrically on the inside of the filterwheel to maintain this temperature. The table for the temperature sensor used can be found in Chapter 7: below tilted RTD Chart.

A manual for the ATHENA temperature controller is provided with the instrument. Using the setup controls, the display can be adjusted to read in °C or °F, and the filterwheel should be auto-tuned when finally mounted, so that the temperature controller can optimize the control feedback parameters used.

Finally, the SMARTMOTOR uses an I^2C serial interface to control external devices. KEO has provided an I^2C parallel I/O board for this purpose. It is called the DIO-116 board and provides 8 bits latched TTL output, and 8 bits TTL input via this card. This board sits as a daughter board on the KEO Intensifier/Shutter Control board and provides the interface to these devices.

The DIO-116 board is set up to use two adjacent port addresses and can be adjusted using the jumpers A1 and A2 providing four separate addresses. These boards can then be hooked up in series allowing the capability for 32 bits of latched output and 32 bits of input. This camera uses one DIO-116 module and it's address is set to 0 using the ports:

PORTA (Output) PORTB (Input)

3.3 Intensifier/Shutter Control Board

The Intensifier/Shutter Control Board built by KEO provides a flexible and expandable interface circuit to control two shutters and the image intensifier. In addition there are four auxiliary signals for additional functions to be added to the interface.

The schematics for this board can be found in Chapter 7: below under the title of Intensifier/Shutter Control Board Schematics Rev B. To accommodate the new Gen-III tubes that require a different HV power-supply, a daughter board was designed. The schematic title for this is called Gen-III Intensifier Gain Control Daughter Board.

Note: A replacement circuit board has been designed and built specifically for the Gen-III version of the Intensifier/Shutter Control and does not need the daughter board option. This has not been installed yet, but the schematics are included in this manual under Gen-III Intensifier/Shutter Control Board Schematics.

3.3.1 Power

The only input power required by the board is 24VDC coming in on P1. Actually, this voltage can be adjusted up to 40VDC and down to 15VDC (in order for the 12V regulator to work). The starting voltage required by the shutter determines this voltage. Voltage regulators on the board derive all other voltages needed for the board.

In this particular system, 24VDC is used to control the filterwheel motor and used as an opening voltage for the Melles-Griot shutter supplied with the system. A 24VDC power supply is mounted inside the camera chassis and supplies this power.

The Control Board creates three different voltages:

5V -- used for the CMOS circuitry on board (U9)
Vsh -- used as the holding voltage for the shutters (U8)
12V -- used for external components (U10)

All three of these voltage regulators can handle up to 1.5A maximum.

The 5V supply can be supplied by either the 5V regulator on board, or can be brought in via the parallel interface on P2 (pin 20). If the on-board regulator is used, then J5 should be installed, and if the external 5V supply is used J6 should be installed. *Under no circumstances should both J5 and J6 be installed at the same time!!!*

Vsh is used to create a holding voltage for the shutters and is set by adjusting the potentiometer VR5. This value should be set at about 1.5V above the desired holding voltage, as there are two diode drops that occur before the voltage reaches the shutter coil.

12V is provided on the board for the convenience of the user in the event that some device is added to the system that requires 12VDC. It has been used in other systems to power RS170 CCD cameras, but has no immediate function in this camera.

3.3.2 Parallel I/O

The Intensifier/Shutter Control Board has an 8 bit In/8 bit Out parallel interface coming in on P2. The schematics can be found in Chapter 7: below. The IN/OUT definitions are from the DIO-116/SMARTMOTOR's point of view. Thus an Output for the DIO-116 is really an input for the Intensifier/Shutter Control Board. Figure 3-1 shows the I/O definitions for the Control Board. Their functions are described here.

Shutter Control Signals: SHTR1* and SHTR2* are negative going pulses that control the shutter outputs. On the initial down-going transition of these signals, a 50 msec pulse (determined by R1*C1 or R2*C2) switches the 24VDC onto the shutters. For the remaining duration of the low state of SHTR1*, the voltage set by Vsh is switched onto the shutters. A longer input pulse can be set by changing the R1*C1, R2*C2 combinations if more input power is necessary to open the shutters.

Note: The SHTR1 channel is over-ridden by the state of the Light Detector and so there will no shutter driver output unless the Light Detector determines that it is *dark enough* to safely open the shutter. SHTR2 has no such disable function.

STROBE*: The strobe pulse is a negative going pulse that clocks in the intensifier gain data into the input latches on U5. The signals REM:GAIN0 and REM:GAIN1 determine the gain setting for the image intensifier and should be set before setting the STROBE* signal low. The new gain state is clocked in on the negative going transition.

Intensifier Power: The intensifier power can be switched on and off remotely by this signal. A low state turns the power off, and a high state turns the power on. To actually have power applied to the image intensifier, the light detector must be in a valid state (dark enough for safe operation of the intensifier tube). There is no way to override this detector as long as it is connected to the Control Board.

DIO-116	Intensifier/Shutter Control Board
OUT0	SHTR1*
OUT1	SHTR2*
OUT2	STROBE*

OUT3	REM:INT_POWER
OUT4	REM:GAIN0
OUT5	REM:GAIN1
OUT6	OUT:TTL0
OUT7	OUT:TTL1
IN0	GAIN0
IN1	GAIN1
IN2	INT_POWER:STS*
IN3	LIGHT_DET*
IN4	SH1:STATUS
IN5	SH2:STATUS
IN6	IN:TTL0
IN7	IN:TTL1
5V	5V
GND	GND

Figure 3-1 Intensifier/Shutter Control Board I/O Definitions

REM:GAIN0 and REM:GAIN1: These are the bits that determine the gain state of the image intensifier. Valid values are thus 0 to 3 and adjust the intensifier gain over its full range, which is usually about 8:1. Gains are usually adjusted so that from maximum gain, each gain below this reduces the output by a factor of 2 (much like the f-stop in a camera lens).

OUT:TTL0 and OUT:TTL1: These are two undefined signals that can be used to control additional devices that may be added to the system at a later time. They are available on P3, which provides an interface via ribbon cable or daughter board configuration to an expansion circuit.

GAIN0 and GAIN1: These bits read the current status of the intensifier gain and send it to the interface device. These bits are used to read the current intensifier gain to verify that it is set correctly. This feature is especially important when used with a 'Manual Control Panel'.

INT_POWER:STS*: This signal gives feedback to the actual current running through the image intensifier and thus give independent confirmation that the intensifier power is actually on. A 5Ω resistor (R11) is hooked up in series with the ground signal of the intensifier and the voltage across this resistor is amplified at U7.A to give an indication of intensifier current. Because this signal is buffered with an additional Schmidt Inverter (U1.F), the signal is active low. A HIGH signal indicates that there is no intensifier power present, and a LOW signal indicates that there is intensifier power present.

LIGHT_DET:STS: This signal reads the inverted value of the light detector amplifier (U7.B). This signal is used to enable or disable the intensifier power

circuit. Thus, a HIGH signal indicates that it is dark enough to operate the intensifier, and a LO signal indicates that it is too light to operate the intensifier. This signal must be adjusted in the field to meet the requirements of the instrument. KEO usually advises that the intensifier power is enabled at a solar depression angle of around 6° (civil twilight). This adjustment is made by changing the value of the potentiometer,VR4, until the light detector switches states.

SH1:STATUS and SH2:STATUS: Some shutters such as the one supplied with your instrument have status switches installed in them. The Intensifier/Shutter Control Board provides the means to read the status of these switches and report them back to the host computer. This feature is useful to check that the shutter is working reliably. Unfortunately, KEO's experience in the field has found that more often than not, the shutter status switch will fail to open reliably even when the shutter is opening correctly. It is up to the users of this instrument to determine whether this status signal is a useful feature or not. In the hope that someday reliable shutters will be a reality, KEO provides this feature!

IN:TTL0 and IN:TTL1: These two undefined inputs to the host can be used to read additional devices that can be added to the system at a later date. They are available on P3 which provides an interface via ribbon cable or daughter board configuration to an expansion ciruit.

5V and GND: GND must be connected between the Control Board and the Host. The 5V signal is optional. 5V for the control board can be derived from the onboard voltage regulator (U9), or from the host interface. Jumpers J5 and J6 must be set accordingly as described above under the POWER section. In some rare cases, the 5V derived on the Control Board might be used to power an interface card through this parallel I/O port. In this case, both J5 and J6 may be installed.

3.3.3 Shutter Driver

There is provision for two shutter drivers on the Intensifier/Shutter Control Board. As described in the Parallel I/O section, both shutters are controlled by ACTIVE-LOW signals. A mono-stable vibrator (U2) creates a 50 msec pulse which is used to momentarily switch the 24VDC power to the shutter. This gives the shutter a powerful starting pulse to open it fully. After 50 msec, only the Vsh voltage is applied to the shutter providing a lower voltage holding current through the shutter coil.

The shutter circuit is set up for your camera to open at 24VDC and hold at about 8VDC which is typical operation voltages for the Modified PRONTOR shutter provided. To get 8VDC on the shutter coil, Vsh must be adjust to about

9.5VDC as there are two diode drops between the voltage regulator and the shutter coil.

The provision for manual operation at a remote control panel is provided via P3, but if no control panel is connected, the jumper J1 must be installed to provide for REMOTE-ONLY operation. Your system was delivered with J1 installed.

In some cases (as in the Prontor E40 shutter), shutters are designed to operate on a opening and holding voltage of 24VDC. The Control Board can be modified slightly to accommodate this case by removing U2 and shorting out pins (SH1) 4 and 6, (SH2) 12 and 10. This will switch in the starting voltage (24V) for the duration of the ACTIVE-LOW signal.

Shutter status capability is provided on the Control Board to accommodate shutters that have micro-switches mounted in them to signal a fully opened shutter. These switches can be interfaced to the host via buffers (U1.C, U1.D) and are ACTIVE-HIGH (i.e. they are high when the switches are closed/shutter is open). In addition, an LED can be brought out to the manual control panel via P3 to give a physical indication of the shutter status. *The modified PRONTOR shutter on your system does not have status feedaback.*

3.3.4 Intensifier Circuit

The intensifier gain can be controlled remotely by the host or from the manual control panel via P3. Four gain settings are set via the potentiometers VR1 - VR4. Gains can be calibrated by using a constant light source and adjusting the values of VR1 - VR4 so that their values cut the gain by 1/2 for each setting. The maximum dynamic range of gain adjustment for image intensifiers is usually on the order of 8-14.

Remote gains from the HOST must be clocked into a latch using the STROBE* signal. DATA is latched on the negative going transition. To program this interface, set the next data state with the STROBE* signal high, then set the STROBE* low and high again. This will clock in the next gain state.

For remote-only operation, J3 should be installed which keeps the input latch selected on the remote data rather than allowing manual selecting of remote/manual gain settings.

3.3.5 Intensifier Power

Intensifier power can be controlled remotely by the host via the REM:INT_POWER line. This signal is ACTIVE-HIGH: and LOW state disables

the intensifier power, and a HIGH state enables it. Intensifier power will only be enabled when BOTH the POWER state is HIGH and the LIGHT_DETECTOR state is high. This provides a safety mechanism for protecting the image intensifier tube which can be damaged easily.

Manual control of the intensifier power is available through P3, but J4 should be installed if the Control Board is going to be used in a remote-only configuration.

Current through the intensifier is monitored and returned via the INT:STATUS bit. Because this is buffered through a Schmidt Inverter for interfacing purposes (U1.F), this signal is ACTIVE-LOW: i.e. the intensifier has power when this signal is LOW, and does not have power when the signal is HIGH.

This status bit is useful in determining if the Intensifier circuit is working correctly and to confirm whether or not the Light Override has disabled the intensifier.

3.3.6 Light Detector

A photo resistive light element is supplied with the camera and is mounted in a LEMO connector hanging off the front of the camera. This detector can be mounted anywhere in the dome or taped to the lens of the instrument. It should be in position to easily detect changes in ambient light level.

The photo resistive element is connected to a voltage follower (U7.B) and subsequently buffered through a Schmidt Inverter. This signal can be read by the host through the LIGHT_DETECTOR signal. A HIGH signal means that the ambient light level is dark enough to allow safe operation of the image intensifier, and a LOW signal means that there is too much light for safe operation.

The trigger point of this circuit is determined by adjusting the value of the potentiometer VR4. KEO recommends setting this potentiometer to enable the image intensifier around civil twilight (or solar depression of 6°).

Chapter 4: Software

4.1 Overview

The software platform to operate ASIP is based upon the Linux operating system. We chose this mainly for its reliability, cost, and built-in network features. The available camera driver software at the time only supported the Red Hat 7.1 distribution of Linux. The application software was developed and coded in C for maintainability and portability. This executable program, called piAuto, only requires two text files for its operation. One file, param.txt, describes the configuration of the filters and the desired sequencing of those filters, in a one-minute acquisition cycle. The other file, schedule.txt, describes the start and stop times for unattended operation. The acquired image data are written to hard disk as data files in both raw scientific grade format and in quick browser compatible JPEG format. Detailed analysis of the raw image data is available with the GUI-based IDL program allsky.pro. Perhaps the most salient feature is the ability to control and access real-time ASIP data when a network connection is available.

4.2 Running the Acquisition Software: piAuto

4.2.1 Login and Using X-Window

As with any Linux or UNIX-based operating system, the usual procedure is to login and launch X-windows, then interact with the computer via shells and other GUI displays. This manual assumes that the operator has some familiarity with Linux and X-Windows. Our Red Hat 7.1 distribution utilizes GNOME as the X-windows manager and GNOME encapsulated bash flavor shells. Those not familiar with this environment can source a wealth of information from the internet. At the completion of software system boot-up the user will be prompted with a GUI-based login and password prompt. For our system it is "root" and "aurora" respectively. This will automatically launch the X-windows display manager. Technically, the acquisition software piAuto can run from any single bash shell and without X-windows. The advantage of having multiple bash windows is you can manage other required applications such as text editors to change the configuration and schedule files. You can use emacs, vi or any other installed text editor derivatives. This is a matter of personal preference and the choice is left up to the operator. So with the X-windows display manager opened, we require one bash shell opened by clicking on the gnome-terminal icon in the shape of a barefoot. From this shell you can spawn many other displays.

4.2.2 Managing the Display

X-windows provide many ways to display the operation of the camera during data acquisition. We will suggest here how one can setup up the display to monitor the status of the acquisition program piAuto and the display of various wavelengths in real-time. This is meant for real-time viewing and operation while the user is in front of the imager computer. Network or remote operation will be discussed later. At a minimum a bash shell window is available to run the piAuto program. But before starting the acquisition, the user will want to setup displaying the raw imager data sorted by filter. This is accomplished by displaying the latest filter specific PGM files generated by the piAuto program. PGM files can be displayed using the shareware program "xv". In the polling mode it automatically updates the screen when the PGM file is updated. Figure 4-1 shows a text script file that generates image displays for all five available filters. Specifially, it places the 6300Å filter in the upper right corner and the 5577Å filter in the lower right corner. The other three filters, 4278Å, 7774Å and 6560Å, initially come up iconified on the bottom task bar. It is left up to the user to click on its icon and open and position the display for these three filters. Of course at anytime during the course of the acquisition, the user can manipulate these individual filter display windows. This script can be launch from the opened bash shell window prompt by typing:"./root/camctl/display.sh".

```
#!/bin/bash

# This file: display.sh

xv -poll -geometry +675+0   6300.pgm &
xv -poll -geometry +675+350 5577.pgm &
xv -poll -iconic 4278.pgm &
xv -poll -iconic 7774.pgm &
xv -poll -iconic 6560.pgm &
```

Figure 4-1 Filter Display Script: display.sh

The typical screen display might look like Figure 4-2. The two primary wavelengths of interest, 6300Å and 5577Å, are placed to the right by default. An optional wavelength is displayed in the top center. Then the primary bash shell window is stretched horizontally to accommodate wider text line displays. Inside this bash window the piAuto program can be launched. As the program is running, information for each acquisition is displayed and scrolled off screen as the data collection progresses. What can be displayed in the available display areas could be a large viewable clock or a small editor window for either the param.txt or schedule.txt files. Alternatively, you can move the current display

screen "off-screen" and work on something else in another screen. This may include view other relevant real-time data using the Netscape browser.

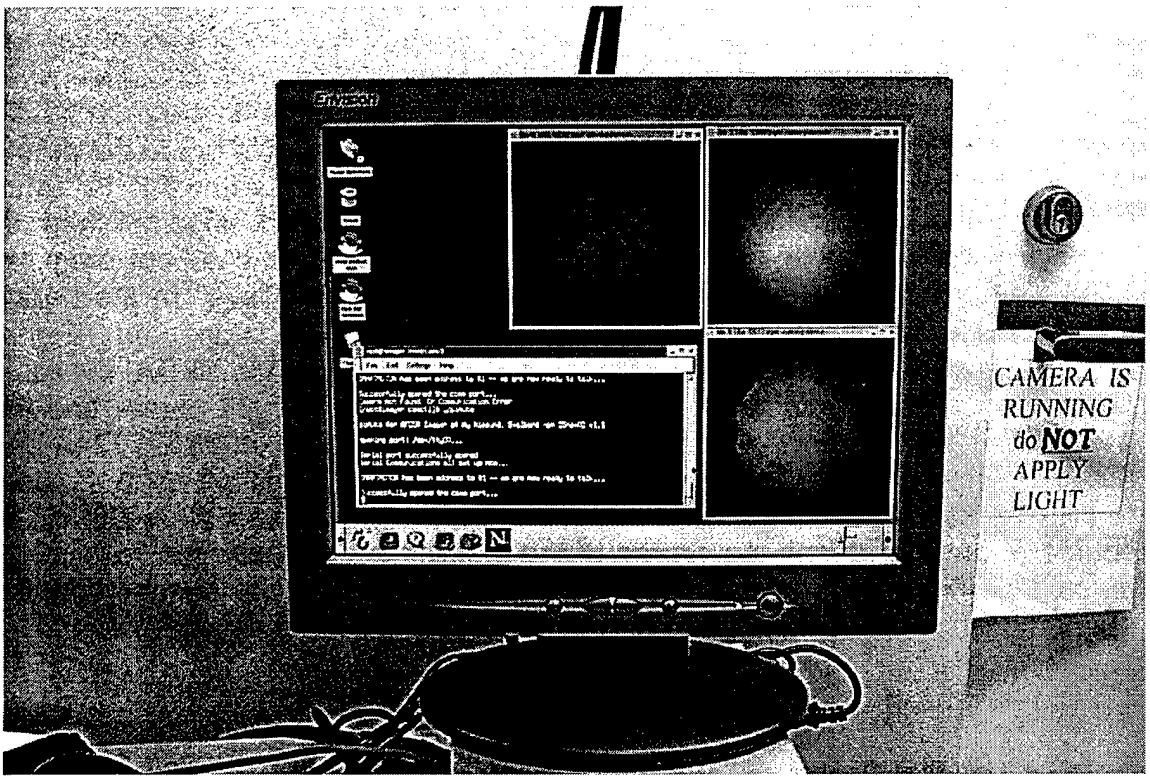


Figure 4-2 Typical Screen Display

4.2.3 Configuring Camera Configuration File: param.txt

The parameter file is located under the subdirectory /root/camctl and is an editable text file read by the piAuto acquisition program. The first line of the file designates three characters as a location or experiment reference. The second line specifies the filter wavelength sequentially placed in the filter wheel, starting with position number 1. A "9999" designation is to used specify no-filter in that position. Following these first two required lines are individual lines that each represent a single image acquisition entry referenced as a time offset within a 60 second cycle by the number in the first column. The second column specifies the filter position to use. The third and fourth columns represent the initial intensifier gain and exposure setting, respectively. The intensifier gain ranges from a minimum of 0 to a maximum of 3. The exposure time units are entered as 0.1 seconds. Figure 4-3 is an example of our typical routine mode configuration. We cycle all five available filters sequentially at 12-second intervals, all with gain 3 intensifier and 3 second starting exposures. This will be automatically adjusted

for each entry in the next minute's cycle. "NYA" designates Ny Alesund and note that filter position six has no filter.

```
NYA
6300 5577 4278 7774 6560 9999
 0 1 3 30
12 2 3 30
24 3 3 30
36 4 3 30
48 5 3 30
```

Figure 4-3 Routine Acquire Example of /root/camctl/param.txt

Another mode commonly used is when a high time resolution is required. In this case several frames of the same filter are acquired within the 60-second cycle. Figure 4-4 is an example of covering high time resolution for 5577Å but requiring less frequently 6300Å frames. The 6300Å frames occur on the minute and on the half-minute, while the 5577Å fills the rest of the 60-second cycle at 5 second intervals. So every minute we have a total of ten 5577Å and two 6300Å images.

```
NYA
6300 5577 4278 7774 6560 9999
 0 1 3 20
 5 2 3 30
10 2 3 30
15 2 3 30
20 2 3 30
25 2 3 30
30 1 3 20
35 2 3 30
40 2 3 30
45 2 3 30
50 2 3 30
55 2 3 30
```

Figure 4-4 Fast Acquire Example of /root/camctl/param.txt

Any sequence can be accommodated as long as enough time is allocated to move the filter wheel (normally up to two seconds) after the exposure time has elapsed, otherwise the next entry or entries will be skipped if the actual time exceeds the indexed offset times.

4.2.4 Configuring the Schedule File: schedule.txt

The schedule file is located under the subdirectory /root/camctl and is an editable text file read by the piAuto acquisition program. This file contains a start and stop time for each line and must be entered chronologically. The number of start and stop time lines is effectively unrestricted, up to the array dimensions specified in the piAuto.c program listing, but at minimum, there should be at least one line.

Each line has two time-date pair columns. The first two columns represent the time-of-day and date for the start time and last two columns represent the time-of-day and date for the corresponding stop time. Time-of-day is in hhmmss format and date is in yymmdd format. Figure 4-5 shows three lines of a schedule.txt file representing three start and stop time pairs. Using this schedule, the acquisition starts on November 22, 2002 at 15:11:00 UTC and stops at 22:06:00 UTC the same day. The next acquisition period starts on November 29, 2002 at 13:48:00 UTC and stops on November 30, 2002 at 01:23:00 UTC. The last acquisition starts on November 30, 2002 at 12:2000 UTC and stops on December 1, 2002 at 05:00:00 UTC.

```
151100 021128 220600 021128
134800 021129 012300 021130
122000 021130 050000 021201
```

Figure 4-5 Sample Schedule File with 3 Start and Stop Entries

4.2.5 Executing and Monitoring piAuto

The acquisition program piAuto can be started if both the param.txt and schedule.txt files have been edited properly. To launch the application, you must be in a bash shell under the subdirectory /root/camctl. To start the program, type "./piAuto". Be sure to place the dot and slash character before "piAuto". At this point everything should be automated. The operator requires no further action unless he needs to stop the program or change either the param.txt or schedule.txt files. Details of the program execution will be displayed on the bash shell output screen and are described below.

The program first initializes the CCD camera hardware and waits till the CCD temperature drops below –25C. It will display the current temperature as it waits to reach this threshold temperature. Then the program proceeds to read the schedule.txt file followed by the param.txt file. If the current time is within one of the schedule acquisition periods, then it will begin acquiring images, otherwise it will display a message "Waiting for Start Time: xxxx" and readout the instantaneous CCD temperature continuously until the next available start time

has been reached. When a stop time has been reached it will proceed to wait for the next available start-stop period from the schedule table and repeat for each period until it has reached the end of the schedule table. The piAuto program terminates when it displays "Finished List - Terminating Program!"

Information pertaining to each frame acquisition will be displayed instantaneously in the bash shell script as piAuto cycles through all the filter sequences. Figure 4-6 shows a typical screen output for a single frame acquisition. The first line describes the parameter settings and data output filename. In this example, the "30:" is the minute cycle offset in seconds from param.txt, "F: 6300" is the filter, "G: 2" is intensifier gain 2, "E: 20" is exposure time of 2 seconds, and the filename is "021218_042030_6300_NYA.keo. The next two line prints the the average and standard deviation statistics based on all the image pixel counts in that particular acquisition frame. The upper and lower numbers are the three sigma bounds from the average and the status describes whether the image is "UNDEREXPOSED", "OVEREXPOSED", or "NO CHANGE NEEDED". The last line displays the gain and exposure adjustments for the next minute cycle acquisition and the current CCD temperature.

```
30: F: 6300 G: 2 E: 20 – 021218_042030_6300_NYA.keo
Average: 12301.43  Std Dev: 4361.86
Upper: 16371  Lower: 8324  Status: UNDEREXPOSED
Computed Adjustment GAIN: 2 E:  35.75 Temp: -40.00
```

Figure 4-6 Typical Run-time Image Acquisition Information

Should it become necessary for the operator to terminate the program manually, he can type the program abort sequence, "Control-C" key in the active bash shell. This should put you back at the shell prompt. You can resume operation by re-running "./piAuto".

4.2.6 Modifying and Compiling piAuto.c

The source code for the piAuto acqusition program is written in the C programming language. This code has evolved over the years and continues to be modified as needed. The source file piAuto.c can be found in Chapter 6:below.

```
# makefile
# Note: this makefile expects to find libpvcam.so in the standard library
# search path, and both pvcam.h and master.h in the standard include search
# path.
```

```
MOTIFHOME = /usr/X11R6
LIBS    = -L${MOTIFHOME}/lib \
        -L${OPENWINHOME}/lib \
        -lpvcam \
        -ljpeg \
        -lm

INCLUDE = -I$(MOTIFHOME)/include -I$(OPENWINHOME)/include
OPTIM = -O2
#OPTIM = -g
CC = gcc
FILES.h= master.h pvcam.h complxw.h

CFLAGS = $(OPTIM) $(INCLUDE)

PHOTO_OBJECTS = piAuto.o SerialLib.o

piAuto : ${PHOTO_OBJECTS}
        gcc ${OPTIM} -o piAuto ${PHOTO_OBJECTS} ${LIBS}

SerialLib.o : SerialLib.c
        gcc ${OPTIM} -c SerialLib.c

%.o: %.c $(FLAGS.h)
        echo 'making $@'
        $(COMPILE.c) $<
```

Figure 4-7 Makefile to Compile piAuto.c

Figure 4-7 shows the contents of the "makefile" needed to automatically compile and link the source file piAuto.c with the associated driver and library files. All one needs to type to create the new executable piAuto is "make" in the subdirectory /root/camctl. This will most likely be necessary as many parameters such as binning and location coordinates are currently hard coded in piAuto.c. These should be moved out of the source code and into readable text files read in by the piAuto program. Expanding param.txt would be a good place start.

4.3 Data Output Files

The piAuto program generates two data output files for each image acquisition and places them all under a predefined directory and subdirectory called "/imagedir/yymmdd", where yymmdd varies with year, month and day respectively. Each file name has the date, time, filter, and a three-letter

identification encoded in the following format: yymmdd_hhmmss_ffff_iii.ext, where ext is either "jpg" or "keo". These extensions describe the file type. For example, the file "021218_235955_6560_NYA.keo" is a raw data file for an image that was acquired December 18, 2002 at 23:59:55 UTC using a 6560Å filter at Ny Alesund. The corresponding JPEG file would be "021218_235955_6560_NYA.jpg".

The typical file sizes in bytes are summarized in Table 4-1. Each pixel digitized from the CCD is a 16-bit unsigned integer ranging from 0 to 65535 and takes up two bytes. Note that the JPEG file sizes will vary based on the actual pixel values whereas the raw file sizes are fixed and not compressed. The former is meant as an instantaneous means to quickly lookup images without any

Binning	Horizontal Dimension	Vertical Dimension	.*,jpg JPEG File (Typical)	*.keo Raw File (Actual)
1 x 1	1340	1300	~200000 bytes	3484000 bytes
2 x 2	670	650	27500 bytes	871000 bytes
4 x 4	335	325	~5000 bytes	217750 bytes

Table 4-1 Data File Sizes

special software other than a web browser while the later provides the raw unaltered data for further analysis and manipulation.

4.3.1 Low Resolution JPEG files

These low-resolution JPEG files take little disk space and provide an excellent quick image lookup without any special analysis or processing software. The JPEG compression algorithm is applied to the high order byte of each pixel value (top 8-bits of 16-bits). As you can see the morphology of the aurora in Figure 4-8 is preserved quite well using just 256 shades of gray. The orientation is mirror flipped along the vertical axis, that is left is east and right is west and the top is north and the bottom is south when the camera itself is aligned such that the top of the image is pointing true north.

Figure 4-8 Sample JPEG Image: 030208_051512_5577_NYA.jpg

4.3.2 Raw Data Files

The raw data files are used for scientific analysis and contains the unaltered pixel values. They do take a considerable amount of disk space depending on the binning mode. Like most image file formats, the rows are stored from top to bottom which is opposite of how the y-axis is oriented in Cartesian coordinates. For our images we have developed an embedded parameter scheme where one can decode information related to this image in the image data itself. The first few pixels have been designated for this purpose since they would otherwise be useless as they are outside of the optical field of view. We summarize these parameters in Table 4-2. The number of parameters can be expanded as necessary to accommodate new parameters. Each parameter, unlike a real pixel count, is encoded as a two-byte signed integer value ranging from –32768 to 32767. Care should be taken, as the byte-order can be different on different computer platforms. This information is encoded in piAuto.c, the source file for the acquisition program, piAuto. Decoding can be found in the various IDL routines that is use for both calibration and analysis.

Number	Name	Byte Offset	Example
0	Year	0	2002
1	Month	2	12
2	Day	4	18
3	Hour	6	23
4	Minute	8	10
5	Second	10	55
6	Filter	12	6300
7	Gain	14	2
8	Exposure (0.1sec)	16	25
9	Lat-Deg	18	78
10	Lat-Min	20	55
11	Lat-Sec	22	24
12	Lon-Deg	24	348
13	Lon-Min	28	4
14	Lon-Sec	30	12
15	Altitude (m)	32	63
16	CCD Temp (0.01C)	34	-4100

Table 4-2 Image Parameters Header Information

4.4 Using IDL to Analyze and Display Data

The software tools to analyze the RAW data files with the extension ".keo" are written in IDL (Interactive Data Language). This commercial data analysis and visualization package from Research Systems, a Kodak company, is being use by our research collaborators and us. Our collaborators contributed and modified several pieces of the code as it has evolved over the years. As the code was written to accommodate several different types of imagers, we have modified its content to support our particular ASIP image formats.

Use of the IDL software tools assumes familiarity with installing and running IDL. This is a cross-platform package that has been originally developed under Linux, but can run under Windows as well. Care should be taken about handling system differences, such as byte order and display drivers such as X-windows vs. Microsoft Windows.

4.4.1 Procedures for Image Annotation and Animation

The core procedures for both image annotation and animation can be found in the IDL source listing, asiptools.pro found in Chapter 6:below. Routines used for calibration is also included in this listing but discussed later. The main ones are described here.

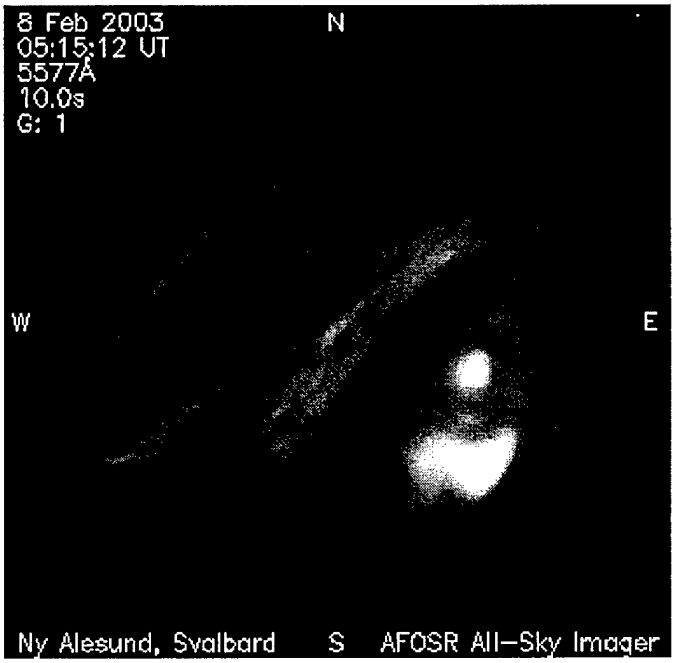


Figure 4-9 Annotated Image Display Example

The procedure, "tvImage" take a *.keo filename and displays the image with parameter annotations. IDL allows you to save the image in various output formats, such as GIF, TIFF, JPEG, and PNG. One caveat is that IDL version 5.3 and later no longer support GIF format. Figure 4-9 shows an annotated image saved as a PNG file. Note that the image orientation is flipped along the vertical axis and properly labeled.

Probably the most powerful way to analyze the temporal dynamics of ASIP data is with animation of a sequence of static images. This is accomplished with the routine "makeMpeg". It takes two arguments. The first is a text list of image files that make up a sequence of images to be analyzed (usually of the same filter). The second argument is the MPEG movie file name that will be created. This routine effectively takes a sequence of annotated images and builds it into an MPEG file. This file can be view by any MPEG movie player software, but one with single-step forward or reverse and frame rate control is preferred. One shareware Linux version recommended is the package

"MpegTV". Sample MPEG files of ASIP data generated with makeMpeg can be retrieved from "http://www.fys.uio.no/~ning/movies".

4.4.2 Interactive Analysis Using GUI-based ALLSKY.PRO

A very powerful interactive GUI-based IDL application has been ported for use on ASIP data. This tool allows the researcher to manipulate and display the ASIP data interactively and project calibrated data over geographic coordinates. The interface is fairly intuitive but some of its key features will be described here. The source code can be found in Chapter 6:below. Hardcopy Postscript file can be generated by this application.

The single panel screen displaying raw data is show in Figure 4-10. When the "QUICK VIEW ON" button is "ON" selecting the *.keo file in the File List slider window will display that file will all the relevant parameters above and below the image. The white background does not represent the edge of the field of view but rather a clipping of the lower 1/256 scaling of pixel intensity. This mode allows the user to quickly select and view individual image frames for further analysis. This is often called viewing the "untransformed" all-sky images. It does not take into account any calibration adjustments.

The second type of view accomplishes all the "magic" when the "QUICK VIEW ON" button is deactived and the Graph Xwindow is selected. Here the Altitude, Scale, and Viewing Altitude values are used. The Altitude is the assumed altitude that you want to project the data down onto geographic coordinates. It is often associated with the emission height of the desired phenomenon. Figure 4-11 shows the same image transformed from 250km. The observers view is set at 300km. Note that the edge is cut off at 20 degrees above the horizon. Projecting and transforming pixel data below this value is undesirable.

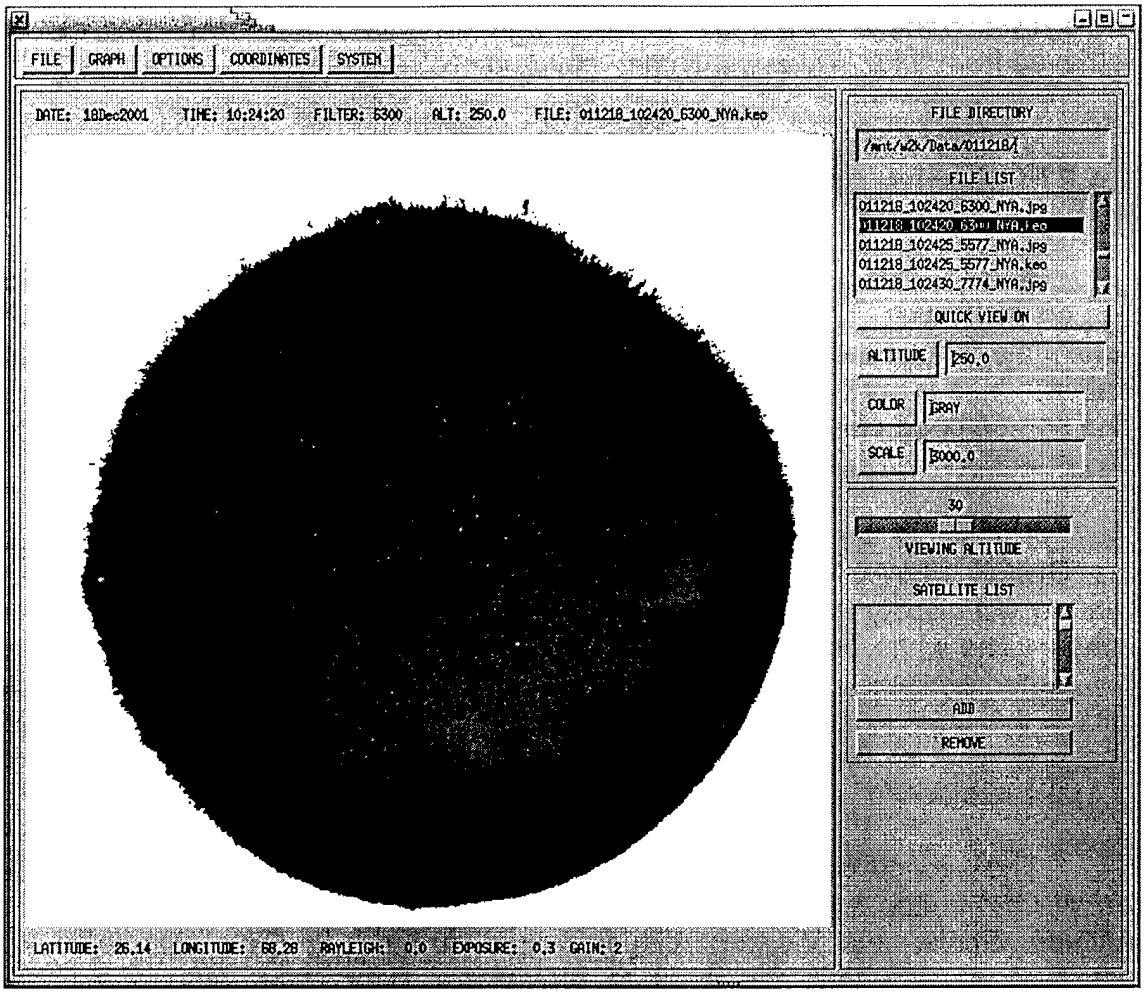


Figure 4-10 Untransformed Display using ALLSKY.PRO

The Scale value is used to scale the calibrated pixel intensity in terms of Rayleigh units. Points of interest in the transformed area can be selected using the cursor as it updates both the position in latitude and logitude coordinates and the Rayleigh count when moving over the display. The image can be saved as a Postcript file for printout or publication. There is also a provision in the code to overlay satellite passes so other sensor data can be compared with the ASIP data. All the results of the calibration process have been incorporated into allsky.pro and will be described further in the Chapter 5:below. We welcome suggestions and feedback from users so we can expand on this application by adding more features for analysis and visualization. Perhaps what is needed immediately and is currently being worked on is the ability to track specific features and measure velocities. It can be done by manually by comparing the difference in time and changes in latitude and longitude coordinates.

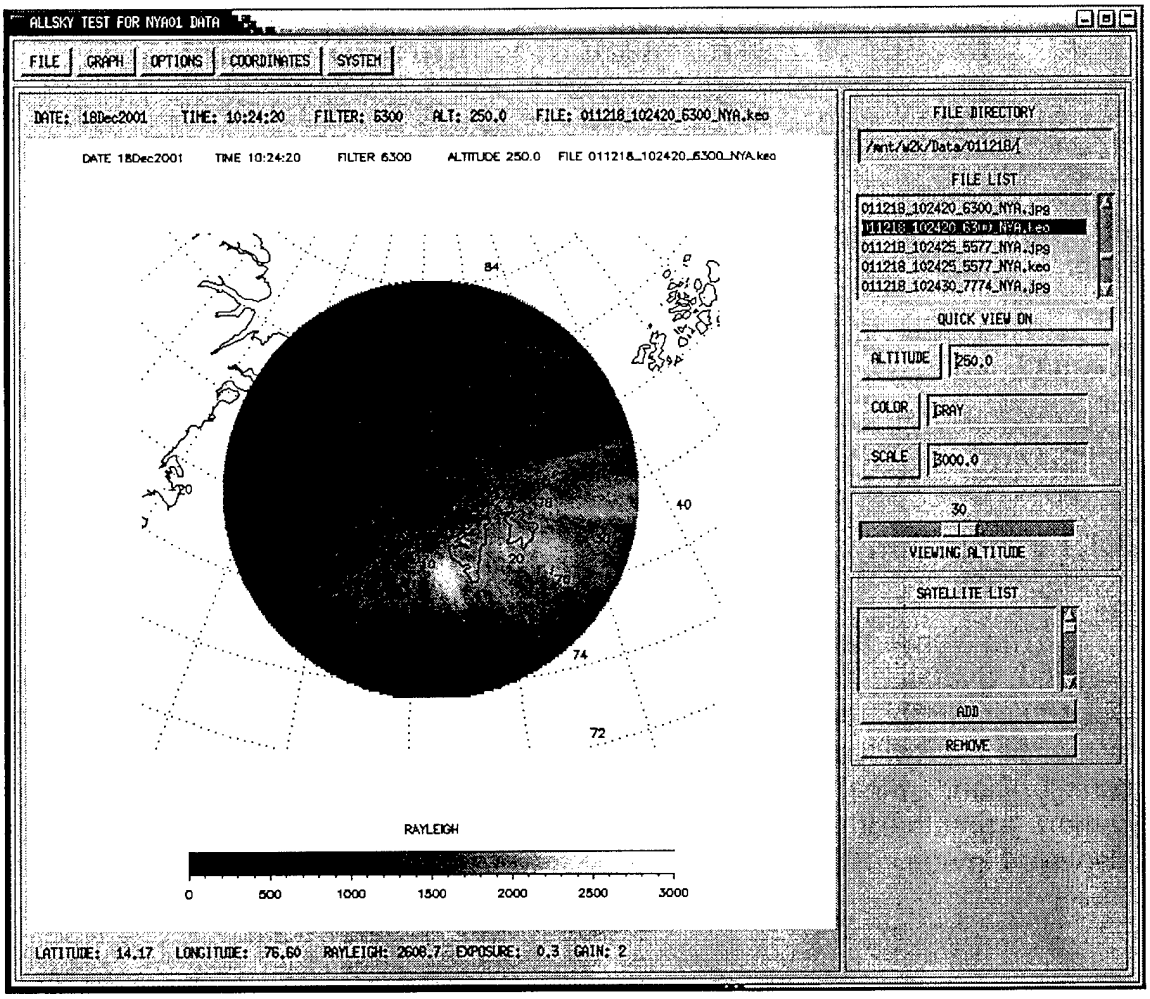


Figure 4-11 Transformed and Calibrated Display using ALLSKY.PRO

Chapter 5: Performance, Calibration, and Data

5.1 Field Evaluation

This ASIP instrument was designed for field use. We took ASIP to Ny Alesund, Svalbard, installed and operated it during two winter periods between December 2001 and February 2002 and between December 2002 and March 2003.

5.1.1 Installation

The Norwegian Polar Institute hosted our instrument at their facility in Ny Alesund, Svalbard as part of our collaborative research effort with the Physics Department at the University of Oslo. A room with a dome and mounting poles was already provided. We had to make mounting brackets to fit ASIP onto their configuration. Final adjustments were made using a level and rotating the camera azimuthally until true north, using a starmap, aligned with the top of the displayed image. 120Volt power on UPS was available.

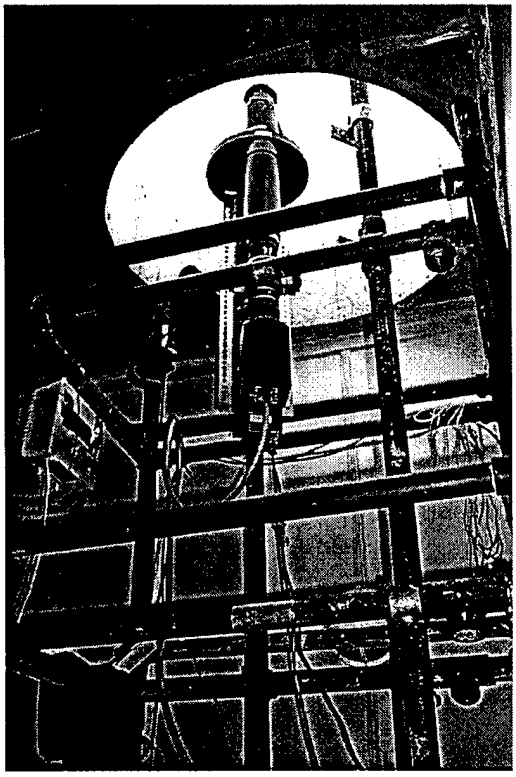


Figure 5-1 Mounted ASIP at Ny Alesund

5.1.2 Host Computer

Our initial selection criteria for ASIP's host computer system was to be light and small to minimize transportation costs. During the first winter's operation, we used a 15-inch flat panel LCD monitor and a "BiscuitPC" type computer. This was a 700MHz Pentium III CPU computer the size of a large shoe box, but had moderate hard disk space (20 Megabytes) and limited archiving support, such as a CDRW. The disk had to be cleared monthly between moon down periods. As a result we then opted for a different computer to meet these shortfalls. In the second winter, we replaced the BiscuitPC with a miniaturized version of a fully functional desktop. This SV24 model from Shuttle, Inc. is a 1.0 GHz CPU configured with a 120 Gigabyte hard disk, a floppy drive, and a CD-RW. The motherboard has built in graphics, networking, USB, FireWire, and one PCI-slot support. Figure 5-2 shows ASIP's host computer in its final operating configuration. Note that the LCD monitor fits well on top of the SV24 unit and takes very little desktop space.

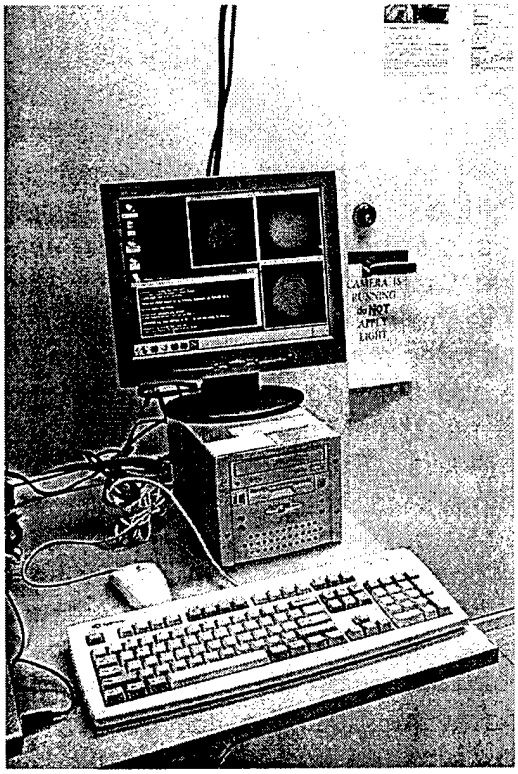


Figure 5-2 Host Computer for ASIP

5.1.3 CCD Temperature & Intensifier Issues

The ideal operating temperature for the CCD is –40C. However we found out that the vacuum seal had leaked in the CCD evacuated chamber and maintaining a 65C differential from ambient was not achievable. The unit was returned to the vendor and the vacuum was resealed using nitrogen backfilling. This did not pose a problem in Ny Alesund because the ambient was often well below 20C in the room with the dome. However this does require external cooling where ambient temperatures exceed 25C, as often is the case when operating in tropical climates.

The initial delivery of the instrument from Keo used a modified electronic circuitry to drive the new Gen-III image intensifer tube. The drive electronics prevented using maximal gain setting provided by Gen-III newer sensitivity. However the settings used in Ny Alesund was sufficient to detect patches, even as the highest gain setting, 3, was unusable. This deficiency has been corrected with a new Gen-III driver electronics design. It has been fabricated and the schematics included here in this manual, but has yet to be field tested. Traditionally, the gain settings on the circuitry should be adjusted so than a change in gain changes the sensitivity by a factor of two, but in our case was determined to be a factor of ten.

5.1.4 Campaign Support

The late awarding of this contract and the subsequent delay of the instrument from Keo prevented us from deploying and testing ASIP in the winter period of December 2000 and February 2001. Our first available opportunity to support a major campaign was in December 2001. During this period we operated the imager both in the routine, one filter per minute mode, as well as customized high acquisition rates of multiple same-filter images per minute, up to twelve images per minute. At the same time data was made available via the internet to researchers in near real-time to support operating modes for the nearby incoherent scatter radars. This same support was repeated in February 2002, after which the instrument was returned to Scion facilities. An extension of the contract was requested to make up for missing the first campaign and would allow us to test out remote operation of the instrument via the internet. This was granted and we deployed the instrument back to Ny Alesund in time to support campaigns in both December 2002 and January 2003. Measurements continued through March 2003, when the instrument was finally delivered to AFRL at Hanscom AFB as required at the closing of the contract. This entire winter's worth of data was able to be stored on the computer's 120G hard disk. Both winter's data were archived onto CD's and delivered to AFRL along with analysis software and calibration data.

5.1.5 Network Operation

A key design feature for ASIP camera operations is the ability to control the imager remotely via the internet. The Linux operating system, by design provide this capability. All Linux, and UNIX derivative operating systems, allow remote login and display using shells and X-windows. During the first winter, the imager data as it was being acquired, were transferred manually when requested, in real-time over the internet to researchers to view the quick look-up JPEG images. During the second winter, remote operations were tested. We logged into the computer over the internet via "ssh" and edited the param.txt and schedule.txt files. We launched the piAuto program by typing "./piAuto > /dev/null", then logging out. The "/dev/null" redirects the program text output so it won't be displayed anywhere. We were able to retrieved both the raw and JPEG data files using "sftp". Additionally, while the imager was acquiring data, we ran the following shell script every five minutes to automate sending the current imager data to a web page hosted by the University of Oslo.

```
#!/bin/sh

#This file: transfer.sh

todaydir=$(date +%y%m%d)
nowdir=$(date +%y%m%d_%H%M)*.jpg
file6300=$(date +%y%m%d_%H%M)00_6300_NYA.jpg

pingok=$(ping -nc 1 129.240.86.90 | grep -c ' 0% packet loss')

sleep 60

cd /imagedir

if [ -d $todaydir ]; then
   cd $todaydir
if [ -f $file6300 ] && [ $pingok ]; then
   ftp -i aurora.uio.no <<-EndFTP
   cd www_docs/$todaydir
   mput $nowdir
EndFTP
fi

fi

cd ~
```

Figure 5-3 Script to Send JPEG's to Web Server

5.1.6 Bare CCD Test

We had the opportunity to test this camera technology in the bare CCD mode without the image intensifier. This was conduction in March 2002 at Ascension Island along with other AFRL scientists. A sample image is shown in Figure 1-4 Bare CCD Mode Data Example - Spread-F Bubbles. As expected, the clarity of the image is evident when the graininess effect contributed by the image intensifier is removed. Of course the trade off is in sensitivity as most of the bare CCD exposure times are often over a minute, in this case, two minutes.

5.2 Calibration

The following sections describe the various calibration techniques required to extract geophysical measurements from the ASIP camera. Most of them were conducted in the field with a known light source standard. The calibration measurements were made in the 2x2 binning mode. The details for these measurements can be extracted and reconstructed from the IDL routines found in asiptools.pro and applied to the actual raw imager data files acquired during the calibration process. The calibrated results are embedded in the GUI-based analysis tool allsky.pro.

5.2.1 Spatial Calibration Using Starmaps

A spatial calibration was carried out to correct for azimuthal installation offsets and optical distortions, mostly attributed to the curvature of the lenses.

Figure 5-4 summarize the spatial calibration equation and constants. From this we derived the optical center of the image in pixel space and the radial mapping based upon best fit with the computer generated starmap from the program Xephem in Figure 5-5. Note that during this particular installation we were fortunate to have no azimuthal offset which is often not the case.

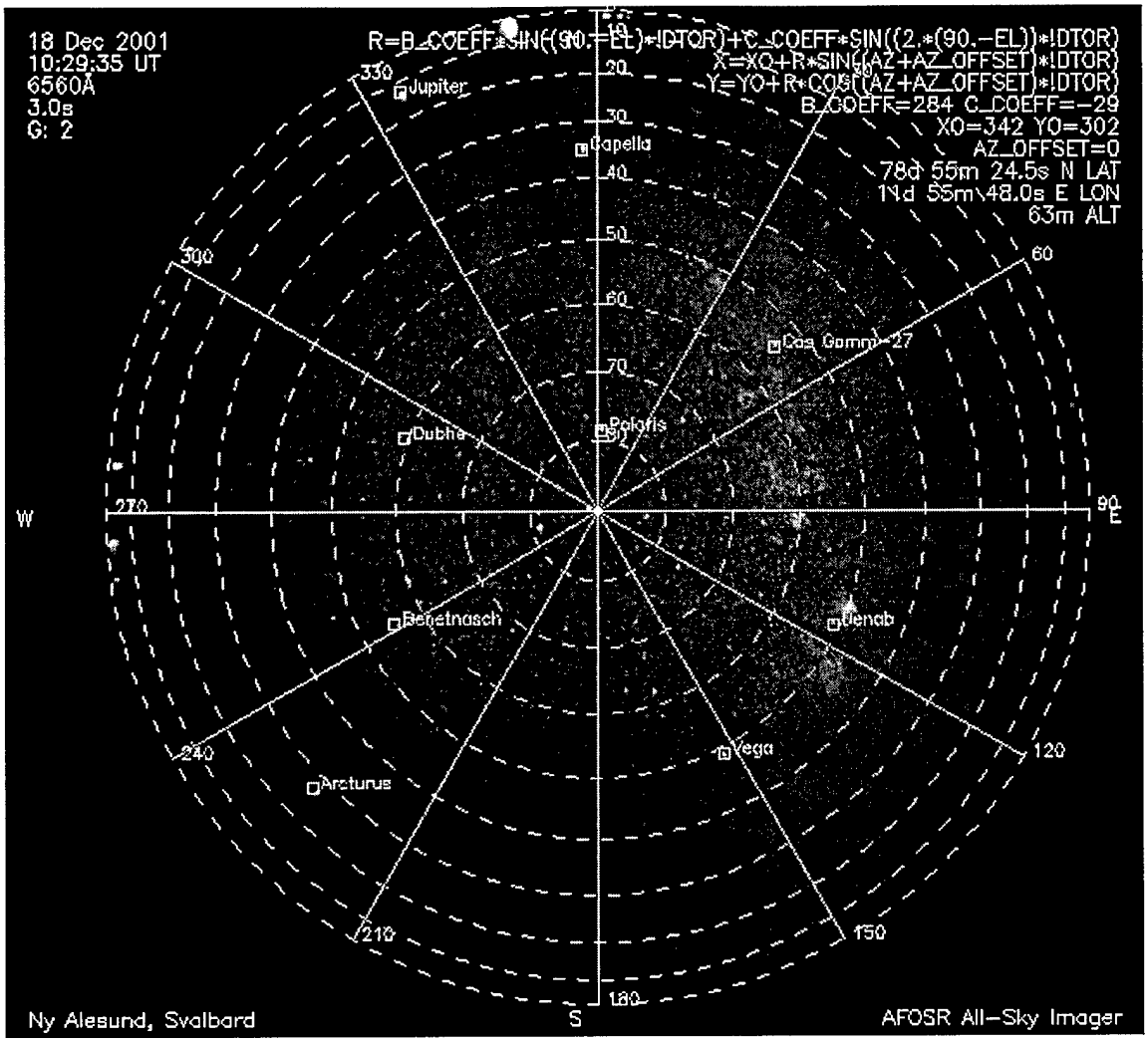
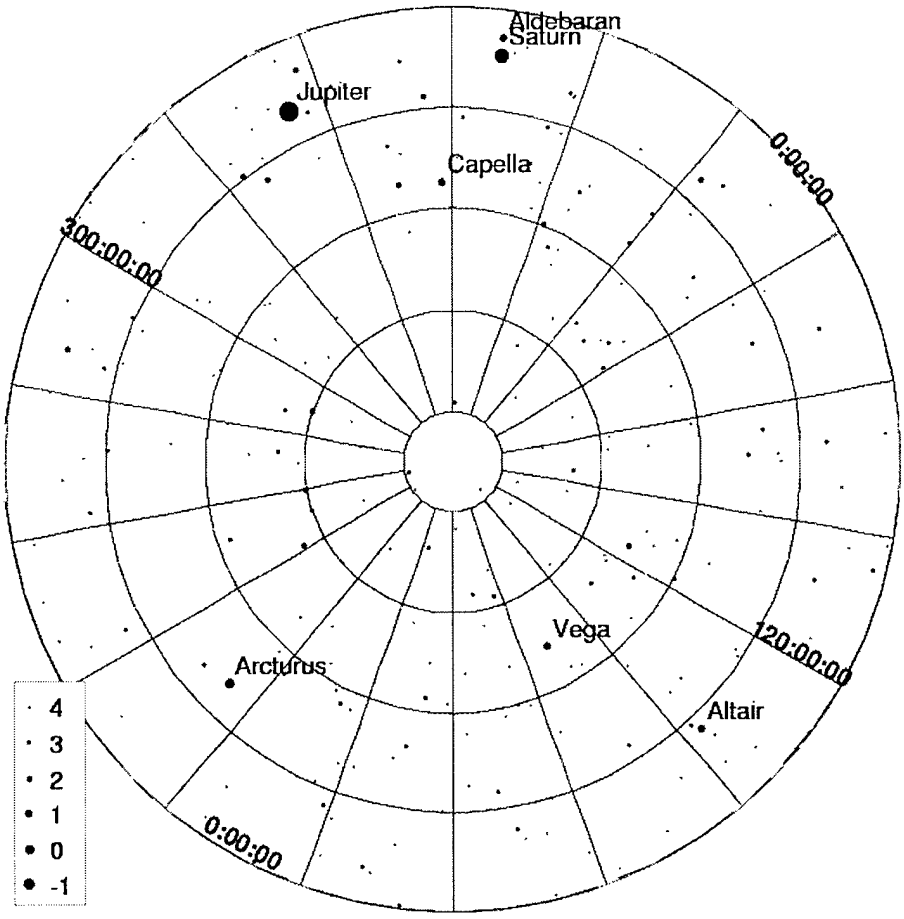


Figure 5-4 Starmap Calibration



XEphem Alt/Az Sky View
Ny Alesund, Norway

RA:	17:05:55.9			Julian Date:	2452261.93721
Declination:	78:55:35			Sidereal Time:	17:05:43
Epoch:	2000.00	Grid Steps:		UTC Date:	12/18/2001
Altitude:	90:00:00	Alt:	20:00:00	UTC Time:	10:29:35
Azimuth:	180:00:00	Az:	20:00:00	Latitude:	78:55:24 N
Field Width:	180:00			Longitude:	11:55:48 E

Created by XEphem Version 3.4 4 Dec 2000
(c) 1996-2000 Elwood Charles Downey
http://www.ClearSkyInstitute.com
Generated Tue Apr 29 20:52:35 2003 UTC

Figure 5-5 XEphem Starmap Output

5.2.2 Light Source

The light source used to make absolute calibration was designed by Keo Consultants and owned by our collaborating colleagues at the University of Oslo, Department of Physics. It was available for us to use in field where the ASIP calibaration was conducted. Below are the two tables required to calculate the light output in units of Rayleighs per Angstrom (R/Å). There are two dial settings on the light source, knobs A and B, that varies the relative attenuation from full open or maximum light output. Table 5-1 assigned an attenuation factor that is multiplied from the maximum output value derived from Table 5-2 as function wavelength. Intermediate wavelength values can be interpolated.

Norway Light Source
Relative Attenuation Calibration

A →

B ↓	#2	#1	#0
#11	1.000E+00	1.900E-01	2.100E-02
#10	5.367E-01	1.020E-01	1.127E-02
#9	2.821E-01	5.360E-02	5.925E-03
#8	1.487E-01	2.826E-02	3.123E-03
#7	7.593E-02	1.443E-02	1.595E-03
#6	3.934E-02	7.475E-03	8.262E-04
#5	2.109E-02	4.007E-03	4.429E-04
#4	1.073E-02	2.040E-03	2.254E-04
#3	5.080E-03	9.652E-04	1.067E-04
#2	2.628E-03	4.992E-04	5.518E-05
#1	1.510E-03	2.869E-04	3.170E-05
#0	0.00E+00	0.00E+00	0.00E+00

Table 5-1 Norway Light Source Relative Attenuation

NORWAY Light Source Calibration
(Set at Maximum Output)

Wavelength (nm)	SLS Output (R/Å)	Wavelength (nm)	SLS Output (R/Å)
350	2.64E+02	600	6.62E+04
360	4.12E+02	610	7.01E+04
370	7.83E+02	620	8.04E+04
380	1.41E+03	630	8.61E+04
390	2.46E+03	640	9.15E+04
400	3.46E+03	650	9.56E+04
410	4.88E+03	660	1.03E+05
420	5.89E+03	670	1.09E+05
430	7.11E+03	680	1.15E+05
440	8.34E+03	690	1.24E+05
450	9.82E+03	700	1.31E+05
460	1.23E+04	710	1.40E+05
470	1.44E+04	720	1.45E+05
480	1.71E+04	730	1.46E+05
490	1.99E+04	740	1.47E+05
500	2.26E+04	750	1.45E+05
510	2.65E+04	760	1.45E+05
520	2.99E+04	770	1.48E+05
530	3.35E+04	780	1.40E+05
540	3.81E+04	790	1.39E+05
550	4.19E+04	800	1.28E+05
560	4.66E+04		
570	5.09E+04		
580	5.59E+04		
590	6.15E+04		

Table 5-2 Norway Light Source at Maximum Output

5.2.3 Vignetting

This is a characteristic intrinsic to commercial lenses. The main contributor to system vignetting is the primary fish-eye lens. We took a light source and scanned it from the center of the image to its edge. A composite image superimposing just the maximum pixel values is show in Figure 5-6. Note that the discontinuity near 45 degrees is to be interpolated. A line plot of its intensity results in a linear equation to the first order characterizing the slope of intensity as a function of zenith angle. Basically the light source is reduced

linearly by a factor of 20.93 from overhead to the edge of the field of view. Thus the vignetting correction divisor VIGFN is:

$$VIGFN = 1 - 0.952252 * (\text{zenith angle} / 90.)$$

Figure 5-6 Composite Image Showing Vignetting

5.2.4 CCD Bias

The CCD bias is the number of counts attributed by the thermal noise within the CCD sensor electronics. We found this number to average out to:

$$CCD\ BIAS = 71.79 \text{ counts}$$

This measurement was taken with the intensifier and shutter turned off.

5.2.5 Dark Noise

The dark noise counts are the contribution from the image intensifer electronics. As such they are dependent on the gain setting of the image intensifer. Table 5-3 summarize the average counts measured with the intensifier on for one second, the shutter closed, and the removal of the CCD bias.

Gain	Counts
0	11.16
1	125.92
2	1147.72

Table 5-3 Dark Noise Measurements

5.2.6 Counts to Rayleigh's Conversion

The desired scientific unit of measure of energy from optical observations are Rayleigh units. With the light source described above, we conducted a series of measurements to derive the following calibration chart describing the ASIP system response to light at different wavelengths and intensifer gains. The measure counts are a function of filter wavelength, filter bandwidth, intensifer gain and exposure time. All values are normalized to unity exposure time of one second.

Table 5-4 summarize the correction divisor to counts (normalized to unity exposure) in order to obtain the absolute Rayleigh value.

Filter	Stock #	Bandwidth(Å)	Gain 0	Gain 1	Gain 2
6300	Lot 0501	16	0.653516	7.88324	74.9065
5577	Lot 0601	16	0.709080	8.36213	81.2303
4278	Lot 4801	25	0.398004	4.82669	42.9298
7774	Lot 0501	16	0.461177	5.55789	50.5229
6560	Lot 0401	30	0.705478	8.60690	79.8694

Table 5-4 Counts to Rayleigh Conversion

5.3 Data

During the course of developing and evaluating the ASIP, we have acquired two winter's worth of ASIP data at a unique location covering measurements of both the dayside cusp and nighttime aurora oval. Ongoing efforts will continue at analyzing this unique data set with other measurements. This is the first time four inch optics were used to acquire images from five different filters on a routine basis. The traditional 6300Å and 5577Å wavelengths have been complemented with 4278Å, 7774Å, and 6560Å wavelengths. With the same four inch optics during campaign modes, we provided relative high frame rates of up to twelve exposures per minute for any combination of filters. This will allow us to resolve temporal events in more detail than the traditional one filter per minute rate.

The quick look up JPEG images and the raw data image files stored on the host computer's hard disk have been archived onto 700 MB Data CD's. The complete set for the two winter periods are distributed in five CD cases. Two complete sets are to be delivered to AFRL. The contents are summarized in Table 4-1.

CASE NUMBER	DATE	BINNING MODE	NUMBER OF CD's
1	13dec01-19dec01	2 x 2	38
2	12jan02-21jan02	2 x 2	51
3	06feb02-11feb02	2 x 2	27
4	06dec02-28jan03	4 x 4	48
5	28jan03-07mar03	4 x 4	30
Total CD's per Set:			194

Table 5-5 Summary of Data CD's

The camera operated during the moon down periods at Ny Alesund, Svalbard during it's winter season when the sun remains below the horizon. The tables for the start and stop times are listed in the schedule.txt files found in Chapter 6:below. Any missing periods may have been the result of intentionally not operating the camera because the sky conditions were unfavorable for collecting optical data, i.e. clouds or snowing.

The quick look up JPEG's and some animation files can also be found and downloaded from the web site: http://www.fys.uio.no/~ning

The IDL software tool allsky.pro have been used to analyze this data set and generate calibrated hardcopy outputs. Two hardcopy outputs normally in Postscript format are displayed here as TIFF files and show the comparison between looking at two separate wavelengths at different intensity scales and at different assumed altitude of emissions.

DATE 18Dec2001 TIME 10:24:20 FILTER 6300 ALTITUDE 250.0 FILE 011218_102420_6300_NYA.keo

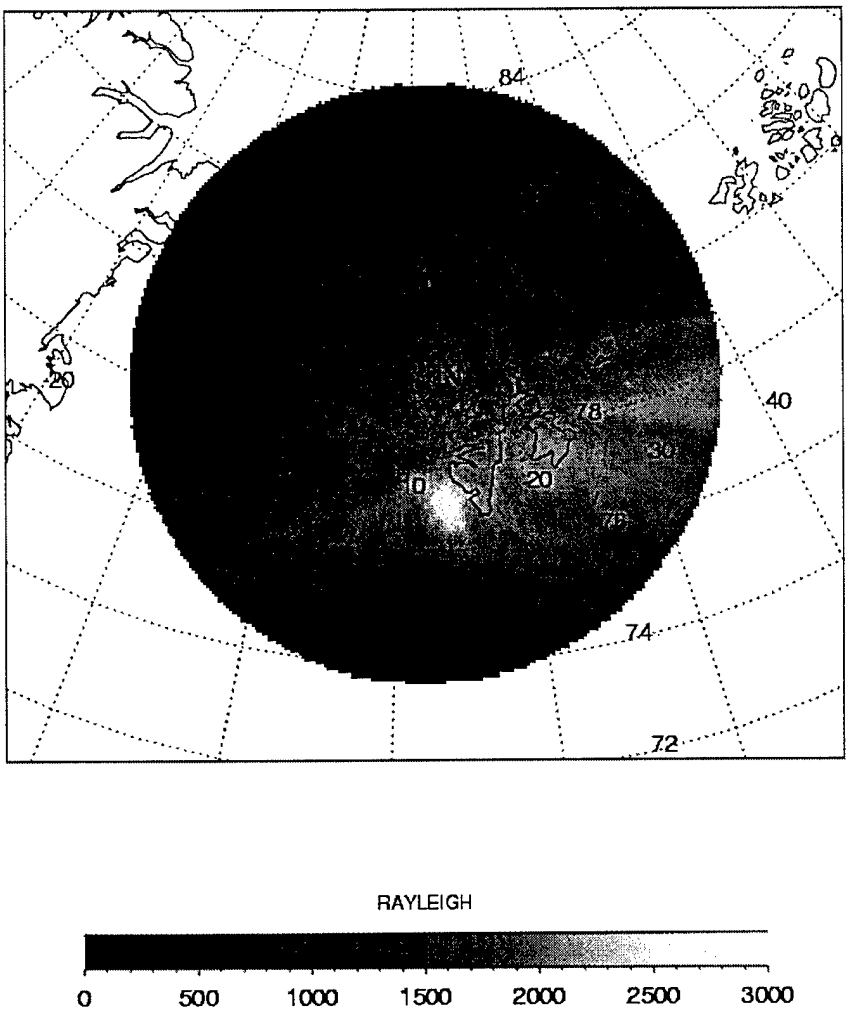


Figure 5-7 Sample Hardcopy Output of 6300Å at 250km

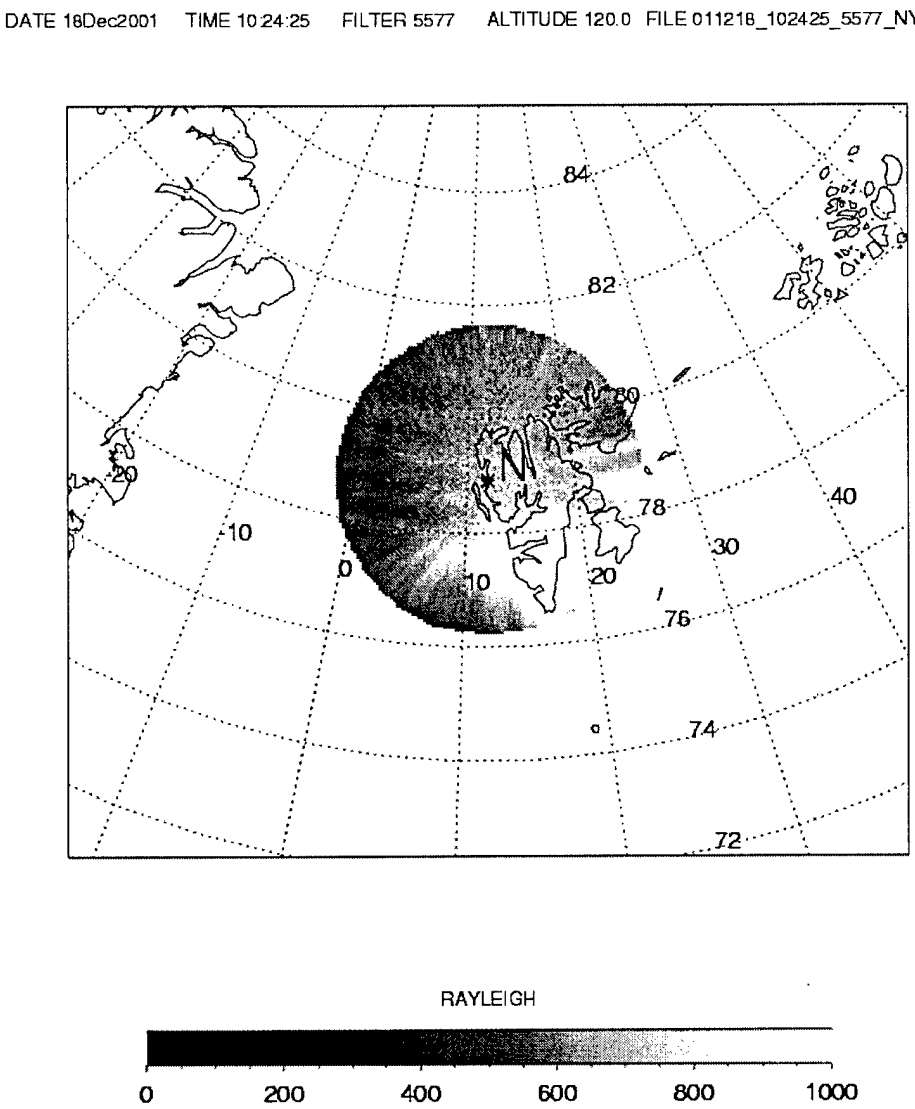


Figure 5-8 Sample Hardcopy Output of 5577Å at 120km

Chapter 6: Source File Listings

6.1 Acquistion Program: /root/camctl/piAuto.c

```
/* This file: piAuto.c  version 1.1   pn 28nov02      */
/*

29nov02 /dev/cua0 -> /dev/ttyS0
      corrected JPEG CF 255/65355 -> 255/MAX_COUNTS
07dec01 Built on anfauto.c template from ANF Imager upgrade
*/

#include <termios.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/signal.h>
#include <math.h>
#include <limits.h>
#include <string.h>
#include <asm/io.h>
#include <time.h>

#include "jconfig.h"
#include "jpeglib.h"

#include "SerialLib.h"
#include "master.h"
#include "pvcam.h"

#define CCD_X_PIXELS   1340
#define CCD_Y_PIXELS   1300
#define CCD_X_BINNING  4
#define CCD_Y_BINNING  4
#define CCD_BYTES_PER_PIXEL 2

#define IMAGE_X_SIZE   CCD_X_PIXELS / CCD_X_BINNING
#define IMAGE_Y_SIZE   CCD_Y_PIXELS / CCD_Y_BINNING

#define BUFSIZE IMAGE_X_SIZE * IMAGE_Y_SIZE      // short unsigned (2-
byte)
```

```
#define PARAMFILE "param.txt"
#define TEMPFILE "current.filename"

#define MAX_COUNTS 65535     // HRP: 4095
#define SIGMA 3.0
#define MAX_EXPOSURE 100.    // HRP: 100.
#define MIN_EXPOSURE 2.
#define MAX_GAIN 3           // Gain 3 not usable on AFOSR imager! - AFRL OK!
#define MIN_GAIN 0

#define _POSIX_SOURCE 1 /* POSIX compliant source */

struct entry {
  int filter_pos;
  int gain;
  int exp_time;
} entry[30];

void do_entry();
void write_jpeg();

int hh,mm,ss,yy,mo,dd;
time_t now,lasttime,starttime,stoptime,startlist[365],stoplist[365];
int time_idx;
int i,j;
long k;
struct tm *tmtime, tmStartTime, tmStopTime;
FILE *fp,*fpOutput,*fpPGM, *fptemp;
char filter[6][10];
int sec_start,filter_pos,gain,exp_time;
int cycle_index[60];
unsigned int seconds_in_cycle;
int entry_idx, schedule_idx, current_tm_day;
char sDirname[30],sFullFilename[70];
char sPgmFilename[10];

int nparam, nNextFilter, nCurrentFilter=-1, nStatus;
int nNextGain, nCurrentGain=-1;
int nLight=0, nIntPower=0, nNextPower;
int nShutter=0, nNextShutter;

//PI Camera variables
boolean status,exposing;
rgn_type rgn;
unsigned short x,y;
```

```
int16 hCam;
uns16_ptr pimg;
uns32 stream_size, bytes;
int16 expStatus;
static int16 tempC;
static int16  ccs_status;
static boolean  cam_status = FALSE;
long imagePixels;

char c;
int fd;

short unsigned buf[BUFSIZE];       // 2x2 bin: 435500, 4x4 bin: 108875
short int *ip;

int int_on_flag = 0;
FILE *f;
//FILE *f=stdin;
int fd;
int tot;
unsigned char ctlByte,ucByte;

char outbyte;
int port_addr, chip;
unsigned char ucdata;
short unsigned uidata, uHiByte, uLoByte;

char sFilename[70],sSet_Intgain[5],sSet_Filpos[6],sObs[10];
char sFilePath[30] = "";
char sLocation[4] = "XXX";

unsigned lat_deg,lat_min,lat_sec,lon_deg,lon_min,lon_sec,alt_m;

unsigned *pBufferIdx;

// Statistics variables
float ave,sdev;
float data[100];            // Max samples: 100
void moment(),findindex();
unsigned randindex[100];
int upper,lower;

int newgain;
float newexp;

main(argc,argv,env)
```

```
int argc;
char *argv[];
char *env[];
{

printf("\r\npiAuto for AFOSR Imager at Ny Alesund, Svalbard -pn 28nov02
v1.1\n\n");

// Initialize Serial Port
if ( openSerial("/dev/ttyS0") )
   printf("Successfully opened the comm port...\n");
   else {
      printf("Error opening comm port, terminating program!\n");
      exit(-1);
   }

// Make sure intensifier is OFF!
  nStatus = intensifierPower(0);
  nStatus = intensifierGain(3);  // Lowest setting - inverted

// Initialize the PVCam Library
if (!pl_pvcam_init())
  {
   printf("Error initializing PVCAM\n");
   return;
  }

  pl_cam_open( "rspipci0", &hCam, OPEN_EXCLUSIVE );
  if (pl_error_code() != 0 ) {
     printf( "Camera Not Found, Or Communication Error\n");
     exit(0);
  }

  pl_ccd_set_tmp_setpoint(hCam,-4000);      //Set to -40 C
  pl_ccd_get_tmp_setpoint(hCam,&tempC);
  printf("Set Temperature: %5.2f C\n",(float)tempC/100.);

  printf("Waiting to reach at least -25C to operate...\n\n");
  pl_ccd_get_tmp(hCam,&tempC);
  while ( tempC > -2500 ) {
     pl_ccd_get_tmp(hCam,&tempC);
     printf("Current Temperature: %5.2f C\r",(float)tempC/100.);
  }
  printf("Minimum temperature reached!\n\n");

  pl_ccd_get_ser_size(hCam, &x);  // Get Sensor Dimensions
```

```
  pl_ccd_get_par_size(hCam, &y);
  printf("Sensor size: %d x %d\n", x, y);
  //  rgn.sbin = CCD_X_BINNING; rgn.s1=0; rgn.s2 = x / rgn.sbin - 1;
  // rgn.pbin = CCD_Y_BINNING; rgn.p1=0; rgn.p2 = y / rgn.pbin - 1;
  rgn.sbin = CCD_X_BINNING; rgn.s1=0; rgn.s2 = x  - 1;
  rgn.pbin = CCD_Y_BINNING; rgn.p1=0; rgn.p2 = y  - 1;

  printf("Store image: X=%d Y=%d with %d x %d binning\n",
         rgn.s2+1,rgn.p2+1,rgn.sbin,rgn.pbin);

  imagePixels = ( (rgn.s2 +1) / rgn.sbin ) *( (rgn.p2 +1) / rgn.pbin );


  pl_exp_init_seq();
  pl_shtr_set_open_mode(hCam, OPEN_PRE_EXPOSURE);
  pl_exp_setup_seq(hCam,1,1,&rgn,TIMED_MODE, 500, &stream_size );
  pimg = (uns16_ptr) malloc( imagePixels * 2 );

// Variable Initialization
for (i=0; i < 60; i++) cycle_index[i] = -1;

lat_deg =  78; lat_min = 55; lat_sec = 24;
lon_deg = 348; lon_min =  4; lon_sec = 12;  // NYA E11d55'48'   348    4  12
alt_m = 63;


//Read schedule File

  time_idx = 0;
  fp = fopen("schedule.txt","r");
  if (fp == NULL) {perror("Can't open schedule file!"); exit(-1); }

  while(
    fscanf(fp,"%2d%2d%2d %2d%2d%2d %2d%2d%2d %2d%2d%2d",
           &tmStartTime.tm_hour,
           &tmStartTime.tm_min,
           &tmStartTime.tm_sec,
           &tmStartTime.tm_year,
           &tmStartTime.tm_mon,
           &tmStartTime.tm_mday,
           &tmStopTime.tm_hour,
           &tmStopTime.tm_min,
           &tmStopTime.tm_sec,
           &tmStopTime.tm_year,
           &tmStopTime.tm_mon,
           &tmStopTime.tm_mday) != EOF) {
```

```
    tmStartTime.tm_mon--;
    tmStopTime.tm_mon--;

tmStartTime.tm_year = tmStartTime.tm_year + 100;
tmStopTime.tm_year = tmStopTime.tm_year + 100;

    starttime = mktime(&tmStartTime);
    stoptime  = mktime(&tmStopTime);

    startlist[time_idx] = starttime;
    stoplist[time_idx] = stoptime;
    time_idx++;

    printf("\rStart Time: %s",asctime(&tmStartTime));
    printf("\rStop  Time: %s",asctime(&tmStopTime));
  }
  fclose(fp);

  // Read Parameter File

  fp = fopen(PARAMFILE,"r");
  if (fp == NULL) {perror("Can't open parameter file!"); exit(-1); }

  fscanf(fp,"%s",sLocation);

  fscanf(fp,"%s %s %s %s %s %s",
         filter[0],filter[1],filter[2],filter[3],filter[4],filter[5]);

  i = 0;
  while(fscanf(fp,"%u %u %u %u",&sec_start,&filter_pos,&gain,&exp_time) == 4)
{
    cycle_index[sec_start] = i;
    entry[i].filter_pos = filter_pos - 1;
    entry[i].gain = gain;
    entry[i].exp_time = exp_time;
    i++;
    printf("\rEntry: %2u %s %2u %3u\n",sec_start,filter[filter_pos-
 1],gain,exp_time);
  }

  fclose(fp);

  for (i=0; i < 6; i++) printf("\rFilter %u: %s\n",i+1,filter[i]);

  current_tm_day = 0;
```

```
 for ( schedule_idx = 0; schedule_idx < time_idx; schedule_idx++) {

 starttime = startlist[schedule_idx];
 stoptime = stoplist[schedule_idx];

 printf("\rWaiting for Start Time: %s\r...\n",ctime(&starttime));
 time(&now);
 while (now < starttime) {
  time(&now);
  pl_ccd_get_tmp(hCam,&tempC);
  printf("\rCurrent Temperature: %5.2f C\r",(float)tempC/100.);
 }

if (now < stoptime) {
// Initialize FrameGrabber

// Turn on Intensifier
  nStatus = intensifierPower(1);
  int_on_flag = 1;
}

while ( now < stoptime) {
 time(&now);
 if(lasttime != now) {
  tmtime = gmtime(&now);
  seconds_in_cycle = tmtime->tm_sec;
  //   printf("%2u: ", seconds_in_cycle);
  if ((entry_idx = cycle_index[seconds_in_cycle]) != -1) {
   printf("\r%2u: F: %s G:%2d E: %3d", seconds_in_cycle,
          filter[entry[entry_idx].filter_pos],
          entry[entry_idx].gain,entry[entry_idx].exp_time);
   sprintf(sFilename,"%02d%02d%02d_%02d%02d%02d_%s_%s.keo",
           tmtime->tm_year-100,tmtime->tm_mon+1,tmtime->tm_mday,
           tmtime->tm_hour,tmtime->tm_min,tmtime->tm_sec,
           filter[entry[entry_idx].filter_pos],sLocation);
   printf(" - %s\n",sFilename);

      // Handle Day Directory Creation
      if (tmtime->tm_mday != current_tm_day) {
       sprintf(sDirname,"//imagedir//%02d%02d%02d",
            tmtime->tm_year-100,tmtime->tm_mon+1,tmtime->tm_mday);
       mkdir(sDirname,0755);
       strcat(sDirname,"/");
      current_tm_day = tmtime->tm_mday;
      }
```

```
    do_entry(entry_idx);

    write_jpeg();

//if ( (fptemp = fopen( TEMPFILE, "wb" )) == NULL ) {
//  printf( "\n\nCouldn't open file to pass current filename!  Aborting the
program...\n" );
//  fclose( fptemp );
//exit(0);
//  }
//sFullFilename[0] = 0;
//strcat(sFullFilename,sDirname);
//strcat(sFullFilename,sFilename);
//fprintf(fptemp,"%s\n",sFullFilename);
//fclose(fptemp);


    printf("\rAverage: %7.2f  Std Dev: %7.2f\n",ave,sdev);

    upper = ave + SIGMA * sdev;
    lower = ave - SIGMA * sdev;

    printf("\rUpper: %6d  Lower: %6d  Status: ",upper,lower);

    newgain = entry[entry_idx].gain;
    newexp = entry[entry_idx].exp_time;

//**** Calculate next frame parameters...
    if (upper > MAX_COUNTS || lower > (int) (MAX_COUNTS/2.0)) {
       printf("OVEREXPOSED\n");
              if (newgain > MIN_GAIN) {
                newgain--;
                newexp *= 1.25;
              } else if (newexp > (MIN_EXPOSURE * 2.)) newexp /= 2.;
           else newexp = MIN_EXPOSURE;
    }
       else if (upper < (int) (MAX_COUNTS/2.0)) {
         printf("UNDEREXPOSED\n");
         if (newexp < MAX_EXPOSURE) {
           // printf ("\rExp < MAX_EXP\n");

           if (newexp <= (MAX_EXPOSURE /2.)) {// printf("\rExp <=
MAX_EXPOSURE/2\n");
                                    newexp *= 1.8; }
         else  { // printf("\rNew = MAX\n");
```

```
                newexp = MAX_EXPOSURE; }
            }

            else if (newgain < MAX_GAIN) { newgain++; newexp = .75 *
MAX_EXPOSURE; }
          }

    else printf("NO CHANGES NEEDED\n");

    if (newgain > MAX_GAIN) newgain = MAX_GAIN;
    if (newexp  > MAX_EXPOSURE) newexp = MAX_EXPOSURE;
    if (newgain < MIN_GAIN) newgain = MIN_GAIN;
    if (newexp  < MIN_EXPOSURE) newexp = MIN_EXPOSURE;

//*** End calculation of next frame's parameters.
//    if (entry_idx != 0) {
      entry[entry_idx].gain = newgain;
      entry[entry_idx].exp_time = newexp;
//    }
     printf ("\rComputed Adjustment GAIN: %d EXP: %6.2f Temp: %5.2f\n",
            newgain, newexp, (float)tempC/100.);
    }
}
  lasttime = now;
}

if (int_on_flag != 0) {
// Turn off Intensifier
  nStatus = intensifierPower(0);
  printf("\rIntensifer Off - Stop Time Reached!\n");

  int_on_flag = 0;
}

 } // End for loop on scheduler index

  printf("\r\nFinished List - Terminating Program!\n");


// Close Parallel Port here!

pl_exp_uninit_seq();
free(pimg);
exit(0);
}  /* End Main */
```

```
void do_entry()
 {
//*** BEGIN ACQUISITION
 nStatus = intensifierGain(3 - (entry[entry_idx].gain & 0x0003));
 // printf("Intensifier%d\n", nStatus);
 nStatus = filterWheel( entry[entry_idx].filter_pos + 1 );
 nStatus = shutter(1); // Open shutter!

 pl_exp_setup_seq(hCam,1,1,&rgn,TIMED_MODE,entry[entry_idx].exp_time *
100 , &stream_size );
 pl_exp_start_seq(hCam,pimg);
 expStatus = 0;
 while (expStatus != READOUT_COMPLETE)
    pl_exp_check_status( hCam, &expStatus, &bytes);
 nStatus = shutter(0); // Close shutter!
// read image here

// Write PGM file
 if ((fpPGM = fopen("image.pgm","wb")) == NULL) {
  printf( "\n\nCouldn't open PGM file! Aborting the program...\n" );
  fclose( fpPGM );
  exit(-1);
 }
// fprintf(fpPGM,"%s","P5 670 650 65535 "); // for 2x2
 fprintf(fpPGM,"%s","P5 335 325 65535 "); // for 4x4
 for(k=0;k<imagePixels;k++) {
   uidata = *(pimg+k);
   buf[k] = uidata;
   uHiByte = (uidata >> 8) & 0x00FF;
   uLoByte = uidata & 0x00FF;
   fputc(uLoByte,fpPGM);
   fputc(uHiByte,fpPGM);
 }
 fclose(fpPGM);


// Write Filter Specific PGM file
 sPgmFilename[0] = 0;
 strcat(sPgmFilename,filter[entry[entry_idx].filter_pos]);
 strcat(sPgmFilename,".pgm");

 if ((fpPGM = fopen(sPgmFilename,"wb")) == NULL) {
  printf( "\n\nCouldn't open Filter Specific PGM file! Aborting the program...\n" );
  fclose( fpPGM );
  exit(-1);
 }
```

```
//  fprintf(fpPGM,"%s","P5 670 650 65535 ");  // for 2x2
  fprintf(fpPGM,"%s","P5 335 325 65535 ");  // for 4x4
  for(k=0;k<imagePixels;k++) {
     uidata = *(pimg+k);
     buf[k] = uidata;
     uHiByte = (uidata >> 8) & 0x00FF;
     uLoByte = uidata & 0x00FF;
     fputc(uLoByte,fpPGM);
     fputc(uHiByte,fpPGM);
  }
  fclose(fpPGM);


// Handle Statistics...
  findindex(100,125);    // 2x2: 250
  for (k=0;k < 100; k++) data[k] = buf[randindex[k]];
  moment(data,100,&ave,&sdev);

// Get CCD temperature
  pl_ccd_get_tmp(hCam,&tempC);

// Insert parameters into Image array.
 ip = &buf[0];
 *ip = tmtime->tm_year + 1900;          // Index 0
 *++ip = tmtime->tm_mon+1;              // Index 1
 *++ip = tmtime->tm_mday;               // Index 2
 *++ip = tmtime->tm_hour;               // Index 3
 *++ip = tmtime->tm_min;                // Index 4
 *++ip = tmtime->tm_sec;                // Index 5
 *++ip = atoi(filter[entry[entry_idx].filter_pos]); // Index 6
 *++ip = entry[entry_idx].gain;         // Index 7
 *++ip = entry[entry_idx].exp_time;     // Index 8
 *++ip = lat_deg;                       // Index 9
 *++ip = lat_min;                       // Index 10
 *++ip = lat_sec;                       // Index 11
 *++ip = lon_deg;                       // Index 12
 *++ip = lon_min;                       // Index 13
 *++ip = lon_sec;                       // Index 14
 *++ip = alt_m;                         // Index 15
 *++ip = tempC;                         // Index 16 - .01 Centigrade units

  sFullFilename[0] = 0;
  strcat(sFullFilename,sDirname);
  strcat(sFullFilename,sFilename);

  if ( ( fpOutput = fopen( sFullFilename, "wb" ) ) == NULL ) {
```

```
    printf( "\n\nCouldn't open Output file!  Aborting the program...\n" );
    fclose( fpOutput );
    exit(-1);
    }

  for(k=0;k<imagePixels;k++) {
     uidata = buf[k];
     uHiByte = (uidata >> 8) & 0x00FF;
     uLoByte = uidata & 0x00FF;
     fputc(uLoByte,fpOutput);
     fputc(uHiByte,fpOutput);
  }
  fclose( fpOutput );

  pl_exp_finish_seq( hCam, pimg, 0 );

  }   //End do_entry();

//*************************************************************************
//write_jpeg
//*************************************************************************
void write_jpeg() {

const unsigned int width = IMAGE_X_SIZE ;
const unsigned int height = IMAGE_Y_SIZE ;
const unsigned int wxh = IMAGE_X_SIZE * IMAGE_Y_SIZE ;

const float CF = (255.0/MAX_COUNTS);     // HRP: (255.0/4095.0);

/*boolean typedef in jpeglib.h, must be TRUE (1) or FALSE (0)*/
const boolean FALSE_COLOR = TRUE;

const int pixparts = (FALSE_COLOR == TRUE) ? 3 : 1;
unsigned short headerdata[40];   // 20
char comment[70];
JSAMPLE pixeldata[width*height*pixparts];
JSAMPROW row_pointer[1];
int row_stride;

struct jpeg_compress_struct imanimage;
struct jpeg_error_mgr      jpegerror;
FILE *jpegfile;
char jfilename[70];
char *dot;
int i = 0;
int j;
```

```
if (pixparts == 3)
/*
 *initialize pixeldata for tri-component pixels
 */
 while(i < wxh ) {
  j = i*3;
  if (i < 40) {
    headerdata[i] = *(&buf[i*2]);
    pixeldata[j] = pixeldata[j+1] = pixeldata[j+2] = 0;
    i++;
  }
  else {
    switch (headerdata[6]) {
    case 4278 :
      pixeldata[j] = pixeldata[j+1] = 0;
      pixeldata[j+2] = (JSAMPLE)(buf[i]*CF);
      break;
    case 5577 :
      pixeldata[j] = pixeldata[j+2] = 0;
      pixeldata[j+1] = (JSAMPLE)(buf[i]*CF);
      break;
    case 6300 :
      pixeldata[j+1] = pixeldata[j+2] = 0;
      pixeldata[j] = (JSAMPLE)(buf[i]*CF);
      break;
    case 7774 :
    default :
      pixeldata[j] = pixeldata[j+1] = pixeldata[j+2] = (JSAMPLE)(buf[i]*CF);
      break;
    }
    i++;
  }
 }
 else
/*
 *initialize the pixeldata for single compnent pixels
 */
 while (i < wxh) {
  if (i < 40) {
    headerdata[i] = buf[i];
    pixeldata[i] = 0;
    i++;
  }
  else {
```

```
    pixeldata[i] = (JSAMPLE)(buf[i]*CF);
    i++;
  }
 }

/*
 *make the jpeg filename equal the keo filename,
 *but change the suffix
 */

 strcpy(jfilename, sFullFilename);
 dot = strrchr(jfilename, '.');
 strcpy(dot, ".jpg\0");
// printf("filename=%s\n", jfilename);

/*
 *initialize the comment string by converting headerdata
 *to a text string, add "hd:" in front as an identifier
 */

 sprintf(comment, "hd: ");
 for (i = 0; i < 40; i++) {
   dot = strrchr(comment, ' ');
   sprintf(++dot, "%d ", headerdata[i]);
 }
// printf("the comment is: %s\n", comment);

imanimage.err = (jpeg_std_error(&jpegerror));
jpeg_create_compress(&imanimage);

 if ((jpegfile = fopen(jfilename, "wb")) == NULL) {
   fprintf(stderr, "cannot open file: %s\n", jfilename);
   exit(1);
 }

jpeg_stdio_dest(&imanimage, jpegfile);

imanimage.image_width = width;
imanimage.image_height = height;
imanimage.input_components = pixparts;
imanimage.in_color_space = (FALSE_COLOR == TRUE) ? JCS_RGB :
JCS_GRAYSCALE;
jpeg_set_defaults(&imanimage);
jpeg_set_quality(&imanimage, 75, TRUE);

jpeg_start_compress(&imanimage, TRUE);
```

```
row_stride = (width*pixparts);
jpeg_write_marker(&imanimage, JPEG_COM, comment, strlen(comment));

 while (imanimage.next_scanline < imanimage.image_height) {
   row_pointer[0] = &pixeldata[imanimage.next_scanline * row_stride];
  (void) jpeg_write_scanlines(&imanimage, row_pointer, 1);
 }

//printf("False Color: %s\n", (FALSE_COLOR == TRUE) ? "ON" : "OFF");

jpeg_finish_compress(&imanimage);

fclose(jpegfile);

jpeg_destroy_compress(&imanimage);
}

void moment(float data[], int n, float *ave, float *sdev)
{
  int j;
  float ep=0.0,s,p,var;
  s=0.0;
  var=0.0;
  for (j=0;j<n;j++) s += data[j];
  *ave=s/n;

  for (j=0;j<n;j++) {
    s = data[j];
    ep += s;
    var += s*s;
  }
  var=(var-ep*ep/n)/(n-1);
  *sdev= sqrt(var);

}

void findindex(int n_points,int pixel_radius)
{
  int i;
  float distance,frand,x,y;

  for (i=0; i<n_points; i++) {
    distance = pixel_radius + 1;
    while( distance > pixel_radius) {
      frand = rand()/(float) RAND_MAX;
      x = 2.0 * pixel_radius * frand - pixel_radius;
```

```
      frand = rand()/(float) RAND_MAX;
      y = 2.0 * pixel_radius * frand - pixel_radius;
      distance = sqrt( x*x + y*y);
    }
//    randindex[i] = 335 + (long) x + 670 * ((long) y + 325);
    randindex[i] = (IMAGE_X_SIZE / 2) + (long) x + IMAGE_X_SIZE * ((long) y +
(IMAGE_Y_SIZE / 2)) ;
  }
}
```

6.2 Schedule File for 2001-2002 Winter: schedule.txt

```
154800 011207 192300 011207
135300 011208 231200 011208
124100 011209 021700 011210
114800 011210 061800 011211
092300 011211 103600 011211
113000 011211 160000 011221
173600 011221 133600 011222
213000 011222 123000 011223
001200 011224 113000 011224
025400 011225 101800 011225
123000 020104 202300 020104
123000 020105 233600 020105
124100 020106 030000 020107
083000 020107 094800 020107
124800 020107 094100 020108
125300 020108 093600 020109
130600 020109 093000 020110
131100 020110 092300 020111
131800 020111 091800 020112
132300 020112 091100 020113
133600 020113 090000 020114
134100 020114 085300 020115
134800 020115 084800 020116
135300 020116 084100 020117
140000 020117 083600 020118
183000 020118 083000 020119
211700 020119 082300 020120
235300 020120 081800 020121
030000 020122 073000 020122
154100 020131 171200 020131
154800 020201 204200 020201
155300 020202 000000 020203
160000 020203 064800 020204
160600 020204 064100 020205
161200 020205 063500 020206
161700 020206 063000 020207
163000 020207 062400 020208
163600 020208 061100 020209
164200 020209 060500 020210
164700 020210 060000 020211
165300 020211 055400 020212
170000 020212 054800 020213
```

170600 020213 054100 020214
171700 020214 053500 020215
183600 020215 052400 020216
211200 020216 051800 020217
000000 020218 051100 020218

6.3 Schedule File for 2002-2003 Winter: schedule.txt

```
151100 021128 220600 021128
134800 021129 012300 021130
122000 021130 050000 021201
105300 021201 151800 021211
200000 021211 134100 021212
230600 021212 124100 021213
014200 021214 113600 021214
043500 021215 101800 021215
140600 021225 190600 021225
122300 021226 223600 021226
111100 021227 014700 021228
100000 021228 122300 030108
201200 030108 111100 030109
225300 030109 101100 030110
013600 030111 090600 030111
133000 030121 154100 030121
133600 030122 195300 030122
134800 030123 230600 030123
082300 030124 085300 030124
135300 030124 031200 030125
061100 030125 084800 030125
140600 030125 083600 030126
141100 030126 083000 030127
142300 030127 082300 030128
143000 030128 081800 030129
143600 030129 080600 030130
144100 030130 080000 030131
145000 030131 075400 030201
150000 030201 074800 030202
150600 030202 074100 030203
151100 030203 073000 030204
170600 030204 072400 030205
200600 030205 071800 030206
224200 030206 071100 030207
014700 030208 060500 030208
170600 030219 201200 030219
171700 030220 235300 030221
051800 030221 053000 030221
172300 030221 052400 030222
173000 030222 051800 030223
173600 030223 051100 030224
174200 030224 050500 030225
```

174700 030225 045400 030226
180000 030226 044800 030227
180600 030227 044100 030228
181000 030228 043000 030301
081100 030301 081100 030301
181700 030301 042400 030302
183000 030302 041800 030303
183600 030303 041100 030304
184200 030304 040000 030305
195300 030305 035400 030306
224700 030306 034200 030307
212300 030320 011200 030321
051800 030321 051800 030321
214200 030321 005400 030322
220000 030322 003600 030323
223000 030323 000600 030324
045400 030324 045400 030324

6.4 Camera controller Serial Driver: SerialLib.c

```
/* This file: SerialLib.c version 1.0  CAL 08-Nov-01

Revision History:

Code is written to communicate with the ANIMATICS servo-motor controller (SMARTMOTOR)
that
is running the KEO version controller software

      Serial interface is COM1-COM4  (cua0-cua3)  @ 9600,8,N,1

Code sends the SMARTMOTOR commands in Immediate mode, and then polls for input
until a <cr> (\r or \13) is recieved as a terminating character.  A timeout of MAX_TIMEOUT
is checked (in milliseconds) and terminates the called routine.

SerialLib.c contains the libraries to initializes and sets up the comm port for communications with
the SMARTMOTOR controller.
*/

#include <termios.h>
#include <stdio.h>
#include <fcntl.h>
#include <string.h>
#include <time.h>

#include "SerialLib.h"



//GLOABL VARIABLES declarations

   char c;                          // character read into by read from serial port buffer
   int fd;                          // pointer to the serial port
   struct termios    oldtio,newtio;      // old and new terminal i/o structures
   unsigned char buf[MAXDATABYTES];

// end of GLOBAL VARIABLE declarations


int openSerial( char *port )
{

//  Open modem device for reading and writing and not as controlling tty because we don't want to
get
//  killed if linenoise sends CTRL-C.

   if ( strlen(port) < 6 )
      {
      printf("opening default port...\n");
         fd = open(MODEMDEVICE, O_RDWR | O_NOCTTY);    // not a valid port assignment
      }
   else
      {
```

```
      printf("opening port: %s...\n", port);
         fd = open( port, O_RDWR | O_NOCTTY);
      }

   if (fd <0)
    {
    printf( "\nCouldn't open serial port!" );
    return 0;                          // return 0 to indicate error reading port;
    }
   else
    printf( "\nSerial port successfully opened" );

   tcgetattr(fd,&oldtio);                  /* save current modem settings */

/*  Set bps rate and hardware flow control and 8n1 (8bit,no parity,1 stopbit). Also don't hangup
    automatically and ignore modem status. Finally enable receiving characters. */

   newtio.c_cflag = B9600 | CS8 | CLOCAL | CREAD;    /* (CAL) no hardware control for the
SMARTMOTOR

/*  Ignore bytes with parity errors and make terminal raw and dumb. */

   newtio.c_iflag = IGNPAR;

/* Raw output. */

   newtio.c_oflag = 0;

/*  Don't echo characters because if you connect to a host it or your modem will echo characters
for you.
     Don't generate signals. */

   newtio.c_lflag = 0;

/* now clean the modem line and activate the settings for modem */

   tcflush(fd, TCIFLUSH);
   tcsetattr(fd,TCSANOW,&newtio);

   newtio.c_cflag = BAUDRATE | CS8 | CLOCAL | CREAD;
   tcflush(fd, TCIFLUSH);
   tcsetattr(fd,TCSANOW,&newtio);

   printf( "\nSerial Communications all set up now...\n" );

   // Setup the address for the motor (#1) by sending an ASCII 129 to the motor...

   sprintf( buf, "\129" );                 /* Send an ASCII to 129 to address the motor
   tcflush(fd, TCIOFLUSH);
   write( fd, buf, strlen(buf) );             /* and send the whole command string */
   printf( "\nSMARTMOTOR has been address to #1 -- we are now ready to talk...\n\n" );

   return 1;
}            // end of 'openSerial()'
```

```
void closeSerial( void )
{
   tcsetattr(fd, TCSANOW, &oldtio );                    // reset to old terminal attributes
   close( fd );                    // and close the terminal
   return;
}        // end of 'closeSerial()'


int filterWheel( int next )
{
   int i,j, k, n_bytes_read, n_rets;
   unsigned char msg[25];
   clock_t start, end;
   double darg, elapsed;

   /* Try sending "move to filter #3 to the SMARTMOTOR */

   sprintf( msg, "g=%d\r", next );
   tcflush(fd, TCIOFLUSH);
   write( fd, msg, strlen(msg) );              /* and send the whole command string */

   sprintf( msg, "GOSUB4\r" );
   tcflush(fd, TCIOFLUSH);
   write( fd, msg, strlen(msg) );              /* and send the whole command string */

   start = clock();                          /* initialize timeout clock */
   elapsed = 0.0;                             /* initialize read duration */

   // Now read the entire buffer for SMARTMOTOR return

   i=0;
   j = 0;
   c = 0;
   n_rets = 1;                               //  SMARTMOTOR will only return one carriage return

   // printf ("\nSending command to SMARTMOTOR..." );

   while ((j < n_rets) && (elapsed < MAX_TIMEOUT))
      {
         n_bytes_read = read(fd,&c,1);            // read the SMARTMOTOR port

         if (n_bytes_read == 1)
            if ( (buf[i++] = c) == 13 )
               j++;                          // check for carriage returns
         end = clock();
         elapsed = ((double)(end-start)) * 1000 / CLOCKS_PER_SEC;
         }
   buf[i] = 0;                             // terminate the character string

   // printf( "\nFinished reading SMARTMOTOR.  %d bytes read in %2.1f msec, string is: %s\n", i,
elapsed, buf );

   if (elapsed < MAX_TIMEOUT)
     j = buf[strlen(buf)-2] - 48;
   else
     j = -1;
```

```
   return j;
}              // end of 'filterwheel'


int intensifierGain( int next )
{
   int i,j, k, n_bytes_read, n_rets;
   unsigned char msg[25];
   clock_t start, end;
   double darg, elapsed;

   // Set up the commands for the next intensifier gain

   sprintf( msg, "e=%d\r", next );
   tcflush(fd, TCIOFLUSH);
   write( fd, msg, strlen(msg) );                /* and send the whole command string */

   sprintf( msg, "GOSUB2\r" );
   tcflush(fd, TCIOFLUSH);
   write( fd, msg, strlen(msg) );                /* and send the whole command string */

   start = clock();                             /* initialize timeout clock */
   elapsed = 0.0;                               /* initialize read duration */

   // Now read the entire buffer for SMARTMOTOR return

   i=0;
   j = 0;
   c = 0;
   n_rets = 1;                                  // SMARTMOTOR will only return one carriage return

   // printf ("\nSending command to SMARTMOTOR..." );

   while ((j < n_rets) && (elapsed < MAX_TIMEOUT))
      {
         n_bytes_read = read(fd,&c,1);              // read the SMARTMOTOR port

         if (n_bytes_read == 1)
            if ( (buf[i++] = c) == 13 )
               j++;                             /* check for carriage returns */
         end = clock();
         elapsed = ((double)(end-start)) * 1000 / CLOCKS_PER_SEC;
         }
   buf[i] = 0;                                  // terminate the character string

   // printf( "\nFinished reading SMARTMOTOR.  %d bytes read in %2.1f msec, string is: %s\n", i,
elapsed, buf );

   if (elapsed < MAX_TIMEOUT)
     j = buf[strlen(buf)-2] - 48;
   else
     j = -1;

   // printf("intgain returns: %d\n", j);
   return j;
```

```
}               // end of 'intensifiergain'

int intensifierPower( int next )
{
   int i,j, k, n_bytes_read, n_rets;
   unsigned char msg[25], temp[25];
   clock_t start, end;
   double darg, elapsed;

   // Set up the commands for the next intensifier power

   sprintf( msg, "f=%d\r", next );
   tcflush(fd, TCIOFLUSH);
   write( fd, msg, strlen(msg) );                /* and send the whole command string */

   sprintf( msg, "GOSUB3\r" );
   tcflush(fd, TCIOFLUSH);
   write( fd, msg, strlen(msg) );                /* and send the whole command string */

   start = clock();                             /* initialize timeout clock */
   elapsed = 0.0;                               /* initialize read duration */

   // Now read the entire buffer for SMARTMOTOR return

   i=0;
   j = 0;
   c = 0;
   n_rets = 1;                                  // SMARTMOTOR will only return one carriage return

   // printf ("\nSending command to SMARTMOTOR..." );

   while ((j < n_rets) && (elapsed < MAX_TIMEOUT))
      {
         n_bytes_read = read(fd,&c,1);          // read the SMARTMOTOR port

         if (n_bytes_read == 1)
            if ( (buf[i++] = c) == 13 )
               j++;                             /* check for carriage returns */
         end = clock();
         elapsed = ((double)(end-start)) * 1000 / CLOCKS_PER_SEC;
         }
   buf[i] = 0;                                  // terminate the character string

   // printf( "\nFinished reading SMARTMOTOR.  %d bytes read in %2.1f msec, string is: %s\n", i,
elapsed, buf );

   if (elapsed < MAX_TIMEOUT)                   // received good return from SMARTMOTOR
    {
    strcpy( temp, buf+7 );
    // printf("\nthe temp string is %s\n", temp );

    if ( strncmp(temp, "OF", 2) )
       j = 1;
    else
       j = 0;
    }
```

```
   else                          // SMARTMOTOR timed out, return error
    j = -1;

   return j;
}              // end of 'intensifierPower'


int shutter( int next )
{
   int i,j, k, n_bytes_read, n_rets;
   unsigned char msg[25], temp[25];
   clock_t start, end;
   double darg, elapsed;

   // Set up the commands for the next intensifier power

   sprintf( msg, "d=%d\r", next );
   tcflush(fd, TCIOFLUSH);
   write( fd, msg, strlen(msg) );              /* and send the whole command string */

   sprintf( msg, "GOSUB1\r" );
   tcflush(fd, TCIOFLUSH);
   write( fd, msg, strlen(msg) );              /* and send the whole command string */

   start = clock();                          /* initialize timeout clock */
   elapsed = 0.0;                             /* initialize read duration */

   // Now read the entire buffer for SMARTMOTOR return

   i=0;
   j = 0;
   c = 0;
   n_rets = 1;                               // SMARTMOTOR will only return one carriage return

   // printf ("\nSending command to SMARTMOTOR..." );

   while ((j < n_rets) && (elapsed < MAX_TIMEOUT))
     {
        n_bytes_read = read(fd,&c,1);          // read the SMARTMOTOR port

        if (n_bytes_read == 1)
           if ( (buf[i++] = c) == 13 )
              j++;                          /* check for carriage returns */
        end = clock();
        elapsed = ((double)(end-start)) * 1000 / CLOCKS_PER_SEC;
        }
   buf[i] = 0;                              // terminate the character string
   // pn 07dec01 next line commented out
   //    printf( "\nFinished reading SMARTMOTOR.  %d bytes read in %2.1f msec, string is: %s\n",
i, elapsed, buf );

   if (elapsed < MAX_TIMEOUT)          // received good return from SMARTMOTOR
    {
    strcpy( temp, buf+4 );
    // printf("\nthe temp string is %s\n", temp );
```

```
    if ( strncmp(temp, "???", 3) )
       j = next;
    else
       j = -2;                    // read error?
    }
   else                          // SMARTMOTOR timed out, return error
    j = -1;

   return j;
}              // end of 'shutter'

int lightStatus( int next )
{
   int i,j, k, n_bytes_read, n_rets;
   unsigned char msg[25];
   clock_t start, end;
   double darg, elapsed;

   // Set up the commands for the next intensifier power

   sprintf( msg, "GOSUB0\r" );
   tcflush(fd, TCIOFLUSH);
   write( fd, msg, strlen(msg) );              /* and send the whole command string */

   start = clock();                           /* initialize timeout clock */
   elapsed = 0.0;                              /* initialize read duration */

   // Now read the entire buffer for SMARTMOTOR return

   i=0;
   j = 0;
   c = 0;
   n_rets = 1;                                // SMARTMOTOR will only return one carriage return

   // printf ("\nSending command to SMARTMOTOR..." );

   while ((j < n_rets) && (elapsed < MAX_TIMEOUT))
      {
         n_bytes_read = read(fd,&c,1);          // read the SMARTMOTOR port

         if (n_bytes_read == 1)
            if ( (buf[i++] = c) == 13 )
               j++;                          /* check for carriage returns */
         end = clock();
         elapsed = ((double)(end-start)) * 1000 / CLOCKS_PER_SEC;
         }
   buf[i] = 0;                              // terminate the character string

   // printf( "\nFinished reading SMARTMOTOR.  %d bytes read in %2.1f msec, string is: %s\n", i,
elapsed, buf );

   if (elapsed < MAX_TIMEOUT)
     i = buf[6] - 48;
   else
     i = -1;
```

```
   return i;
}              // end of 'lightStatus'

int sendReset( void )
{
   int i,j, k, n_bytes_read, n_rets;
   unsigned char msg[25];
   clock_t start, end;
   double darg, elapsed;

   // Set up the commands for the next intensifier power

   sprintf( msg, "Z\r" );
   tcflush(fd, TCIOFLUSH);
   write( fd, msg, strlen(msg) );                  /* and send the whole command string */

   start = clock();                               /* initialize timeout clock */
   elapsed = 0.0;                                  /* initialize read duration */

   // Now read the entire buffer for SMARTMOTOR return

   i=0;
   j = 0;
   c = 0;
   n_rets = 1;                                    //  SMARTMOTOR will only return one carriage return

   // printf ("\nSending command to SMARTMOTOR..." );

   while ((j < n_rets) && (elapsed < 20000))       // long timeout for HOME routine
      {
         n_bytes_read = read(fd,&c,1);             // read the SMARTMOTOR port

         if (n_bytes_read == 1)
            if ( (buf[i++] = c) == 13 )
               j++;                                /* check for carriage returns */
         end = clock();
         elapsed = ((double)(end-start)) * 1000 / CLOCKS_PER_SEC;
         }
   buf[i] = 0;                                    // terminate the character string

   // printf( "\nFinished reading SMARTMOTOR.  %d bytes read in %2.1f msec, string is: %s\n", i,
elapsed, buf );

   if (elapsed < 20000)
     j = 1;
   else
     j = -1;

   // printf("\nj = %d\n", j );
   return j;
}              // end of 'sendReset'
```

6.5 Include file for SerialLib.c: SerialLib.h

```
#define MAXDATABYTES 255

#define BAUDRATE B9600                  // Data collecting baud rate
#define MODEMDEVICE "/dev/cua0"         // currently set for COM1 (cua0)

// #define _POSIX_SOURCE 1              // POSIX compliant source

#define MAX_TIMEOUT 2000.0              // TIMEOUT parameter in milliseconds
#define MAX_FPOS 6                      // Maximum filterwheel position
#define MIN_FPOS 1                      // Minimum filterwheel position
#define MAX_GAIN 3                      // Maximum intensifier gain
#define MIN_GAIN 0                      // Minimum intensifier gain

int  filterWheel( int );            // call to set filterwheel position
int  intensifierGain( int );        // call to set intensifier gain
int  intensifierPower(int );        // call to set intensifier power
int  lightStatus( int );            // call to check Light Status
int  shutter( int );                // call to the shutter control
int  sendReset( void );             // call to do a software reset
int  openSerial( char * );          // call to open the serial port
void closeSerial( void );           // call to close the serial port
```

6.6 IDL Analysis Routines: asiptools.pro

```
PRO tvImage,fileName
XSIZE=670 & YSIZE=650
i16Image=UINTARR(XSIZE,YSIZE)
i16ImgParams=INTARR(16)
OPENR,lunImageFile,fileName,/GET_LUN
READU,lunImageFile,i16Image
FREE_LUN,lunImageFile
i16ImgParams=i16Image(0:20)
;WINDOW,0,XSIZE=335,YSIZE=325 & wset,0    ; For MPEG movies
;TVSCL,ROTATE(REBIN(i16Image,335,325),2)  ;
TVSCL,ROTATE(i16Image,2)    ; Flip both X and Y dimensions
;TVSCL,i16Image
;XYOUTS,0.02,0.96,align=0.,STRING(i16ImgParams(3:5),FORMAT='(I2.2,":",I2.2,":",I2.2)'),/norm
al,charsize=1.0
;XYOUTS,0.02,1.96,align=0.,STRING(i16ImgParams(0),FORMAT='(I4.4)'),/normal,charsize=1.0

myCharSize=1.25
monthname=['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec']
header=i16Image(0:25)
year=string(header(0))
if header(1) LE 12 and header(1) GT 0 then month=' '+monthname(header(1)-1) else month=' Err'
day=strcompress(string(header(2)),/remove)
hour=header(3)
minute=header(4)
second=header(5)
time=string(hour,format='(I2.2)')+':'+string(minute,format='(I2.2)')+':'$
        +string(second,format='(I2.2)')+' UT'
wave=strcompress(string(header(6)),/rem)+string(byte(197))
gain=strcompress(string(header(7)),/rem)
expos=strcompress(string(float(header(8)/10.),format='(F6.1)'),/rem)
xyouts,0.02,.96,align=0.,strcompress(day+month+year)+'!C'+time $
        +'!C'+wave+'!C'+expos+'s'+'!CG: '+gain,/normal,charsize=myCharSize
;xyouts,0.98,.96,align=1.,charsize=0.75,'Expos:'+strcompress(expos)+'!C'+'Filt:'+strcompress(wav
e)+'!C'+'Gain:'+strcompress(gain),/normal
;xyouts,0.02,0.02,align=0.,'Expos:'+strcompress(expos),/normal
if n_elements(location) NE 0 then
xyouts,0.02,0.01,align=0.,location,/normal,charsize=myCharSize
if n_elements(equip) NE 0 then xyouts,.98,.01,align=1.,'AFRL '+equip+'
Imager',/normal,charsize=myCharSize $
        else xyouts,.98,.01,align=1.,'AFOSR All-Sky Imager',/normal,charsize=myCharSize
;xyouts,0.98,0.02,align=1.,'Gain:'+strcompress(gain),/normal
xyouts,0.5,0.96,/normal,align=.5,'N',charsize=myCharSize
xyouts,0.5,0.01,/normal,align=0.5,'S',charsize=myCharSize
xyouts,0.01,0.5,/normal,'W',charsize=myCharSize
xyouts,0.98,0.5,/normal,align=1.,'E',charsize=myCharSize
xyouts,.02,.01,align=0.,'Ny Alesund, Svalbard',/normal,charsize=myCharSize
END


PRO MARKAZEL,XSIZE,YSIZE,XO,YO,B_COEFF,C_COEFF,AZ_OFFSET,AZ,EL,LABEL
  R=B_COEFF*SIN((90.-EL)*!DTOR)+C_COEFF*SIN((2.*(90.-EL))*!DTOR)
  X=XO+R*SIN((AZ+AZ_OFFSET)*!DTOR)
```

```
  Y=YO+R*COS((AZ+AZ_OFFSET)*!DTOR)
  PLOTS,X,Y,PSYM=6,/DEVICE                    ; '+' = PSYM=1
  XYOUTS,X,Y,LABEL,/DEVICE
END


PRO MAKEGRID,XSIZE,YSIZE,XO,YO,B_COEFF,C_COEFF,AZ_OFFSET

FOR EL=90,0,-10 DO BEGIN
  R=B_COEFF*SIN((90.-EL)*!DTOR)+C_COEFF*SIN((2.*(90.-EL))*!DTOR)
  FOR AZ=0,360 DO BEGIN
       X=XO+R*SIN((AZ+AZ_OFFSET)*!DTOR)
       Y=YO+R*COS((AZ+AZ_OFFSET)*!DTOR)
    IF AZ EQ 0 THEN BEGIN
         IF EL NE 90 THEN XYOUTS,X,Y,STRCOMPRESS(EL),/DEVICE
       PLOTS,X,Y,/DEVICE
       ENDIF
    PLOTS,X,Y,/CONTINUE,LINESTYLE=2,/DEVICE
  ENDFOR
ENDFOR

FOR AZ=0,330,30 DO BEGIN
  PLOTS,XO,YO,/DEVICE
  X=XO+R*SIN((AZ+AZ_OFFSET)*!DTOR)
  Y=YO+R*COS((AZ+AZ_OFFSET)*!DTOR)
  PLOTS,X,Y,/CONTINUE,/DEVICE
  XYOUTS,X,Y,STRCOMPRESS(AZ),/DEVICE
ENDFOR

END


PRO STARMAP

XSIZE=670 & YSIZE=650 & XO=342 & YO=302 & B_COEFF=284 & C_COEFF=-29 &
AZ_OFFSET=0

TVIMAGE,'~/nya/starmap/011218_102935_6560_NYA.keo'

MARKAZEL,XSIZE,YSIZE,XO,YO,B_COEFF,C_COEFF,AZ_OFFSET,334.5,13,' Jupiter'
MARKAZEL,XSIZE,YSIZE,XO,YO,B_COEFF,C_COEFF,AZ_OFFSET,152.5,49,' Vega'
MARKAZEL,XSIZE,YSIZE,XO,YO,B_COEFF,C_COEFF,AZ_OFFSET,226,27,' Arcturus'
MARKAZEL,XSIZE,YSIZE,XO,YO,B_COEFF,C_COEFF,AZ_OFFSET,357.5,35,' Capella'
MARKAZEL,XSIZE,YSIZE,XO,YO,B_COEFF,C_COEFF,AZ_OFFSET,2,78.5,' Polaris'
MARKAZEL,XSIZE,YSIZE,XO,YO,B_COEFF,C_COEFF,AZ_OFFSET,48,54.5,' Cas Gamm-27'
MARKAZEL,XSIZE,YSIZE,XO,YO,B_COEFF,C_COEFF,AZ_OFFSET,241.5,55.5,' Benetnasch'
MARKAZEL,XSIZE,YSIZE,XO,YO,B_COEFF,C_COEFF,AZ_OFFSET,290,59.5,' Dubhe'
MARKAZEL,XSIZE,YSIZE,XO,YO,B_COEFF,C_COEFF,AZ_OFFSET,115.5,51,' Denab'

MAKEGRID,XSIZE,YSIZE,XO,YO,B_COEFF,C_COEFF,AZ_OFFSET

myCharSize=1.25
xyouts,.98,.96,align=1.,'R=B_COEFF*SIN((90.-EL)*!!DTOR)+C_COEFF*SIN((2.*(90.-
EL))*!!DTOR)'+ $
  '!CX=XO+R*SIN((AZ+AZ_OFFSET)*!!DTOR)'+'!CY=YO+R*COS((AZ+AZ_OFFSET)*!!DTOR)'+
$
```

```
  '!CB_COEFF=284 C_COEFF=-29!CXO=342 YO=302!CAZ_OFFSET=0'+ $
  '!C78d 55m 24.5s N LAT!C11d 55m 48.0s E LON!C63m ALT',/normal,charsize=myCharSize

T=TVRD()
WRITE_JPEG,'~/nya/starmap/011218_starmap.jpg',T,QUALITY=100
END


PRO MAKEMPEG,imageListFilename,outFileName                ;pn 16jan02

;CD,'~/mpegtest/images'

filename=''
OPENR,imageListFile,imageListFileName,/GET_LUN
mpegID = MPEG_OPEN([335,325])
i =0
WHILE (NOT EOF(imageListFile)) DO BEGIN
  READF,imageListFile,filename
  PRINT,filename
  tvimage,filename
  MPEG_PUT,mpegID,WINDOW=0 ,FRAME=i,/ORDER
  i = i + 1
ENDWHILE

FREE_LUN,imageListFile

MPEG_SAVE,mpegID,FILENAME=outFileName
MPEG_CLOSE,mpegID
WDELETE,0
END                              ;End of program

PRO GETSPE,fileName,SPEIMAGE
i16Image=UINTARR(670,650)
bImgParams=BYTARR(4100)
OPENR,lunImageFile,fileName,/GET_LUN
READU,lunImageFile,bImgParams,i16Image
SPEIMAGE=i16Image
FREE_LUN,lunImageFile
END


PRO SPEDIFF,AFILE,BFILE,MEAN
  GETSPE,AFILE,AIMAGE
  GETSPE,BFILE,BIMAGE
  IAI=ROTATE(LONG(AIMAGE)-LONG(BIMAGE),2)

IMAGE_STATISTICS,IAI[330:360,260:290],MEAN=MEAN,STDDEV=SD,MINIMUM=MIN,MAXIM
UM=MAX
  PRINT,MEAN,SD,MIN,MAX,FORMAT='(%"MEAN=%9.2f SD=%9.2f MIN=%9.2f MAX=%9.2f")'
END


PRO CCDBIAS
  CD,'/mnt/w2k/NYA FEB02 CAL/CCD NOISE'
  GETSPE,'G2_e1_ccd_f1.SPE',SIMAGE
  IAI=ROTATE(SIMAGE,2)
```

```
IMAGE_STATISTICS,IAI[330:360,260:290],MEAN=MEAN,STDDEV=SD,MINIMUM=MIN,MAXIM
UM=MAX
  PRINT,MEAN,SD,MIN,MAX,FORMAT='(%"MEAN=%9.2f SD=%9.2f MIN=%9.2f MAX=%9.2f")'
END

PRO DARKNOISE

  PRINT,"GAIN 0: "
  CD,'/mnt/w2k/NYA FEB02 CAL/INT NOISE'
  SPEDIFF,'/mnt/w2k/NYA FEB02 CAL/INT NOISE/G0_e1_int_f1.SPE', $
       '/mnt/w2k/NYA FEB02 CAL/CCD NOISE/G2_e1_ccd_f1.SPE', MEAN
  PRINT,"GAIN 1: "
  CD,'/mnt/w2k/NYA FEB02 CAL/INT NOISE'
  SPEDIFF,'/mnt/w2k/NYA FEB02 CAL/INT NOISE/G1_e1_int_f1.SPE', $
       '/mnt/w2k/NYA FEB02 CAL/CCD NOISE/G2_e1_ccd_f1.SPE', MEAN
  PRINT,"GAIN 2: "
  CD,'/mnt/w2k/NYA FEB02 CAL/INT NOISE'
  SPEDIFF,'/mnt/w2k/NYA FEB02 CAL/INT NOISE/G2_e1_int_f1.SPE', $
       '/mnt/w2k/NYA FEB02 CAL/CCD NOISE/G2_e1_ccd_f1.SPE', MEAN
END

PRO CAL6300                    ; 6300/16A PO:00-168 LOT 0501
  PRINT,"GAIN 0: "
  SPEDIFF,'/mnt/w2k/NYA FEB02 CAL/GAIN 0 SRC & BKGD/G0_E1_A0B09_F1.SPE', $
       '/mnt/w2k/NYA FEB02 CAL/GAIN 0 SRC & BKGD/G0_e1_BKGD_f1.SPE', MEAN
  PRINT,"LCR=", MEAN / (5.925E-3 * 8.61E4 * 16.)
  PRINT,"GAIN 1: "
  SPEDIFF,'/mnt/w2k/NYA FEB02 CAL/GAIN 1 SRC & BKGC/G1_E1_A0B05_F1.SPE', $
       '/mnt/w2k/NYA FEB02 CAL/GAIN 1 SRC & BKGC/G1_E1_BKGD_F1.SPE',MEAN
  PRINT,"LCR=", MEAN / (4.429E-4 * 8.61E4 * 16.)
  PRINT,"GAIN 2: "
  SPEDIFF,'/mnt/w2k/NYA FEB02 CAL/GAIN 2 SRC & BKGD/G2_E1_A0B02_F1.SPE', $
       '/mnt/w2k/NYA FEB02 CAL/GAIN 2 SRC & BKGD/G2_E1_BKGD_F1.SPE',MEAN
  PRINT,"LCR=", MEAN / (5.518E-5 * 8.61E4 * 16.)
END

PRO CAL5577                    ; 5577/16A PO:00-168 LOT 0601
  PRINT,"GAIN 0: "
  SPEDIFF,'/mnt/w2k/NYA FEB02 CAL/GAIN 0 SRC & BKGD/G0_E1_A0B09_F2.SPE', $
       '/mnt/w2k/NYA FEB02 CAL/GAIN 0 SRC & BKGD/G0_e1_BKGD_f2.SPE', MEAN
  PRINT,"LCR=", MEAN / (5.925E-3 * 4.4579E4 * 16.)
  PRINT,"GAIN 1: "
  SPEDIFF,'/mnt/w2k/NYA FEB02 CAL/GAIN 1 SRC & BKGC/G1_E1_A0B05_F2.SPE', $
       '/mnt/w2k/NYA FEB02 CAL/GAIN 1 SRC & BKGC/G1_E1_BKGD_F2.SPE',MEAN
  PRINT,"LCR=", MEAN / (4.429E-4 * 4.4579E4 * 16.)
  PRINT,"GAIN 2: "
  SPEDIFF,'/mnt/w2k/NYA FEB02 CAL/GAIN 2 SRC & BKGD/G2_E1_A0B02_F2.SPE', $
       '/mnt/w2k/NYA FEB02 CAL/GAIN 2 SRC & BKGD/G2_E1_BKGD_F2.SPE',MEAN
  PRINT,"LCR=", MEAN / (5.518E-5 * 4.4579E4 * 16.)
END

PRO CAL4278                    ; 4278/25A PO:00-168 LOT 4801
  PRINT,"GAIN 0: "
  SPEDIFF,'/mnt/w2k/NYA FEB02 CAL/GAIN 0 SRC & BKGD/G0_E1_A0B09_F3.SPE', $
       '/mnt/w2k/NYA FEB02 CAL/GAIN 0 SRC & BKGD/G0_e1_BKGD_f3.SPE', MEAN
```

```
  PRINT,"LCR=", MEAN / (5.925E-3 * 6.8416E3 * 25.)
  PRINT,"GAIN 1: "
  SPEDIFF,'/mnt/w2k/NYA FEB02 CAL/GAIN 1 SRC & BKGC/G1_E1_A0B05_F3.SPE', $
       '/mnt/w2k/NYA FEB02 CAL/GAIN 1 SRC & BKGC/G1_E1_BKGD_F3.SPE',MEAN
  PRINT,"LCR=", MEAN / (4.429E-4 * 6.8416E3 * 25.)
  PRINT,"GAIN 2: "
  SPEDIFF,'/mnt/w2k/NYA FEB02 CAL/GAIN 2 SRC & BKGD/G2_E1_A0B02_F3.SPE', $
       '/mnt/w2k/NYA FEB02 CAL/GAIN 2 SRC & BKGD/G2_E1_BKGD_F3.SPE',MEAN
  PRINT,"LCR=", MEAN / (5.518E-5 * 6.8416E3 * 25.)
END

PRO CAL7774                    ; 7774/16A PO:00-168 LOT 0501
  PRINT,"GAIN 0: "
  SPEDIFF,'/mnt/w2k/NYA FEB02 CAL/GAIN 0 SRC & BKGD/G0_E1_A0B09_F4.SPE', $
        '/mnt/w2k/NYA FEB02 CAL/GAIN 0 SRC & BKGD/G0_e1_BKGD_f4.SPE', MEAN
  PRINT,"LCR=", MEAN / (5.925E-3 * 1.4208E5 * 16.)
  PRINT,"GAIN 1: "
  SPEDIFF,'/mnt/w2k/NYA FEB02 CAL/GAIN 1 SRC & BKGC/G1_E1_A0B05_F4.SPE', $
       '/mnt/w2k/NYA FEB02 CAL/GAIN 1 SRC & BKGC/G1_E1_BKGD_F4.SPE',MEAN
  PRINT,"LCR=", MEAN / (4.429E-4 * 1.4208E5 * 16.)
  PRINT,"GAIN 2: "
  SPEDIFF,'/mnt/w2k/NYA FEB02 CAL/GAIN 2 SRC & BKGD/G2_E1_A0B02_F4.SPE', $
       '/mnt/w2k/NYA FEB02 CAL/GAIN 2 SRC & BKGD/G2_E1_BKGD_F4.SPE',MEAN
  PRINT,"LCR=", MEAN / (5.518E-5 * 1.4208E5 * 16.)
END

PRO CAL6560                    ; 6560/30A PO: 00-168 LOT 0401
  PRINT,"GAIN 0: "
  SPEDIFF,'/mnt/w2k/NYA FEB02 CAL/GAIN 0 SRC & BKGD/G0_E1_A0B09_F5.SPE', $
        '/mnt/w2k/NYA FEB02 CAL/GAIN 0 SRC & BKGD/G0_e1_BKGD_f5.SPE', MEAN
  PRINT,"LCR=", MEAN / (5.925E-3 * 9.93E4 * 30.)
  PRINT,"GAIN 1: "
  SPEDIFF,'/mnt/w2k/NYA FEB02 CAL/GAIN 1 SRC & BKGC/G1_E1_A0B05_F5.SPE', $
       '/mnt/w2k/NYA FEB02 CAL/GAIN 1 SRC & BKGC/G1_E1_BKGD_F5.SPE',MEAN
  PRINT,"LCR=", MEAN / (4.429E-4 * 9.93E4 * 30.)
  PRINT,"GAIN 2: "
  SPEDIFF,'/mnt/w2k/NYA FEB02 CAL/GAIN 2 SRC & BKGD/G2_E1_A0B02_F5.SPE', $
       '/mnt/w2k/NYA FEB02 CAL/GAIN 2 SRC & BKGD/G2_E1_BKGD_F5.SPE',MEAN
  PRINT,"LCR=", MEAN / (5.518E-5 * 9.93E4 * 30.)
END

PRO VIGNETTE

  GETSPE,'/mnt/w2k/NYA FEB02 CAL/VIGNETTING/G0_e4_vig_f1.SPE',SIMAGE
  GETSPE,'/mnt/w2k/NYA FEB02 CAL/VIGNETTING/G0_e4_vig_f2.SPE',BIMAGE
  VIMAGE= SIMAGE > BIMAGE
  GETSPE,'/mnt/w2k/NYA FEB02 CAL/VIGNETTING/G0_e4_vig_f4.SPE',BIMAGE
  VIMAGE= VIMAGE > BIMAGE
  GETSPE,'/mnt/w2k/NYA FEB02 CAL/VIGNETTING/G0_e4_vig_f6.SPE',BIMAGE
  VIMAGE= VIMAGE > BIMAGE
  GETSPE,'/mnt/w2k/NYA FEB02 CAL/VIGNETTING/G0_e4_vig_f8.SPE',BIMAGE
  VIMAGE= VIMAGE > BIMAGE
  GETSPE,'/mnt/w2k/NYA FEB02 CAL/VIGNETTING/G0_e4_vig_f9.SPE',BIMAGE
  VIMAGE= VIMAGE > BIMAGE
  GETSPE,'/mnt/w2k/NYA FEB02 CAL/VIGNETTING/G0_e4_vig_F10.SPE',BIMAGE
  VIMAGE= VIMAGE > BIMAGE
```

```
  GETSPE,'/mnt/w2k/NYA FEB02 CAL/VIGNETTING/G0_e4_vig_F11.SPE',BIMAGE
  VIMAGE= VIMAGE > BIMAGE
  GETSPE,'/mnt/w2k/NYA FEB02 CAL/VIGNETTING/G0_e4_vig_F12.SPE',BIMAGE
  VIMAGE= VIMAGE > BIMAGE
  GETSPE,'/mnt/w2k/NYA FEB02 CAL/VIGNETTING/G0_e4_vig_F13.SPE',BIMAGE
  VIMAGE= VIMAGE > BIMAGE

  GETSPE,'/mnt/w2k/NYA FEB02 CAL/INT NOISE/G0_e4_int_f1.SPE',INTBACKGD

  VIMAGE=ROTATE(LONG(VIMAGE)-LONG(INTBACKGD),2)


IMAGE_STATISTICS,VIMAGE[330:360,260:290],MEAN=MEAN,STDDEV=SD,MINIMUM=MIN,M
AXIMUM=MAX
  PRINT,MEAN,SD,MIN,MAX,FORMAT='(%"MEAN=%9.2f SD=%9.2f MIN=%9.2f MAX=%9.2f")'
  ZCOUNT=MEAN

IMAGE_STATISTICS,VIMAGE[440:470,260:290],MEAN=MEAN,STDDEV=SD,MINIMUM=MIN,M
AXIMUM=MAX
  PRINT,MEAN,SD,MIN,MAX,FORMAT='(%"MEAN=%9.2f SD=%9.2f MIN=%9.2f MAX=%9.2f")'
  COUNTAT103=MEAN

IMAGE_STATISTICS,VIMAGE[605:615,260:290],MEAN=MEAN,STDDEV=SD,MINIMUM=MIN,M
AXIMUM=MAX
  PRINT,MEAN,SD,MIN,MAX,FORMAT='(%"MEAN=%9.2f SD=%9.2f MIN=%9.2f MAX=%9.2f")'


  ;y=A+Bx -> A=y-Bx, B=(ZCOUNT-COUNTAT103)/103.
  PRINT,'A=', ZCOUNT-(ZCOUNT-COUNTAT103)/103.*284.
  PRINT,'1-NORMALIZED=',1.-(ZCOUNT-(ZCOUNT-COUNTAT103)/103.*284.)/ZCOUNT

END

PRO GETCOEFF

  Y=FLTARR(91)
  X=FLTARR(91)
  XSIZE=670 & YSIZE=650 & XO=342 & YO=302 & B_COEFF=284 & C_COEFF=-29 &
AZ_OFFSET=0
  FOR I=0,90 DO BEGIN
    X(I)=B_COEFF*SIN((I)*!DTOR)+C_COEFF*SIN((2.*(I))*!DTOR)
      Y(I)=I
;    PRINT,Y(I),X(I)
  ENDFOR

  COEFFS=POLY_FIT(X,Y,4,/DOUBLE)
  PRINT,COEFFS

END
```

6.7 IDL Interactive GUI-based program: allsky.pro

```
;;; Modified from version 1.5 for new ASIP    Peter Ning 13Mar03

;TUESDAY, DECEMBER 22, 1998, 8:55:19 AM -- version 1.5

;;;;;;;;;;;;;;;;;;
;;; PRO ALLSKY ;;;
;;;;;;;;;;;;;;;;;;

  PRO ALLSKY

  BASE=WIDGET_BASE(TITLE='ALLSKY TEST FOR NYA01 DATA', /COLUMN)

  TOPROW=WIDGET_BASE(BASE, /FRAME, /ROW)
  BOTROW=WIDGET_BASE(BASE, /FRAME, /ROW)

  W_FILE=WIDGET_BUTTON(TOPROW, VALUE='FILE', /MENU)
  W_GRAPH=WIDGET_BUTTON(TOPROW, VALUE='GRAPH', /MENU)
  W_OPTION=WIDGET_BUTTON(TOPROW, VALUE='OPTIONS', /MENU)
  W_COORD=WIDGET_BUTTON(TOPROW, VALUE='COORDINATES', /MENU)
  W_SYSTEM=WIDGET_BUTTON(TOPROW, VALUE='SYSTEM', /MENU)

  W_FILE_B1=WIDGET_BUTTON(W_FILE, VALUE='OPEN', UVALUE='OPEN')
  W_FILE_B2=WIDGET_BUTTON(W_FILE, VALUE='QUIT', UVALUE='QUIT')

  W_GRAPH_B1=WIDGET_BUTTON(W_GRAPH, VALUE='XWINDOW', $
    UVALUE='XWINDOW')
  W_GRAPH_B2=WIDGET_BUTTON(W_GRAPH, VALUE='PS FILE', $
    UVALUE='PS FILE')
  W_GRAPH_B3=WIDGET_BUTTON(W_GRAPH, VALUE='SAMPLE', $
    UVALUE='SAMPLE')

  OPTION=FLTARR(3)
  W_OPTION_B1=WIDGET_BUTTON(W_OPTION, VALUE='  MERIDIAN', $
    UVALUE='MERIDIAN')
  OPTION(0)=0
  W_OPTION_B2=WIDGET_BUTTON(W_OPTION, VALUE='+ NY ALESUND', $
    UVALUE='ALESUND')
  OPTION(1)=1
  W_OPTION_B3=WIDGET_BUTTON(W_OPTION, VALUE='  LONGYEARBYEN', $
    UVALUE='LONGYEAR')
  OPTION(2)=0

  W_COORD_B1=WIDGET_BUTTON(W_COORD, VALUE='+ GEOGRAPHIC', $
    UVALUE='GEOGRAPHIC')
  W_COORD_B2=WIDGET_BUTTON(W_COORD, VALUE='  MAGNETIC', $
    UVALUE='MAGNETIC')

  W_SYS_B1=WIDGET_BUTTON(W_SYSTEM, VALUE='+ BIT SWAP', $
    UVALUE='BITSWAP')

  LCOL=WIDGET_BASE(BOTROW, /FRAME, /COLUMN)
  RCOL=WIDGET_BASE(BOTROW, /FRAME, /COLUMN)
```

```
  TRCOL=WIDGET_BASE(RCOL, /FRAME, /COLUMN)
  MRCOL=WIDGET_BASE(RCOL, /FRAME, /COLUMN)
  BRCOL=WIDGET_BASE(RCOL, /FRAME, /COLUMN)

  MIDROW=WIDGET_BASE(LCOL, /ROW)

  WMR_DATE_BASE=WIDGET_BASE(MIDROW, /ROW)
  W_DATE_LABEL=WIDGET_LABEL(WMR_DATE_BASE, VALUE="DATE:")
  W_DATE_TEXT=WIDGET_LABEL(WMR_DATE_BASE, VALUE=" ?????????")

  WMR_TIME_BASE=WIDGET_BASE(MIDROW, /ROW)
  W_TIME_LABEL=WIDGET_LABEL(WMR_TIME_BASE, VALUE="  TIME:")
  W_TIME_TEXT=WIDGET_LABEL(WMR_TIME_BASE, VALUE="??:??:??")

  WMR_FILTER_BASE=WIDGET_BASE(MIDROW, /ROW)
  W_FILTER_LABEL=WIDGET_LABEL(WMR_FILTER_BASE, VALUE="  FILTER:")
  W_FILTER_TEXT=WIDGET_LABEL(WMR_FILTER_BASE, VALUE="????")

  WMR_ALT_BASE=WIDGET_BASE(MIDROW, /ROW)
  W_ALT_LABEL=WIDGET_LABEL(WMR_ALT_BASE, VALUE="  ALT:")
  W_ALT_TEXT=WIDGET_LABEL(WMR_ALT_BASE, VALUE="???.?")

  WMR_FILE_BASE=WIDGET_BASE(MIDROW, /ROW)
  W_FILE_LABEL=WIDGET_LABEL(WMR_FILE_BASE, VALUE="  FILE:")
  W_FILE_TEXT=WIDGET_LABEL(WMR_FILE_BASE,
VALUE="??????????????????????????")

  WXSZ=670 ;pn - was 500
  WYSZ=650 ;pn - was 500
  WX=0
  WY=0
  W_DRAW=WIDGET_DRAW(LCOL, XSIZE=WXSZ, YSIZE=WYSZ, $
    /MOTION_EVENTS, RETAIN=2)

  BOTROW=WIDGET_BASE(LCOL, /ROW)

  W_LAT_BASE=WIDGET_BASE(BOTROW, /ROW)
  W_LAT_LABEL=WIDGET_LABEL(W_LAT_BASE, VALUE="LATITUDE:")
  W_LAT_TEXT=WIDGET_LABEL(W_LAT_BASE, VALUE=' 000.00')

  W_LON_BASE=WIDGET_BASE(BOTROW, /ROW)
  W_LON_LABEL=WIDGET_LABEL(W_LON_BASE, VALUE="LONGITUDE:")
  W_LON_TEXT=WIDGET_LABEL(W_LON_BASE, VALUE=' 000.00')

  W_RAY_BASE=WIDGET_BASE(BOTROW, /ROW)
  W_RAY_LABEL=WIDGET_LABEL(W_RAY_BASE, VALUE="RAYLEIGH:")
  W_RAY_TEXT=WIDGET_LABEL(W_RAY_BASE, VALUE='0000.0')

  W_EXP_BASE=WIDGET_BASE(BOTROW, /ROW)
  W_EXP_LABEL=WIDGET_LABEL(W_EXP_BASE, VALUE="EXPOSURE:")
  W_EXP_TEXT=WIDGET_LABEL(W_EXP_BASE, VALUE='00.0')

  W_GAIN_BASE=WIDGET_BASE(BOTROW, /ROW)
  W_GAIN_LABEL=WIDGET_LABEL(W_GAIN_BASE, VALUE="GAIN:")
  W_GAIN_TEXT=WIDGET_LABEL(W_GAIN_BASE, VALUE='0')
```

```
;DEFAULT READ DIRECTORY
  DIR='~/nya/starmap/'
  PDIR=DIR
  W_DIRECTORY_LABEL=WIDGET_LABEL(TRCOL, VALUE='FILE DIRECTORY')
  W_DIRECTORY_TEXT=WIDGET_TEXT(TRCOL, VALUE=DIR, /EDITABLE)

  IMGLIST=FINDFILE(DIR)
  FILE=IMGLIST(0)
  PFILE=FILE
  W_IMAGEFILE_LABEL=WIDGET_LABEL(TRCOL, VALUE='FILE LIST')
  W_IMAGEFILE_LIST=WIDGET_LIST(TRCOL, VALUE=IMGLIST, $
     UVALUE=IMGLIST, YSIZE=5)

  W_FISHEYE=WIDGET_BUTTON(TRCOL, VALUE='QUICK VIEW ON', $
    UVALUE='QUICKVIEW')

  ALT=''
  W_ALT_BASE=WIDGET_BASE(TRCOL, /ROW)
  W_ALTITUDE_LABEL=WIDGET_BUTTON(W_ALT_BASE, $
    VALUE='ALTITUDE', /MENU)
  W_ALTITUDE_TEXT=WIDGET_TEXT(W_ALT_BASE, $
    VALUE=ALT, /EDITABLE)

  W_ALT_B1=WIDGET_BUTTON(W_ALTITUDE_LABEL, VALUE='120 KM', $
    UVALUE='a120')
  W_ALT_B2=WIDGET_BUTTON(W_ALTITUDE_LABEL, VALUE='160 KM', $
    UVALUE='a160')
  W_ALT_B3=WIDGET_BUTTON(W_ALTITUDE_LABEL, VALUE='200 KM', $
    UVALUE='a200')
  W_ALT_B4=WIDGET_BUTTON(W_ALTITUDE_LABEL, VALUE='250 KM', $
    UVALUE='a250')
  W_ALT_B5=WIDGET_BUTTON(W_ALTITUDE_LABEL, VALUE='300 KM', $
    UVALUE='a300')
  W_ALT_B6=WIDGET_BUTTON(W_ALTITUDE_LABEL, VALUE='350 KM', $
    UVALUE='a350')

  COLOR='GRAY'
  W_COLOR_BASE=WIDGET_BASE(TRCOL, /ROW)
  W_COLOR_LABEL=WIDGET_BUTTON(W_COLOR_BASE, $
    VALUE='COLOR', /MENU)
  W_COLOR_TEXT=WIDGET_TEXT(W_COLOR_BASE, $
    VALUE=COLOR)

  W_COL_B1=WIDGET_BUTTON(W_COLOR_LABEL, VALUE='GRAY', $
    UVALUE='GRAY')
  W_COL_B2=WIDGET_BUTTON(W_COLOR_LABEL, VALUE='RED', $
    UVALUE='RED')
  W_COL_B3=WIDGET_BUTTON(W_COLOR_LABEL, VALUE='GREEN', $
    UVALUE='GREEN')
  W_COL_B4=WIDGET_BUTTON(W_COLOR_LABEL, VALUE='BLUE', $
    UVALUE='BLUE')
  W_COL_B5=WIDGET_BUTTON(W_COLOR_LABEL, VALUE='RAINBOW1', $
    UVALUE='RAINBOW1')
  W_COL_B6=WIDGET_BUTTON(W_COLOR_LABEL, VALUE='RAINBOW2', $
    UVALUE='RAINBOW2')
```

```
scl='4000'
W_scl_BASE=WIDGET_BASE(TRCOL, /ROW)
W_scl_LABEL=WIDGET_BUTTON(W_scl_BASE, $
  VALUE='SCALE', /MENU)
W_scl_TEXT=WIDGET_TEXT(W_scl_BASE, $
  VALUE=scl, /EDITABLE)

W_scl_B1=WIDGET_BUTTON(W_scl_LABEL, VALUE='500', $
  UVALUE='s500')
W_scl_B2=WIDGET_BUTTON(W_scl_LABEL, VALUE='1000', $
  UVALUE='s1000')
W_scl_B3=WIDGET_BUTTON(W_scl_LABEL, VALUE='1500', $
  UVALUE='s1500')
W_scl_B4=WIDGET_BUTTON(W_scl_LABEL, VALUE='2000', $
  UVALUE='s2000')
W_scl_B5=WIDGET_BUTTON(W_scl_LABEL, VALUE='3000', $
  UVALUE='s3000')
W_scl_B6=WIDGET_BUTTON(W_scl_LABEL, VALUE='4000', $
  UVALUE='s4000')

VIEW=20
W_VIEW_ALT=WIDGET_SLIDER(MRCOL, VALUE=VIEW, $
  TITLE='     VIEWING ALTITUDE     ', $
  MAXIMUM=60,MINIMUM=5)

SATLIST=STRARR(20)
SATLIST(0)=DIR
W_SAT_LABEL=WIDGET_LABEL(BRCOL, $
  VALUE='      SATELLITE LIST      ')
W_SAT_LIST=WIDGET_LIST(BRCOL, VALUE=SATLIST(1:*), $
  UVALUE=SATLIST, YSIZE=4)
W_SAT_B1=WIDGET_BUTTON(BRCOL, VALUE='ADD', $
  UVALUE='ADD')
W_SAT_B2=WIDGET_BUTTON(BRCOL, VALUE='REMOVE', $
  UVALUE='REMOVE')

WIDGET_CONTROL, BASE, /REALIZE

WIDGET_CONTROL, /HOURGLASS

WIDGET_CONTROL, W_DRAW, GET_VALUE=INDEX

SET_COLOR,COLOR,TP

WDGT=INTARR(40)

WDGT(4)=W_DIRECTORY_TEXT

WDGT(6)=W_IMAGEFILE_LIST
WDGT(7)=W_ALTITUDE_TEXT
WDGT(8)=W_DATE_TEXT
WDGT(9)=W_TIME_TEXT
WDGT(10)=W_FILTER_TEXT
WDGT(11)=W_VIEW_ALT
WDGT(12)=W_SYS_B1
```

```
WDGT(15)=W_ALT_TEXT
WDGT(16)=W_FILE_TEXT
wdgt(17)=w_scl_text
WDGT(18)=W_FISHEYE
WDGT(19)=W_SAT_LIST
WDGT(20)=W_LAT_TEXT
WDGT(21)=W_LON_TEXT
WDGT(22)=W_RAY_TEXT
WDGT(23)=W_EXP_TEXT
WDGT(24)=W_GAIN_TEXT
WDGT(25)=TOPROW

WDGT(28)=RCOL

WDGT(31)=W_COLOR_TEXT
WDGT(32)=W_COORD_B1
WDGT(33)=W_COORD_B2
WDGT(34)=W_OPTION_B1
WDGT(35)=W_OPTION_B2
WDGT(36)=W_OPTION_B3


FLAGS=INTARR(10)
FLAGS(0)=0   ;1 IF A VALID FILE HAS BEEN LOADED INTO MEMORY
FLAGS(1)=0   ;1 IF AN ALTITUDE HAS BEEN SELECTED
FLAGS(2)=1   ;1 IF AN I/O ERROR OCCURED WHILE
        ;  READING OR WRITING A FILE
FLAGS(3)=0   ;pn - changed to 0 - 1 TO SWAP THE BIT ORDER
FLAGS(4)=0   ;1 TO READ PIXEL VALUES AT MOUSE LOCATION
FLAGS(5)=1   ;1 TO PLOT RAW FISHEYE IMAGE ON SCREEN
FLAGS(6)=0   ;1 IF CURRENT IMAGE AND ALTITUDE HAVE ALREADY
        ;  BEEN PLOTED ON SCREEN
FLAGS(7)=0   ;1/0 FOR PLOTTING GEO/MAG COORDS. OVER IMAGE

PWIN=INTARR(6)
PWIN(0)=WXSZ
PWIN(1)=WYSZ
PWIN(2)=INDEX
PWIN(3)=TP
PWIN(4)=WX
PWIN(5)=WY

;NY ALESUND
  SITEELEV=0.063   ; KM
  SITELAT=78.92    ; DEGREES
  SITELON=11.93    ; DEGREES

  DELLAT=((7.0)/325.0)*VIEW*10.0
  DELLON=((27.0)/325.0)*VIEW*10.0
  SITELATMIN=SITELAT-DELLAT
  SITELATMAX=SITELAT+DELLAT
  SITELONMIN=SITELON-DELLON
  SITELONMAX=SITELON+DELLON

  PLOC=FLTARR(9)
```

```
PLOC(0)=SITEELEV
PLOC(1)=SITELAT
PLOC(2)=SITELON
PLOC(3)=ALT
PLOC(4)=SITELATMIN
PLOC(5)=SITELATMAX
PLOC(6)=SITELONMIN
PLOC(7)=SITELONMAX
ploc(8)=scl

STATE={WDGT:WDGT, FLAGS:FLAGS, PWIN:PWIN, $
  PLOC:PLOC, OPTION:OPTION, DIR:DIR, $
  FILE:FILE, PDIR:PDIR, PFILE:PFILE, $
  LAT:0.0, LON:0.0, TRIANGLES:0.0, $
  IAI:0.0, INFO:0.0, MAPIMAGE:0.0, MAP:0.0}

WIDGET_CONTROL, BASE, SET_UVALUE=STATE

XMANAGER, "ALLSKY", BASE

END

;;;;;;;;;;;;;;;;;;;;;;;;
;;; PRO ALLSKY_EVENT ;;;
;;;;;;;;;;;;;;;;;;;;;;;;

PRO ALLSKY_EVENT, EV

WIDGET_CONTROL, EV.TOP, GET_UVALUE=STATE

TYPE=TAG_NAMES(EV, /STRUCTURE_NAME)

IF (EV.ID EQ STATE.WDGT(4)) THEN TYPE="DIRECTORY"

IF (EV.ID EQ STATE.WDGT(7)) THEN TYPE="ALTITUDE"

IF (EV.ID EQ STATE.WDGT(6)) THEN TYPE="IMAGE_LIST"

IF (EV.ID EQ STATE.WDGT(17)) THEN TYPE="SCALE"

IF (EV.ID EQ STATE.WDGT(19)) THEN TYPE="SAT_LIST"

CASE TYPE OF

  "WIDGET_SLIDER" : BEGIN
    WIDGET_CONTROL,EV.ID,GET_VALUE=VIEW
    DELLAT=(7.0/325.0)*VIEW*10.0
    DELLON=(27.0/325.0)*VIEW*10.0
    STATE.PLOC(4)=STATE.PLOC(1)-DELLAT
    STATE.PLOC(5)=STATE.PLOC(1)+DELLAT
    STATE.PLOC(6)=STATE.PLOC(2)-DELLON
    STATE.PLOC(7)=STATE.PLOC(2)+DELLON
    STATE.FLAGS(6)=0
    WIDGET_CONTROL,EV.TOP,SET_UVALUE=STATE
  ENDCASE
```

```
"WIDGET_DRAW" : BEGIN
  IF STATE.FLAGS(4) EQ 1 THEN BEGIN
    XSTACK=!X
    YSTACK=!Y
    !X=STATE.MAP.X
    !Y=STATE.MAP.Y
    LLPOS=CONVERT_COORD(EV.X,EV.Y,/DEVICE,/TO_DATA)
    IF (STATE.FLAGS(7) EQ 0) THEN BEGIN
     WIDGET_CONTROL, STATE.WDGT(20), $
      SET_VALUE=STRTRIM(STRING(FORMAT='(F7.2)', $
      LLPOS(0)),2)
     WIDGET_CONTROL, STATE.WDGT(21), $
      SET_VALUE=STRTRIM(STRING(FORMAT='(F7.2)', $
      LLPOS(1)),2)
    ENDIF ELSE BEGIN
     GEOMAG,LLPOS(1),LLPOS(0),POS1,POS2,1
     WIDGET_CONTROL, STATE.WDGT(20), $
      SET_VALUE=STRTRIM(STRING(FORMAT='(F7.2)', $
      POS1),2)
     WIDGET_CONTROL, STATE.WDGT(21), $
      SET_VALUE=STRTRIM(STRING(FORMAT='(F7.2)', $
      POS2),2)
    ENDELSE
    PIC=FLOAT(TVRD(EV.X,EV.Y,1,1))
    TP=FLOAT(STATE.PWIN(3))
    IF ((PIC(0) GT 0) AND (PIC(0) LT TP-1)) THEN BEGIN
      RAY=((PIC(0)-1.0)/(TP-3.0))*STATE.PLOC(8)
    ENDIF ELSE BEGIN
      RAY=0.0
    ENDELSE
    WIDGET_CONTROL, STATE.WDGT(22), $
      SET_VALUE=STRTRIM(STRING(FORMAT='(F6.1)', $
        RAY(0,0)),2)
    !X=XSTACK
    !Y=YSTACK
  ENDIF
ENDCASE

"WIDGET_BUTTON" : BEGIN
  WIDGET_CONTROL, EV.ID, GET_UVALUE=UV

  IF ((UV EQ 'GRAY') OR (UV EQ 'RED') OR $
     (UV EQ 'GREEN') OR (UV EQ 'BLUE') OR $
     (UV EQ 'RAINBOW1') OR (UV EQ 'RAINBOW2')) $
     THEN BEGIN
    COLOR=UV
    UV='COLORTABLE'
  ENDIF

  IF ((UV EQ 'a120') OR (UV EQ 'a160') OR $
    (UV EQ 'a200') OR (UV EQ 'a250') OR $
    (UV EQ 'a300') OR (UV EQ 'a350')) THEN BEGIN
    ALT=strmid(UV,1,4)
    UV='SELECTALT'
  ENDIF
```

```
IF ((UV EQ 's500') OR (UV EQ 's1000') OR $
   (UV EQ 's1500') OR (UV EQ 's2000') OR $
   (UV EQ 's3000') OR (UV EQ 's4000')) THEN BEGIN
  SCL=strmid(UV,1,5)
  UV='SELECTSCL'
ENDIF

CASE UV OF

  "QUIT" : BEGIN
    M_BASE=WIDGET_BASE(TITLE='MESSAGE', /COLUMN, $
        GROUP_LEADER=EV.TOP, /MODAL)
    M_TEXT=WIDGET_label(M_BASE, VALUE='DO YOU '+ $
        'WANT TO QUIT ALLSKY ?')
    M_SUBASE=WIDGET_BASE(M_BASE, /ROW)
    M_B1=WIDGET_BUTTON(M_SUBASE, VALUE='YES', $
        UVALUE='YES')
    M_B2=WIDGET_BUTTON(M_SUBASE, VALUE='NO', $
        UVALUE='NO')
    M_B3=WIDGET_BUTTON(M_SUBASE, VALUE='CANCEL', $
        UVALUE='CANCEL')
    WIDGET_CONTROL, M_BASE, /REALIZE
    widget_control, m_b1, /input_focus
    EVENT=WIDGET_EVENT(M_BASE)
    WIDGET_CONTROL,EVENT.ID,GET_UVALUE=ANS
    widget_control, m_base, /destroy
    IF ANS EQ 'YES' THEN $
      WIDGET_CONTROL, /DESTROY, EV.TOP
  ENDCASE

  "XWINDOW" : BEGIN
    IF (STATE.FLAGS(0) AND STATE.FLAGS(1)) THEN BEGIN
    WIDGET_CONTROL, /HOURGLASS
    WIDGET_CONTROL, STATE.WDGT(28), SENSITIVE=0
    WIDGET_CONTROL, STATE.WDGT(25), SENSITIVE=0
    WIDGET_CONTROL, STATE.WDGT(19), GET_UVALUE=SATLIST
    W_PLOT_IMAGE,STATE,SATLIST
    STATE.FLAGS(4)=1
    STATE.FLAGS(6)=1
    WIDGET_CONTROL, EV.TOP, SET_UVALUE=STATE
    WIDGET_CONTROL, STATE.WDGT(28), SENSITIVE=1
    WIDGET_CONTROL, STATE.WDGT(25), SENSITIVE=1
    ENDIF ELSE BEGIN
     IF (NOT STATE.FLAGS(0) AND NOT STATE.FLAGS(1)) $
       THEN BEGIN
      RESULT=DIALOG_MESSAGE(['BEFOR YOU CAN MAKE AN '+ $
        'XWINDOW IMAGE YOU MUST SELECT: ', ' ', $
        '(1) AN IMAGE FILE AND', $
        '(2) A PROJECTION ALTITUDE.'],/INFORMATION)
     ENDIF ELSE BEGIN
      IF (NOT STATE.FLAGS(0)) THEN BEGIN
       RESULT=DIALOG_MESSAGE(['BEFOR YOU CAN MAKE AN '+ $
        'XWINDOW IMAGE YOU MUST SELECT: ', ' ', $
        '(1) AN IMAGE FILE.'],/INFORMATION)
      ENDIF
      IF (NOT STATE.FLAGS(1)) THEN BEGIN
```

```
    RESULT=DIALOG_MESSAGE(['BEFOR YOU CAN MAKE AN '+ $
      'XWINDOW IMAGE YOU MUST SELECT:', ' ', $
      '(1) A PROJECTION ALTITUDE.'],/INFORMATION)
   ENDIF
  ENDELSE
 ENDELSE
ENDCASE

"PS FILE" : BEGIN
 IF (STATE.FLAGS(0) AND STATE.FLAGS(1)) THEN BEGIN
   WIDGET_CONTROL, STATE.WDGT(28), SENSITIVE=0
   WIDGET_CONTROL, STATE.WDGT(25), SENSITIVE=0
   PFLNM=STATE.PFILE
   PPTH=STATE.PDIR
   PFLNM=DIALOG_PICKFILE(/WRITE, FILE=PFLNM, $
     FILTER=PPTH, GET_PATH=PPTH)
   IF (STRLEN(PFLNM) GT 0) THEN BEGIN
     WIDGET_CONTROL, /HOURGLASS
     PFLNM=STRMID(PFLNM,STRLEN(PPTH), $
       (STRLEN(PFLNM)-STRLEN(PPTH)))
     STATE.PFILE=PFLNM
     STATE.PDIR=PPTH
     WIDGET_CONTROL, STATE.WDGT(19), $
       GET_UVALUE=SATLIST
     W_PRINT_IMAGE,STATE,SATLIST
     WIDGET_CONTROL, EV.TOP, SET_UVALUE=STATE
   ENDIF
   WIDGET_CONTROL, STATE.WDGT(28), SENSITIVE=1
   WIDGET_CONTROL, STATE.WDGT(25), SENSITIVE=1
 ENDIF ELSE BEGIN
  IF (NOT STATE.FLAGS(0) AND NOT STATE.FLAGS(1)) $
    THEN BEGIN
   RESULT=DIALOG_MESSAGE(['BEFOR YOU CAN MAKE A '+ $
     'POST SCRIPT FILE YOU MUST SELECT: ', ' ', $
     '(1) AN IMAGE FILE AND', $
     '(2) A PROJECTION ALTITUDE.'],/INFORMATION)
  ENDIF ELSE BEGIN
   IF (NOT STATE.FLAGS(0)) THEN BEGIN
    RESULT=DIALOG_MESSAGE(['BEFOR YOU CAN MAKE A '+ $
      'POST SCRIPT FILE YOU MUST SELECT: ', ' ', $
      '(1) AN IMAGE FILE.'],/INFORMATION)
   ENDIF
   IF (NOT STATE.FLAGS(1)) THEN BEGIN
    RESULT=DIALOG_MESSAGE(['BEFOR YOU CAN MAKE A '+ $
      'POST SCRIPT FILE YOU MUST SELECT:', ' ', $
      '(1) A PROJECTION ALTITUDE.'],/INFORMATION)
   ENDIF
  ENDELSE
 ENDELSE
ENDCASE


"SAMPLE" : BEGIN
 SAMPLE_IMAGE
ENDCASE
```

```
"MERIDIAN" : BEGIN
  IF (STATE.OPTION(0) EQ 0) THEN BEGIN
    WIDGET_CONTROL, STATE.WDGT(34), $
      SET_VALUE='+ MERIDIAN'
    STATE.OPTION(0)=1
    WIDGET_CONTROL, EV.TOP, SET_UVALUE=STATE
  ENDIF ELSE BEGIN
    WIDGET_CONTROL, STATE.WDGT(34), $
      SET_VALUE=' MERIDIAN'
    STATE.OPTION(0)=0
    WIDGET_CONTROL, EV.TOP, SET_UVALUE=STATE
  ENDELSE
ENDCASE

"ALESUND" : BEGIN
  IF (STATE.OPTION(1) EQ 0) THEN BEGIN
    WIDGET_CONTROL, STATE.WDGT(35), $
      SET_VALUE='+ NY ALESUND'
    STATE.OPTION(1)=1
    WIDGET_CONTROL, EV.TOP, SET_UVALUE=STATE
  ENDIF ELSE BEGIN
    WIDGET_CONTROL, STATE.WDGT(35), $
      SET_VALUE=' NY ALESUND'
    STATE.OPTION(1)=0
    WIDGET_CONTROL, EV.TOP, SET_UVALUE=STATE
  ENDELSE
ENDCASE

"LONGYEAR" : BEGIN
  IF (STATE.OPTION(2) EQ 0) THEN BEGIN
    WIDGET_CONTROL, STATE.WDGT(36), $
      SET_VALUE='+ LONGYEARBYEN'
    STATE.OPTION(2)=1
    WIDGET_CONTROL, EV.TOP, SET_UVALUE=STATE
  ENDIF ELSE BEGIN
    WIDGET_CONTROL, STATE.WDGT(36), $
      SET_VALUE=' LONGYEARBYEN'
    STATE.OPTION(2)=0
    WIDGET_CONTROL, EV.TOP, SET_UVALUE=STATE
  ENDELSE
ENDCASE

"QUICKVIEW" : BEGIN
  IF (STATE.FLAGS(5) EQ 0) THEN BEGIN
    WIDGET_CONTROL, STATE.WDGT(18), $
      SET_VALUE='QUICK VIEW ON'
    STATE.FLAGS(5)=1
    WIDGET_CONTROL, EV.TOP, SET_UVALUE=STATE
  ENDIF ELSE BEGIN
    WIDGET_CONTROL, STATE.WDGT(18), $
      SET_VALUE='QUICK VIEW OFF'
    STATE.FLAGS(5)=0
    WIDGET_CONTROL, EV.TOP, SET_UVALUE=STATE
  ENDELSE
ENDCASE
```

```
"BITSWAP" : BEGIN
  IF (STATE.FLAGS(3) EQ 1) THEN BEGIN
    WIDGET_CONTROL, STATE.WDGT(12),SET_VALUE='  BIT SWAP'
    STATE.FLAGS(3)=0
  ENDIF ELSE BEGIN
    WIDGET_CONTROL, STATE.WDGT(12),SET_VALUE='+ BIT SWAP'
    STATE.FLAGS(3)=1
  ENDELSE
  WIDGET_CONTROL, EV.TOP, SET_UVALUE=STATE
ENDCASE

"GEOGRAPHIC" : BEGIN
  STATE.FLAGS(7)=0
  WIDGET_CONTROL, STATE.WDGT(32),SET_VALUE='+ GEOGRAPHIC'
  WIDGET_CONTROL, STATE.WDGT(33),SET_VALUE='  MAGNETIC'
  WIDGET_CONTROL, EV.TOP, SET_UVALUE=STATE
ENDCASE

"MAGNETIC" : BEGIN
  STATE.FLAGS(7)=1
  WIDGET_CONTROL, STATE.WDGT(32),SET_VALUE='  GEOGRAPHIC'
  WIDGET_CONTROL, STATE.WDGT(33),SET_VALUE='+ MAGNETIC'
  WIDGET_CONTROL, EV.TOP, SET_UVALUE=STATE
ENDCASE

"OPEN" : BEGIN
  PTH=STATE.DIR
  FLNM=DIALOG_PICKFILE(/MUST_EXIST, /READ, $
    GET_PATH=PTH, FILTER=PTH)
  IF (STRLEN(FLNM) GT 0) THEN BEGIN
    WIDGET_CONTROL, /HOURGLASS
    STATE.FLAGS(6)=0
    STATE.DIR=PTH
    FLNM=STRMID(FLNM,STRLEN(PTH), $
      (STRLEN(FLNM)-STRLEN(PTH)))
    STATE.FILE=FLNM
    STATE.PFILE=FLNM+'.ps'
    IMGLIST=FINDFILE(PTH)
    FLTOP=WHERE(IMGLIST EQ FLNM)
    WIDGET_CONTROL, STATE.WDGT(6), $
      SET_VALUE=IMGLIST
    WIDGET_CONTROL, STATE.WDGT(6), $
      SET_UVALUE=IMGLIST
    WIDGET_CONTROL, STATE.WDGT(6), $
      SET_LIST_TOP=FLTOP(0), $
      SET_LIST_SELECT=FLTOP(0)
    WIDGET_CONTROL, STATE.WDGT(4), $
      SET_VALUE=PTH
    W_LOAD_IMAGE,STATE
    IF (STATE.FLAGS(2) EQ 1) THEN BEGIN
      STATE.FLAGS(0)=1
      WIDGET_CONTROL, STATE.WDGT(8), $
        SET_VALUE=STATE.INFO(0)
      WIDGET_CONTROL, STATE.WDGT(9), $
        SET_VALUE=STATE.INFO(1)
      WIDGET_CONTROL, STATE.WDGT(10), $
```

```
        SET_VALUE=STATE.INFO(2)
      WIDGET_CONTROL, STATE.WDGT(16), $
        SET_VALUE=FLNM
      WIDGET_CONTROL, STATE.WDGT(23), $
        SET_VALUE=STRING(FORMAT='(F4.1)', $
        STATE.INFO(3))
      WIDGET_CONTROL, STATE.WDGT(24), $
        SET_VALUE=STRING(FORMAT='(I1)', $
        STATE.INFO(4))
    ENDIF ELSE BEGIN
      STATE.FLAGS(0)=0
      WIDGET_CONTROL, STATE.WDGT(8), $
        SET_VALUE='???????'
      WIDGET_CONTROL, STATE.WDGT(9), $
        SET_VALUE='??:??:??'
      WIDGET_CONTROL, STATE.WDGT(10), $
        SET_VALUE='????'
      WIDGET_CONTROL, STATE.WDGT(16), $
        SET_VALUE='?????'
      WIDGET_CONTROL, STATE.WDGT(23), $
        SET_VALUE='??.?'
      WIDGET_CONTROL, STATE.WDGT(24), $
        SET_VALUE='?'
    ENDELSE
    WIDGET_CONTROL, EV.TOP, SET_UVALUE=STATE
  ENDIF
ENDCASE

'SELECTALT' : BEGIN
  WIDGET_CONTROL, /HOURGLASS
  WIDGET_CONTROL, STATE.WDGT(28), SENSITIVE=0
  WIDGET_CONTROL, STATE.WDGT(25), SENSITIVE=0
  WIDGET_CONTROL, STATE.WDGT(7), $
    SET_VALUE=STRING(ALT,FORMAT='(F5.1)')
  WIDGET_CONTROL, STATE.WDGT(15), $
    SET_VALUE=STRING(ALT,FORMAT='(F5.1)')
  STATE.PLOC(3)=ALT
  W_PROJECT_IMAGE,STATE
  STATE.FLAGS(1)=1
  STATE.FLAGS(6)=0
  WIDGET_CONTROL, EV.TOP, SET_UVALUE=STATE
  WIDGET_CONTROL, STATE.WDGT(28), SENSITIVE=1
  WIDGET_CONTROL, STATE.WDGT(25), SENSITIVE=1
ENDCASE

'SELECTSCL' : BEGIN
  WIDGET_CONTROL, /HOURGLASS
  WIDGET_CONTROL, STATE.WDGT(28), SENSITIVE=0
  WIDGET_CONTROL, STATE.WDGT(25), SENSITIVE=0
  WIDGET_CONTROL, STATE.WDGT(17), $
    SET_VALUE=STRING(SCL,FORMAT='(F6.1)')
  STATE.PLOC(8)=SCL
  STATE.FLAGS(6)=0
  WIDGET_CONTROL, EV.TOP, SET_UVALUE=STATE
  WIDGET_CONTROL, STATE.WDGT(28), SENSITIVE=1
  WIDGET_CONTROL, STATE.WDGT(25), SENSITIVE=1
```

```
    ENDCASE

    "COLORTABLE" : BEGIN
      WIDGET_CONTROL, /HOURGLASS
      WIDGET_CONTROL, STATE.WDGT(31), SET_VALUE=COLOR
      SET_COLOR,COLOR,TP
      STATE.PWIN(3)=TP
      WIDGET_CONTROL, EV.TOP, SET_UVALUE=STATE
    ENDCASE

    "ADD" : BEGIN
      WIDGET_CONTROL, STATE.WDGT(19), $
        GET_UVALUE=SATLIST
      PTH=SATLIST(0)
      FLNM=DIALOG_PICKFILE(/MUST_EXIST, /READ, $
        GET_PATH=PTH, FILTER=PTH+'V*')
      IF (STRLEN(FLNM) GT 0) THEN BEGIN
        SATLIST(0)=PTH
        SATLIST(N_ELEMENTS(WHERE(SATLIST(0:18) $
          NE ''))=STRMID(FLNM,STRLEN(PTH), $
          (STRLEN(FLNM)-STRLEN(PTH)))
        SATLIST(19)=''
        WIDGET_CONTROL, STATE.WDGT(19), $
          SET_VALUE=SATLIST(1:18)
        WIDGET_CONTROL, STATE.WDGT(19), $
          SET_UVALUE=SATLIST
      ENDIF
    ENDCASE

    "REMOVE" : BEGIN
      WIDGET_CONTROL, STATE.WDGT(19), $
        GET_UVALUE=SATLIST
      IF ((SATLIST(19) NE '') AND $
        (SATLIST(SATLIST(19)) NE '')) THEN BEGIN
        ISATLIST=N_ELEMENTS(WHERE(SATLIST(0:18) $
          NE ''))-1
        SATLIST(SATLIST(19))=''
        SATLIST(SATLIST(19):17)= $
          SATLIST((SATLIST(19)+1):18)
        SATLIST(19)=''
        WIDGET_CONTROL, STATE.WDGT(19), $
          SET_VALUE=SATLIST(1:18)
        WIDGET_CONTROL, STATE.WDGT(19), $
          SET_UVALUE=SATLIST
      ENDIF
    ENDCASE

    ELSE: PRINT, "UNKNOWN BUTTON! ", UV
  ENDCASE

ENDCASE

"SAT_LIST" : BEGIN
  WIDGET_CONTROL, STATE.WDGT(19), GET_UVALUE=SATLIST
  SATLIST(19)=EV.INDEX+1
  WIDGET_CONTROL, STATE.WDGT(19), SET_UVALUE=SATLIST
```

```
ENDCASE

"DIRECTORY" : BEGIN
  WIDGET_CONTROL, EV.ID, GET_VALUE=DIRS
  DIR=STRING(DIRS(0))
  IMGLIST=FINDFILE(DIR)
  WIDGET_CONTROL, STATE.WDGT(6), SET_VALUE=IMGLIST
  WIDGET_CONTROL, STATE.WDGT(6), SET_UVALUE=IMGLIST
  STATE.DIR=DIR
  WIDGET_CONTROL, EV.TOP, SET_UVALUE=STATE
ENDCASE

"IMAGE_LIST" : BEGIN
  WIDGET_CONTROL, /HOURGLASS
  STATE.FLAGS(6)=0
  WIDGET_CONTROL, EV.ID, GET_UVALUE=IMGLIST
  FLNM=IMGLIST(EV.INDEX)
  STATE.FILE=FLNM
  STATE.PFILE=FLNM+'.ps'
  W_LOAD_IMAGE,STATE
  IF (STATE.FLAGS(2) EQ 1) THEN BEGIN
    STATE.FLAGS(0)=1
    WIDGET_CONTROL, STATE.WDGT(8), $
      SET_VALUE=STATE.INFO(0)
    WIDGET_CONTROL, STATE.WDGT(9), $
      SET_VALUE=STATE.INFO(1)
    WIDGET_CONTROL, STATE.WDGT(10), $
      SET_VALUE=STATE.INFO(2)
    WIDGET_CONTROL, STATE.WDGT(16), $
      SET_VALUE=FLNM
    WIDGET_CONTROL, STATE.WDGT(23), $
      SET_VALUE=STRING(FORMAT='(F4.1)',STATE.INFO(3))
    WIDGET_CONTROL, STATE.WDGT(24), $
      SET_VALUE=STRING(FORMAT='(I1)',STATE.INFO(4))
  ENDIF ELSE BEGIN
    STATE.FLAGS(0)=0
    WIDGET_CONTROL, STATE.WDGT(8), $
      SET_VALUE='???????'
    WIDGET_CONTROL, STATE.WDGT(9), $
      SET_VALUE='??:??:??'
    WIDGET_CONTROL, STATE.WDGT(10), $
      SET_VALUE='????'
    WIDGET_CONTROL, STATE.WDGT(16), $
      SET_VALUE='?????'
    WIDGET_CONTROL, STATE.WDGT(23), $
      SET_VALUE='??.?'
    WIDGET_CONTROL, STATE.WDGT(24), $
      SET_VALUE='?'
  ENDELSE
  WIDGET_CONTROL, EV.TOP, SET_UVALUE=STATE
ENDCASE

"ALTITUDE" : BEGIN
  WIDGET_CONTROL, /HOURGLASS
  WIDGET_CONTROL, STATE.WDGT(28), SENSITIVE=0
  WIDGET_CONTROL, STATE.WDGT(25), SENSITIVE=0
```

```
    WIDGET_CONTROL, EV.ID, GET_VALUE=ALTS
    ALT='0'+ALTS(0)
    IF ((ALT GE 50.0) AND (ALT LE 600.0)) THEN BEGIN
      STATE.PLOC(3)=ALT
      WIDGET_CONTROL, STATE.WDGT(15), $
        SET_VALUE=STRING(ALT,FORMAT='(F5.1)')
      WIDGET_CONTROL, STATE.WDGT(7), $
        SET_VALUE=STRING(ALT,FORMAT='(F5.1)')
      W_PROJECT_IMAGE,STATE
      STATE.FLAGS(1)=1
    ENDIF ELSE BEGIN
      RESULT=DIALOG_MESSAGE(['YOU MUST SELECT AN ALTITUDE', $
        'BETWEEN 50.0 AND 600.0 KM'],/INFORMATION)
      WIDGET_CONTROL, STATE.WDGT(15), $
        SET_VALUE='???.?'
      WIDGET_CONTROL, STATE.WDGT(7), $
        SET_VALUE='???.?'
      STATE.FLAGS(1)=0
    ENDELSE
    STATE.FLAGS(6)=0
    WIDGET_CONTROL, EV.TOP, SET_UVALUE=STATE
    WIDGET_CONTROL, STATE.WDGT(28), SENSITIVE=1
    WIDGET_CONTROL, STATE.WDGT(25), SENSITIVE=1
  ENDCASE

  "SCALE" : BEGIN
    WIDGET_CONTROL, /HOURGLASS
    WIDGET_CONTROL, STATE.WDGT(28), SENSITIVE=0
    WIDGET_CONTROL, STATE.WDGT(25), SENSITIVE=0
    WIDGET_CONTROL, EV.ID, GET_VALUE=SCLS
    SCL='0'+SCLS(0)
    IF ((SCL GE 10.0) AND (SCL LE 6000.0)) THEN BEGIN
      STATE.PLOC(8)=SCL
      WIDGET_CONTROL, STATE.WDGT(17), $
        SET_VALUE=STRING(SCL,FORMAT='(F6.1)')
    ENDIF ELSE BEGIN
      RESULT=DIALOG_MESSAGE(['YOU MUST SELECT A VALUE', $
        'BETWEEN 10.0 AND 6000.0 R'],/INFORMATION)
      WIDGET_CONTROL, STATE.WDGT(17), $
        SET_VALUE='???.?'
    ENDELSE
    STATE.FLAGS(6)=0
    WIDGET_CONTROL, EV.TOP, SET_UVALUE=STATE
    WIDGET_CONTROL, STATE.WDGT(28), SENSITIVE=1
    WIDGET_CONTROL, STATE.WDGT(25), SENSITIVE=1
  ENDCASE

  ELSE:

ENDCASE

END

;;;;;;;;;;;;;;;;;;;;;;;;
;;; PRO W_LOAD_IMAGE ;;;
;;;;;;;;;;;;;;;;;;;;;;;;
```

```
PRO W_LOAD_IMAGE,STATE

FILENAME=STATE.DIR+STATE.FILE
INDEX=STATE.PWIN(2)
TP=STATE.PWIN(3)

FLAGS=STATE.FLAGS

FLAG2=FLAGS(2)
FLAG3=FLAGS(3)
FLAG5=FLAGS(5)

LOAD_IMAGE,FILENAME,INDEX,TP, $
  IAI,FDATE,FTIME,FFILTER, $
  FEXPOSURE,FGAIN, $
  FLAG2,FLAG3,FLAG5

IF (FLAG2 EQ 1) THEN BEGIN

  FLAGS(2)=1
  IF (FLAGS(5) EQ 1) THEN FLAGS(4)=0

  INFO=STRARR(5)
  INFO(0)=FDATE
  INFO(1)=FTIME
  INFO(2)=FFILTER
  INFO(3)=FEXPOSURE
  INFO(4)=FGAIN

  STATE={WDGT:STATE.WDGT, FLAGS:FLAGS, $
    PWIN:STATE.PWIN, PLOC:STATE.PLOC, $
    OPTION:STATE.OPTION, DIR:STATE.DIR, $
    FILE:STATE.FILE, PDIR:STATE.PDIR, $
    PFILE:STATE.PFILE, LAT:STATE.LAT, $
    LON:STATE.LON, TRIANGLES:STATE.TRIANGLES, $
    IAI:IAI, INFO:INFO, MAPIMAGE:STATE.MAPIMAGE, $
    MAP:STATE.MAP}

ENDIF ELSE BEGIN

  STATE.FLAGS(2)=0

ENDELSE

END

;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; PRO W_PROJECT_IMAGE ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;

PRO W_PROJECT_IMAGE,STATE

ALT=STATE.PLOC(3)

SITEELEV=STATE.PLOC(0)
```

```
  SITELAT=STATE.PLOC(1)
  SITELON=STATE.PLOC(2)

  PROJECT_IMAGE,SITELAT,SITELON,SITEELEV,ALT, $
    LAT,LON,TRIANGLES

  STATE={WDGT:STATE.WDGT, FLAGS:STATE.FLAGS,$
    PWIN:STATE.PWIN, PLOC:STATE.PLOC, $
    OPTION:STATE.OPTION, DIR:STATE.DIR, $
    FILE:STATE.FILE, PDIR:STATE.PDIR, $
    PFILE:STATE.PFILE, LAT:LAT, LON:LON, $
    TRIANGLES:TRIANGLES, IAI:STATE.IAI, $
    INFO:STATE.INFO, MAPIMAGE:STATE.MAPIMAGE, $
    MAP:STATE.MAP}

  END

;;;;;;;;;;;;;;;;;;;;;;;;
;;; PRO W_PLOT_IMAGE ;;;
;;;;;;;;;;;;;;;;;;;;;;;;

  PRO W_PLOT_IMAGE,STATE,SATLIST

  WXSZ=STATE.PWIN(0)
  WYSZ=STATE.PWIN(1)
  INDEX=STATE.PWIN(2)
  TP=STATE.PWIN(3)
  WX=STATE.PWIN(4)
  WY=STATE.PWIN(5)
  OPTION=STATE.OPTION

  FILENAME=STATE.FILE

  SITELAT=STATE.PLOC(1)
  SITELON=STATE.PLOC(2)

  SITELATMIN=STATE.PLOC(4)
  SITELATMAX=STATE.PLOC(5)
  SITELONMIN=STATE.PLOC(6)
  SITELONMAX=STATE.PLOC(7)

  SCL=STATE.PLOC(8)

  LAT=STATE.LAT
  LON=STATE.LON
  TRIANGLES=STATE.TRIANGLES
  IAI=STATE.IAI

  FDATE=STATE.INFO(0)
  FTIME=STATE.INFO(1)
  FFILTER=STATE.INFO(2)
  FEXPOSURE=STATE.INFO(3)
  FGAIN=STATE.INFO(4)

  ALT=STATE.PLOC(3)
```

```
  MAPIMAGE=STATE.MAPIMAGE
  FLAG1=STATE.FLAGS(6)
  FLAG2=STATE.FLAGS(7)

  PLOT_IMAGE,WXSZ,WYSZ,INDEX,TP,OPTION, $
    FILENAME,SITELAT, $
    SITELON,SITELATMIN,SITELATMAX,SITELONMIN, $
    SITELONMAX,LAT,LON,TRIANGLES,IAI, $
    FDATE,FTIME,FFILTER,FEXPOSURE,FGAIN,ALT, $
    MAPIMAGE,WX,WY,FLAG1,FLAG2,MAP,SATLIST,SCL

  PWIN=STATE.PWIN
  PWIN(4)=WX
  PWIN(5)=WY

  STATE={WDGT:STATE.WDGT, FLAGS:STATE.FLAGS, $
    PWIN:PWIN, PLOC:STATE.PLOC, $
    OPTION:STATE.OPTION, DIR:STATE.DIR, $
    FILE:STATE.FILE, PDIR:STATE.PDIR, $
    PFILE:STATE.PFILE, LAT:STATE.LAT, $
    LON:STATE.LON, TRIANGLES:STATE.TRIANGLES, $
    IAI:STATE.IAI, INFO:STATE.INFO, $
    MAPIMAGE:MAPIMAGE, MAP:MAP}

  END

;;;;;;;;;;;;;;;;;;;;;;;;
;;; PRO W_PRINT_IMAGE ;;;
;;;;;;;;;;;;;;;;;;;;;;;;

  PRO W_PRINT_IMAGE,STATE,SATLIST

  WXSZ=STATE.PWIN(0)
  WYSZ=STATE.PWIN(1)
  INDEX=STATE.PWIN(2)
  TP=STATE.PWIN(3)
  OPTION=STATE.OPTION

  FILENAME=STATE.FILE

  PFILENAME=STATE.PDIR+STATE.PFILE

  SITELAT=STATE.PLOC(1)
  SITELON=STATE.PLOC(2)

  SITELATMIN=STATE.PLOC(4)
  SITELATMAX=STATE.PLOC(5)
  SITELONMIN=STATE.PLOC(6)
  SITELONMAX=STATE.PLOC(7)

  SCL=STATE.PLOC(8)

  LAT=STATE.LAT
  LON=STATE.LON
  TRIANGLES=STATE.TRIANGLES
  IAI=STATE.IAI
```

```
  FDATE=STATE.INFO(0)
  FTIME=STATE.INFO(1)
  FFILTER=STATE.INFO(2)
  FEXPOSURE=STATE.INFO(3)
  FGAIN=STATE.INFO(4)

  ALT=STATE.PLOC(3)

  FLAGS=STATE.FLAGS
  FLAG1=FLAGS(2)
  FLAG2=FLAGS(7)

  PRINT_IMAGE,WXSZ,WYSZ,INDEX,TP,OPTION, $
    FILENAME,PFILENAME, SITELAT, $
    SITELON,SITELATMIN,SITELATMAX,SITELONMIN, $
    SITELONMAX,LAT,LON,TRIANGLES,IAI, $
    FDATE,FTIME,FFILTER,FEXPOSURE,FGAIN, $
    ALT,FLAG1,FLAG2,SATLIST,SCL

  STATE.FLAGS(2)=FLAG1

  END

;;;;;;;;;;;;;;;;;;;;;;
;;; PRO PLOT_IMAGE ;;;
;;;;;;;;;;;;;;;;;;;;;;

  PRO PLOT_IMAGE,WXSZ,WYSZ,INDEX,TP,OPTION, $
    FILENAME,SITELAT,SITELON,SITELATMIN, $
    SITELATMAX,SITELONMIN,SITELONMAX,LAT, $
    LON,TRIANGLES,IAI,FDATE,FTIME,FFILTER, $
    FEXPOSURE,FGAIN,ALT,WARPIMAGECG,WX,WY, $
    FLAG1,FLAG2,MAP,SATLIST,SCL

  PSTACK=[!P]
  XSTACK=[!X]
  YSTACK=[!Y]

  WSET,INDEX

  IF FLAG1 EQ 0 THEN BEGIN

;   CALHAARP,IAI,FEXPOSURE,FFILTER,FGAIN,ALT,CIAI

   CALASIP,IAI,FEXPOSURE,FFILTER,FGAIN,ALT,CIAI

;   CIAI=IAI/16  ;pn
   KER=FLTARR(3,3)+1.0
   CIAI=CONVOL(FLOAT(CIAI),KER,TOTAL(KER))

   BLANK=WHERE((LAT EQ -999.0)OR(LON EQ -999.0))
   CIAI(BLANK)=-1

  ENDIF
```

```
ALT_VALUE=STRING(ALT,FORMAT='(F5.1)')
TITLE='DATE '+FDATE+'    TIME '+FTIME+'     FILTER ' $
  +FFILTER+'     ALTITUDE '+ALT_VALUE+'   FILE ' $
  +FILENAME

IF FLAG1 EQ 0 THEN BEGIN

  GS=[0.05,0.05]

  GOOD=WHERE((LAT NE -999.0)AND(LON NE -999.0))

  XRANGE=[MIN(LON(GOOD)),MAX(LON(GOOD))]
  YRANGE=[MIN(LAT(GOOD)),MAX(LAT(GOOD))]

  LIMITS=[XRANGE(0),YRANGE(0),XRANGE(1),YRANGE(1)]

  MAPIMAGE=TRIGRID(LON(GOOD),LAT(GOOD),CIAI(GOOD), $
    TRIANGLES,GS,LIMITS,MISSING=-1.0)

ENDIF

!P.BACKGROUND=-1
!P.COLOR=0
ERASE

; ERASE,COLOR=-1 pn needed?

RMAX=SCL
RMIN=0.0


PRINT, 'RMAX-SCL: ',SCL

IF FLAG1 EQ 0 THEN BEGIN

  MAPIMAGES=1+BYTSCL(MAPIMAGE,TOP=(TP-3),MIN=RMIN,MAX=RMAX)
  MAPIMAGES(WHERE(MAPIMAGE EQ -1.0))=TP-1

ENDIF

IPOS=[0.0,0.23125,1.0,0.88125]

MAP_SET,SITELAT,SITELON,LIMIT=[SITELATMIN,SITELONMIN, $
  SITELATMAX,SITELON,SITELATMIN,SITELONMAX, $
  SITELATMIN,SITELON], $
  /ISOTROPIC,POS=IPOS,/NOBORDER

MAP={X:!X,Y:!Y}

IF FLAG1 EQ 0 THEN BEGIN

  WARPIMAGE=MAP_IMAGE(MAPIMAGES,WX,WY,WXSIZE, $
    WYSIZE,LATMIN=YRANGE(0),LATMAX=YRANGE(1), $
    LONMIN=XRANGE(0),LONMAX=XRANGE(1), $
    MISSING=255B,COMPRESS=1)
```

```
    WARPIMAGECG=CONGRID(WARPIMAGE,WXSIZE,WYSIZE)

  ENDIF

  TV,WARPIMAGECG,WX/FLOAT(WXSZ), $
    WY/FLOAT(WYSZ),/NOR

  MAP_SET,SITELAT,SITELON,LIMIT=[SITELATMIN,SITELONMIN, $
    SITELATMAX,SITELON,SITELATMIN,SITELONMAX, $
    SITELATMIN,SITELON], $
    /CONTINENT,/ISOTROPIC,/NOBORDER, $
    /NOERASE,POS=IPOS,/HIRES

; ADD ANNOTATION

  IF OPTION(1) THEN ANNOTATE,SITELON,SITELAT,' N',2
  IF OPTION(2) THEN ANNOTATE,15.83,78.2,' L',2
  XYOUTS,0.5,0.97,TITLE,ALIGN=0.5,/NOR

; OVERLAY LINE TRACES

  IF (N_ELEMENTS(WHERE(SATLIST(0:18) NE '')) GT 1) $
    THEN BEGIN

    FOR I=1,(N_ELEMENTS(WHERE(SATLIST(0:18) NE ''))-1) $
      DO BEGIN

      SATFILE=SATLIST(0)+SATLIST(I)

      IF STRUPCASE(STRMID(SATLIST(I),0,1)) EQ 'V' $
          THEN BEGIN
       VSATELLITE,FDATE,FTIME,SATFILE
      ENDIF
      IF STRUPCASE(STRMID(SATLIST(I),0,1)) EQ 'F' $
          THEN BEGIN
       FSATELLITE,FDATE,FTIME,SATFILE
      ENDIF

    ENDFOR

  ENDIF

; PLOT NORTH-SOUTH MERIDIAN LINE

  IF OPTION(0) THEN MERIDIAN,ALT

; PLOT COORDINATE GRID

  IF (FLAG2 EQ 0) THEN BEGIN

; PLOT GEOGRAPHIC COORDINATE GRID
    GEOCOORD

  ENDIF ELSE BEGIN

; PLOT MAGNETIC COORDINATE GRID
```

```
    MAGCOORD

  ENDELSE

; MAKE COLOR BAR

  CL=FINDGEN(1000)
  CLBR=[[CL],[CL]]
  CLBRS=1+BYTSCL(CLBR,TOP=(TP-3))

  TV,CONGRID(CLBRS,0.6*FLOAT(WXSZ),0.03*FLOAT(WYSZ)), $
    0.2,0.07,XSIZE=0.6,YSIZE=0.03,/NOR

  PLOT,[RMIN,RMAX],[0,1],POS=[0.2,0.07,0.8,0.1], $
    /NOR,YSTY=5,XSTY=5,/NODATA,/NOERASE
  AXIS,XAXIS=0,XSTY=1,TICKLEN=-.2
  AXIS,XAXIS=1,XTITLE='RAYLEIGH',XTICKS=1,XTICKLEN=0,$
    XTICKN=[' ',' ']

; WRITE GIF IMAGE FILE

  TVLCT,QWR,QWG,QWB,/GET

;  WRITE_IMAGE,FILENAME+'.gif','JPEG',TVRD(),QWR,QWG,QWB

  WRITE_JPEG,FILENAME+'.jpg',TVRD()
; RESTORE PARAMETERS

  !P=PSTACK(0)
  !X=XSTACK(0)
  !Y=YSTACK(0)

  END

;;;;;;;;;;;;;;;;
;;; ANNOTATE ;;;
;;;;;;;;;;;;;;;;

  PRO ANNOTATE,SITELON,SITELAT,TITLE,SHAPE

  XYOUTS,SITELON,SITELAT,TITLE,CHARSIZE=1.7
  PLOTS,SITELON,SITELAT,PSYM=SHAPE,THICK=4.0

  END

;;;;;;;;;;;;;;;;;;;;;;
;;; PRO FSATELLITE ;;;
;;;;;;;;;;;;;;;;;;;;;;

  PRO FSATELLITE,FDATE,FTIME,SATFILE

  I=-1
  DATA=FLTARR(3,5000)
  SDATE=STRARR(5000)
  SSECOND=DBLARR(5000)
```

```
RAD=!PI/180.0

D0=0
D1='D'
D2=0
D3=0
D4=0
D5=0
D6=0.0
D7=0.0
D8=0.0

OPENR,1,SATFILE

WHILE NOT EOF(1) DO BEGIN

I=I+1

READF,1,FORMAT='(I2,A3,I4,1X,I2,1X,I2,1X,F6.3,3(1X,F10.6))', $
  D0,D1,D2,D3,D4,D5,D6,D7,D8
DATA(0,I)=D6
DATA(1,I)=D7
DATA(2,I)=D8

SSECOND(I)=(D3*3600.0D)+(D4*60.0D)+D5

ENDWHILE

CLOSE,1

DATA=DATA(*,0:I)
SSECOND=SSECOND(0:I)

FSECOND=(DOUBLE(STRMID(FTIME,0,2))*3600.0D)+ $
  (DOUBLE(STRMID(FTIME,3,2))*60.0D)+ $
  DOUBLE(STRMID(FTIME,6,2))

USERSYM,1.0*COS(FINDGEN(16)*(!PI*2/15.)), $
  1.0*SIN(FINDGEN(16)*(!PI*2/15.)),THICK=2.5

IPOS=[0.0,0.23125,1.0,0.88125]

IF ((FSECOND GE MIN(SSECOND)) AND $
 (FSECOND LE MAX(SSECOND))) THEN BEGIN

 CDATA=CONVERT_COORD(DATA(2,*),DATA(1,*),/DATA,/TO_NORMAL)

 HPOS=INTERPOL(CDATA(0,*),SSECOND,FSECOND)
 VPOS=INTERPOL(CDATA(1,*),SSECOND,FSECOND)

 DDATA=CONVERT_COORD(HPOS,VPOS,/NORMAL,/TO_DEVICE)
 IF ((DDATA(0) GT !P.CLIP(0)) AND (DDATA(0) LT !P.CLIP(2)) $
      AND (DDATA(1) GT !P.CLIP(1)) AND $
      (DDATA(1) LT !P.CLIP(3))) THEN BEGIN

  PLOTS,HPOS,VPOS,PSYM=8,/NOR,NOCLIP=0
```

```
    PLOTS,CDATA(0,*),CDATA(1,*),/NOR,NOCLIP=0

   ENDIF

  ENDIF

  END

;;;;;;;;;;;;;;;;;;;;;;
;;; PRO VSATELLITE ;;;
;;;;;;;;;;;;;;;;;;;;;;

  PRO VSATELLITE,FDATE,FTIME,SATFILE

  I=-1
  DATA=FLTARR(5,5000)
  SDATE=STRARR(5000)
  SSECOND=DBLARR(5000)

  RAD=!PI/180.0

  D0=0
  D1='D'
  D2=0
  D3=0
  D4=0
  D5=0
  D6=0.0
  D7=0.0
  D8=0.0
  D9=0.0
  D10=0.0

  OPENR,1,SATFILE

  WHILE NOT EOF(1) DO BEGIN

  I=I+1

  READF,1,FORMAT='(I2,A3,I4,1X,I2,1X,I2,1X,F6.3,5(1X,F10.6))', $
    D0,D1,D2,D3,D4,D5,D6,D7,D8,D9,D10
  DATA(0,I)=D6
  DATA(1,I)=D7
  DATA(2,I)=D8
  DATA(3,I)=D9
  DATA(4,I)=D10

  SSECOND(I)=(D3*3600.0D)+(D4*60.0D)+D5

  ENDWHILE

  CLOSE,1

  DATA=DATA(*,0:I)
  SSECOND=SSECOND(0:I)
```

```
FSECOND=(DOUBLE(STRMID(FTIME,0,2))*3600.0D)+ $
  (DOUBLE(STRMID(FTIME,3,2))*60.0D)+ $
  DOUBLE(STRMID(FTIME,6,2))

USERSYM,1.0*COS(FINDGEN(16)*(!PI*2/15.)), $
  1.0*SIN(FINDGEN(16)*(!PI*2/15.)),THICK=2.5

IPOS=[0.0,0.23125,1.0,0.88125]

IF ((FSECOND GE MIN(SSECOND)) AND $
 (FSECOND LE MAX(SSECOND))) THEN BEGIN

 CDATA=CONVERT_COORD(DATA(2,*),DATA(1,*),/DATA,/TO_NORMAL)

 HPOS=INTERPOL(CDATA(0,*),SSECOND,FSECOND)
 VPOS=INTERPOL(CDATA(1,*),SSECOND,FSECOND)

 DDATA=CONVERT_COORD(HPOS,VPOS,/NORMAL,/TO_DEVICE)
 IF ((DDATA(0) GT !P.CLIP(0)) AND (DDATA(0) LT !P.CLIP(2)) $
      AND (DDATA(1) GT !P.CLIP(1)) AND $
      (DDATA(1) LT !P.CLIP(3))) THEN BEGIN

  PLOTS,HPOS,VPOS,PSYM=8,/NOR,NOCLIP=0
  PLOTS,CDATA(0,*),CDATA(1,*),/NOR,NOCLIP=0

  DLAT=INTERPOL(DATA(3,*),SSECOND,FSECOND)
  DLON=INTERPOL(DATA(4,*),SSECOND,FSECOND)

  IF (DLAT(0)+DLON(0)) NE 0.0 THEN BEGIN

   DLAT=DLAT*0.0035
   DLON=DLON*0.0035

   EDATA=CONVERT_COORD(HPOS,VPOS,/NORMAL,/TO_DATA)
   FDATA=CONVERT_COORD(EDATA(0),EDATA(1), $
         /DATA,/TO_NORMAL)
   GDATA=CONVERT_COORD(EDATA(0),EDATA(1)+1.0, $
         /DATA,/TO_NORMAL)

   ANG=-ATAN(GDATA(0)-FDATA(0),GDATA(1)-FDATA(1))

   ELAT=DLAT*COS(ANG)+DLON*SIN(ANG)
   ELON=-DLAT*SIN(ANG)+DLON*COS(ANG)

   ARROW,HPOS,VPOS,HPOS+ELON,VPOS+ELAT,/NOR,THICK=2.5,/SOLID

  ENDIF

 ENDIF

ENDIF

END

;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;
;;; PRO MERIDIAN ;;;
```

```
;;;;;;;;;;;;;;;;;;;;

  PRO MERIDIAN,ALT

  RAD=!PI/180.0
  LAT_MAG=75.4295
  LON_MAG=130.713
;  LAT_MAG=81.3
;  LON_MAG=110.8
  DIPANG=70.0-((ASIN((6360.0/(6360.0+ALT))* $
    SIN(110.0*RAD)))/RAD)

  ILAT=FLTARR(21)
  ILON=FLTARR(21)

  FOR I=0,20 DO BEGIN

    ILAT_MAG=LAT_MAG+((I/10.0)*DIPANG)-DIPANG

    GEOMAG,LAT_GEO,LON_GEO,ILAT_MAG,LON_MAG,-1

    ILAT(I)=LAT_GEO
    ILON(I)=LON_GEO

  ENDFOR

  OPLOT,ILON,ILAT

  DZ_MAG_LAT=7.9-((ASIN((6360.0/(6360.0+ALT))* $
    SIN(172.1*RAD)))/RAD)

  Z_MAG_LAT=LAT_MAG-DZ_MAG_LAT
  Z_MAG_LON=LON_MAG

  GEOMAG,Z_LAT_GEO,Z_LON_GEO,Z_MAG_LAT,Z_MAG_LON,-1

  USERSYM,1.0*COS(FINDGEN(16)*(!PI*2/16.)), $
    1.0*SIN(FINDGEN(16)*(!PI*2/16.))

  ;PLOTS,Z_LON_GEO,Z_LAT_GEO,PSYM=8

  END

;;;;;;;;;;;;;;;;;;;;
;;; PRO MAGCOORD ;;;
;;;;;;;;;;;;;;;;;;;;

  PRO MAGCOORD

  LONDEL=10.0
  LATDEL=2.0
  LONLAB=130.0+(0.5*LONDEL)
  LATLAB=74.0+(0.5*LATDEL)

  FOR MLON=0,360.0-LONDEL,LONDEL DO BEGIN
```

```
  GEOMAG,GLAT,GLON,0.0,MLON,-1
  PLOTS,GLON,GLAT

  FOR MLAT=-90,90.0,1.0 DO BEGIN

    GEOMAG,GLAT,GLON,MLAT,MLON,-1

    PLOTS,GLON,GLAT,/CONT,LINESTYLE=1

  ENDFOR

ENDFOR

FOR MLAT=-90,90.0,LATDEL DO BEGIN

  GEOMAG,GLAT,GLON,MLAT,0.0,-1

  PLOTS,GLON,GLAT

  FOR MLON=0,360.0,5.0 DO BEGIN

    GEOMAG,GLAT,GLON,MLAT,MLON,-1
    PLOTS,GLON,GLAT,/CONT,LINESTYLE=1

  ENDFOR

ENDFOR

FOR MLON=0,360.0-LONDEL,LONDEL DO BEGIN
  IF MLON GT 180.0 THEN BEGIN
    CMLON=MLON-360.0
  ENDIF ELSE BEGIN
    CMLON=MLON
  ENDELSE
  GLAT=LATLAB
  GEOMAG,GLAT,GLON,LATLAB,MLON,-1
  CDATA=CONVERT_COORD(GLON+1.0,GLAT,/DATA,/TO_DEVICE)
  IF ((CDATA(0) GT !P.CLIP(0)) AND (CDATA(0) LT !P.CLIP(2)) $
    AND (CDATA(1) GT !P.CLIP(1)) AND $
    (CDATA(1) LT !P.CLIP(3))) THEN $
  XYOUTS,CDATA(0),CDATA(1),STRTRIM(STRING(FORMAT='(I3)', $
    CMLON),2),/DEVICE,ALIGN=0.0
ENDFOR

FOR MLAT=-90,90.0,LATDEL DO BEGIN
  GEOMAG,GLAT,GLON,MLAT,LONLAB,-1
  CDATA=CONVERT_COORD(GLON+0.5,GLAT-0.2,/DATA,/TO_DEVICE)
  IF ((CDATA(0) GT !P.CLIP(0)) AND (CDATA(0) LT !P.CLIP(2)) $
    AND (CDATA(1) GT !P.CLIP(1)) AND $
    (CDATA(1) LT !P.CLIP(3))) THEN $
  XYOUTS,CDATA(0),CDATA(1),STRTRIM(STRING(FORMAT='(I3)', $
    MLAT),2),/DEVICE,ALIGN=0.0
ENDFOR

END
```

```
;;;;;;;;;;;;;;;;;;;;
;;; PRO GEOCOORD ;;;
;;;;;;;;;;;;;;;;;;;;

  PRO GEOCOORD

  LONDEL=10.0
  LATDEL=2.0
  LONLAB=20.0+(0.5*LONDEL)
  LATLAB=76.0+(0.5*LATDEL)

  FOR GLON=0,360.0,LONDEL DO BEGIN

    GLAT=0.0

    PLOTS,GLON,GLAT

    FOR GLAT=0,90.0,1.0 DO BEGIN

      PLOTS,GLON,GLAT,/CONT,LINESTYLE=1

    ENDFOR

  ENDFOR

  FOR GLAT=0,90.0,LATDEL DO BEGIN

    GLON=0.0

    PLOTS,GLON,GLAT

    FOR GLON=0,360.0,5.0 DO BEGIN

      PLOTS,GLON,GLAT,/CONT,LINESTYLE=1

    ENDFOR

  ENDFOR

  FOR GLON=0,360.0,LONDEL DO BEGIN
    IF GLON GT 180.0 THEN BEGIN
      CGLON=GLON-360.0
    ENDIF ELSE BEGIN
      CGLON=GLON
    ENDELSE
    GLAT=LATLAB
    CDATA=CONVERT_COORD(GLON,GLAT,/DATA,/TO_DEVICE)
    CDATA(0)=CDATA(0)+5.0
    IF ((CDATA(0) GT !P.CLIP(0)) AND (CDATA(0) LT !P.CLIP(2)) $
      AND (CDATA(1) GT !P.CLIP(1)) AND $
      (CDATA(1) LT !P.CLIP(3))) THEN $
    XYOUTS,CDATA(0),CDATA(1),STRTRIM(STRING(FORMAT='(I3)', $
      CGLON),2),/DEVICE,ALIGN=0.0
  ENDFOR

  FOR GLAT=0,90.0,LATDEL DO BEGIN
```

```
    GLON=LONLAB
    CDATA=CONVERT_COORD(GLON,GLAT,/DATA,/TO_DEVICE)
    CDATA(1)=CDATA(1)+8.0
    IF ((CDATA(0) GT !P.CLIP(0)) AND (CDATA(0) LT !P.CLIP(2)) $
      AND (CDATA(1) GT !P.CLIP(1)) AND $
      (CDATA(1) LT !P.CLIP(3))) THEN $
    XYOUTS,CDATA(0),CDATA(1),STRTRIM(STRING(FORMAT='(I3)', $
      GLAT),2),/DEVICE,ALIGN=0.0
  ENDFOR

  END

;;;;;;;;;;;;;;;;;;;;;;
;;; PRO PRINT_IMAGE ;;;
;;;;;;;;;;;;;;;;;;;;;;

  PRO PRINT_IMAGE,WXSZ,WYSZ,INDEX,TP,OPTION, $
    FILENAME,PFILENAME,SITELAT, $
    SITELON,SITELATMIN,SITELATMAX,SITELONMIN, $
    SITELONMAX,LAT,LON,TRIANGLES,IAI, $
    FDATE,FTIME,FFILTER,FEXPOSURE,FGAIN, $
    ALT,FLAG1,FLAG2,SATLIST,SCL

  PSTACK=[!P]
  XSTACK=[!X]
  YSTACK=[!Y]

  ON_IOERROR, ERR
  FLAG1=0

  SET_PLOT,'PS'
  !P.FONT=0
  !P.CHARSIZE=1.0
  !P.THICK=2.0
  !X.THICK=2.0
  !Y.THICK=2.0
  DEVICE,FILENAME=PFILENAME
  DEVICE,BITS_PER_PIXEL=8
  DEVICE,COLOR=0              ; set to 1 for reverse video
  DEVICE,/PORTRAIT
  DEVICE,XSIZE=8.0,YSIZE=8.0,XOFFSET=0.25, $
    YOFFSET=1.5,/INCHES

;  CALHAARP,IAI,FEXPOSURE,FFILTER,FGAIN,ALT,CIAI

  CIAI=IAI/16

  KER=FLTARR(3,3)+1.0
  CIAI=CONVOL(FLOAT(CIAI),KER,TOTAL(KER))

  BLANK=WHERE((LAT EQ -999.0)OR(LON EQ -999.0))
  CIAI(BLANK)=-1

  ALT_VALUE=STRING(ALT,FORMAT='(F5.1)')
  TITLE='DATE '+FDATE+'    TIME '+FTIME $
    +'    FILTER '+FFILTER+'    ALTITUDE ' $
```

```
  +ALT_VALUE+'  FILE '+FILENAME

GS=[0.05,0.05]

GOOD=WHERE((LAT NE -999.0)AND(LON NE -999.0))

XRANGE=[MIN(LON(GOOD)),MAX(LON(GOOD))]
YRANGE=[MIN(LAT(GOOD)),MAX(LAT(GOOD))]

LIMITS=[XRANGE(0),YRANGE(0),XRANGE(1),YRANGE(1)]

MAPIMAGE=TRIGRID(LON(GOOD),LAT(GOOD),CIAI(GOOD), $
  TRIANGLES,GS,LIMITS,MISSING=-1.0)

!P.BACKGROUND=TP-1
!P.COLOR=0
ERASE

RMAX=SCL
RMIN=0.0

MAPIMAGES=1+BYTSCL(MAPIMAGE,TOP=(TP-3),MIN=RMIN,MAX=RMAX)
MAPIMAGES(WHERE(MAPIMAGE EQ -1.0))=TP-1

IPOS=[0.0,0.23125,1.0,0.88125]

MAP_SET,SITELAT,SITELON,LIMIT=[SITELATMIN,SITELONMIN, $
  SITELATMAX,SITELON,SITELATMIN,SITELONMAX, $
  SITELATMIN,SITELON], $
  /ISOTROPIC,POS=IPOS

WARPIMAGE=MAP_IMAGE(MAPIMAGES,WX,WY,WXSIZE, $
  WYSIZE,LATMIN=YRANGE(0),LATMAX=YRANGE(1), $
  LONMIN=XRANGE(0),LONMAX=XRANGE(1), $
  MISSING=255B,COMPRESS=1)

TV,WARPIMAGE,WX,WY,XSIZE=WXSIZE,YSIZE=WYSIZE

MAP_SET,SITELAT,SITELON,LIMIT=[SITELATMIN,SITELONMIN, $
  SITELATMAX,SITELON,SITELATMIN,SITELONMAX, $
  SITELATMIN,SITELON], $
  /CONTINENT,/ISOTROPIC, $
  /NOERASE,POS=IPOS,/HIRES

; ADD ANNOTATION

IF OPTION(1) THEN ANNOTATE,SITELON,SITELAT,' N',2
IF OPTION(2) THEN ANNOTATE,15.83,78.2,' L',2
XYOUTS,0.5,0.95,TITLE,ALIGN=0.5,/NOR,CHARSIZE=0.8

; OVER LAY LINE TRACES

IF (N_ELEMENTS(WHERE(SATLIST(0:18) NE '')) GT 1) $
  THEN BEGIN

  FOR I=1,(N_ELEMENTS(WHERE(SATLIST(0:18) NE ''))-1) DO BEGIN
```

```
      SATFILE=SATLIST(0)+SATLIST(I)

      IF STRUPCASE(STRMID(SATLIST(I),0,1)) EQ 'V' $
         THEN BEGIN
       VSATELLITE,FDATE,FTIME,SATFILE
      ENDIF
      IF STRUPCASE(STRMID(SATLIST(I),0,1)) EQ 'F' $
         THEN BEGIN
       FSATELLITE,FDATE,FTIME,SATFILE
      ENDIF

    ENDFOR

  ENDIF

; PLOT NORTH-SOUTH MERIDIAN LINE

  IF OPTION(0) THEN MERIDIAN,ALT

; PLOT COORDINATE GRID

  IF (FLAG2 EQ 0) THEN BEGIN

; PLOT GEOGRAPHIC COORDINATE GRID
    GEOCOORD

  ENDIF ELSE BEGIN

; PLOT MAGNETIC COORDINATE GRID
    MAGCOORD

  ENDELSE

; MAKE COLOR BAR

  CL=FINDGEN(1000)
  CLBR=[[CL],[CL]]
  CLBRS=1+BYTSCL(CLBR,TOP=(TP-3))

  TV,CLBRS,0.2,0.05,XSIZE=0.6,YSIZE=0.03,/NOR

  PLOT,[RMIN,RMAX],[0,1],POS=[0.2,0.05,0.8,0.08], $
    /NOR,YSTY=5,XSTY=5,/NODATA,/NOERASE
  AXIS,XAXIS=0,XSTY=1,TICKLEN=-.2
  AXIS,XAXIS=1,XTITLE='RAYLEIGH',XTICKS=1,XTICKLEN=0,$
    XTICKN=[' ',' '],CHARSIZE=0.8

  FLAG1=1
  ERR:DEVICE,/CLOSE_FILE
  SET_PLOT,'X'

  !P=PSTACK(0)
  !X=XSTACK(0)
  !Y=YSTACK(0)
```

```
  END

;;;;;;;;;;;;;;;;;;;;;;;;;
;;; PRO PROJECT_IMAGE ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;

;;; THIS FILE: CONVERT_IMAGE.PRO
;;;
;;; AUTHOR:  PETER NING
;;;
;;; REV:

  PRO PROJECT_IMAGE,SITELAT,SITELON,SITEELEV,ALT, $
    LAT,LON,TRIANGLES

;ROUTINE TO FIND GEOGRAPHIC COORDINATES FOR ALL-SKY IMAGES
;FROM AZ, EL, AND ASSUMED ALTITUDE.  DOES NOT DISPLAY AN OUTPUT
;IMAGE--ONLY DETERMINES GEOGRAPHIC COORDINATES OF EACH PIXEL.
;USE TRIANGULATE AND TRIGRID TO DISPLAY IMAGE DATA.
;BASED ON ALGORITHM AND CODE FROM KEOCGMMAPS BY DAN WEIMER.
;WRITTEN UP AS INDEPENDENT PROCEDURE BY TODD PEDERSEN 4/20/98.
;THIS IMPLEMENTATION ALLOWS NON-LINEAR PIXEL-ELEVATION RELATIONS
;AS WELL AS A VARIETY OF IMAGE SIZES.
;FOR EACH IMAGER ORIENTATION/SITING AND ASSUMED ALTITUDE,
;THIS ROUTINE NEEDS TO BE RUN ONLY ONCE--
;THE GEOGRAPHIC COORDINATES OF EACH PIXEL WILL BE THE SAME
;FOR EACH IMAGE.
;VERIFICATION (UPDATED 4/27/98):
;ROUTINE ASSUMES IMAGE IS IN MAP FORMAT
;(EAST ON THE RIGHT--DEFAULT FOR HAARP)
;INPUT PARAMETER AZOFFSET PRODUCES A *CLOCKWISE* ROTATION
;FOR POSITIVE INPUT VALUES.  NOT YET TESTED AGAINST CONVERTED
;IMAGES FROM OTHER ROUTINES.
;INPUTS:
;    IMAGESIZE: INTEGER GIVING SIZE OF IMAGE (USUALLY 256)
;    XCENTER: X COORDINATE OF IMAGE CENTER, IN PIXELS
;    YCENTER: Y COORDINATE OF IMAGE CENTER, IN PIXELS
;    IMAGERADIUS: RADIUS OF THE IMAGE, IN PIXELS
;    ELCUTOFF: ELEVATION ANGLE BELOW WHICH NO CONVERSION
;      IS CARRIED OUT (DEGREES)
;    AZOFFSET: PARAMETER ALLOWING FOR ROTATION IN AZIMUTH,
;      IN DEGREES.  POSITIVE NUMBERS GIVE *CLOCKWISE* ROTATION.
;    SITELAT: LATITUDE OF OBSERVATION SITE, IN DEGREES
;    SITELON: LONGITUDE OF OBSERVATION SITE, IN DEGREES
;    SITEELEV: ALTITUDE OF OBSERVATION SITE, IN KILOMETERS
;    ALT: ASSUMED ALTITUDE OF THE OPTICAL EMISSIONS
;    RADIALFN: SET OF POLYNOMIAL COEFFICIENTS TO MODEL DEPENDENCE
;      OF ELEVATION ANGLE ON RADIAL PIXEL DISTANCE
;      (ONLY LINEAR COEFFICIENTS USED AS OF PRESENT,
;      BUT POLYNOMIAL MODELING POSSIBLE)
;      (FOR LINEAR MODEL, RADIALFN SHOULD HAVE THE FORM [0.,90./R]
;      WHERE R IS THE IMAGE RADIUS AT 0 DEG ELEVATION ANGLE)

;OUTPUTS:
;    LAT: ARRAY OF LATITUDES FOR EACH POINT IN THE IMAGE
;      (IN DEGREES)
```

```
;   LON: ARRAY OF LONGITUDES FOR EACH POINT IN THE IMAGE
;     (IN DEGREES)
;*** NOTE: POINTS IN THE ORIGINAL ARRAY FALLING OUTSIDE THE ACTUAL
;     AREA OF THE IMAGE OR BELOW THE MINIMUM ELEVATION ANGLE
;     SPECIFIED BY INPUT PARAMETER ELCUTOFF ARE FLAGGED WITH
;     LAT AND LON COORDINATES OF -999.  USE "WHERE" TO THROW
;     THESE OUT IN SUBSEQUENT PROCESSING.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; SOME OF THE FIXED INPUTS

;  IMAGESIZE=256
;
;  XCENTER=127.0
;  YCENTER=125.0
;  IMAGERADIUS=145.7
;  ELCUTOFF=20.0
;  AZOFFSET=0.0
;  RADIALFN=[0.0,90.0/145.7]

  IMAGESIZE_X=670
  IMAGESIZE_Y=650
  XCENTER=342.0
  YCENTER=302.0
  IMAGERADIUS=284.0
  ELCUTOFF=20.0
  AZOFFSET=0.0
  RADIALFN=[0.0,90.0/IMAGERADIUS]


;SIMPLE MODEL OF VARIATION OF EARTH'S RADIUS WITH LATITUDE.

  EARTHRADIUS=6356.77+21.39*COS(SITELAT*!DTOR)

;SET UP INPUT VARIABLES TO MATCH VARIABLE NAMES IN OLD CODE


  ;SITEELEV=SITEALT
  SITELAT=SITELAT
  SITELON=SITELON
  ;ALT=TARGETALT

;CONVERT TO RADIANS AND CALCULATE FACTORS

  SITELATRAD=SITELAT*!DTOR
  SITELONRAD=SITELON*!DTOR
  SLAT=SIN(SITELATRAD)
  CLAT=COS(SITELATRAD)
  SLON=SIN(SITELONRAD)
  CLON=COS(SITELONRAD)
  PX=CLAT*CLON
  PY=CLAT*SLON

  PZ=SLAT
```

```
;CALCULATE UP, EAST, AND NORTH UNIT VECTORS IN GEOGRAPHIC COORDINATES

  UPX=PX
  UPY=PY
  UPZ=PZ
  RHO=SQRT(PX^2+PY^2)
  EASTX=-PY/RHO
  EASTY=PX/RHO
  EASTZ=0.0
  NORTHX=-PX*PZ/RHO
  NORTHY=-PZ*PY/RHO
  NORTHZ=RHO

  PX=EARTHRADIUS*PX
  PY=EARTHRADIUS*PY
  PZ=EARTHRADIUS*PZ

;MORE VARIABLE NAME CONVERSIONS

  CXIMAGECENTER=XCENTER
  CYIMAGECENTER=YCENTER
  CRADIUSOF90FOV=IMAGERADIUS

;  XINIMAGE=LINDGEN(IMAGESIZE,IMAGESIZE) MOD IMAGESIZE
  XINIMAGE=LINDGEN(IMAGESIZE_X,IMAGESIZE_Y) MOD IMAGESIZE_X
  YINIMAGE=ROTATE(XINIMAGE,1)

;AZ/EL COMPUTATION

  XDELTA=XINIMAGE-CXIMAGECENTER
  YDELTA=YINIMAGE-CYIMAGECENTER
  RADIALPIXELDISTANCE=SQRT(XDELTA^2+YDELTA^2)

  AZIMUTH=FLTARR(IMAGESIZE_X,IMAGESIZE_Y)
  NOTZERO=WHERE(RADIALPIXELDISTANCE GT 0.0)

;REDEFINE AZIMUTH ANGLE SO THAT ZERO IS DUE EAST, INCREASING
;TOWARD NORTH (MATHEMATICAL CONVENTION)

  AZIMUTH(NOTZERO)=ATAN(YDELTA(NOTZERO),XDELTA(NOTZERO))

;REDEFINE TEMPORARY ARRAY VARIABLES TO FREE UP MEMORY

  XINIMAGE=0.0 & YINIMAGE=0.0 & XDELTA=0.0 & YDELTA=0.0

;CORRECT FOR AZIMUTH OFFSET, IF ANY

  AZIMUTH=AZIMUTH-AZOFFSET*!DTOR

  ELVD=90.0-POLY(RADIALPIXELDISTANCE,RADIALFN)

;FILTER OUT POINTS OUTSIDE OF IMAGE OR BELOW ELEVATION CUTOFF LIMIT.

  MASKBAD=WHERE((RADIALPIXELDISTANCE GT IMAGERADIUS) OR $
    (ELVD LT ELCUTOFF))
```

```
;FREE UP MEMORY AFTER VARIABLE NO LONGER NEEDED

  RADIALPIXELDISTANCE=0.0
  ELV=ELVD*!DTOR
  ELVD=0.0

;LAT/LON COMPUTATION

;CALCULATE X,Y,Z COORDINATES OF AURORAL IMAGE AT GIVEN ALT.
;IN A LOCAL COORDINATE SYSTEM WHERE X=EAST, Y=NORTH, AND Z=UP

  RSIN=EARTHRADIUS*SIN(ELV)
  RA=SQRT(RSIN^2 + 2.0*EARTHRADIUS*ALT + ALT^2)-RSIN

  UP=RA*SIN(ELV)
  RHO=RA*COS(ELV)
  EAST=RHO*COS(AZIMUTH)
  NORTH=RHO*SIN(AZIMUTH)

;FREE UP MEMORY

  RHO=0.0 & AZIMUTH=0.0 & ELV=0.0 & RA=0.0

;NOW ADD VECTORS TOGETHER TO GET GEOGRAPHIC COORDINATES OF
;THE IMAGE POINT

  XG=UPX*UP+EASTX*EAST+NORTHX*NORTH+PX
  YG=UPY*UP+EASTY*EAST+NORTHY*NORTH+PY
  ZG=UPZ*UP+EASTZ*EAST+NORTHZ*NORTH+PZ

;FREE UP MEMORY

  UP=0.0 & EAST=0.0 & NORTH=0.0

;NOW FIND LATITUDE AND LONGITUDE OF THESE VECTORS

  RG=SQRT(XG^2+YG^2+ZG^2)
  LAT=!RADEG*ASIN(ZG/RG)
  LON=!RADEG*ATAN(YG,XG)

;RETURN OUTPUT VARIABLES TO CALLLING PROCEDURE,
;FLAGGING POINTS OUTSIDE THE IMAGE OR BELOW THE MINIMUM
;ELEVATION ANGLE WITH VALUES OF -999.

  LAT(MASKBAD)=-999.0
  LON(MASKBAD)=-999.0

  GOOD=WHERE((LAT NE -999.0)AND(LON NE -999.0))
  TRIANGULATE,LON(GOOD),LAT(GOOD),TRIANGLES

  END

;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;
;;; PRO LOAD_IMAGE ;;;
;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;
```

```
  PRO LOAD_IMAGE,FILENAME,INDEX,TP, $
    IAI,FDATE,FTIME,FFILTER, $
    FEXPOSURE,FGAIN, $
    FLAG2,FLAG3,FLAG5

  ON_IOERROR, ERR
  FLAG2=0

;  HDR=BYTARR(74)
;  ITI_COMMENT=BYTARR(185)
;  IAI=INTARR(256,256)
   IAI=UINTARR(670,650)        ;pn

;  DUMMY4BYTES=BYTARR(4)
  OPENR,1,FILENAME  ;OPEN IMAGE FILE
;  STATUS=FSTAT(1)
;  READU,1,HDR
;  READU,1,ITI_COMMENT
; IF (STATUS.SIZE EQ 131335) THEN READU,1,DUMMY4BYTES
  READU,1,IAI
  CLOSE,1
  IF (FLAG3 EQ 1) THEN BEGIN
  UPPER_BYTE=ISHFT(IAI,8)
  LOWER_BYTE=ISHFT(IAI,-8)
;  IAI=(UPPER_BYTE OR LOWER_BYTE) AND '0FFF'X
  IAI=(UPPER_BYTE OR LOWER_BYTE)
  ENDIF

;  IAI=ROTATE(IAI,7)  ;FLIP UPSIDE DOWN
   i16ImgParams=INTARR(16)
  i16ImgParams=IAI(0:25)
  IAI=ROTATE(IAI,2)  ;FLIP UPSIDE DOWN & LEFT RIGHT


;  FDATE=STRUPCASE(STRCOMPRESS(STRMID(ITI_COMMENT,1,9), $
;    /REMOVE_ALL))
;  FTIME=STRMID(ITI_COMMENT,11,8)
;  FFILTER=STRMID(ITI_COMMENT,30,4)
;  FEXPOSURE=STRMID(STRING(ITI_COMMENT),37,4)/10.0
;  FGAIN=STRMID(ITI_COMMENT,22,1)

  monthname=['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec']
  MONTH=monthname[i16ImgParams(1)-1]
  DAY=strcompress(string(i16ImgParams(2)),/remove)
  YEAR=strcompress(string(i16ImgParams(0)),/remove_all)
  FDATE=STRCOMPRESS(DAY+MONTH+YEAR)
  PRINT,FDATE
  FTIME=STRING(i16ImgParams(3:5),FORMAT='(I2.2,":",I2.2,":",I2.2)')
  FFILTER=STRCOMPRESS(STRING(i16ImgParams(6)),/REMOVE_ALL)
  FEXPOSURE=strcompress(string(float(i16ImgParams(8)/10.),format='(F6.1)'),/rem)
  FGAIN=STRING(i16ImgParams(7))


  IF (FLAG5 EQ 1) THEN BEGIN
    WSET,INDEX
;    BIAI=1+BYTSCL(IAI,TOP=(TP-3),MIN=200.0,MAX=4100.0)
```

```
   BIAI=1+BYTSCL(IAI,TOP=(TP-3),MIN=512.0,MAX=65535.)
   BIAI(WHERE(BIAI EQ 1))=TP-1
;  TV,CONGRID(BIAI,500,500)
    TV,BIAI
  ENDIF

  FLAG2=1
  ERR:IF FLAG2 EQ 0 THEN BEGIN
    CLOSE,1
    RESULT=DIALOG_MESSAGE(['AN ERROR OCCURED WHILE TRYING '+ $
      'TO READ THE IMAGE DATA FILE.', ' ', $
      'BE SURE YOU HAVE SELECTED A VALID FILE '+ $
      'AND TRY AGAIN'],/ERROR)
  ENDIF

  END

;;;;;;;;;;;;;;;;;;
;;; PRO GEOMAG ;;;
;;;;;;;;;;;;;;;;;;

  PRO GEOMAG,LAT_GEO,LON_GEO,LAT_MAG,LON_MAG,J

  RAD=!PI/180.0

  GAMMA=70.905*RAD
  BETA=-11.018*RAD
  ALPHA=0.0

  R11=COS(GAMMA)*COS(BETA)*COS(ALPHA)-SIN(GAMMA)*SIN(ALPHA)
  R12=COS(GAMMA)*COS(BETA)*SIN(ALPHA)+SIN(GAMMA)*COS(ALPHA)
  R13=-COS(GAMMA)*SIN(BETA)
  R21=-SIN(GAMMA)*COS(BETA)*COS(ALPHA)-COS(GAMMA)*SIN(ALPHA)
  R22=SIN(GAMMA)*COS(BETA)*SIN(ALPHA)+COS(GAMMA)*COS(ALPHA)
  R23=SIN(GAMMA)*SIN(BETA)
  R31=SIN(BETA)*COS(ALPHA)
  R32=SIN(BETA)*SIN(ALPHA)
  R33=COS(BETA)

  IF J LT 0 THEN BEGIN
   THETA_MAG=(90.0-LAT_MAG)*RAD
   PHI_MAG=LON_MAG*RAD
   XM=SIN(THETA_MAG)*COS(PHI_MAG)
   YM=SIN(THETA_MAG)*SIN(PHI_MAG)
   ZM=COS(THETA_MAG)
   XG=(R11*XM)+(R12*YM)+(R13*ZM)
   YG=(R21*XM)+(R22*YM)+(R23*ZM)
   ZG=(R31*XM)+(R32*YM)+(R33*ZM)
   THETA_GEO=ATAN(SQRT((XG^2.0)+(YG^2.0)),ZG)
   PHI_GEO=ATAN(YG,XG)
   LAT_GEO=90.0-(THETA_GEO/RAD)
   LON_GEO=PHI_GEO/RAD
  ENDIF ELSE BEGIN
   THETA_GEO=(90.0-LAT_GEO)*RAD
   PHI_GEO=LON_GEO*RAD
   XG=SIN(THETA_GEO)*COS(PHI_GEO)
```

```
    YG=SIN(THETA_GEO)*SIN(PHI_GEO)
    ZG=COS(THETA_GEO)
    XM=(R11*XG)+(R21*YG)+(R31*ZG)
    YM=(R12*XG)+(R22*YG)+(R32*ZG)
    ZM=(R13*XG)+(R23*YG)+(R33*ZG)
    THETA_MAG=ATAN(SQRT((XM^2.0)+(YM^2.0)),ZM)
    PHI_MAG=ATAN(YM,XM)
    LAT_MAG=90.0-(THETA_MAG/RAD)
    LON_MAG=PHI_MAG/RAD
   ENDELSE

   END

;;;;;;;;;;;;;;;;;;;;
;;; PRO SET_COLOR ;;;
;;;;;;;;;;;;;;;;;;;;

   PRO SET_COLOR,COLOR,TP

;   TP=!D.N_COLORS

   TP=!D.TABLE_SIZE   ;pn - changed from above

PRINT,TP,!D.N_COLORS

   HUE=FLTARR(TP)
   SAT=FLTARR(TP)
   VAL=FLTARR(TP)

   CASE COLOR OF

     'GRAY' : BEGIN

       HUE=HUE+0
       SAT=SAT+0.0
       VAL(1:TP-2)=(TP-3-FINDGEN(TP-2))*(1.0/(FLOAT(TP-3)))
       ENDCASE

     'RED' : BEGIN

       HUE=HUE+0
       SAT(1:TP-2)=FINDGEN(TP-2)*(1.0/(FLOAT(TP-3)))
       VAL=VAL+1.0
       ENDCASE

     'GREEN' : BEGIN

       HUE=HUE+120
       SAT(1:TP-2)=FINDGEN(TP-2)*(1.0/(FLOAT(TP-3)))
       VAL=VAL+1.0
       VAL(1:TP-2)=1-FINDGEN(TP-2)*(0.5/(FLOAT(TP-3)))
       ENDCASE

     'BLUE' : BEGIN

       HUE=HUE+240
```

```
      SAT(1:TP-2)=FINDGEN(TP-2)*(1.0/(FLOAT(TP-3)))
      VAL=VAL+1.0
      ENDCASE

    'RAINBOW1' : BEGIN

      HUE(1:TP-2)=240.0-(FINDGEN(TP-2)*(240.0/(FLOAT(TP-3))))
      SAT(1:TP-2)=SAT(1:TP-2)+1.0
      VAL(1:TP-2)=VAL(1:TP-2)+1.0
      ENDCASE

    'RAINBOW2' : BEGIN

      huearr=[240,240,210,180,120,60,40,10,0]
      satarr=[1.0,1.0,1.0,1.0,1.0,1.0,1.0,0.7,1.0]
      valarr=[0.5,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0]

      cindex=fix(findgen(tp-2)*9.0/(tp-2))
      hue(1:tp-2)=huearr(cindex)
      SAT(1:TP-2)=satarr(cindex)
      VAL(1:TP-2)=valarr(cindex)
      ENDCASE

    ELSE:
  ENDCASE

  TVLCT,HUE,SAT,VAL,/HSV

  HUE(0)=0
  SAT(0)=0.0
  VAL(0)=0.0
  HUE(TP-1)=0
  SAT(TP-1)=0.0
  VAL(TP-1)=1.0


  TVLCT,HUE,SAT,VAL,/HSV

  END

PRO CALASIP,IMAGE,EXPOSURE,FILTER,GAIN,ALT,CIMAGE

  EXPOSURE=FLOAT(EXPOSURE)
  GAIN=FIX(GAIN)
  BIAS=71.79
  DCR=[11.16,125.92,1147.72,0.]

  CASE FIX(FILTER) OF
    6300: LCR=[0.653516,7.88324,74.9065,0.0]
    5577: LCR=[0.709080,8.36213,81.2303,0.0]
    4278: LCR=[0.398004,4.82669,42.9298,0.0]
      7774: LCR=[0.461177,5.55789,50.5229,0.0]
      6560: LCR=[0.705478,8.60690,79.8694,0.0]

  ELSE:BEGIN
    PRINT,'CALIBRATION DATA FOR THAT FILTER NOT AVAILABLE.'
```

```
    PRINT,'WARNING: NO CALIBRATION PERFORMED."
    STOP
  END

  ENDCASE

  XO=342 & YO=302 & R=284
  X=(INDGEN(670)-XO)#REPLICATE(1,650)
  Y=REPLICATE(1,670)#(INDGEN(650)-YO)
  RHO=SQRT(X^2.0+Y^2.0)
  X=0.0 & Y=0.0
  BAD=WHERE(RHO GT R)

;POLYNOMIAL COEFFICIENTS FOR ZENITH ANGLE
  COEFFS=[0.55071073,0.20038557,0.0010658430,-7.1024379e-06,1.6577757e-08]

;NEED ZENITH ANGLE TO CALCULATE VIGNETTING AND VAN RHIJN
  ZEN=POLY(RHO,COEFFS)
  RHO=0.

;VIGNETTING FUNCTION--DIVIDE BY THIS LATER TO CORRECT IMAGE
  VIGFN=1.-0.952242/90.*ZEN

;DO VAN RHIJN CORRECTION
  EL=90.0-ZEN
  ZEN=0.0
  RE=6372.0              ; Todd's
  SITELAT=78.92    ; DEGREES
  RE=6356.77+21.39*COS(SITELAT*!DTOR)
;ANGLE BETWEEN LINE OF SIGHT AND VERTICAL, AT OBSERVED VOLUME
  ALPHA=ASIN(RE*COS(!DTOR*EL)/(RE+ALT))

;CORRECTION FACTOR--MULTIPLY BY THIS LATER ON
  V=COS(ALPHA)

  ZEN=0.0 & EL=0.0

  FIMAGE=FLOAT(IMAGE)


IMAGE_STATISTICS,FIMAGE[200:400,200:400],MEAN=MEAN,STDDEV=SD,MINIMUM=MIN,M
AXIMUM=MAX
  PRINT,MEAN,SD,MIN,MAX,FORMAT='(%"MEAN=%9.2f SD=%9.2f MIN=%9.2f MAX=%9.2f")'

  PRINT ,GAIN, EXPOSURE, DCR[GAIN],LCR[GAIN]
  CIMAGE=((FIMAGE-BIAS)/EXPOSURE-DCR[GAIN])/LCR[GAIN]/VIGFN*V

  CIMAGE(BAD)=0.0
END

;;;;;;;;;;;;;;;;;;;;
;;; PRO CALHAARP3 ;;;
;;;;;;;;;;;;;;;;;;;;

  PRO CALHAARP,IMAGE,EXPOSURE,FILTER,GAIN,ALT,CIMAGE
```

```
; PROGRAM TO CALIBRATE ALL-SKY IMAGES FROM THE HAARP IMAGER
; TO GIVE EMISSION INTENSITIES IN RAYLEIGHS.  CURRENT VERSION
; BASED ON EMPIRICAL CALIBRATION PERFORMED BY TODD PEDERSEN
; 7/21/98 USING KEO LIGHT SOURCE #1 OVER A RANGE OF INTENSITIES
; AND EXPOSURE TIMES FOR EACH GAIN AND WAVELENGTH SETTING.
; THIS VERSION TAKES INTO ACCOUNT BACKGROUND COUNTS, CCD BIAS, ETC.
; VIGNETTING IS APPROXIMATED BY A LINEAR FUNCTION, WHICH HAS NOT
; BEEN EMPIRICALLY UPDATED SINCE THE ORIGINAL INSTRUMENT
; CALIBRATION. BECAUSE VIGNETTING AND VAN RHIJN EFFECTS MORE OR
; LESS CANCEL OUT, CALIBRATED IMAGES SUFFER FROM APPARENT
; UNDEREXPOSURE NEAR THE CENTER. SET THE KEYWORD VANRHIJN TO THE
; ASSUMED ALTITUDE (I.E. VANRHIJN=300., ETC.) TO CORRECT FOR THIS.
; THIS VERSION SHOULD WORK WITH BOTH 256X256 AND 512X512 PIXEL
; IMAGES--ACTUAL CALIBRATION WAS DONE WITH 512X512 IMAGES;
; CALIBRATION PARAMETERS ARE THEN CORRECTED FOR 256X256 IMAGES
; (MULTIPLY DCR AND LCR BY 4, LEAVE BIAS UNCHANGED).

; BY TODD PEDERSEN 7/22/98
; BASED ON CALHAARP.PRO BY TODD PEDERSEN
; (***CAUTION: INTENSITY SLOPES ARE DEFINED
; DIFFERENTLY IN THIS VERSION, SO DON'T TRY TO COMPARE DIRECTLY
; WITH OLD VERSION***)

; PRELIMINARILY TESTED 7/22/98:
; RUNS OK, GIVES CORRECT RESULTS FOR BOTH 256X256 AND 512X512 IMAGES
;
; INPUTS:
;    IMAGE:  SQUARE ARRAY CONTAINING THE RAW INPUT IMAGE
;    EXPOSURE:   EXPOSURE TIME IN SECONDS
;    GAIN:   INSTRUMENT GAIN SETTING (INTEGER: 0-3)
;    FILTER:   FILTER WAVELENGTH (INTEGER)--CURVES ARE DIFFERENT
;            FOR EACH FILTER AND GAIN
;
; OUTPUTS:
;    RETURNS CALIBRATED IMAGE AS FUNCTION VALUE,
;      IN UNITS OF RAYLEIGH.
;

;SLOPES OF INTENSITY CURVES FOR EACH GAIN (DEPENDS ON WAVELENGTH)

  LCR=FLTARR(4)

;SLOPES OF DARK COUNT CURVES FOR EACH GAIN (WAVELENGTH INDEPENDENT)

  DCR=[1.28,1.37,1.67,2.58]

;CCD BIAS (INDEPENDENT OF EXPOSURE TIME, GAIN, AND WAVELENTH)

  BIAS=14.46


  CASE FIX(FILTER) OF

   6300:BEGIN
     LCR=[0.140,0.290,0.547,1.900]
   END
```

```
    5577:BEGIN
      LCR=[0.28,0.564,1.084,3.866]
    END

    4278:BEGIN
      LCR=[0.269,0.539,1.014,3.730]
    END

    ELSE:BEGIN
      PRINT,'CALIBRATION DATA FOR THAT FILTER NOT AVAILABLE.'
      PRINT,'WARNING: NO CALIBRATION PERFORMED."
      STOP
    END

  ENDCASE

;IMAGE CENTER AND RADIUS IN PIXELS

  XO=127 & YO=125 & R=122

  X=(INDGEN(256)-XO)#REPLICATE(1,256)
  Y=REPLICATE(1,256)#(INDGEN(256)-YO)
  RHO=SQRT(X^2.0+Y^2.0)
  X=0.0 & Y=0.0
  BAD=WHERE(RHO GT R)

;POLYNOMIAL COEFFICIENTS FOR ZENITH ANGLE
  COEFFS=[0.,0.4<83202,0.00704043,-0.000110095,5.62538E-07]

;NEED ZENITH ANGLE TO CALCULATE VIGNETTING AND VAN RHIJN
  ZEN=POLY(RHO,COEFFS)

  RHO=0.

;VIGNETTING FUNCTION--DIVIDE BY THIS LATER TO CORRECT IMAGE
  VIGFN=1.-0.82/90.*ZEN

;DO VAN RHIJN CORRECTION IF REQUESTED

    ;PRINT,'CORRECTING FOR VAN RHIJN EFFECT FOR ASSUMED '+ $
    ;  'ALTITUDE OF '+STRING(ALT)+' KM.'
    EL=90.0-ZEN
    ZEN=0.0
    RE=6372.0

;ANGLE BETWEEN LINE OF SIGHT AND VERTICAL, AT OBSERVED VOLUME
    ALPHA=ASIN(RE*COS(!DTOR*EL)/(RE+ALT))

;CORRECTION FACTOR--MULTIPLY BY THIS LATER ON
    V=COS(ALPHA)
    ;V=1.0

  ZEN=0.0 & EL=0.0

;DIFFERENT PROCESSING FOR 1X1 AND 2X2 BINNING
```

```
  S=SIZE(IMAGE)

;HANDLING FOR 2X2 BINNING
  IF S(1) EQ 256 THEN BEGIN

    DCR=DCR*4.
    LCR=LCR*4.

;HANDLING FOR 1X1 BINNING
  ENDIF ELSE IF S(1) EQ 512 THEN BEGIN

;CHANGE SIZE ON VIGNETTING, VAN RHIJN AND BAD POINT MASK
    VIGFN=REBIN(VIGFN,512,512)

    IF V NE 1.0 THEN V=REBIN(V,512,512)
      DUMMY=REPLICATE(1.0,256,256) & DUMMY(BAD)=0.
      BAD=WHERE(REBIN(DUMMY,512,512) LT 1.)
    ENDIF ELSE PRINT,'INCORRECT IMAGE SIZE'

  GAIN=FIX(GAIN)

;APPLY CALIBRATION
  CIMAGE=((IMAGE-BIAS)/EXPOSURE-DCR(GAIN))/LCR(GAIN)/VIGFN*V

;MASK POINTS OUTSIDE ACTUAL IMAGE
  CIMAGE(BAD)=0.0

  END

;;;;;;;;;;;;;;;;;;;;;;;;
;;; FUNCTION YTICKS ;;;
;;;;;;;;;;;;;;;;;;;;;;;;

  FUNCTION YTICKS, AXIS,INDEX,VALUE

  POWER=ALOG10(VALUE)
  RETURN, STRING(POWER, FORMAT="('10!U',I2,'!N')")

  END

;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; PRO SAMPLE_IMAGE ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;

  PRO SAMPLE_IMAGE

  READ_GIF,'test.gif',IMAGE,R,G,B

  TVLCT,R,G,B

  TV,IMAGE

  END

;;;;;;;;;;;;;;;;;;;;;;;
;;; PRO ALLSKY1.5 ;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;

  ALLSKY

  END
```

6.8 SMARTMOTOR RFL01.SMS Program Listing

```
'************************************************************'
'                                                            '
'  ARFL01.SMS        2/8/01                                  '
'                      KEO Consultants                       '
'                      Cyril Lance                           '
'                                                            '
'  SMART-MOTOR embedded software for controlling the ARFL    '
'  camera electronics through the DIO-116 Parallel I/O board '
'  from the SMART-MOTOR to the Intensifier/Shutter Control   '
'  board.  Routines in the program are also included to      '
'  control the position of the Filterwheel.  A HOME reset    '
'  code is included to home to the magnetic sensor, and then '
'  move a specified offset to the first FW position.         '
'                                                            '
'  4" Optics, Six Position Filterwheel, Bare CCD AND Intensified '
'  Imager.  Uses Princeton Scientific VersArray CCD Camera Sysetm.'
'  Intensifier, Shutter, Light Sensor use standard KEO Electronics'
'  interface.  Filterwheel uses a 30:1 gear reducer.         '
'                                                            '
'  modified 6/21/00 to have version 4.10 and version 4.11 bug '
'  fix from ANIMATICS                                        '
'                                                            '
'  modified 11/27/01 (Final) to make sure that ALL subroutines are terminated '
'  with the <cr> (#13) character.                            '
'                                                            '
'  Program loops, allowing input from user and then calls    '
'  to the various subroutines.                               '
'                                                            '
'  Variables used:                                           '
'                                                            '
'     a ==>  Holds contents of last read input (DINB0)       '
'     b ==>  Holds contents of output register (DOUTA0)      '
'     c ==>  Holds Light Detector Status 0=Light, 1=Dark     '
'     d ==>  Last read 'Shutter Status Bits' 0=closed, 1=open'
'     e ==>  Holds Intensifier Gain                          '
'     f ==>  Holds Intensifier Power Status 0=off, 1=on      '
'     g ==>  Holds next filter position                      '
'     h ==>  Holds present filter position                   '
'     j ==>  Holds the HOME offset                           '
'                                                            '
'  SUBROUTINES used:                                         '
'                                                            '
'     C0 ==> Read_Light_Detector_State                       '
'     C1 ==> Shutter_Control                                 '
'     C2 ==> Intensifier_Gain_Control                        '
'     C3 ==> Intensifier_Power_Control                       '
'     C4 ==> Filterwheel_Control                             '
'     C5 ==> HOME_Reset                                      '
'                                                            '
'************************************************************'

SADDR1              ' Identifies this as the motor #1
```

```
RUN?                    ' Don't autoload for now...

' Initialize the output port:  Shutters off, Int. Gain = 0

b=7                     ' Set Ouput bits, STROBE hi
DOUTA0,b
b=3                     ' Set STROBE low
DOUTA0,b
b=7                     ' Set STROBE hi to clock data
DOUTA0,b

' Initialize variables for SUBROUTINE calls

d=-1                    ' Shutter Parameter -- just read status
e=-1                    ' Intensifier Gain -- just read status
f=-1                    ' Intensifier Power -- just read status
g=-1                    ' Filter Wheel Report -- just report position

' Initialize the motor variables and move around a bit to get the
' feeling that things are working!

AMPS=300                ' Set the Maximum Current to about 2 Amp
MP                      ' Set the Motor to "Position Mode"

'  version 4.10 and version 4.11 bug fix from ANIMATICS

KGON
KGOFF

' ADJUST THE PWM PARAMETERS FOR SMOOTH, RELIABLE OPERATION

 KP=500
 KD=1000
 F

' SET NORMAL MOVE ACCELERATION AND VELOCITY PARAMETERS

A=1000
V=1000000

' HOME Initialization -- go to magnetic detector and slew to offset
' *** HOME Offset is defined in HOME routine!!!  Don't Change! ***

GOSUB5                  ' Find HOME Position, then slew to position #1

WHILE 1 LOOP            ' Program loops forever until user types "END"
END                     ' End of "MAIN" Program definition

'******************************'
'                              '
'  SUBROUTINE Defintions       '
'                              '
'******************************'

' SUBROUTINE C0:  Read_Light_Detector_State( var c )
```

```
C0                              ' Read light status:  LO=light, HI=dark
  a=DINB0                       ' Get input BYTE (PORT B)
  c=a&8                         ' Mask out B3
  c=c/8                         ' Shift value to B0
  PRINT("LIGHT:",c)             ' Print value to screen
  PRINT(#13)            ' Print TERMINATING character <cr>
RETURN                          ' Return to MAIN routine

' SUBROUTINE C1:  Shutter_Control( var d )   ' Uses Shutter #1

C1
  IF d==0                       ' Close the shutter
    b=b|1                       ' Close -- Shutter 1 bit HI (B0)
    DOUTA0,b                    ' and output to DIO-116 PORTA
    WAIT=41                      ' Wait 10 msec
    d=-1                        ' Set to read status
  ENDIF

  IF d==1                       ' Open the shutter
    b=b&254                     ' Open -- Shutter 1 bit LO (B0)
    DOUTA0,b                    ' and output to DIO-116 PORTA
    WAIT=41                     ' Wait 10 msec
    d=-1                        ' Set to read status
  ENDIF

  IF d==-1
    PRINT("SHTR:???", #13)
  ENDIF                         ' Finished reporting status

  RETURN                      ' End of Subroutine C1

' SUBROUTINE C2:  Intensifier_Gain_Control( var e )

C2
  IF e>3                      ' Check for non-valid gain
    PRINT("GAIN:-1",#13)
    RETURN
  ENDIF

  IF e>=0               ' Check for valid gain
    b=b&207             ' Clear the output bits (B4-B5)
    e=e*16              ' Shift gain over to B4-B5
    b=b|e               ' OR gain into output BYTE
    DOUTA0,b                    ' and output to DIO-116 PORTA

    b=b&251             ' Set the STROBE bit low (B2)
    DOUTA0,b            ' and output to DIO-116 PORTA

    b=b|4                       ' Set the STROBE bit high (B2)
    DOUTA0,b                    ' and output to DIO-116 PORTA
    e=-1                ' Flag to read intensifier gain

  ENDIF                 ' Intensifier gain has been set

  IF e==-1              ' Flagged to read gain
    a=DINB0                     ' Get input BYTE (PORT B)
```

```
    e=a&3                       ' Mask out B0-B1
    PRINT("GAIN:")              ' Print value to screen
    PRINT(e)
    PRINT(#13)
  ENDIF

  RETURN

' SUBROUTINE C3:  Intensifier_Power_Control( var f )

C3
  IF f==1                       ' Command:  Turn Power ON
    b=b|8                     ' Set the Int. Power bit (B3)
    DOUTA0,b                    ' and output to DIO-116 PORTA
    f=-1                        ' Set Flag to read power
  ENDIF

  IF f==0                       ' Command:  Turn Power OFF
    b=b&247                     ' Clear the Int. Power bit (B3)
    DOUTA0,b                    ' and output to DIO-116 PORTA
    f=-1                        ' Set Flag to read power
  ENDIF

  IF f==-1
    a=DINB0                     ' Get input BYTE (PORT B)
    f=a&4                       ' Mask out B2
    IF f                        ' INTSTS* --> Active Low...
      PRINT("INTPWR:OFF") ' Bit HI:  Power is OFF
    ENDIF
    IF f==0
      PRINT("INTPWR:ON")    ' Bit LO:  Power is ON
    ENDIF
    PRINT(#13)                  ' Print CR character
  ENDIF

  RETURN                        ' Return to MAIN routine

' SUBROUTINE C4:  Filterwheel_Control( var g )

C4
  IF g==-1                    ' Request for present position
    h=@P/10000                 ' Update present position
    PRINT("FILT:")            ' and report the position
    Rh                          ' to the user
    RETURN
  ENDIF

  IF g>6
    PRINT("FILT:-1")
    PRINT(#13)
    RETURN
  ENDIF

  IF g<1
    PRINT("FILT:-1")
    PRINT(#13)
```

```
  RETURN
 ENDIF

 IF h==1                        ' Present Position is 10000
   IF g<5
     P=g*10000                  ' Next position = g*10000
   ENDIF
   IF g>=5
    i=g*10000
    i=i-60000
     P=i                        ' Move in opposite direction
   ENDIF
 ENDIF

 IF h==2                        ' Present Position is 20000
   IF g<6
     P=g*10000                  ' Next position is g*10000
   ENDIF
   IF g==6
     P=0                        ' Move in opposite direction
   ENDIF
 ENDIF

IF h==3                         ' Present Position is 30000
    P=g*10000                   ' Next position is g*10000
 ENDIF

IF h==4                         ' Present Position is 40000
    P=g*10000                   ' Next position is g*10000
 ENDIF

 IF h==5                        ' Present Position is 50000
   IF g<3
     i=g*10000          ' Next position is g*10000
     i=i+60000
     P=i
   ENDIF
   IF g>2
     P=g*10000                  ' Next Position is g*10000
   ENDIF
 ENDIF

 IF h==6                ' Present Position is 60000
   IF g<3
     i=g*10000
     i=i+60000
     P=i                        ' Move in positive direction (wrap around)
   ENDIF
   IF g>=3
     P=g*10000          ' Next position is g*10000
   ENDIF
 ENDIF

 G                              ' Make the filter move to next position
 TWAIT                  ' Wait until trajectory is finished
 h=g                            ' Reset the present filter position
```

```
  O=h*10000                    ' and reset the present origin

  PRINT("FILT:")
  Rh
RETURN

' SUBROUTINE C5:  HOME_Reset

C5
  V50000              ' Slower Homing velocity
  A800                         ' Home Acceleration

  i=UAI                        ' Are we at the HOME bit?
  IF i==0                      ' move off it a little bit.
    i=@P-1000         ' to guarantee that we're not already at HOME
    P=i
    MP
    G
    TWAIT
  ENDIF

  MV                           ' Put motor in Velocity Mode
  UAI                          ' Define the Encoder A bit as input

  G                            ' Start slewing the motor
  i=UAI                        ' Look at the HOME bit
  WHILE i==1          ' Slew while this is not set (active_low)
    i=UAI                      ' and keep reading the HOME bit
  LOOP

WHILE i==0
    i=UAI
  LOOP

  A400
  V=-1200             ' Set the velocity to slow reverse
  G                            ' and move backwards until beg. of HOME

  WHILE i==1                   ' go back to beginning of pulse
    i=UAI
  LOOP

  A=4000              ' setup for a very quick stop
  X

  G                            ' Bring the motor to a HALT!!!  (AT HOME)
  TWAIT
  WAIT=400                     ' Wait 0.1 sec just to be safe
  O=5350              ' Set this as offset origin

  MP                           ' Put motor in Position MODE
  V=1000000                    ' Normal slew velocity
  A=1000                       ' Normal slew acceleration

  P=10000                      ' Setup to slew to position #1 from HOME
  G                            ' Make the move
```

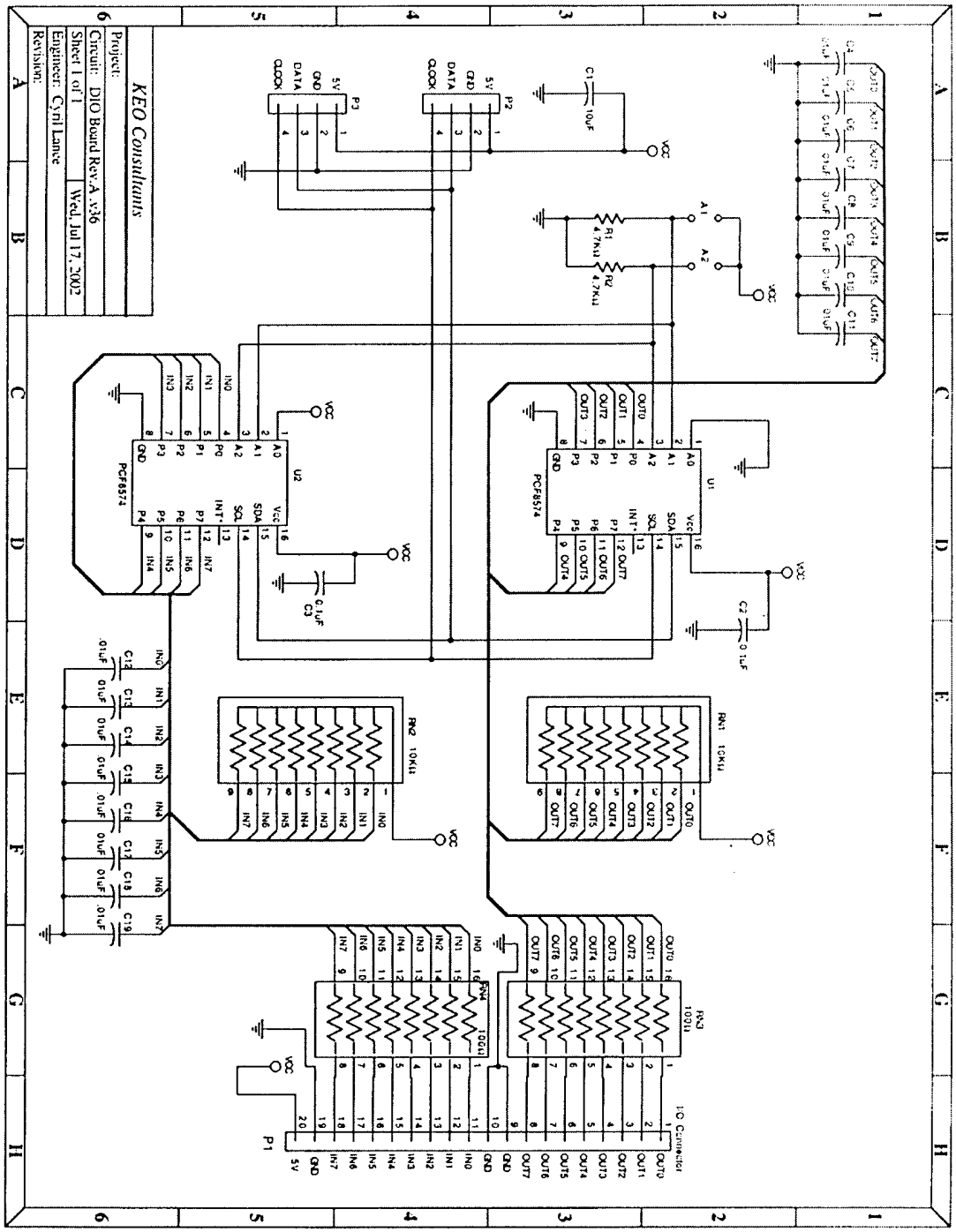
```
  TWAIT                         ' Wait until the trajectory is over
  WAIT=5z00                     ' Wait 1 sec for things to settle

  h=1                           ' Set to current position
  PRINT("HOME:1")
  PRINT(#13)
RETURN

END             ' END of ARFL10.SMS code
```

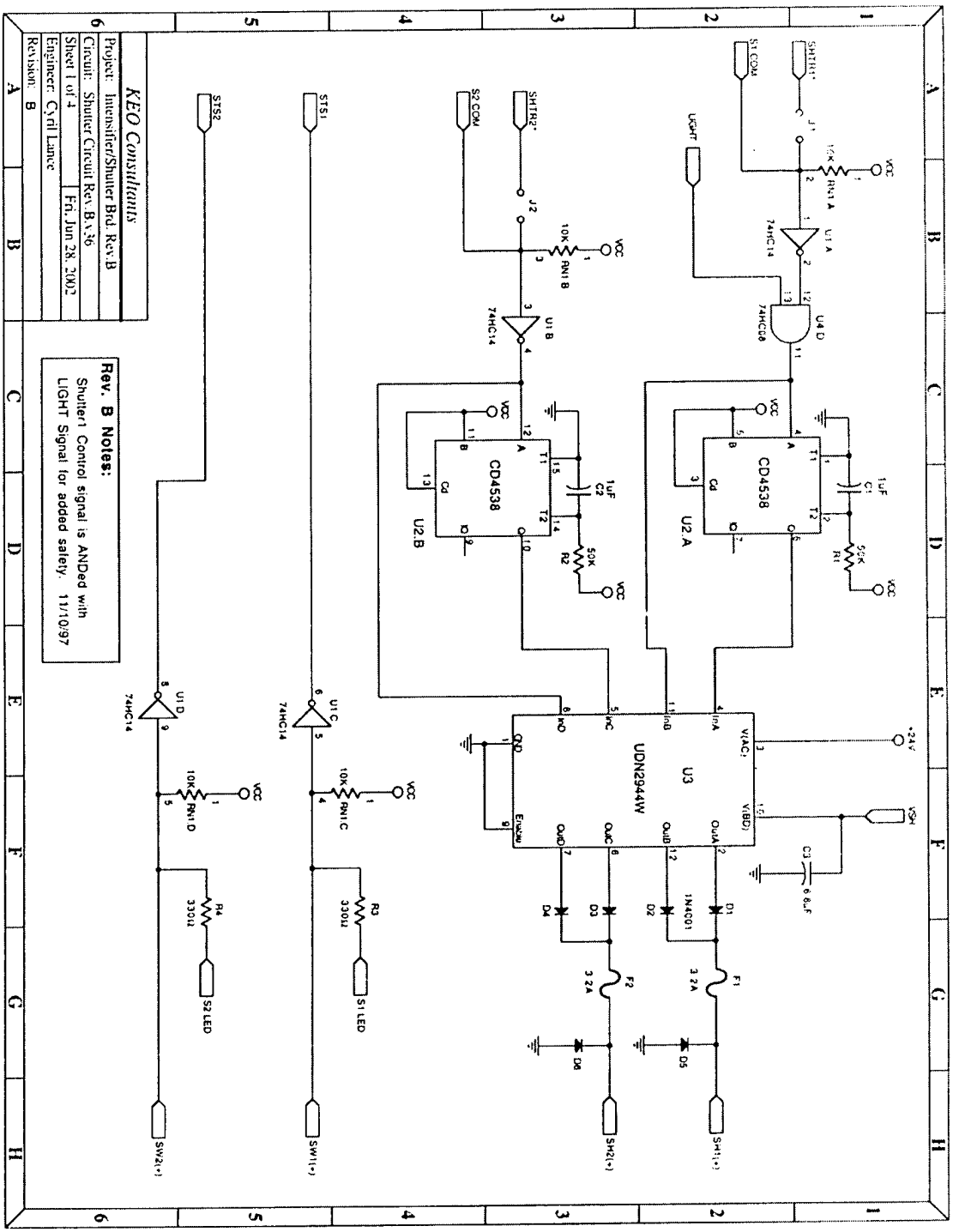
Chapter 7: Hardware References

7.1 Schematics: DIO Board Rev A

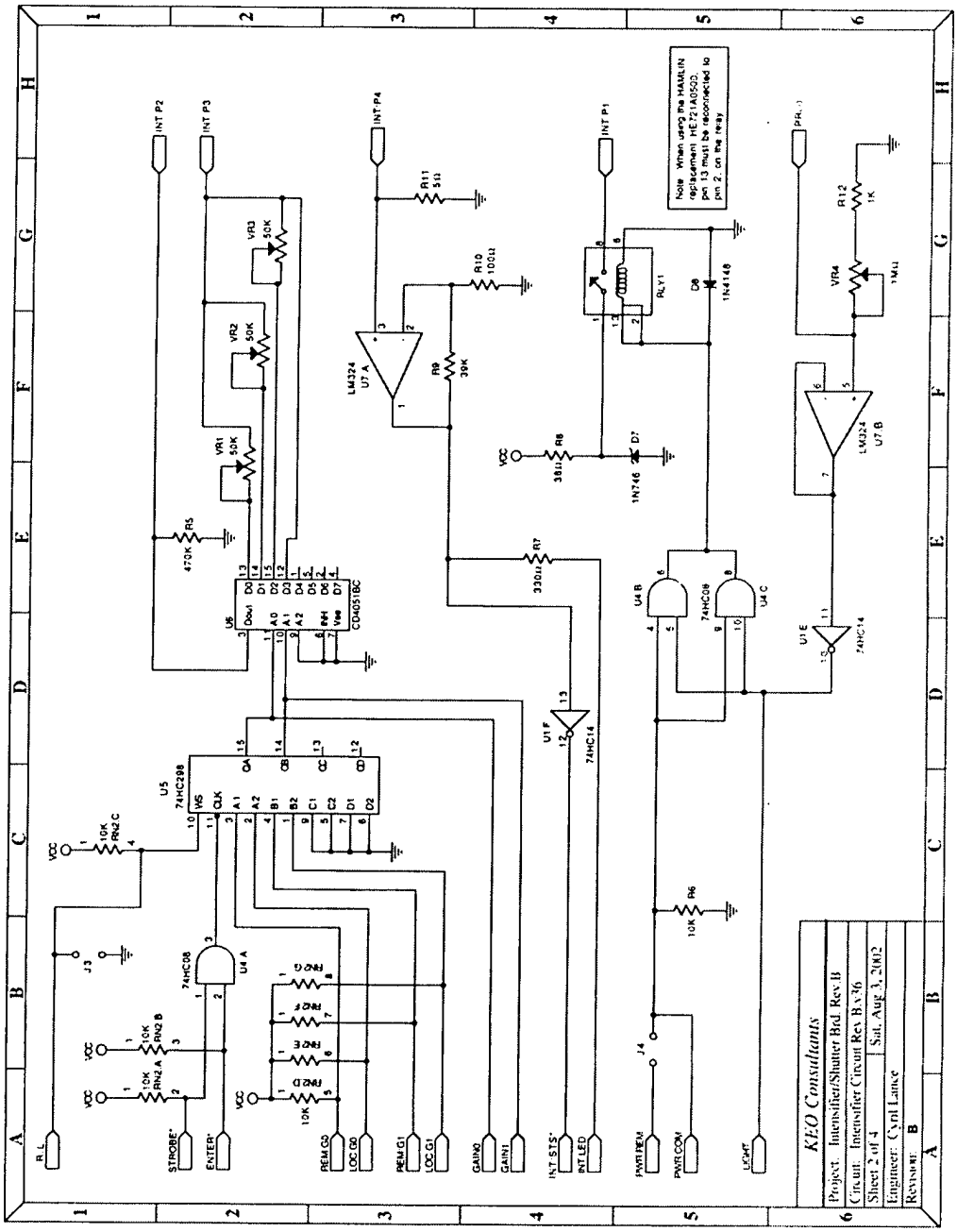


7.2 Intensifier/Shutter Control Board Schematics Rev B

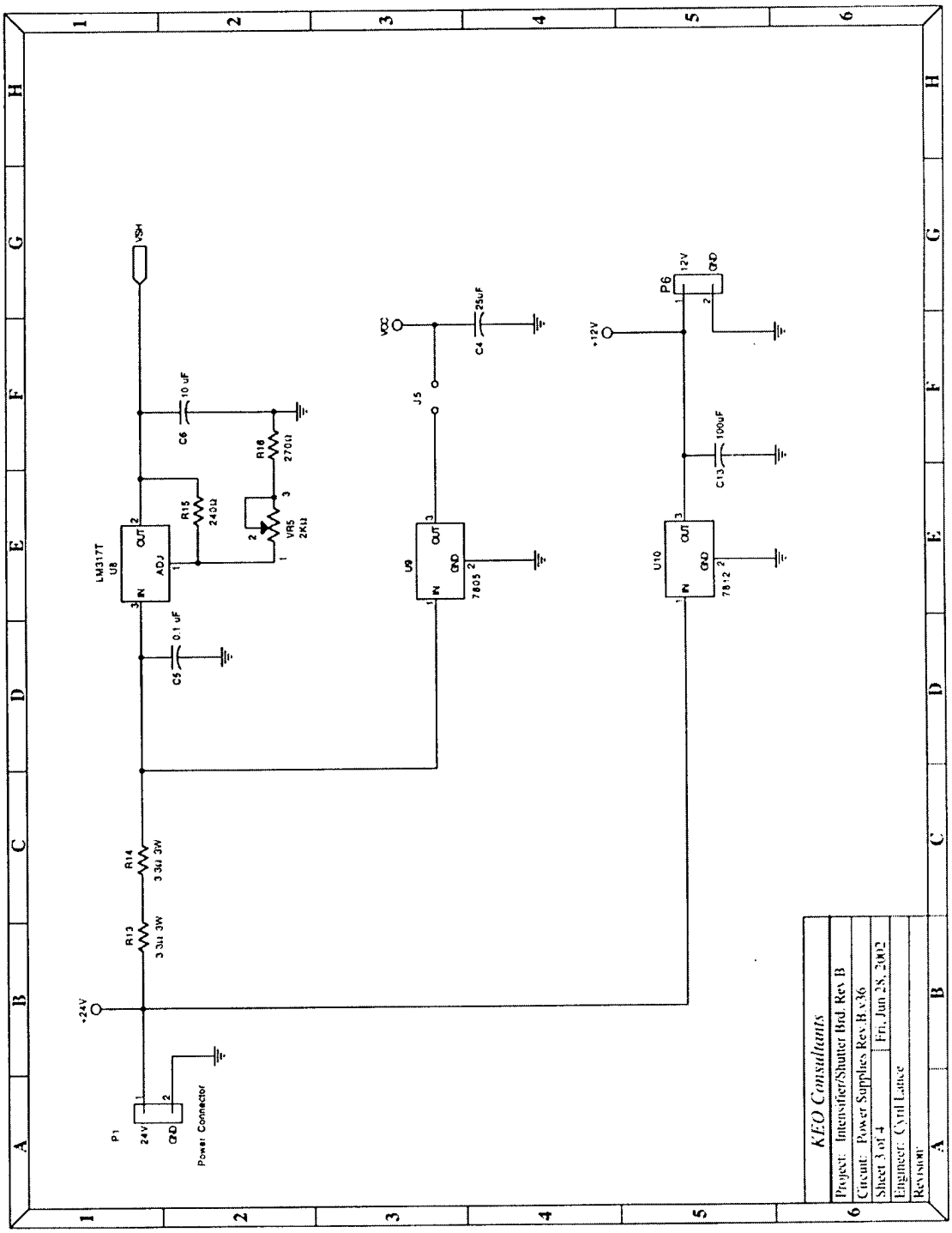
7.2.1 Shutter Control Circuit



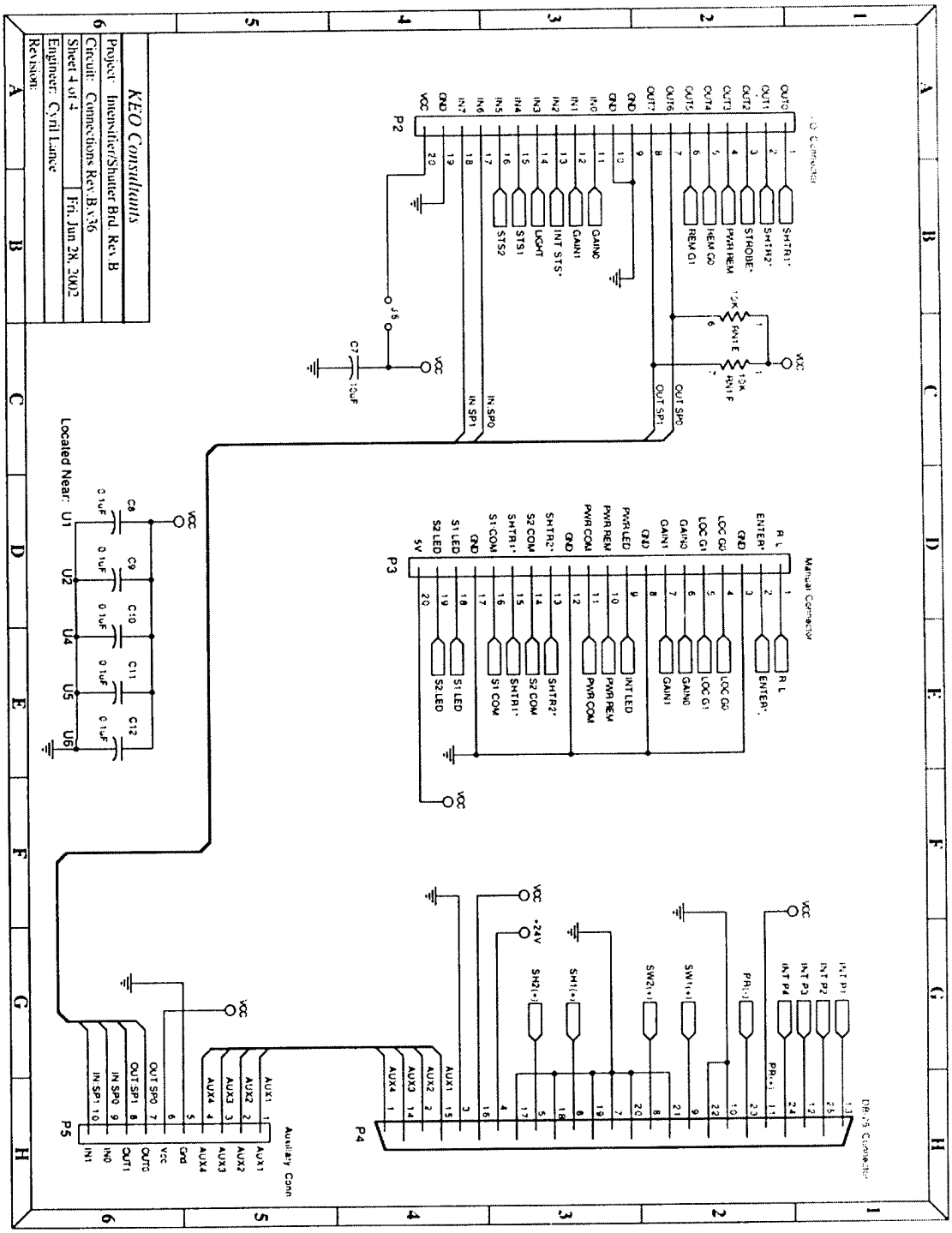
7.2.2 Intensifier/Light Detector Circuit



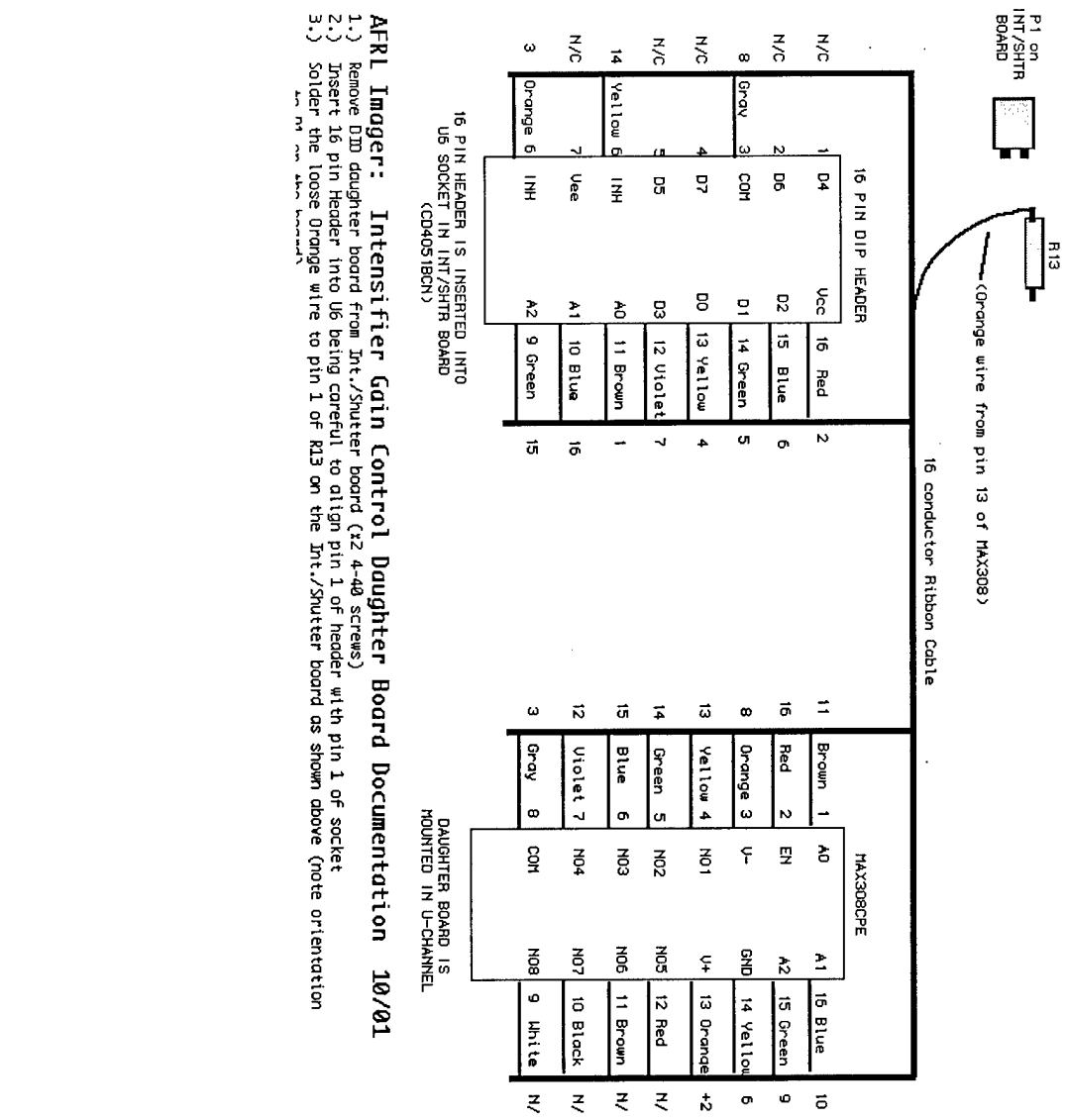
7.2.3 Power Supplies



7.2.4 Connector Schematic



7.3 Gen-III Intensifier Gain Control Daughter Board

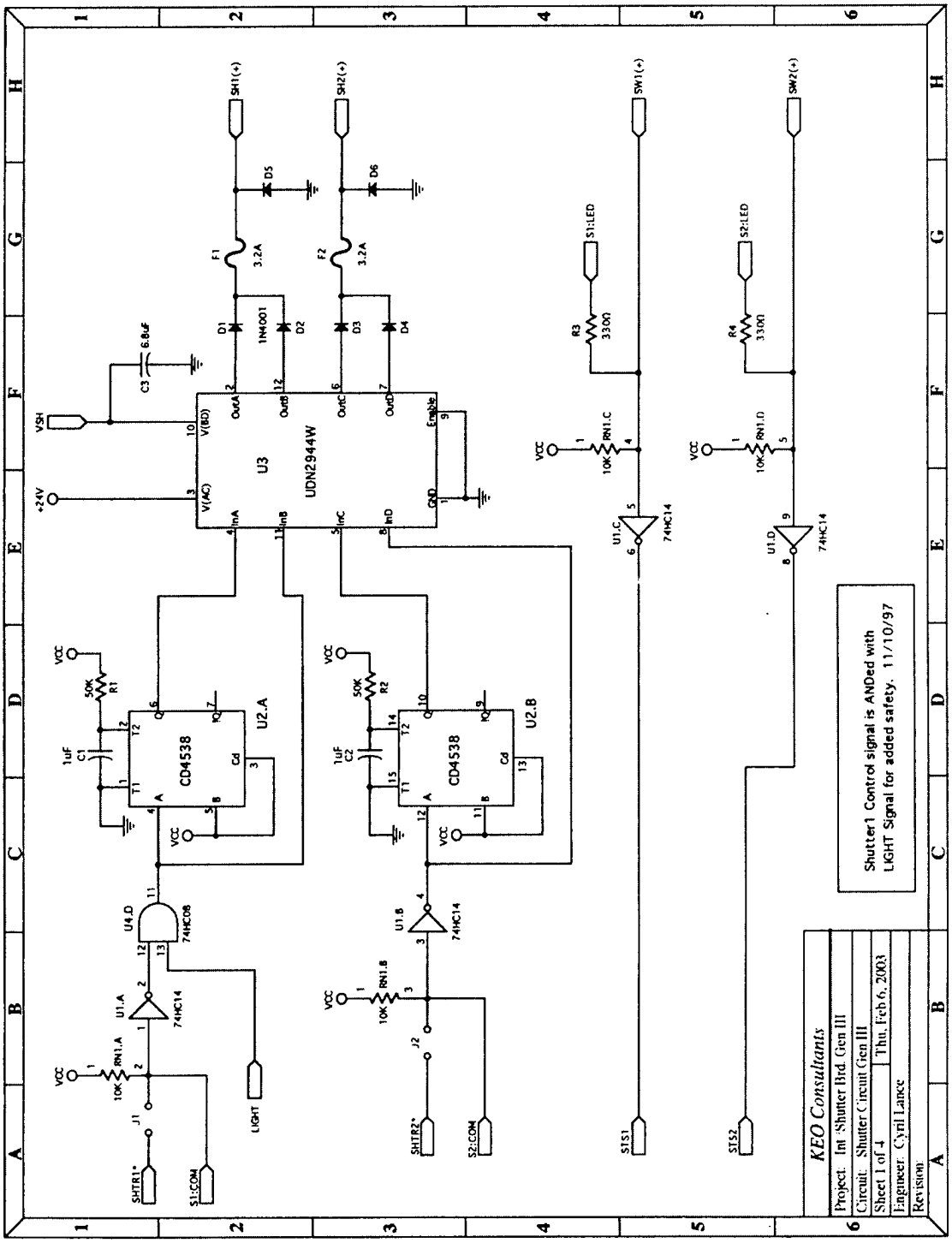


AFRL Imager: Intensifier Gain Control Daughter Board Documentation 10/01

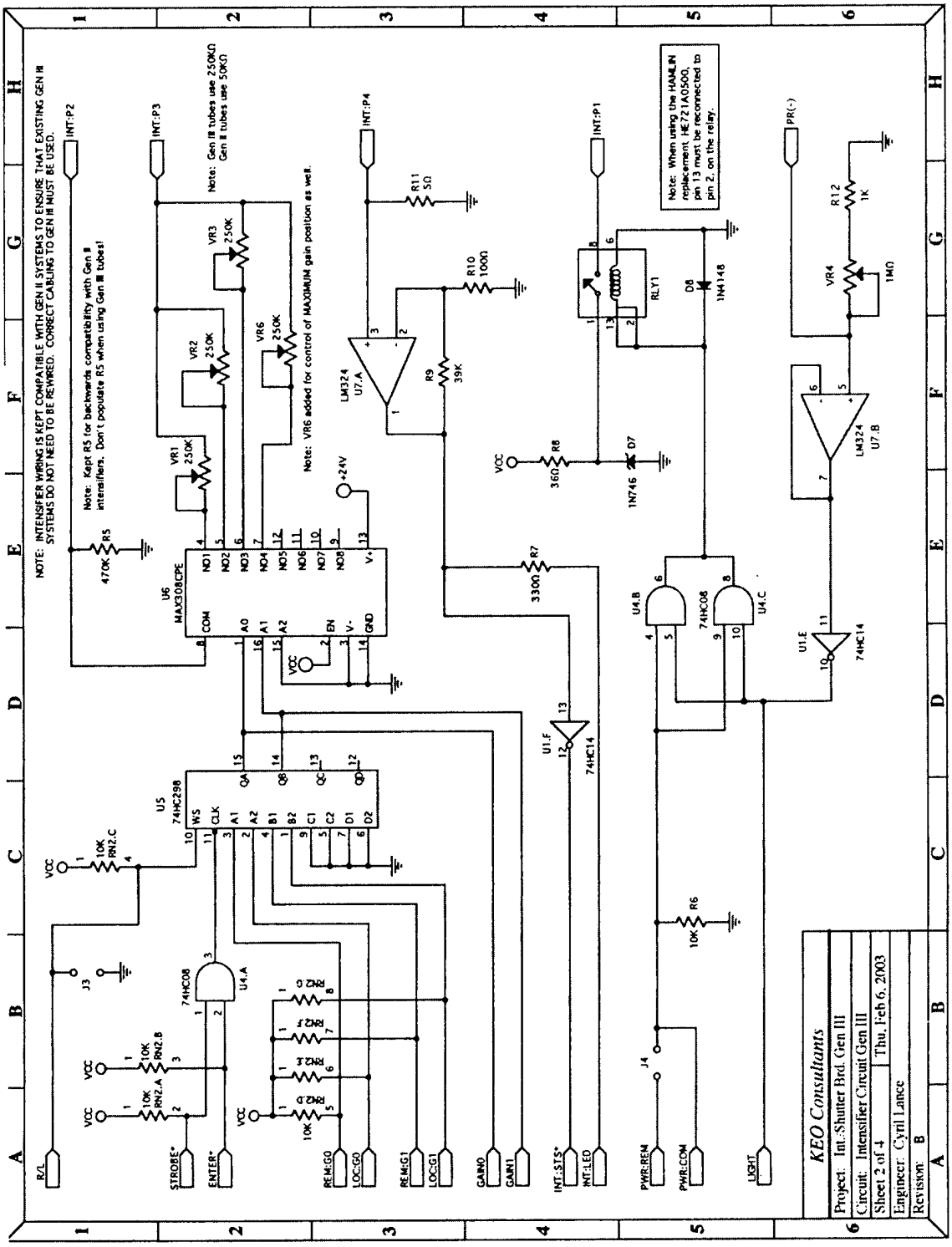
1.) Remove DIP daughter board from Int./Shutter board (x2 4-40 screws)
2.) Insert 16 pin Header into U6 being careful to align pin 1 of header with pin 1 of socket
3.) Solder the loose Orange wire to pin 1 of R13 on the Int./Shutter board as shown above (note orientation

7.4 Gen-III Intensifier/Shutter Control Board Schematics

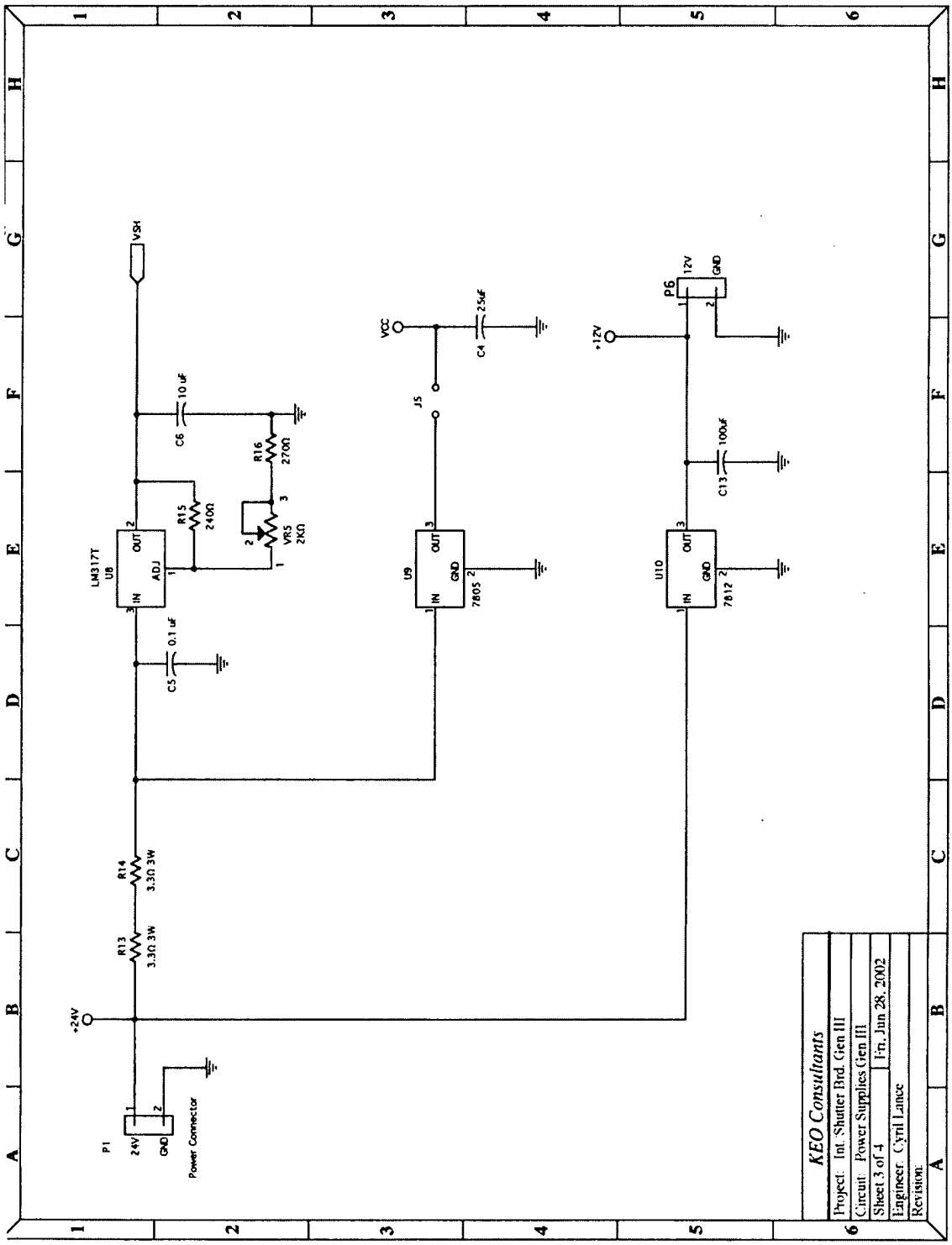
7.4.1 Shutter Control Circuit



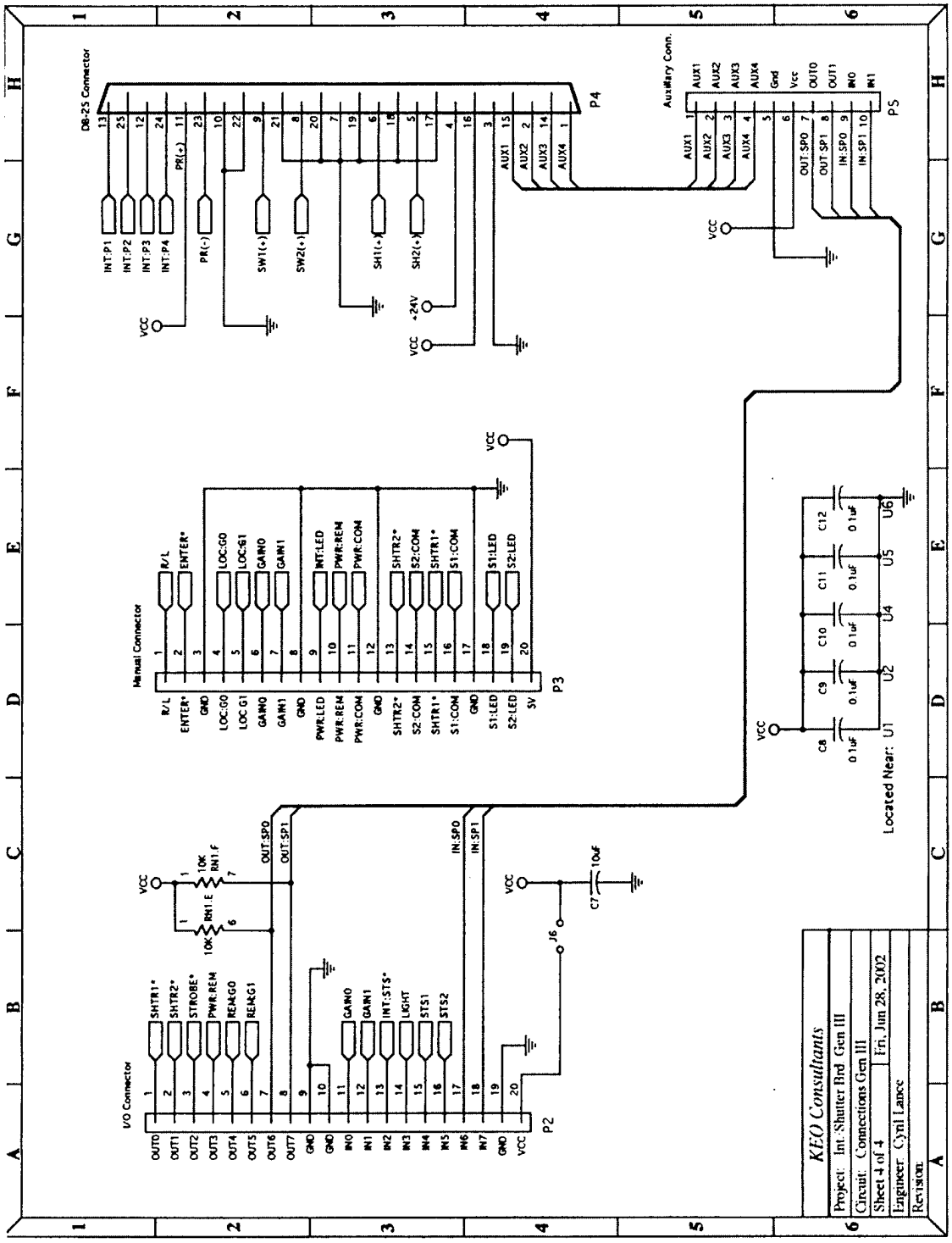
7.4.2 Intensifier/Light Detector Circuit



7.4.3 Power Supplies



7.4.4 Connector Schematic



7.5 SMARTMOTOR Interface

An embedded program was developed for the SMARTMOTOR that allows the host to easily control both the filterwheel and the Intensifier/Control Board through the DIO-116 interface described above. The following section describes this system and the subroutines stored in the SMARTMOTOR EEPROM. Users should refer to the Animatics documentation for a deeper understand of the SMARTMOTOR programming language and capabilities.

7.5.1 Description of ARFL01.SMS

A simple control routine was developed for your application and is stored in the SMARTMOTOR EEPROM. This software does not auto load but rather is initialized from the host by the RUN command. To stop the program at anytime, the END command should be issued from the HOST. The firmware listing can be found in Chapter 6: above.

Upon initializing the software (RUN), the motion parameters for the filterwheel are initialized:

```
AMPS = 300        ' Sets the maximum current to 2 AMP
MP                ' Sets the motor in absolute position mode
A=1000            ' Sets acceleration to 1000
V=1000000         ' Sets Velocity to 1000000
```

An output byte of 7 is STROBED into the Control Board, which sets both shutters to closed and the intensifier gain to 0 and the intensifier power to OFF.

Note that this guarantees that the control board state is in a SAFE mode of operation. Thus, the RUN command should be issued as soon as possible after power up of the unit. Because the default state of the shutter is closed, and the additional safety of the light detector enabled, there is little chance of hurting the image intensifier, but it is good to initialize the program to guarantee this state.

(The SMARTMOTOR can automatically run the software at power up by removing the **RUN?** command in the beginning of ARFL01.SMS. After some experimentation with this control software, the user may want to implement this feature.)

Accelerations/Velocity/PID tuning:

Accelerations and velocities for the filterwheel were selected by trial and error and a medium speed was chosen to change the filterwheels. The servo-motor control system is capable of much higher performance, but the user should experiment with the adjustment of the tuning parameters to find the optimum system parameters for their applications.

```
' version 4.10 and version 4.11 bug fix from ANIMATICS

KGON
KGOFF

KP=500
KD=1000
F
```

Refer to the Animatics manual for definitions of units, motion control commands, and the servo tuning parameters to learn how to change these values.

KEO has tested the filterwheel and control software for over 2000 repetitions with no failures. The current parameter state should suffice for most applications.

Subroutine Definitions:

Once the program has initialized the parameters for the system, a HOME command is issued. When the unit is successfully HOMED, the encoder position is calibrated and we can start issuing commands to the unit. An infinite loop is entered that just sits and waits for commands from the host. Any valid SMARTMOTOR commands may be issued by the host, but it is recommended that only the commands set forth in this manual be issued while the program is running.

To terminate ARFL01.SMS at any time, the command

END

must be issued. The program functions can easily be tested by hooking the SMARTMOTOR to any RS-232 port and using a terminal emulation software package.

ARFL01.SMS is based around the execution 6 subroutines that control the various functions of the camera system. SMARTMOTOR variables are set to set the command state of the subroutines and to hold the return results. The SMARTMOTOR variables for ARFL01.SMS are set up as follows:

a	Holds contents of last read input (DINB0)
b	Holds contents of output register (DOUTA0)

c	Holds light detector status bit
d	Holds shutter status bit
e	Holds Intensifier Gain
f	Holds Intensifier Power status bit
g	Holds next filter position
h	holds present filter position

ARFL01.SMS has the following subroutine definitions:

C0	Read_Light_Detector_State
C1	Shutter_Control
C2	Intensifier_Gain_Control
C3	Intensfier_Power_Control
C4	Filterwheel_Control
C5	HOME_Reset

Subroutines are terminated by emitting a LF character (#10) which is used by the host serial control to terminate the port read.

C0: Read Light Detector State

The command

```
GOSUB0
```

reads the input register and stores the result in variable a. Bit 3 of this register is masked out and stored in variable c and shifted to the LSB position. C0 returns with the string:

```
LIGHT:c                 ' where c is 0 or 1
```

C1: Shutter Control

The command

```
d=x
GOSUB1
```

where:

```
x = -1          ' just reads shutter state
x =  1                  ' opens the shutter
x =  0                  ' closes the shutter
```

controls the shutter state. If d == -1, then the shutter status bit is read and masked from register a and stored in variable d. If d == 16, it indicates that the shutter is open and C1 emits

SHTR:???

If d == 0, then C1 emits

SHTR:???

Because your camera only has one shutter controlled by the Control Board, provision for only one shutter was included in your software package. *Note that the return string contains a ??? indicator for the shutter status. This indicates that there is no valid status-detection system on this shutter.*

To open or close the shutter, the host would send the following commands:

```
d=1                 ' Set shutter cmd to OPEN
GOSUB1              ' Operate shutter subroutine
.
.                   ' get exposure, etc....
.
d=0                 ' Set shutter cmd to CLOSE
GOSUB1              ' Operate shutter subroutine
```

Both calls to C1 will return the status of the shutter just as if the d == -1 parameter had been sent to the subroutine. The user can choose to ignore or use this return state based on their programming needs.

C2 Intensifier Gain Control

The command:

```
e=x
GOSUB2
```

where:

```
e=-1                ' just reads current Intensifier Gain
e= 0 - 3            ' sets the intensifier gain to e
```

controls the gain of the image intensifier tube. If e == -1, then the input register is read and the intensifier gain bits (B0-B1) are masked out and returned to the host:

GAIN:e

If an invalid gain was set in e (values other than 0 through 3), C2 returns:

GAIN:-1

To set the gain, for example, to 2, the following commands should be issued by the host:

```
e=2
GOSUB2
```

If everything is working correctly, the subroutine will return the following string to the host:

```
GAIN:2
```

C3: Intensifier Power Control

The command

```
f=x
GOSUB3
```

where:

```
x = -1          ' just reads int. power state
x =  1          ' turns on intensifier power
x =  0          ' turns off the intensifier power
```

controls the intensifier power state. If f == -1, then the shutter status bit is read and masked from register a and stored in variable f. If f == 4, the intensifier power is OFF (remember the status signal is ACTIVE-LOW), and if f == 0, the intensifier power is on:

```
INTPWR:OFF      ' f == 4
INTPWR:ON       ' f == 0
```

To turn on/off the intensifier power, the host would send the following commands:

```
f=1             ' Set intensifier power to ON
GOSUB3          ' Operate int. power subroutine
.
.               ' do other stuff..
.
f=0             ' Set intensifier power to OFF
GOSUB3          ' Operate int. power subroutine
```

Both calls to C3 will return the status of the intensifier power. Remember that even after turning on the intensifier power from the host, the Control Board might still not enable the intensifier power if the Light Detector Bit is set LOW. This bit is checked by running C0.

C4 Filterwheel Control

The command

```
g=x
GOSUB4
```

where:

```
g = -1                    ' reads the current filterwheel position
g = 1 through 6  ' sets the filterwheel to the position: g
```

commands the filterwheel to move to the selected position stored in g, or to read the current filterwheel position, if g == -1. C4 returns the current filter position to the host:

```
FILT:g
```

where g is a valid filter position between 1 and 5 (for your filterwheel system), or -1 if a non-valid filterwheel position command was issued.

NOTE: the filterwheel position is calculated from the current encoder state. There are 2000 encoder counts per revolution, and 5 motor revolutions per filterwheel position. The encoder position is always set so that the encoder is equal to the filter position * 10000. The servo loop should guarantee that the motor position is always locked onto the correct encoder position. When reading the filter position, C4 reads the current encoder position and divides by 10000 to get the filter number:

```
h=@P/10000
```

If the encoder has drifted off, h might not return an accurate filter position. This signals the user that there might be a problem with the filterwheel (and that the rates/tuning parameters might need to be adjusted). The user should use the Animatics utility SMI and refer to the manual to adjust these values should it become necessary.

When a valid filter position is stored in g, C4 calculates the quickest move for the motor to execute to reach that path position. C4 will then wait for the trajectory to finish, reset the current filter position and report it to the host:

```
FILT:h
```

For example, a command to move to position number 2 would look like:

```
g=2                       ' Host emits: filter position # 2
GOSUB4                    ' Host emits: got to filterwheel routine
.
.
FILT:2                    ' SMARTMOTOR returns
```

C5: HOME Reset

This command is executed upon startup of the program ARFL01.SMS, but is also available to be used at anytime during operation of the motor. This could be useful if there is a suspected problem with the filterwheel position.

The command

```
GOSUB5
```

sets the velocity and acceleration of the motor to a slower rate, and slews until the HOME detector is detected. The motor then stops and slowly moves in the opposite direction until the HOME bit is set again. When the motor first stops, it will slew past the HOME bit due to inertia of the motor and the deceleration process. Thus, when we reverse the motor, the HOME bit is LOW again.

Once the HOME bit is detected again, we turn off the motor instantly and this determines our HOME calibration. C5 then waits 0.1 seconds for the motor to settle to be safe and sets the origin at this point to a predefined offset (current 5350). This offset has been calibrated so that when the motor is slewed to 10000, it will be exactly at the correct position for filter position #1.

Once the origin is set, new motion parameters are set for the filterwheel:

```
MP          ' Absolute position mode
V1000000    ' Velocity setting
A1000       ' Acceleration setting
```

Then the motor is slewed at these rates to position #1 (P2000). After the trajectory is finished, C5 waits for a second to make sure everything has settled. Finally, C5 outputs to the host:

```
HOME:1
```

The offset, and final motion parameters for the filterwheel can all be adjusted in C5 by using the utilities provided by Animatics. Motion can be speeded up or slowed down as deemed necessary by the user. KEO has made an effort to find moderate motion parameters that provide quick but safe operating conditions.

7.6 RS-232 Communication to SMARTMOTOR

The SMARTMOTOR has an RS-232 interface which is connected to a host computer via the DB-9 on the back panel of our imagers. The filterwheel, image intensifier and shutter systems are all controlled by a set of commands issued by the host via the RS-232 interface.

Each SMARTMOTOR is individually addressed. Upon power-up, the SMARTMOTOR used in your camera system is given the address of #1. The SMARTMOTOR continually polls the RS-232 receive line and ignores all characters until it is addressed.

Addressing an individual SMARTMOTOR is achieved by sending out a high-order ASCII character:

ASCII Value	Motor Address
128	0 (All)
129	1
130	2
131	3
.	.
.	.
.	.

In our case, the Host must issue an ASCII character of value #129 (Decimal) before establishing a link with the SMARTMOTOR in the camera. Once this has been established, all commands issued from the Host as described in the preceding sections will be interpreted by the SMARTMOTOR.

Communication Protocol

The SMARTMOTOR communication is set up for:

9600 Baud, 8 bits, 1 stop bit, No Parity, No Flow Control

Note that all communication to the SMARTMOTOR is case sensitive!

7.7 RS-232 Communication to ATHENA XT16

The ATHENA XT16 provided with this imager for controlling the filterwheel temperature has RS-232 remote capabilities. For a full description of how to use this option, please refer to the ATHENA XT16 manual.

KEO has set up and tested the XT16 RS-232 interface with the following protocol:

96.n.1	9600 Baud, 8 bits, 1 stop bit, No Parity, No Flow Control
ID.1	Controller ID # = 1

There are many XT16 commands, but there are only really three commands that you will need to know for normal HOST communication. These are summarized here:

Reading the current temperature:

Send:	**#01R00<cr>**	
XT16 returns:	**<lf>#01R00= 026.7C<cr><lf>**	(17 bytes)

Reading the current set point temperature:

Send:	**#01R01<cr>**	
XT16 returns:	**<lf>#01R01= 026.7C<cr><lf>**	(17 bytes)

Modifying the set point temperature:

Send:	**#01m01 025.0C<cr>**	
XT16 returns:	**<lf>#01M01 025.0C<cr><lf>**	(16 bytes)

You can use the KEO program **PRGTEST.EXE** to test out the XT16. If the ANIMATICS (FW) interface is plugged into COM1, and the Temperature controller (TEMP) interface is plugged into COM2, choose *Disconnect* from the *Connect* menu, then select COM2 from under the *Settings* dialog box (*Connect menu)*, and then *Connect* again from the *Connect* menu. Now you will be able to manually type in the XT16 commands and verify that the interface is working correctly. This assumes that all the communications protocols remain consistent with this appendix.

7.8 RTD Chart

TABLE XVIII Temperature Vs. Resistance Table

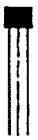
For European Curve, Alpha = .00385 — 1° Celsius Increments

°C	Ohm	Diff.	°C	Ohm	Diff.	°C	Ohm	Diff.	°C	Ohm	Diff.	°C	Ohm	Diff.	°C	Ohm	Diff.
-200	18.49		-140	43.87	0.42	-80	68.33	0.41	-20	92.16	0.39	± 0	100.00	0.39	- 60	123.24	0.38
199	18.93	0.44	139	44.28	0.41	79	68.73	0.40	19	92.55	0.39	+ 1	100.39	0.39	61	123.62	0.38
198	19.36	0.43	138	44.70	0.42	78	69.13	0.40	18	92.95	0.40	2	100.78	0.39	62	124.01	0.38
197	19.79	0.43	137	45.11	0.41	77	69.53	0.40	17	93.34	0.39	3	101.17	0.39	63	124.39	0.38
196	20.22	0.43	136	45.52	0.41	76	69.93	0.40	16	93.73	0.39	4	101.56	0.39	64	124.77	0.38
195	20.65	0.43	135	45.94	0.42	75	70.33	0.40	15	94.12	0.39	5	101.95	0.39	65	125.16	0.39
194	21.08	0.43	134	46.35	0.41	74	70.73	0.40	14	94.52	0.40	6	102.34	0.39	66	125.54	0.38
193	21.51	0.43	133	46.76	0.41	73	71.13	0.40	13	94.91	0.39	7	102.73	0.39	67	125.92	0.38
192	21.94	0.43	132	47.18	0.42	72	71.53	0.40	12	95.30	0.39	8	103.12	0.39	68	126.31	0.38
191	22.37	0.43	131	47.59	0.41	71	71.93	0.40	11	95.69	0.39	9	103.51	0.39	69	126.69	0.38
190	22.80	0.43	130	48.00	0.41	70	72.33	0.40	10	96.09	0.40	10	103.90	0.39	70	127.07	0.38
189	23.23	0.43	129	48.41	0.41	69	72.73	0.40	9	96.48	0.39	11	104.29	0.39	71	127.45	0.38
188	23.66	0.43	128	48.82	0.41	68	73.13	0.40	8	96.87	0.39	12	104.68	0.39	72	127.84	0.38
187	24.09	0.43	127	49.23	0.41	67	73.53	0.40	7	97.26	0.39	13	105.07	0.39	73	128.22	0.38
186	24.52	0.43	126	49.64	0.41	66	73.93	0.40	6	97.65	0.39	14	105.46	0.39	74	128.60	0.38
185	24.94	0.42	125	50.06	0.42	65	74.33	0.40	5	98.04	0.39	15	105.85	0.39	75	128.98	0.38
184	25.37	0.43	124	50.47	0.41	64	74.73	0.40	4	98.44	0.40	16	106.24	0.39	76	129.37	0.39
183	25.80	0.43	123	50.88	0.41	63	75.13	0.40	3	98.83	0.39	17	106.63	0.39	77	129.75	0.38
182	26.23	0.43	122	51.29	0.41	62	75.53	0.40	2	99.22	0.39	18	107.02	0.39	78	130.13	0.38
181	26.65	0.42	121	51.70	0.41	61	75.93	0.40	1	99.61	0.39	19	107.40	0.38	79	130.51	0.38
180	27.08	0.43	120	52.11	0.41	60	76.33	0.40				20	107.79	0.39	80	130.89	0.38
179	27.50	0.42	119	52.52	0.41	59	76.73	0.40				21	108.18	0.39	81	131.27	0.38
178	27.93	0.43	118	52.92	0.40	58	77.13	0.40				22	108.57	0.39	82	131.66	0.39
177	28.35	0.42	117	53.33	0.41	57	77.52	0.39				23	108.96	0.39	83	132.04	0.38
176	28.78	0.43	116	53.74	0.41	56	77.92	0.40				24	109.35	0.39	84	132.42	0.38
175	29.20	0.42	115	54.15	0.41	55	78.32	0.40				25	109.73	0.38	85	132.80	0.38
174	29.63	0.43	114	54.56	0.41	54	78.72	0.40				26	110.12	0.39	86	133.18	0.38
173	30.05	0.42	113	54.97	0.41	53	79.11	0.39				27	110.51	0.39	87	133.56	0.38
172	30.47	0.42	112	55.36	0.41	52	79.51	0.40				28	110.90	0.39	88	133.94	0.38
171	30.90	0.43	111	55.78	0.40	51	79.91	0.40				29	111.28	0.38	89	134.32	0.38
170	31.32	0.42	110	56.19	0.41	50	80.31	0.40				30	111.67	0.39	90	134.70	0.38
169	31.74	0.42	109	56.60	0.41	49	80.70	0.39				31	112.06	0.39	91	135.08	0.38
168	32.16	0.42	108	57.00	0.40	48	81.10	0.40				32	112.45	0.39	92	135.46	0.38
167	32.59	0.43	107	57.41	0.41	47	81.50	0.40				33	112.83	0.38	93	135.84	0.38
166	33.01	0.42	106	57.82	0.41	46	81.89	0.39				34	113.22	0.39	94	136.22	0.38
165	33.43	0.42	105	58.22	0.40	45	82.29	0.40				35	113.61	0.39	95	136.60	0.38
164	33.85	0.42	104	58.63	0.41	44	82.69	0.40				36	113.99	0.38	96	136.98	0.38
163	34.27	0.42	103	59.04	0.41	43	83.08	0.39				37	114.38	0.39	97	137.36	0.38
162	34.69	0.42	102	59.44	0.40	42	83.48	0.40				38	114.77	0.39	98	137.74	0.38
161	35.11	0.42	101	59.85	0.41	41	83.88	0.40				39	115.15	0.38	99	138.12	0.38
160	35.53	0.42	100	60.25	0.40	40	84.27	0.39				40	115.54	0.39	100	138.50	0.38
159	35.95	0.42	99	60.66	0.41	39	84.67	0.40				41	115.93	0.39	101	138.88	0.38
158	36.37	0.42	98	61.06	0.40	38	85.06	0.39				42	116.31	0.38	102	139.26	0.38
157	36.79	0.42	97	61.47	0.41	37	85.46	0.40				43	116.70	0.39	103	139.64	0.38
156	37.21	0.42	96	61.87	0.40	36	85.85	0.39				44	117.08	0.38	104	140.02	0.38
155	37.63	0.42	95	62.28	0.41	35	86.25	0.40				45	117.47	0.39	105	140.39	0.37
154	38.04	0.41	94	62.68	0.40	34	86.64	0.39				46	117.85	0.38	106	140.77	0.38
153	38.46	0.42	93	63.09	0.41	33	87.04	0.40				47	118.24	0.39	107	141.15	0.38
152	38.88	0.42	92	63.49	0.40	32	87.43	0.39				48	118.62	0.38	108	141.53	0.38
151	39.30	0.42	91	63.90	0.41	31	87.83	0.40				49	119.01	0.39	109	141.91	0.38
150	39.71	0.41	90	64.30	0.40	30	88.22	0.39				50	119.40	0.39	110	142.29	0.38
149	40.13	0.42	89	64.70	0.40	29	88.62	0.40				51	119.78	0.38	111	142.66	0.37
148	40.55	0.42	88	65.11	0.41	28	89.01	0.39				52	120.16	0.38	112	143.04	0.38
147	40.96	0.41	87	65.51	0.40	27	89.40	0.39				53	120.55	0.39	113	143.42	0.38
146	41.38	0.42	86	65.91	0.40	26	89.80	0.40				54	120.93	0.38	114	143.80	0.38
145	41.79	0.41	85	66.31	0.40	25	90.19	0.39				55	121.32	0.39	115	144.17	0.37
144	42.21	0.42	84	66.72	0.41	24	90.59	0.40				56	121.70	0.38	116	144.55	0.38
143	42.63	0.42	83	67.12	0.40	23	90.98	0.39				57	122.09	0.39	117	144.93	0.38
142	43.04	0.41	82	67.52	0.40	22	91.37	0.39				58	122.47	0.38	118	145.31	0.38
141	43.45	0.41	81	67.92	0.40	21	91.77	0.40				59	122.86	0.39	119	145.68	0.37

Note: At 100°C resistance is 138.50 ohms.

7.9 SS440 Digital Position Sensor Data Sheet

SS400 Digital position sensors



400SS Series position sensors have a thermally balanced ingrated circuit over full temperature range. The negative compensation slope is optimized to match the negative temperature coefficient of lower cost magnets. Bipolar, latching and unipolar magnetics are available.

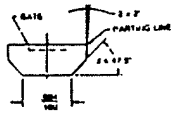
Band gap regulation provides extremely stable operation over 3.8 to 24 VDC supply voltage range. SS400 sensors are capable of continuous 20 mA sinking output, and may be cycled as high as 50 mA maximum.

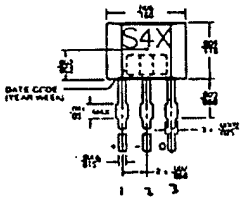
FEATURES

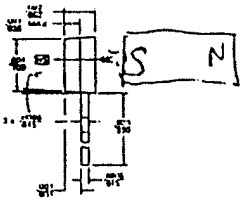
- 3.8-24 VDC supply voltage
- Digital current sinking output
- 3 pin in-line PCB terminals
- Quad-Hall design virtually eliminates mechanical stress effects
- Temperature compensated magnetics
- Operate/release points can be customized
- Bipolar, unipolar, latching magnetics
- High output current capability – 50 mA absolute maximum
- Operate/release points symmetrical around zero gauss (bipolar/latch)
- Operating temperature range of –55 to +150°C (–67 to +302°F)

MOUNTING DIMENSIONS

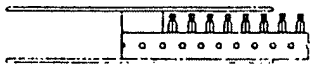
(For reference only)







TAPE AND REEL



ORDER GUIDE

Catalog Listing		SS411A	SS413A	SS441A	SS443A	SS449A	SS461A	SS466A
Magnetic Type		Bipolar	Bipolar	Unipolar	Unipolar	Unipolar	Latching	Latching
Supply Voltage (VDC)		3.8 to 24	3.8 to 24	3.8 to 24	3.8 to 24	3.8 to 24	3.8 to 24	3.8 to 24
Supply Current (max.)		10 mA	10 mA	10 mA	10 mA	10 mA	10 mA	10 mA
Output Type		Sink	Sink	Sink	Sink	Sink	Sink	Sink
Output Voltage (max.)		.40 V	.40 V	.40 V	.40 V	.40 V	.40 V	.40 V
Output Current, max.*		20 mA	20 mA	20 mA	20 mA	20 mA	20 mA	20 mA
Output Leakage Current, max.		10 µA	10 µA	10 µA	10 µA	10 µA	10 µA	10 µA
Output Switching Time								
V_{cc} = 12 V, R_L = 1.6 K, C = 20 pF	Rise (10-90%)	.05 µs typ. 1.5 µs max.	.05 µs typ. 1.5 µs max.	.05 µs typ. 1.5 µs max.	.05 µs typ. 1.5 µs max.	.05 µs typ. 1.5 µs max.	.05 µs typ. 1.5 µs max.	.05 µs typ. 1.5 µs max.
	Fall (90-10%)	.15 µs typ. 1.5 µs max.	.15 µs typ. 1.5 µs max.	.15 µs typ. 1.5 µs max.	.15 µs typ. 1.5 µs max.	.15 µs typ. 1.5 µs max.	.15 µs typ. 1.5 µs max.	.15 µs typ. 1.5 µs max.
Magnetic Characteristics (Gauss)								
–40°C	Max. Op.	70	140	135	215	435	110	200
	Min. Rel.	–70	–140	20	80	210	–110	–200
	Min. Dif.	15	20	15	25	30	50	200
0°C	Max. Op.	65	140	117	190	400	90	185
	Min. Rel.	–65	–140	20	80	230	–90	–185
	Min. Dif.	15	20	18	25	30	50	200
25°C	Max. Op.	60	140	115	180	390	85	180
	Min. Rel.	–60	–140	20	75	235	–85	–180
	Min. Dif.	15	20	20	25	30	50	200
85°C	Max. Op.	60	140	120	180	400	85	180
	Min. Rel.	–60	–140	15	70	215	–85	–180
	Min. Dif.	12	20	15	15	30	50	190
125°C	Max. Op.	65	140	123	190	410	100	180
	Min. Rel.	–65	–140	15	60	200	–100	–180
	Min. Dif.	12	20	8	10	30	50	160
150°C	Max. Op.	70	140	125	200	420	110	185
	Min. Rel.	–70	–140	10	55	185	–110	–185
	Min. Dif.	10	20	5	5	30	50	140

* Absolute maximum output current is 50 mA for all SS400 listings.

Note: For SS400 on tape and reel with formed leads on 0.100" centers, contact your nearest MICRO SWITCH Sales Office. One reel contains 3,000 sensors.

For application help: call your local sales office listed on back cover or call 1-800-537-6945.

7.10 Wiring and Cable Documentation

7.10.1 AC Power

Signal Name	Rear Panel Bendix	Wire Color	Athena Temp. Controller	24V & 12V Supplies
AC(Line)	A	Black	L1, COM#1	AC(+)
AC(Neutral)	C	White	L2	AC(-)
Chassis Gnd	B	Green		GND

7.10.2 24VDC Power

Signal Name	24V Supply	Wire Color	Int./Shutter Board	Smart Motor DB11
24V	Out(+)	Red	P1.1	A1
Gnd	Out(-)	Black	P1.2	A2

7.10.3 12VDC Power TEC and RTD / ATHENA XT32 Connections

Signal Name	12V Supply	Wire Color	TEC LEMO	TEC/ATHENA XT32
12V Fans	Out(+)	Orange	2	TEC.2
12V TEC	Out(+)	Red/Blk	1	XT32.5 (N.O.)
RTD(+)		Yellow	4	XT32.1 (+)
RTD(-)		Brown	5	XT32.2 (-)
Gnd	Out(-)	Black	3	P1.2

7.10.4 RS-232 Interface

Signal Name	Rear Panel DB9	Wire Color	Smart Motor DB11
Transmit	2	Red	3
Receive	3	White	4
Gnd	5	Black	5
			ATHENA XT16
Transmit	7	Orange	14
Receive	8	Purple	13
Gnd	9	Brown	6

ATHENA XT16: [Set by KEO] 9600 Baud, No Parity, 8 Bits, 1 Stop Bit, ID=01
SMARTMOTOR: 9600 Baud, No Parity, 8 Bits, 1 Stop Bit, ADDR=01

7.10.5 Filter Wheel Cable

Signal Name	Filterwheel Bendix 8 pin	Wire Color	SMARTMOTOR I/O Connector	Athena XT16 Controller
Home(5V)	C	Red	6	
Home(Gnd)	D	Brown	7	
Home(Out)	E	Orange	1	
RTD(+)	A	Green		10 RTD(+)
RTD(-)	G	Blue		9 RTD(-)
RTD(S)	H	Yellow		8 RTD(S)
AC(Line)	F	Black		1 N.O. #1
AC(Neutral)	B	White		12 L2

7.10.6 ANILINK Interface (I^2C)

Signal Name	Smart Motor AMP-MTE	Wire Color	DIO-116 AMP-MTE
5V	1	Yellow	1
Gnd	2	Green	2
Data	3	Red	3
Clock	4	Black	4

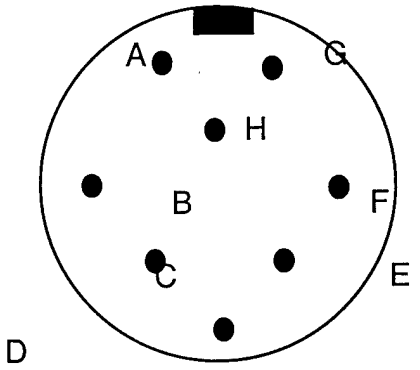
7.10.7 Intensifier/Shutter Controller Board Outputs

Signal Name	Int/Shtr Board DB-25	Wire Color	Light Detector LEMO FGG.0B	Shutter
PR(+)	11	White	1	
PR(-)	23	Gray	2	
SH1(+)	6	Red		1
SH1(-)	18	Brown		2
			Intensifier "Detector"	
P1	13	Red	1 (3VDC)	
P2	25	Yellow	3 (Gain)	
P3	12	Orange	4 (Gain)	
P4	24	Black	2 (GND)	

7.10.8 RS-232 Junction Cable

Signal Name	Rear Panel DB9 (Male)	Wire Color	FW Control DB9 (Female)	Temp. Control DB9 (Female)
Transmit	2	Red	2	
Receive	3	White	3	
Gnd	5	Black	5	
Transmit	7	Red		2
Receive	8	White		3
Gnd	9	Black		5

7.10.9 Filterwheel Wiring Documentation



Filterwheel Connector

Signal Name	Pin
RTD(+)	A
RTD(-)	G
RTD(S)	H
AC(Line)	F
AC(Neutral)	B
5VDC	C
Gnd	D
Home	E

Note #1: AC(Line) and AC(Neutral) are connected to both Heater Pads in parallel. The total parallel resistance should be on the order of 150Ω.

Note #2: There is an internal pull-up resistor of 4.7KΩ between pins C and E for the Hall-Effect Detector.