ESC-TR-2010-071

Project Report
HUSIR-8

Haystack Antenna Control System Design Document

J.V. Eshbaugh
M.T. Clarke
G.W. Maglathlin

7 December 2010

Lincoln Laboratory
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LEXINGTON, MASSACHUSETTS

Prepared for the Department of the Air Force under Contract FA8721-05-C-0002.

This report is based on studies performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology. This work was sponsored by the Department of the Air Force, 850 ELSG/NS, under Contract FA8721-05-C-0002.

Massachusetts Institute of Technology
Lincoln Laboratory

Haystack Antenna Control System Design Document

J.V. Eshbaugh
M.T. Clarke
Group 92

G.W. Maglathlin
Maglathlin Consulting Services, LLC

Project Report HUSIR-8

7 December 2010

Lexington Massachusetts

This page intentionally left blank.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF TABLES

This page intentionally left blank.

1. INTRODUCTION

MIT Lincoln Laboratory is carrying out a program to upgrade the existing Haystack radar by adding a W-band (96 GHz) radar capability. The existing X-band radar (10 GHz) will remain intact and the resulting radar will be capable of either simultaneous dual-band or single-band operation. This document describes the performance requirements of the control system for the HUSIR antenna. A new higher precision antenna is being installed, allowing the antenna to be used efficiently at W-Band. At W-Band, the system will have a very narrow beam that needs to be accurately pointed by a new control system while operating at higher velocities than the old hydraulic system.

This document describes the design of the control system as it is to be implemented, but this document alone isn't enough to completely define the control system design. A large number of drawings go along with this document (and will be referenced throughout this document), as well as several Interface Control Documents (ICDs). These ICDs will define the data that is transferred between the radar or astronomy computer and the antenna control unit (ACU), between the ACU and the maintenance computers, and between the ACU and the Programmable Logic Controller (PLC) system. The ICDs will be referenced, where appropriate.

The control system isn't being designed in a vacuum. Other teams are designing or will be designing other parts of the antenna system that the control system will interface with. As a result, the control system design effort will result in a number of requirements being placed on components that are not the direct responsibility of the control system team. An example of this is the stow pin actuator. The control system has requirements for the control and interlocking of the stow pin, but the actual mechanics and mounting of it are the responsibility of another team.

This page intentionally left blank.

2. DOCUMENTATION

The following documents will provide information required for a complete understanding of the HUSIR antenna control system.

HUSIR Antenna Control System Requirements Document

Pointing Computer to ACU Interface Control Document

Maintenance Computer to ACU Interface Control Document

ACU to Safety PLC system Interface Control Document

Antenna Safety PLC to Stow Pins Interface Control Document

HUSIR Control System Software Simulation

Detailed fabrication drawings for all components of the control system

A-488990: Control System Schematic

This page intentionally left blank.

3. CONTROL SYSTEM OVERVIEW

3.1 DRIVE SYSTEM SIZING

To properly size the drive system for the antenna, the overall inertia of the antenna must be known. The current inertias for the antenna (as of March 2008) are 9.48×10^6 kgm^2 in azimuth, constant with elevation change, and 7.76×10^6 kgm^2 in elevation.

The control system must provide enough torque to accelerate those inertias at 2 deg/sec^2. The total torque per axis required is $T_{az} = I_{az} A_{required} \frac{\pi}{180}$ Nm. For a system with no margin, the required torque in azimuth is 330914 Nm. With eight pinions in azimuth, each pinion must provide 41364 Nm of torque. A gear ratio of 1888:1 is used in azimuth, meaning that each of the eight motors in azimuth must provide 21.9 Nm of torque to accelerate the antenna at the required rate. A similar calculation for elevation using a 4395:1 gear ratio results in the 4 motors needing to each provide 15.4 Nm of torque.

The MHD115A-024 class motors selected for the drive system are capable of producing 24.9 Nm, with higher values for peak and stall when they are operated with a 100K temperature rise allowable using free convection. Therefore motors can accelerate the antenna at the required rates, with some margin. Should the inertias rise beyond the drive capability, the motors will require blowers to develop more torque. The pinion tooth loads will need to be re-evaluated in that case, as they are close to the allowable loads for maximum life. Tooth loads for the current inertias are 6231 lbs and 2831 lbs for azimuth and elevation respectively.

3.2 BASIC ARCHITECTURE

The basic architecture of the control system was set well before the start of this design document. The architecture was updated when MIT LL took control system work over from the vendor to reflect a PLC system and lessons learned from the radiometer (or rooftop) antenna, and the XTR system. The architecture as is stands is shown in Figure 1.

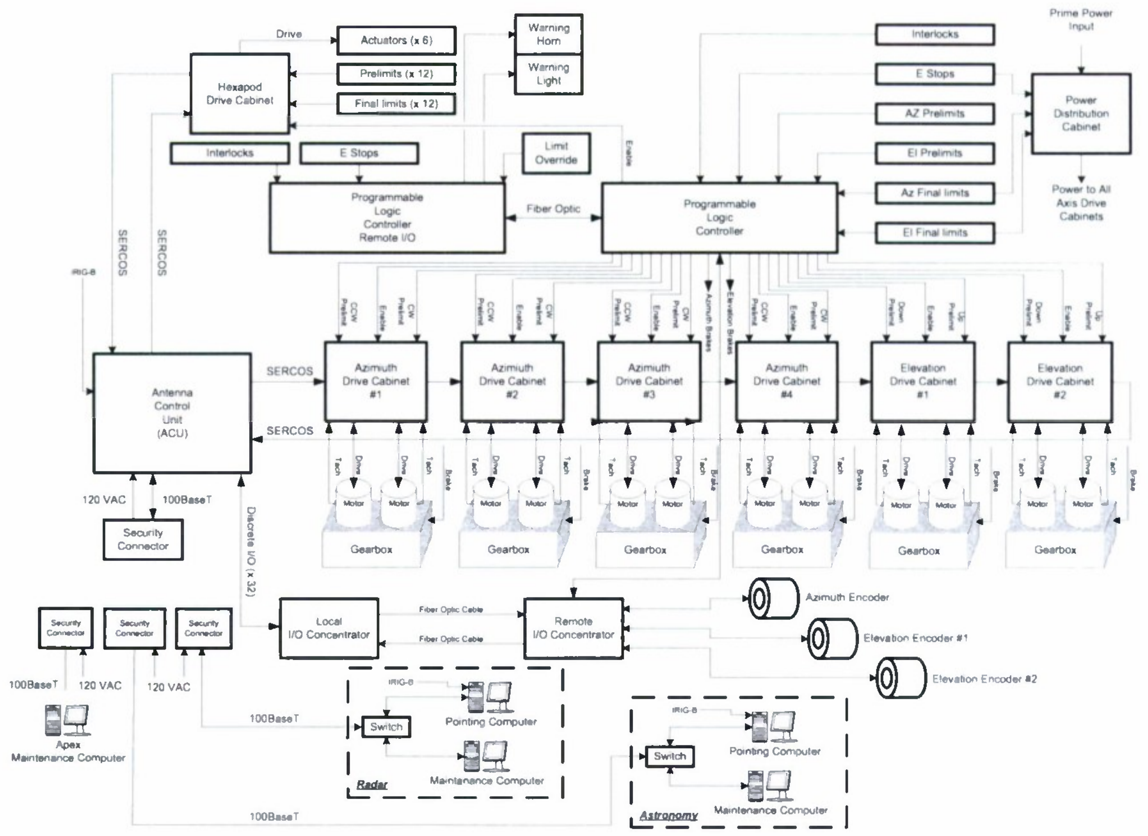


Figure 1. HUSIR control system architecture.

The details of the design and theory of operation will be developed in the following sections.

3.3 CONTROL SYSTEM POINTING ERROR BUDGET

The overall pointing error budget allocation for the control system is 1 millidegree, blind pointing after calibration and 0.5 millidegrees for tracking. That budget allocation is further broken up into a number of terms that are reproduced below:

Control System		
Sensors		
	Encoder accuracy	0.05
	Encoder coupling	0.05
Servo commands		
	time synchronization	0.05

- The drive backlash term is reduced to zero by incorporating torque biasing on pairs of drive motors.
- The command lag is due to the time between the time hack and the actual command output inside the servo loop. This is currently on the order of 300 microseconds on the rooftop and XTR systems. At 5 degrees per second in azimuth, this error is 1.5 millidegrees peak.
- Time synchronization refers to the accuracy of the IRIG timing used by the ACU relative to the radar. IRIG is typically capable of synchronizing multiple systems to 1–10 microseconds accuracy. The budget uses the worst case number.

For the following sections, the reader is referred to the *HUSIR Control System Interconnection Diagram*. This diagram contains chassis and cable numbers that are referenced heavily in the remainder of this document. Locations of components referred to in this document are shown in Figure 2.

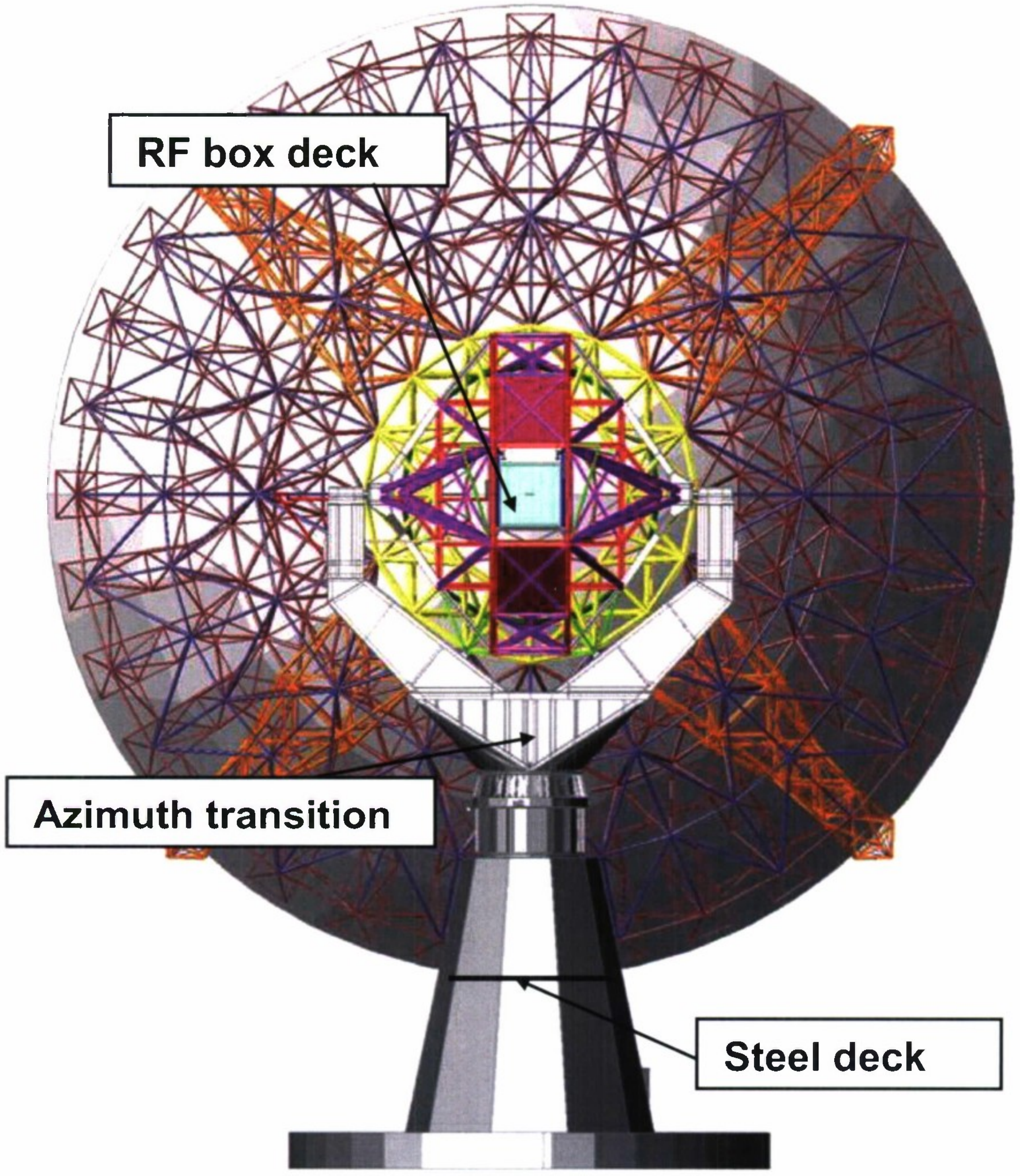


Figure 2. Haystack antenna control system locations.

4. ANTENNA CONTROL UNIT

The Antenna Control Unit (ACU) for the Haystack Ultra-wideband Satellite Imaging Radar (HUSIR) provides position and rate servo control for the antenna pedestal in response to commands from the Pointing Computer or the Maintenance Computer (MC). In addition, the ACU performs the logic decisions necessary to put the antenna pedestal into the requested modes of operation and to supplement the safety features provided by the Programmable Logic Controller (PLC) safety and interlock system. Figure 3 shows a top level block diagram which shows the ACU's relationship to other HUSIR equipment.

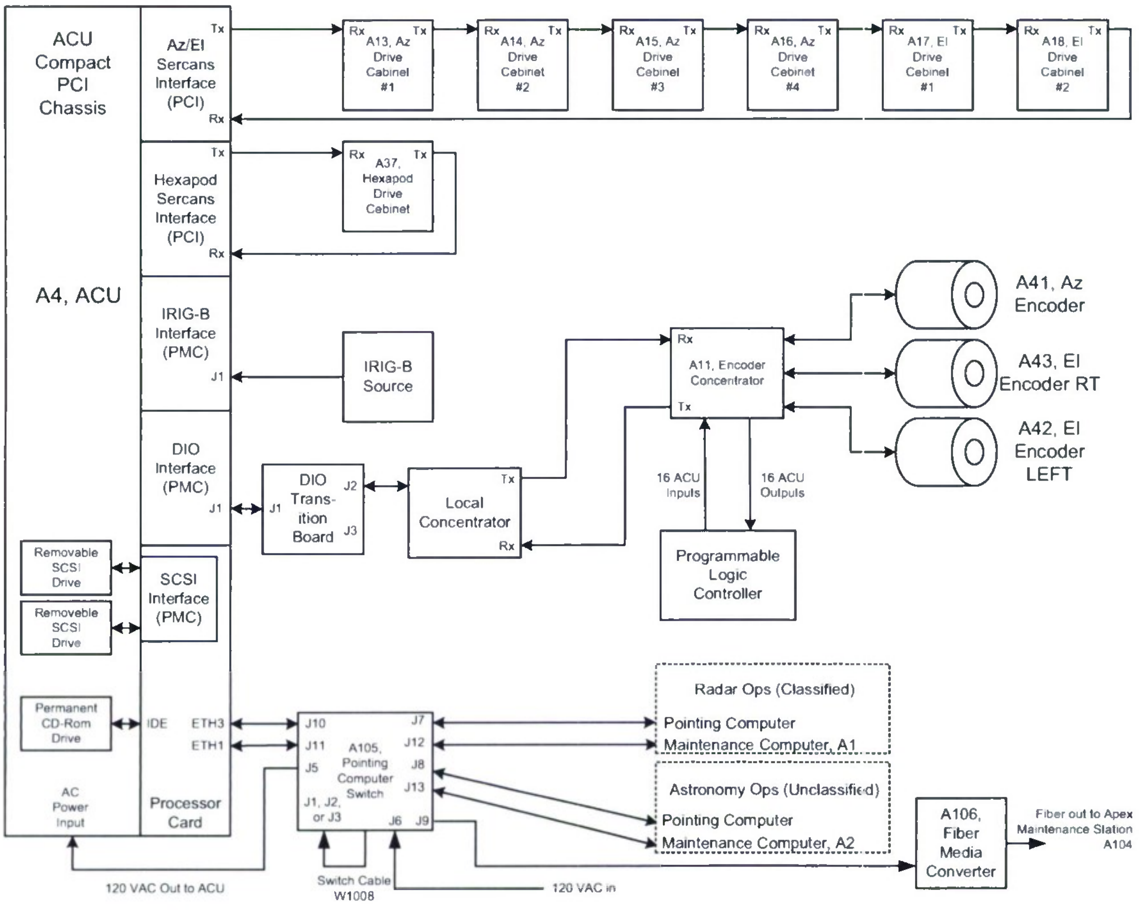


Figure 3. HUSIR ACU top level block diagram.

4.1 ANTENNA CONTROL UNIT HARDWARE

The completed ACU is shown in Figure 4.

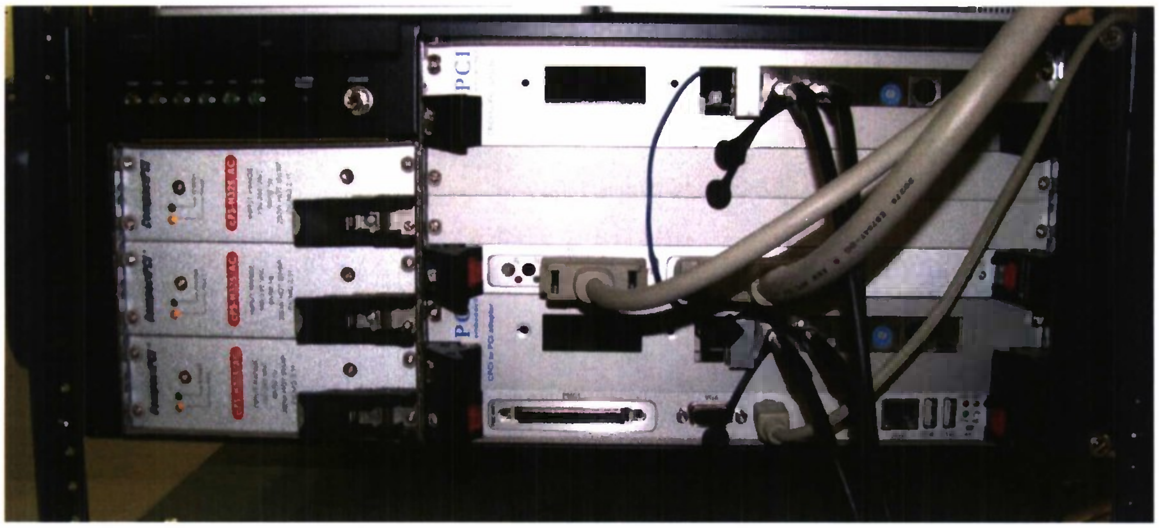


Figure 4. Photograph of completed ACU.

4.1.1 ACU Chassis

The ACU chassis is a standard 19" rack mount commercial-off-the-shelf (COTS) Compact PCI chassis manufactured by Adlink. The cPCIS-6418U/AC has the following features:

- Three redundant power supplies
- Permanent CD-ROM/CD-RW drive
- Card slot space the following complement of cards

o Processor card	1 slot
o PMC Carrier with Time code and DIO	1 slot
o Az/El Sercans Interface	2 slots
o Hexapod Sercans Interface	2 slots

cPCIS-6418U/AC Datasheet

cPCIS-6418U/AC Manual

4.1.2 ACU Processor Card

The ACU Processor Card is an Adlink cPCI-6840V Intel based Compact PCI computer on a card which provides the following capabilities:

- Compatibility with the above specified chassis
- Compatible with Linux
- 1.8 GHZ Pentium-M Processor
- APIC Interrupt processing
- 2 GB System Random Access Memory
- Ability to boot over Ethernet LAN
- IDE Interface (unused)
- SCSI Interface (unused)
- VGA Interface
- 3 100BaseT Ethernet interfaces
- Keyboard and Mouse Interface

cPCI-6840V Datasheet

cPCI-6840V Manual

4.1.3 PMC Carrier Card

The ACU contains an Adlink cPCI-8602 PMC carrier card which will support the installation of two PMC cards. The purpose of this card is to provide the mechanical and electrical interface between the Time Code and DIO cards and the Compact PCI Chassis.

cPCI-8602 Datasheet

cPCI-8602 Manual

4.1.4 ACU Timecode Interface Card

The ACU Timecode Interface Card is utilized to maintain time synchronization with the pointing computer using IRIG-B. The PCI-SyncClock32-UNIV from Brandywine Communications is the PMC card selected.

PCI-SyncClock32-UNIV Datasheet

4.1.5 ACU Digital Input/Output (DIO) Interface Card

This interface must have a minimum of 16 TTL inputs and 16 TTL outputs (16 outputs, 48 inputs desired) arranged in a parallel manner. The response time of these inputs and outputs must be less than 50 nanoseconds. The card selected is the Acromag PMC-464 parallel DIO card.

Acromag PMC-464 Datasheet

4.1.6 ACU Azimuth and Elevation Drive Interface Card

The four azimuth drive cabinets and the two elevation drive cabinets are connected to the ACU via a SERCOS fiber optic ring. In order to guarantee maximum compatibility with the azimuth and elevation drive amplifiers which are manufactured by BOSCH/REXROTH, the ACU utilizes the SERCANS interface produced by BOSCH/REXROTH. This SERCANS interface reduces the processing load of the ACU by handling all the low level SERCOS interface signaling and allowing for high level data transfer via shared memory. The SERCANS card is PCI based and requires a PCI to cPCI adapter board to function. The adapter board is manufactured by PCI-SYSTEMS.

Bosch Rexroth SERCANS PCI Card Manual

PCI-SYSTEMS Adapter Board Datasheet

4.1.7 ACU Hexapod Interface Card

The hexapod drive cabinet is connected to the ACU via a SERCOS fiber optic ring. In order to guarantee maximum software compatibility with the azimuth and elevation drive interface card is the same card as that specified in section 4.1.6.

4.2 ANTENNA CONTROL UNIT INTERFACES

4.2.1 Pointing Computer Switch

To prevent a breach of security, the classified and unclassified networks must never be connected. The method proposed for this system is to have a manual switch in the form of a cable with a connector on it that can be plugged into one of three ports as shown in Figure 2. Each port has the power feed for the ACU and the Ethernet connection to the appropriate network. The ACU itself has no nonvolatile memory on board, and is booted over the network from a maintenance computer (two unclassified and one classified). When it is desired to switch from one security level to another, the connector (A in Figure 2) is removed from one port (labeled 1, 2, or 3 in Figure 2) and connected to another. When the connector is removed, the AC power to the ACU is cut, and in doing so, all potentially classified information lost. When the connector is plugged into the other port, AC power is reapplied after an adjustable time delay (configurable from 1 second to several minutes) and the system booted from whichever network the ACU is connected to. This ensures that the system has been powered off before switching networks.

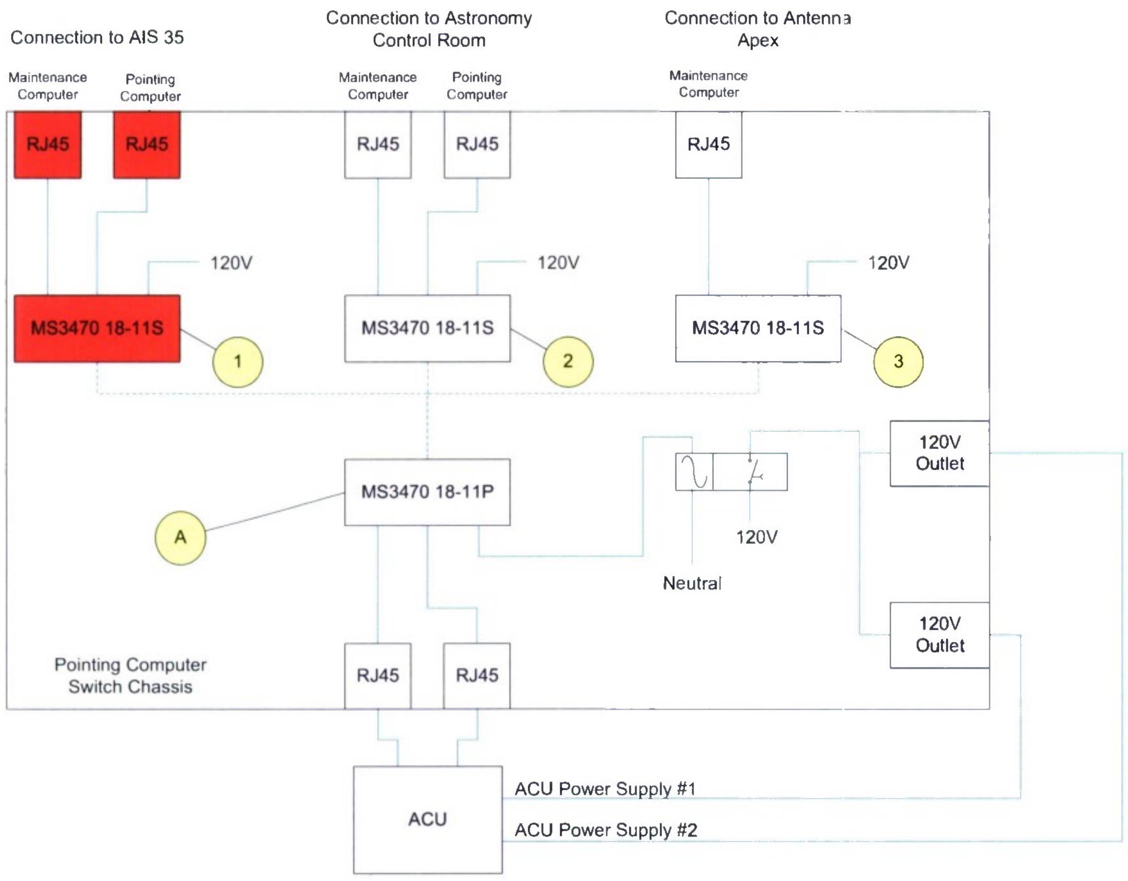


Figure 5. Pointing computer switch schematic.

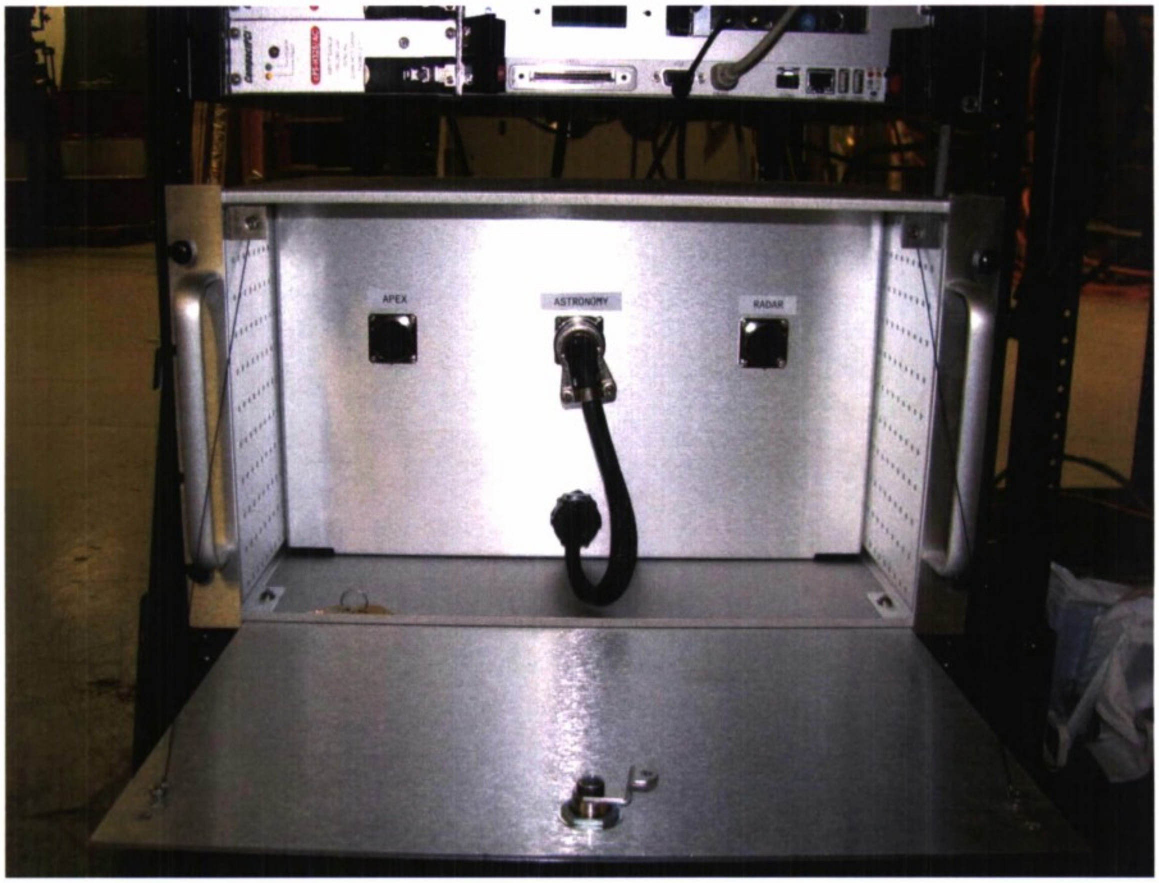


Figure 6. Front view of pointing computer switch with door open.

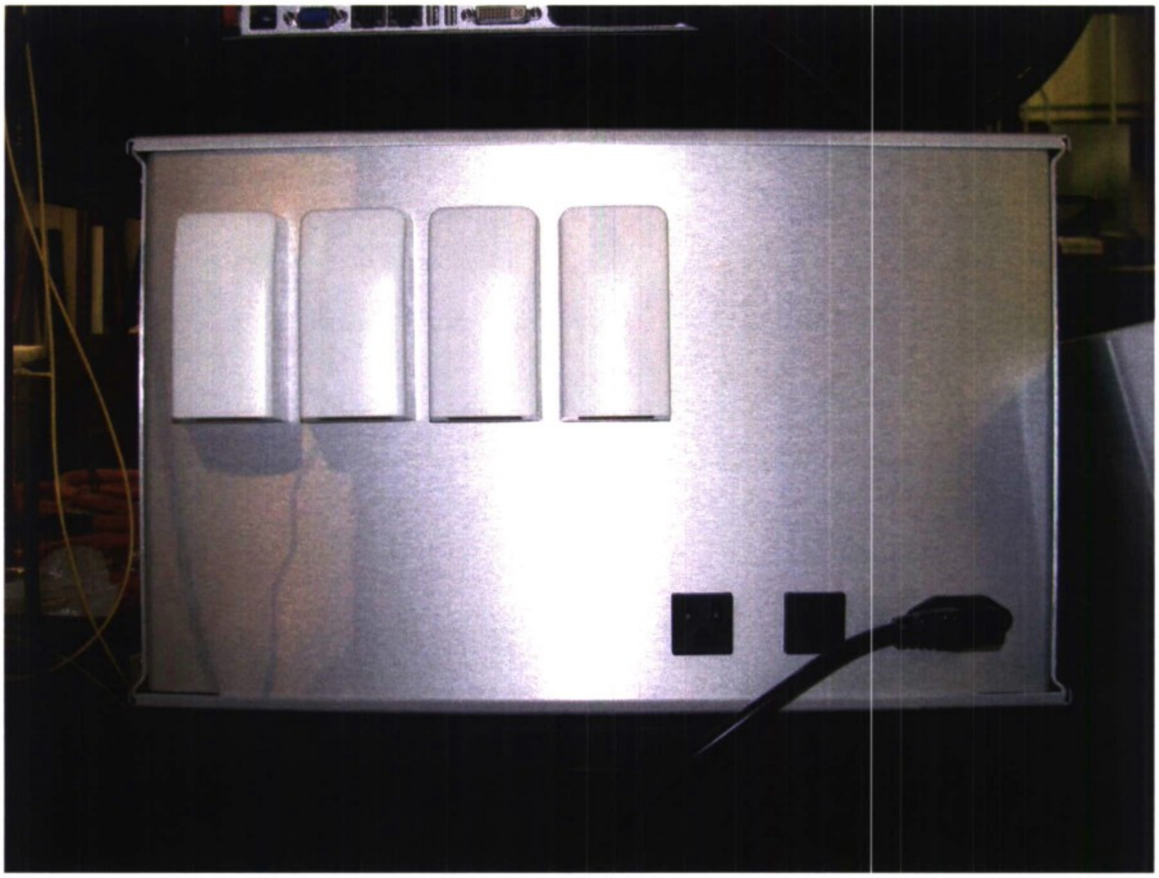


Figure 7. Rear view of the pointing computer switch.

4.2.2 Pointing Computer Interface

The pointing Computer interface shall be supported by Ethernet Interface ETH3 on the ACU processor card. This interface is 100BaseT Ethernet over copper twisted pair wiring in accordance with IEEE 802.3-2002. The ETH3 interface on the ACU is an RG-45 jack (female). This interface shall be connected to the ACU connection Panel via a standard straight through Ethernet cable. The pin outs of this connector are shown in Table 4-1.

Table 4-1

Pointing Computer Interface Pin Outs

Pin Number	Signal Name	Purpose
1	TX+	High side of Ethernet Transmitter
2	TX-	Low side of Ethernet Transmitter
3	RX+	High side of Ethernet Receiver
4	NC	No Connection
5	NC	No Connection
6	RX-	Low side of Ethernet Receiver
7	NC	No Connection
8	NC	No Connection

4.2.3 Maintenance Computer Interface

The Maintenance Computer interface shall be supported by Ethernet Interface ETH1 on the ACU pr ocessor c ard. This i nterface i s 100 BaseT E thernet over c opper twisted p air w iring in accordance with IEEE 802.3-2002. The ETH3 interface on the ACU is an RG-45 jack (female). This in terface s hall be connected to the ACU connection Panel via a standard straight through Ethernet cable. The pin outs of this connector are shown in Table 4-2.

Table 4-2
Maintenance Computer Pin Outs

Pin Number	Signal Name	Purpose
1	TX+	High side of Ethernet Transmitter
2	TX-	Low side of Ethernet Transmitter
3	RX+	High side of Ethernet Receiver
4	NC	No Connection
5	NC	No Connection
6	RX-	Low side of Ethernet Receiver
7	NC	No Connection
8	NC	No Connection

4.2.4 IRIG-B Interface

The IRIG-B interface supported by the PCI-SyncClock32-UNIV described in Paragraph 4.1.4 uses a female SMB connector as an input. The center conductor of this connector carries the IRIG-B signal. The outer shell of this connector provides a ground reference. The cable interfacing to this card must have a male SMB connector.

4.2.5 Digital Input/Output Interface

The PMC-464 card described in Paragraph 4.1.5 above will provide 16 bits of output and 48 bits of input. All inputs and outputs are TTL levels. The 16 bits of output and 16 of the input bits are used to communicate with the Local Concentrator board. All of this I/O is provided on a single female SCSI-3 68 pin connector (AMP 787082-7 or equivalent). The pin outs of this connector and the function of each pin is shown in Table 4-3.

Table 4-3

Digital I/O Interface Pin Outs

Pin Number	Signal Name	Direction	Purpose
1	Digital CH0	Input	2^0 Input bit from Local Concentrator
2	Digital CH1	Input	2^1 Input bit from Local Concentrator
3	Digital CH2	Input	2^2 Input bit from Local Concentrator
4	Digital CH3	Input	2^3 Input bit from Local Concentrator
5	Digital CH4	Input	2^4 Input bit from Local Concentrator
6	Digital CH5	Input	2^5 Input bit from Local Concentrator
7	Digital CH6	Input	2^6 Input bit from Local Concentrator
8	Digital CH7	Input	2^7 Input bit from Local Concentrator
9	Digital CH8	Input	2^8 Input bit from Local Concentrator
10	Digital CH9	Input	2^9 Input bit from Local Concentrator
11	Digital CH10	Input	2^{10} Input bit from Local Concentrator
12	Digital CH11	Input	2^{11} Input bit from Local Concentrator
13	Digital CH12	Input	2^{12} Input bit from Local Concentrator

Pin Number	Signal Name	Direction	Purpose
14	Digital CH13	Input	2^{13} Input bit from Local Concentrator
15	Digital CH14	Input	2^{14} Input bit from Local Concentrator
16	Digital CH15	Input	2^{15} Input bit from Local Concentrator
17	COMMON	N/A	Common Tie Point
18	COMMON	N/A	Common Tie Point
19	Digital CH16	Output	2^{0} Output bit to Local Concentrator
20	Digital CH17	Output	2^{1} Output bit to Local Concentrator
21	Digital CH18	Output	2^{2} Output bit to Local Concentrator
22	Digital CH19	Output	2^{3} Output bit to Local Concentrator
23	Digital CH20	Output	2^{4} Output bit to Local Concentrator
24	Digital CH21	Output	2^{5} Output bit to Local Concentrator
25	Digital CH22	Output	2^{6} Output bit to Local Concentrator
26	Digital CH23	Output	2^{7} Output bit to Local Concentrator
27	Digital CH48	Input	Status Input #16
28	Digital CH49	Input	Status Input #17
29	Digital CH50	Input	Status Input #18
30	Digital CH51	Input	Status Input #19
31	Digital CH52	Input	Status Input #20
32	Digital CH53	Input	Status Input #21
33	Digital CH54	Input	Status Input #22
34	Digital CH55	Input	Status Input #23
35	COMMON	N/A	Common Tie Point
36	COMMON	N/A	Common Tie Point
37	Digital CH24	Output	2^{8} Output bit to Local Concentrator
38	Digital CH25	Output	2^{9} Output bit to Local Concentrator
39	Digital CH26	Output	2^{10} Output bit to Local Concentrator

Pin Number	Signal Name	Direction	Purpose
40	Digital CH27	Output	2^{11} Output bit to Local Concentrator
41	Digital CH28	Output	2^{12} Output bit to Local Concentrator
42	Digital CH29	Output	2^{13} Output bit to Local Concentrator
43	Digital CH30	Output	2^{14} Output bit to Local Concentrator
44	Digital CH31	Output	2^{15} Output bit to Local Concentrator

Sixteen status inputs and sixteen control outputs will be provided by the DIO capability of the Remote Concentrator Module. These input and output points will be connected to the Programmable Logic Controller. The functions of these inputs and outputs are defined in the *ACU to Safety PLC System Interface Control Document.*

4.2.6 Azimuth/Elevation Drive Interface

The Azimuth/Elevation Drive SERCANS interface interfaces to the drives via a fiber optic ring as shown in Figure 3. The signals are carried over 200 mm core hard-clad-silica (HCS) fiber cable manufactured by OFS Specialty Photonics. HCS fiber was chosen because it is more flexible and durable than plastic fiber, with lower loss. The Az/El SERCANS interface operates as the synchronization slave with a 1000 Hz cycle time. The interface is synchronized to the Hexapod SERCANS interface.

4.2.7 Hexapod Interface

The Hexapod Drive SERCANS interface interfaces to the drives via a fiber optic ring as shown in Figure 3. HCS fiber is used for this ring as well. The Hexapod SERCANS interface is the synchronization master and operates at a 10 Hz cycle time.

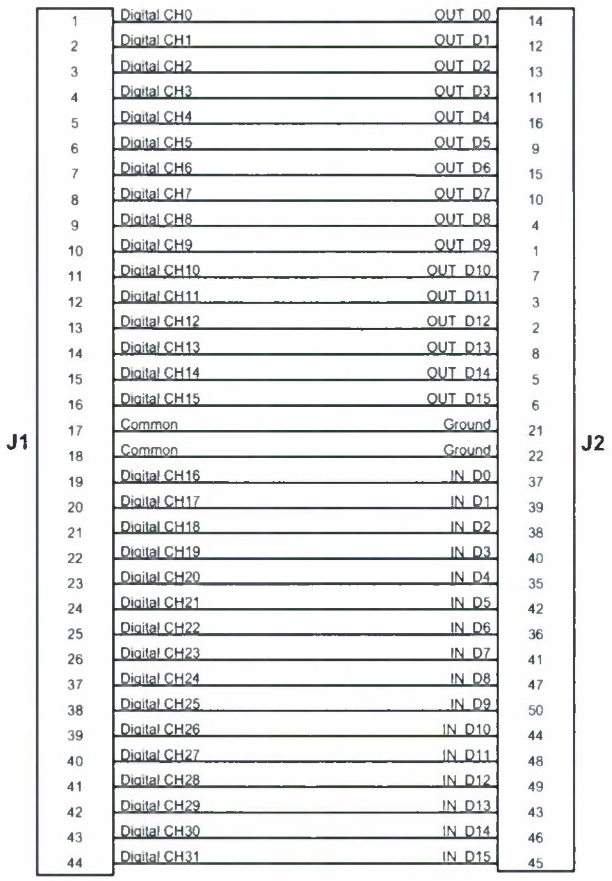


Figure 8. DIO transition board schematic.

4.3 ANTENNA CONTROL UNIT SOFTWARE

The antenna eontrol unit software will be developed in C and C++ utilizing tools available under the GNU Publie Lieense (GPL). Speeifieally the starting Linux distribution will be Fedora 8. The ke rnel w ill be r eeompiled ut ilizing Linux va nilla ke rnel 2.6. 23.15 a s pr ovided b y kernel.org and pa tehed with R eal T ime A pplieations Interfaee ve rsion 3.6, a s pr ovided b y RTAI.org. The kernel shall be eonfigured for minimum stable size and minimum possible use of loadable mo dules to faeilitate its a bility to b e b ooted v ia LAN. All C an d C ++ eo de w ill b e eompiled utilizing the GCC Compiler Version 4.1.2. Later versions of all software may be used.

The souree eode for the ACU is maintained in the version eontrol system used by HUSIR.

4.3.1 Processing Overview

The overall architecture of the ACU application consists of a Real-Time Kernel-Space Module (RTKM), a Real-Time User-Space Module (RTUM) and a Non-Realtime User Space Module (NRTUM) as shown in Figure 9. The RTKM is dynamically linked into the operating Linux kernel when the ACU application is started. The NRTUM is then started which creates a separate thread for each external communications interface and an additional thread for interacting with the RTKM. This thread is warped into Real-Time via RTAI LXRT support thus creating the RTUM. This allows User Space to react as quickly as possible to inputs from the RTKM.

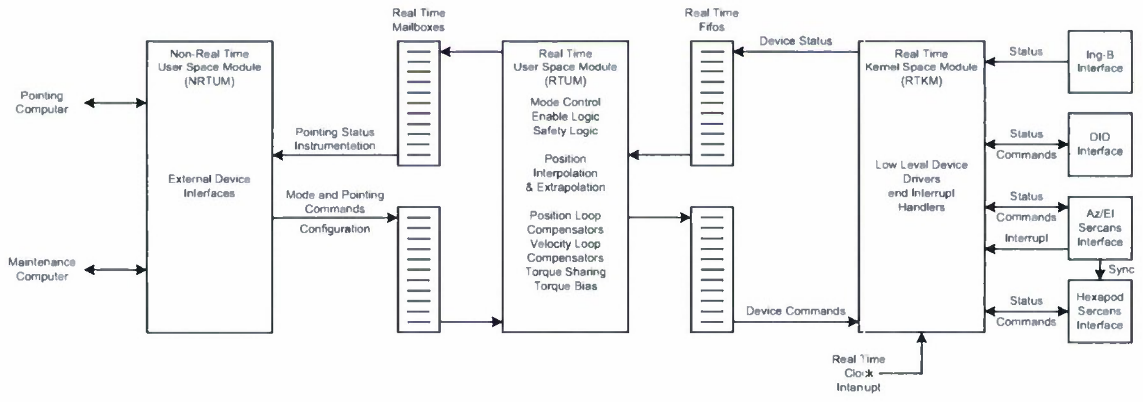

Pointing Computer
Maintenance Computer
Non-Real Time User Space Module (NRTUM)
External Device Interfaces
Real Time Mailboxes
Pointing Status Instrumentation
Mode and Pointing Commands
Configuration
Real Time User Space Module (RTUM)
Mode Control Enable Logic Safety Logic
Position Interpolation & Extrapolation
Position Loop Compensators Velocity Loop Compensators Torque Sharing Torque Bias
Real Time Fifos
Device Status
Device Commands
Real Time Kernel Space Module (RTKM)
Low Level Device Drivers and Interrupt Handlers
Real Time Clock Interrupt
Status
Commands
Interrupt
Irig-B Interface
DIO Interface
Az/El Sercans Interface
Sync
Hexapod Sercans Interface

Figure 9. ACU software top level block diagram.

Real-Time Kernel-Space Module (RTKM) A simplified flow chart of the Real-Time Kernel-Space Module (RTKM) processing is shown in Figure 10. Key to understanding the operation of this module is the fact that during normal operation, the module runs 2000 times a second, alternating between input mode and output mode. This assures that the timing requirements of the SERCANS interface are met. Upon receiving an AT Data ready interrupt from the SERCANS interface when in input mode, the module reads the drive data for all azimuth, elevation and hexapod drives. The encoders, DIO and timecode are then also read. All of this data is packaged and sent to the Real-Time User-Space Module (RTUM). The kernel module then waits for the resulting command data to be returned by the RTUM. The data is stored for use during the next interrupt cycle which will be a write cycle.

Real time interrupts occur 1000 times a second. The processing associated with these interrupts will check to see if the SERCANS interface is operating. If the SERCANS interface is running, the RTC interrupt need not perform any further processing. If not it will attempt to restart the interface and also read all other I/O devices for status reporting.

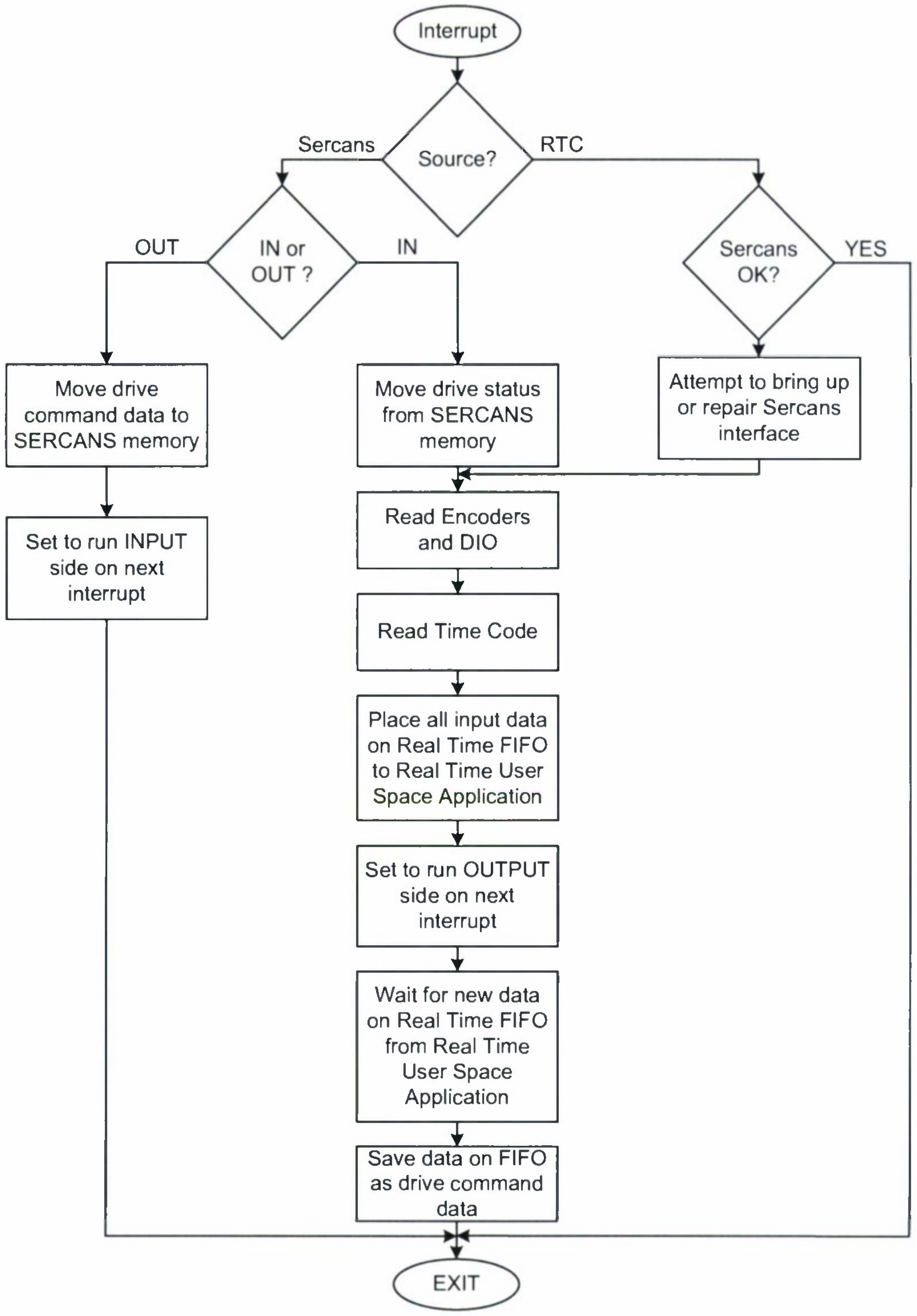
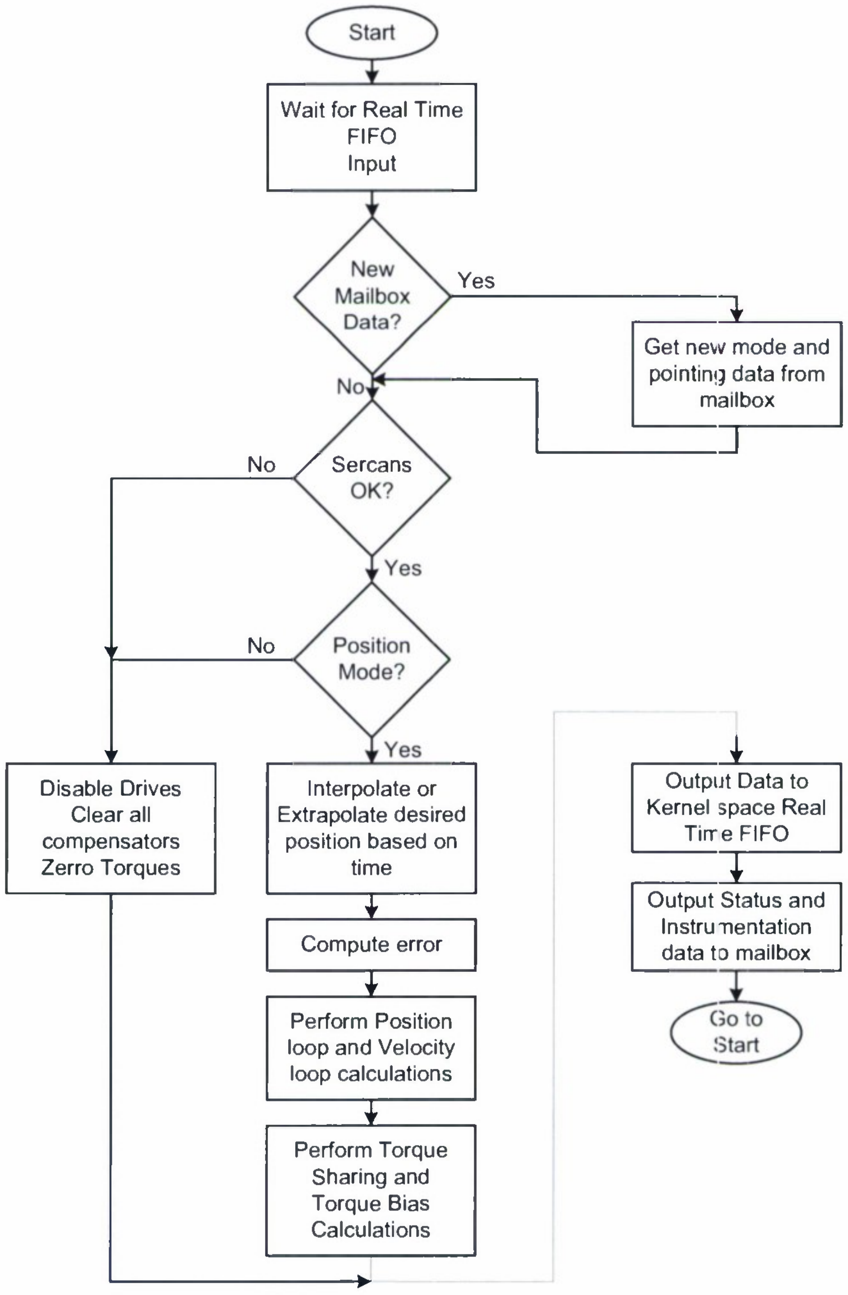

Interrupt
Source?
Sercans
RTC
IN or OUT ?
OUT
IN
Sercans OK?
YES
Move drive command data to SERCANS memory
Set to run INPUT side on next interrupt
Move drive status from SERCANS memory
Attempt to bring up or repair Sercans interface
Read Encoders and DIO
Read Time Code
Place all input data on Real Time FIFO to Real Time User Space Application
Set to run OUTPUT side on next interrupt
Wait for new data on Real Time FIFO from Real Time User Space Application
Save data on FIFO as drive command data
EXIT

Figure 10. Real-Time Kernel-Space Module (RTKM) processing.

Real-Time User-Space Module (RTUM) A simplified flow chart of the Real-Time User-Space Module (RTUM) processing is shown in Figure 11. This module processes the status messages received from the RTKM via the Real Time.



Start
Wait for Real Time FIFO Input
New Mailbox Data?
Yes
Get new mode and pointing data from mailbox
No
Sercans OK?
No
Yes
Position Mode?
No
Yes
Disable Drives Clear all compensators Zerro Torques
Interpolate or Extrapolate desired position based on time
Compute error
Perform Position loop and Velocity loop calculations
Perform Torque Sharing and Torque Bias Calculations
Output Data to Kernel space Real Time FIFO
Output Status and Instrumentation data to mailbox
Go to Start

Figure 11. Real-Time User-Space Module (RTUM) processing.

FIFOs and processes the command data received from the NRTUM module. The RTUM starts execution each time data is placed upon the Real Time FIFO by the RTKM (1000 times per second). Each time it runs, the NRTUM module checks the mailbox connected to the NRTURM to see if any command data has be received from the Pointing Computer or the Maintenance computer. If a command has been received, the RTUM retrieves the data and stores it internally.

Whether or not new command data is received, the NRTUM checks the status it has received from the RTKM to determine if the SERCANS interface is operating. If so, the latest received command data is referenced to determine the current pedestal mode. If the drives are to be disabled, the RTUM clears all memory in all compensators, zeros all torque outputs and sends this data to the RTKM via the Real Time output FIFO. If the drives are up and running, the latest pointing commands received are used to interpolate or extrapolate the current desired position for each axis. These positions along with the current position of each axis are used to generate the position error of each axis. The position error is used as an input to the position and velocity loops which ultimately determine the total torque that is to be applied to each axis. This torque command is the input to the torque sharing and torque bias algorithms which determine the amount of torque that is to be applied to each motor. These torque commands are passed to the RTKM via the Real Time Output FIFO.

Non-Real-Time User-Space Module (NRTUM) A simplified flow chart of the Non-Real-Time User-Space Module (NRTUM) processing is shown in Figure 12. This module processes the command messages received from the Pointing Computer (PC) and Maintenance Computer (MC) and passes status received from the RTUM back to the PC and MC.

The processing performed by this module is straight forward. The module waits for an input. The input can be one of four things. Processing proceeds as follows based upon the source of the input.

New socket connection. The module determines if the connection is from the Pointing Computer or Maintenance based upon the port of the connection. A new TCP (for the MC) or UDP (for the PC) socket is created for the connection.

Input from a UDP socket. This is command data from the Pointing Computer. The command is packaged and sent to the RTUM via the output mailbox.

Input from a TCP socket. This is command data from the Maintenance Computer. The command is packaged and sent to the RTUM via the output mailbox.

Input from the RTUM mailbox. This is status data from the RTUM (and ultimately the RTKM). This data is received at a rate of 1000 times per second. Based upon configuration parameters, this data rate is decimated, repackaged and forwarded to the Pointing Computer and Maintenance Computer as status and instrumentation messages.

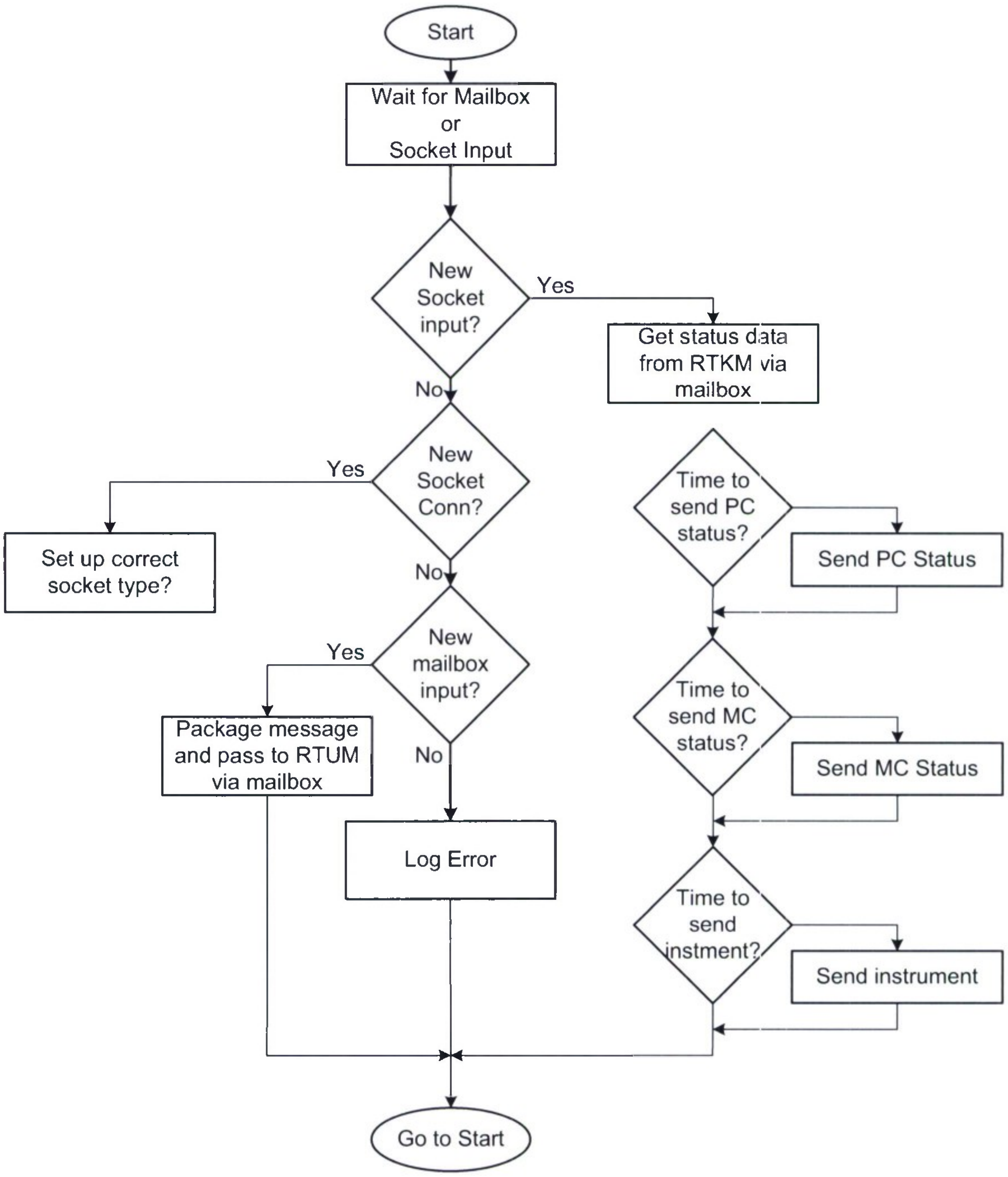


Figure 12. Non-Real-Time User-Space Module (NRTUM) processing.

4.3.2 Detailed Design Information

The following paragraphs detail the design of each of the three modules described above.

Real-Time Kernel-Space Module (RTKM) The purpose of the Real-Time Kernel-Space Module is to perform the low level interaction with the hardware interfaces. The RTKM must be written in C and contains three sections: Initialization (code executed when the module is loaded), Cleanup (code executed when the module is removed), and Interrupt Handling (the code that is executed each time an interrupt occurs.

The processing to be performed by each of these sections is as follows:

<u>Initialization</u>

The initialization section of the RTKM shall take three arguments as follows:

SIMULATE_TIMECODE	(0/1 => Timecode interface present/not present)
SIMULATE_DIO	(0/1 => DIO interface present/not present)
SIMULATE_SERCANS	(0/1 => SERCANS interface present/not present)

Given these inputs, the initialization section of the RTKM shall perform the following processing.

- Initialize the Time Code Module with the SIMULATE_TIMECODE argument and a pointer to a data structure to hold time code data.
- Initialize the DIO Module with the SIMULATE_DIO argument and a pointer to a data structure to hold DIO data.
- Initialize the SERCANS Module with the SIMULATE_SERCANS argument, a pointer to a data structure to hold SERCANS data and a pointer to an interrupt handler for the SERCANS AT data ready interrupt.
- Initialize the RTC service of RTAI to cause an RTC interrupt every 1 ms.
- Create and input and output real time FIFO utilizing RTAI services.

<u>SERCANS Interrupt Handler</u>

The SERCANS Interrupt Handler is called every 500 µs. Upon each call it will alternate between read mode and write mode, hence the drives shall be effectively updated at a rate of 1000 times per second. When in read mode, the SERCANS interface shall perform the following processing.

- Call the Time Code Module for the Current time.
- Transfer all pertinent drive data from the SERCANS dual ported memory.
- Call the DIO module for the current state of the encoders and digital inputs.
- Create the following data structure and place on the output real time FIFO for pick up by the RTUM. All units are native to the device. Any units conversion will be performed by the RTUM. In the following structure, the Pseudo variable is used to hold

pseudo-input bits that are generated by internal processing. Examples of these bits are time code status, encoder ready status, etc.

```
typedef struct RTKMOutputData {
        int nanosecond;                 //Time of validity
        int second;
        int minute;
        int hour;
        int jday;
        int year;
        int AzEncoderCounts;            //Binary az encoder values
        int ElEncoder Counts[2];        //Two binary el encoder values
        int MotorVelocity[12];          //Velocity off all motors
        int motorTorque[12];            //Reported torque from all motors
        int motorVolts[12];             //Bus Voltage at each motor
        int motorWatts[12];             //Power consumption of each motor
        int motorTemp[12];              //Temperature of each motor
        int ampTemp[12];                //Temperature of each amplifier
        int shaftPosition[6];           //Position of each hexapod shaft
        unsigned char inputs[DIGITAL_INPUT_CNT];        //State
  of discrete ins  int
        unsigned char outputs[DIGITAL_OUTPUT_CNT]; //State of discrete outs
        unsigned char Pseudo[PSEUDO_INPUT_BYTES]; //State of generated bits
        } RTKMOutputData_t;
```

- Wait for a response of the following form on the input Real Time FIFO from the RTUM. Both desired velocity and desired torque are passed to the RTUM because thc BOSCH drive revert to velocity mode when in a prelimit condition.

```
        typedef struct RTKMInputData {
                int pseudoDigitalOutputs;
                int desiredVelocity[12];
                int desiredTorque[12];
                int desiredPosition[6];
                unsigned char outputs[DIGITAL_OUTPUT_CNT]; //State of discrete outs
        RTKMInputData_t;
```

- Move the data from the Real Time FIFO to internal memory.
- Exit and wait for another interrupt.

When in write mode, the SERCANS interface shall move the azimuth and elevation data received from the RTUM during the previous read cycle to the SERCANS dual ported RAM for transmission t o t he 8 a zimuth dr ivcs a nd t he 4 e levation dr ivcs. This data s hall i nelude t hc following:

Drive M ode (Standby or T orque M ode) for the 8 azimuth m otors a nd the four elevation motors.

- If no prelimit is set for azimuth, the desired torque for eaeh of 8 motors.
- If a prelimit is set for azimuth, the desired veloeity for eaeh of 8 motors.
- If no prelimit is set for elevation , the desired torque for eaeh of 4 motors.
- If a prelimit is set for elevation, the desired veloeity for eaeh of 4 motors.
- Reset the SERCANS timeout timer.

Real Time Cloek Interrupt Handler

The real Time Interrupt Handler will run ever 1 ms. The purpose of this handler is to eheck the S ERCANS t imeout to v erify th at th e S ERCANS in terfaee is o perating. This i s done b y eheeking the SERCANS timeout timer. If this timer/eountcr reaches 10, the SERCANS interrupt has n ot o eeurred for 1 0 m illiseeonds. In t his ease, t he R eal T ime C loek Interrupt H andlcr attempts to r estart th e S ERCANS in terface. All other da ta i s r ead and s ent t o t he R TUM. In addition, a pseudo input is set to refleet the state of the SERCANS interfaee as down.

Cleanup

When the kernel module is unloaded, all interrupts are diseonneeted and set baek to their quieseent s tates. Eaeh h ardware m odules (SERCOS, T imeCodc a nd D IO) r elease m cthod i s ealled to release their respeetive memories and hardware devices.

Time Code Module

The Time Code Module shall be initialized by the RTKM when it is loaded into thc activc Linux Kernel. The initialization shall allow for the startup of the Timc Code Module in normal or simulation mode. If started in simulation mode no furthcr initialization need oeeur. If started in normal mode, the following operations shall bc performed:

- The timeeode deviee shall be found on the PCI bus by calling the ***pci_get_subsys*** Linux service with the following parameters:

VENDOR	0x10B5
DEVICE	0x9030
SUBVENDOR	0x10B5
SUBDEVICE	0x1042

- The timecode device I/O memory shall be mapped into kernel spacc with the following three ealls:

  pci_resource_start
  request_mem_rcgion
  ioremap_noeaehe

The timecode module shall provide a routine to allow the RTKM to request the current time at any time. If in simulation mode, the Time Code Module shall return the time from the internal PC clock and compute the current nanosecond from RTAI resources. If not in simulation mode, the Time Code Module shall return the time as read from the IRIB-B interface. The returned data shall at a minimum include the following data:

```
typedef struct timecodeModuleData {
        int nanosecond;
        int second;
        int minute;
        int hour;
        int jday;
        int year;
        short syncStatus
} timecodeModuleData_t;
```

The timecode module shall include a method to set the current year as this is not provided by the IRIG-B Interface.

The timecode module shall be terminated upon request from the RTKM. When terminated, the Time Code Module shall release all resources (memory and devices) and then exit.

DIO Module

The Dio Module shall be initialized by the RTKM when it is loaded into the active Linux Kernel. The initialization shall allow for the startup of the Dio Module in normal or simulation mode. If started in simulation mode no further initialization need occur. If started in normal mode, the following operations shall be performed:

- The Dio device shall be found on the PCI bus by calling the ***pci_get_subsys*** Linux service with the following parameters:

VENDOR	0x16D5
DEVICE	0x4248

- The Dio device I/O memory shall be mapped into kernel space with the following three calls:
  pci_resource_start
  request_mem_region
  ioremap_nocache

The Dio module shall provide a routine to allow the RTKM to request the current status of the inputs to the concentrator subsystem at any time. If in simulation mode, the DIO module shall return zeros for all inputs, because the RTUM will simulate their values. If not in simulation mode, the Time Code Module shall return all data as read from the local concentrator module. The returned data shall at a minimum include the following data:

```
typedef struct dioModuleData {
        unsigned int encoderBits[3];          //El1, El2, Az Encoders (in order)
        unsigned int localInputs;             //32 Below wrap digital inputs
        unsigned short remoteInputs;          //16 Above wrap digital inputs
        unsigned short digitalOutputs;        //Current state of 16 above wrap outputs
        unsigned short encoder3Status;        //Azimuth Encoder status
        unsigned short status;                //Status of both elevation encoders
        unsigned short spins;                 //Loops waiting for encoder data
} dioModuleData_t;
```

The Dio module shall provide a routine to allow the RTKM to change the state of the 16 above wrap digital outputs.

The Dio module shall be terminated upon request from the RTKM. When terminated, the Dio Module shall release all resources (memory and devices). and then exit.

Sercans Module

The Sercans Module shall be initialized by the RTKM when it is loaded into the active Linux Kernel. The initialization shall allow for the startup of the Sercans Module in normal or simulation mode. If started in simulation mode no further initialization need occur. If started in normal mode, the following operations shall be performed:

- The two SERCANS devices shall be found on the PCI bus by calling the ***pci_get_subsys*** Linux service twice with the following parameters:

VENDOR	0x104A
DEVICE	0x4000

- The SERCANS device I/O memory shall be mapped into kernel space for each of the devices with the following three calls:
  pci_resource_start
  request_mem_region
  ioremap_nocache
- The SERCANS device 0 (as set by the front panel address) shall be set to Phase zero and configured for 12 drives. The following read and write data shall be configured for automatic transfer between the SERCANS device and each drive:
  - Torque Command (Write to drive)
  - Velocity Command (Write to drive)
  - Current Torque (Read from Drive)
  - Current Velocity (Read from drive)
  - Bus Voltage (Read from drive)
  - Bus Power (Read from drive)

    - Motor Temperature (Read from drive)
    - Amplifier Temperature (Read from drive)
- The SERCANS device 1 (as set by the front panel address shall be set to Phase zcro and configured f or 6 dr ives. The following r ead a nd w rite d ata s hall be configured f or automatic transfer between the SERCANS device and each drivc:
    - Position Command (Write to drive)
    - Current Velocity (Read from drive)
    - Bus Voltage (Read from drive)
    - Bus Power (Read from drive)
    - Motor Temperature (Read from drive)
    - Amplifier Temperaturc (Read from drive)
- Both S ERCANS i nterfaces s hall b e r un up t o phase t wo. At a ll th e drives on bot h SERCANS interfaces shall be configured from the file SERCANS.CFG thc format of this f ile s hall b e s imilar to th e f ollowing e xample. Note th at th is a llows f or th c configuration of any drive parameter on either of the two SERCANS interfaces.

```
#Card Configuration
#Card, 0, Parameter, Value, Length, Comment
0, 0, Y-0-0017, 2, 2,            Sercos Baud rate (2 MBaud)
0, 0, Y-0-0004, 1000, 2,         Sercos cycle time
0, 0, Y-0-0005, 1000, 2,         NC cycle time
0, 0, Y-0-0009, 0, 2,            Life counter difference
0, 0, Y-0-0023, 1, 2,            Language selection
0, 0, Y-0-0029, 4, 2,            Power on target phase

#Drive Configuration
#Card, Drive, Parameter, Value, Length, Comment
#Note: Drive of -1 sends parameter to all drives on card
0, -1, P-0-1350, 2, 2,           Stop PLC
0, -1, S-0-0032, 2, 2,           Primary mode of operation (Velocity)
0, -1, S-0-0033, 1, 2,           Secondary operation mode 1 (Torque)
0, -1, S-0-0034, 0, 2,           Secondary operation mode 2 (Undefined)
0, -1, S-0-0035, 0, 2,           Secondary operation mode 3 (Undefined)
0, -1, S-0-0208, 0, 2,           Celsius Temperature Scaling
0, -1, P-0-0109, 1200, 2,        Torque/Force Peak Limit (0.1 of a percent)
0, -1, S-0-0082, 1200, 2,        Torque/force positive limit (0.1 of a percent)
0, -1, S-0-0083, -1200, 2,       Torque/force negative limit (0.1s of a
percent)
0, -1, S-0-0091, 60000000, 4,    Vclocity bipolar limit (0.0001s of an RPM)
0, -1, S-0-0092, 1200, 2,        Torque/force bipolar limit (0.1s of a percent)
0, -1, S-0-0823, 0, 2,           Torque/force ramp (0.1s of a percent)
0, -1, S-0-0822, 0, 2,           Torque/force ramp time (ms)
```

- Both SERCANS interfaces shall be run up to phase four and the SERCANS AT interrupt shall be enabled.

The SERCANS module shall provide a routine to allow the RTKM to query the state of all drives. The RTKM shall utilize this routine immediately upon receipt of an AT data ready interrupt (when in read mode) to assure correct synchrony with the SERCANS interface. The data supplied by the SERCANS interface shall at a minimum include the following:

```
typedef struct SERCANSInData {
        int MotorVelocity[12];          //Velocity off all motors
        int motorTorque[12];            //Reported torque from all motors
        int motorVolts[12];             //Bus Voltage at each motor
        int motorWatts[12];             //Power consumption of each motor
        int motorTemp[12];              //Temperature of each motor
        int ampTemp[12];                //Temperature of each amplifier
        int shaftPosition[6];           //Position of each hexapod shaft
        int localDigitalInputs;         //State of 32 local concentrator
inputs
        int pseudoInputs;               //State of 32 pseudo inputs
        short remoteDigitalInputs;      //State of 16 remote concentrator
inputs
} SERCANSInData_t;
```

The SERCANS module shall provide a routine to allow the RTKM to command the state of all drives. The RTKM shall utilize this routine immediately upon receipt of an AT data ready interrupt (when in write mode) to assure correct synchrony with the SERCANS interface. The data supplied to the SERCANS interface shall at a minimum include the following:

```
typedef struct SERCANSOutData {
        int desiredVelocity[12];    //Desired velocity of Az and El drives
        int desiredTorque[12];      //Desired torque applied to Az and El drives
        int desiredPosition[6];     //Desired position of hexapod drives
        char driveState[18];        //Desired state of all drives
SERCANSOutData_t;
```

The SERCANS module shall be terminated upon request from the RTKM. When terminated, the SERCANS Module shall release all resources (memory and devices) and then exit.

Real Time User-Space Module (RTUM) The Real Time User-Space Module (RTUM) performs the bulk of the processing within the ACU. There are many complex data structures that are supported and utilized by this processing. For this reason, the significant data structures of the RTUM are discussed in the following paragraphs and then the processing of the RTUM is described in subsequent paragraphs.

Pedestal Object

One Pedestal Object is created by the RTUM to represent the features of HUSIR antenna that apply to the pedestal as a whole. This Pedestal Object instantiates other objects (axis, compensator, etc.) to complete the definition of the antenna.

Pedestal Object Data

The ***pedestalDefinition*** structure holds all data describing the pedestal that may be configured on startup but may not change once the RTUM has commenced normal operation. In other words, the ***pedestalDefinition*** structure data is loaded once during system startup and then does not change thereafter. The purpose of this structure is to hold data that describes the physical characteristics of the pedestal that do not change with time or performance modifications such as number of axis and coordinate system organization. The ***pedestalDefinition*** structure is defined below.

```
typedef struct pedestalDefinition {
        string strName;                    //Ascii Name of pedestal
        bool bSimulateDiscreteInputs;      //Simulate discrete inputs if true
        bool bSimulateDiscreteOutputs;     //Simulate discrete outputs if true
        bool bSimulateAnalogInputs;        //Simulate analog inputs if true
        bool bSimulatePositionInputs;      //Simulate position inputs if true
        bool bSimulateAnalogOutputs;       //Simulate analog outputs if true
        int iServoUpdateRate;              //Servo loop update rate (Updates per
Second)
        int iAxisCount;                    //Number of driven axis on pedestal
        int iReceiverCount;                //Number of driven axis on pedestal
        int eCoordinateSystem;             //Coordinate Transform type
} pedestalDefinition_t;
```

The ***pedestalConfiguration*** structure holds all data describing the pedestal that may be changed at any time. This data may change due to real-time events (in the case of orientation sensors) or to design, implementation or test modifications. The ***pedestalConfiguration*** structure is loaded with initial data on startup and may be modified at any time via a call to the ***configurePedestal*** method. The ***pedestalConfiguration*** structure is defined below.

```
typedef struct pedestalConfiguration {
        double dLatitude;                  //Latitude of fixed pedestal (degrees north)
        double dLongitude;              //Longitude of fixed pedestal (degrees east)
        double dAltitude;               //Altitude of fixed pedestal (meters)
        double dPiteh;                  //Piteh of pedestal base (RH Coord, degrees)
        double dRoll;                   //Roll of pedestal base (RH Coord, degrces)
        double dHeading;                //Heading of ped base (RH Coord, degrees)
} pedestalConfiguration_t;
```

The ***pedestalStatus*** strueture holds the eurrent state of any variables assoeiated with the pedestal that may ehange with time. These variables inelude pedestal orientation parameters to allow for the future addition of sensors that may bc incorporated into a mount model loeated within the ACU itself. Additionally, the raw data reeeived from the RTKM and the raw data sent to the RTKM is stored here. The raw data reeeived from the RTKM is utilized by the RTUM to determine the current pedestal state and to eompute the desired state and torques for each drive. This data is sent to the RTKM and also stored in the ***pedestalStatus*** strueturc. The ***rawIn*** and ***rawOut*** variables are available to be streamed to any storage deviee to eapture the eomplete state of the pedestal during each eomputation period. The ***pedestalStatus*** strueture is dcfined below.

```
typedef struct pedestalStatus {
        RTKMOutputData_t rawIn;   //Raw data input from the RTKM
        RTKMInputData_t rawOut;   //Raw data output to the RTKM
        double dAzimuth;          //Current geo az position (degrees east of truc north)
        double dElevation;        //Current geo el position (degrees up from horizon)
        double dLatitude;         //Currcnt geographic latitude (degrees east)
        double dPitch;            //Current pitch of ped base (RH Coord, dcgrees)
        double dRoll;             //Currcnt roll of ped basc (RH Coord, degrees)
        double dHeading;          //Current heading of ped base (RH Coord, dcgrces)
} pedestalStatus_t;
```

Pedestal Objeet Methods

The following methods, at a minimum, are provided by the Pedestal Object.

CONFIGUREPEDESTAL

The ***configurePedestal*** method ean be invoked asynehronously with respeet to the servo loop calculations. Thus it can be ealled from the RTUM or the NRTUM. Normally it will bc callcd from the NRTUM as the result of receiving new configuration from the maintcnance eomputer. It is called with a ***pedestalConfiguration*** structure whieh holds the new information. All this method does is eopy the ***pedestalConfiguration*** strueture to an internal mcmory arca for latcr uscd by the ***updateServo*** method. In this way therc arc no conflicts due to thc asynchronous naturc of the updates.

CONFIGUREAXIS

The ***configureAxis*** method can be invoked asynchronously with respect to the servo loop calculations. Thus it can be called from the RTUM or the NRTUM. Normally it will be called from the NRTUM as the result of receiving new configuration from the maintenance computer. It is called with an axis index and an axisConfiguration structure which holds the new information. All this method does is call the specified axis object's ***configureAxis*** method. Providing this method in the pedestal object helps to effectively encapsulate the axis object.

UPDATESERVO

The updateServo method is called each time the servo loop is run. This first function this method performs is to copy any new configuration data that has been asynchronously (from the SERCANS interrupt handler) received into the variables that are actually used for each calculation Next, the updateServo method for each axis (azimuth and elevation) is called to actually perform the processing.

REPORTPEDESTALSTATUS

The reportPedesatalStatus may be called at any time to determine the status of the pedestal. This returns the status of all devices and internal computations not associated with any particular axis but associated with the with the pedestal as a whole. . The units reported by this method are compatible with the maintenance computer and pointing computer Interface Control Documents, they are not in units that are necessarily compatible with the SERCOS drives.

REPORTAXISSTATUS

The reportAxisStatus method may be called at any time to determine the status of an axis. This method takes an index representing azimuth or elevation as an input and returns the status of all devices and internal computations associated with the requested axis. The units reported by this method are compatible with the maintenance computer and pointing computer Interface Control Documents, they are not in units that are necessarily compatible with the SERCOS drives. This method is implemented by the Pedestal Object by calling the requested axis object's reportAxisStatus method.

Pedestal Object Configuration File

The pedestal is instantiated with a configuration object that has been created from a configuration file (pedestal.cfg). A sample configuration file for a two axis pedestal is shown below. Note that the configuration file is divided into two sections: DEFINITION and CONFIGURATION. Those items in the definition section cannot be modified during ACU runtime. The items in the CONFIGURATION section may be modified at any time. If a configuration items are modified, the ACU application shall update the pedestal configuration file so that the modification persist through to the next ACU startup. Note that the pedestal configuration file contains an include definition for each of the defined axis (azimuth.cfg and elevation.cfg).

```
//Pedestal definitions
[PEDESTAL DEFINITION]
NAME         =        TEST_PEDESTAL    //Name of Pedestal (31 characters or less)

SIMULATE_DISCRETE_INPUTS   = true  //Simulate pedestal discrete inputs if true
SIMULATE_DISCRETE_OUTPUT   = true  //Simulate pedestal discrete outputs if true
SIMULATE_ANALOG_INPUTS     = true  //Simulate pedestal analog inputs if true
SIMULATE_ANALOG_OUTPUTS    = true  //Simulate pedestal analog outputs if true
SIMULATE_POSITION_INPUTS   = true  //Simulate pedestal position inputs if true
SERVO_RATE                 = 1000 //Servo computations per second
RECEIVER_COUNT             = 4     //Number of monitored receivers
AXIS_COUNT                 = 2     //Number of driven axes
COORDINATE_SYSTEM          = 0     //Elevation - Azimuth

[PEDESTAL CONFIGURATION]
LATITUDE                           = 0.0   //Degrees north
LONGITUDE                          = 0.0   //Degrees east
ALTITUDE                           = 0.0   //Meters
PITCH                              = 0.0   //Degrees (RH Coordinate system
ROLL                               = 0.0   //Degrees (RH Coordinate system)
HEADING                            = 0.0 //Degrees east of true north

#include azimuth.cfg                       //Axis 0
#include elevation.cfg                     //Axis 1
```

Axis Object

An Axis Object is instantiated for each axis defined for the pedestal (as held in the pedestalDefinition structure within the Pedestal Object). The axis object embodies the data structures and methods necessary to represent and control each individual axis of the pedestal.

Axis Object Data

The ***axisDefinition*** structure holds all data describing the axis that may be configured on startup but may not change once the RTUM has commenced normal operation. In other words, the ***axisDefinition*** structure data is loaded once during system startup and then does not change thereafter. The purpose of this structure is to hold data that describes the physical characteristics of the axis that do not change with time or performance modifications such as inertias, motor performance characteristics and which input bits represent which functions. Many of these parameters are included so that the performance of the pedestal simulator within the ACU can be realistic.

```
typedef struct axisDefinition {
        string strName;
        double dAxisInertia;                //Axis inertia (ft-lb-sec^2)
        double dAxisLrf;                    //Axis lowest resonant frequency (Hz)
        double dGearBoxInertia;             //Axis gear box inertia (ft-lb-sec^2)
        double dShaftInertia;               //Motor shaft inertia (ft-lb-sec^2)
        double dViscousFriction;            //Axis viscous friction (ft-lb / rad / sec)
        double dCoulumbFriction;            //Coulumb friction (ft-lb)
        double dMotorFriction;              //Motor shaft Friction (ft-lb / rad /sec)
        double dGearRatio;                  //Axis gear ratio (Shaft to Axis)
        double dDampingRatio;               //Axis damping ratio (Q)
        double dPositionSensorGearing;      //Gear ratio of two sensors (0 means
average)
        int iLowerPrelimitBit;              //Discrete input bit for lower prelimit
        int iLowerFinalLimitBit;            //Discrete input bit for lower final limit
        int iUpperPrelimitBit;              //Discrete input bit for upper prelimit
        int iUpperFinalLimitBit;            //Discrete input bit for upper final limit
        int iSectorBit;                     //Discrete input bit for sector switch
        vector<positionSensor_t> positionSensor;    //Position sensor inputs
        double dMotorKv;                    //Deg/Sec per volt if non-zero (i.e. no vel
loop)
        vector<motor_t> motor;              //Physical offsets of position sensors
        unsigned char ucPowerOnControl[DIGITAL_OUTPUT_BYTES];
        unsigned char ucPowerOnSense[DIGITAL_INPUT_BYTES];
        unsigned char ucEnableControl[DIGITAL_OUTPUT_BYTES];
        unsigned char ucEnableSense[DIGITAL_INPUT_BYTES];
        unsigned char ucBrakeControl[DIGITAL_OUTPUT_BYTES];
        unsigned char ucBrakeSense[DIGITAL_INPUT_BYTES];
        unsigned char ucInterlock[DIGITAL_INPUT_BYTES];
        unsigned char ucPseudoInterlock[PSEUDO_INPUT_BYTES];
        double dLowerPreLimit;       //Location of lower directional limit (degrees)
        double dLowerFinalLimit;     //Location of lower final interlock limit (degrees)
        double dUpperPreLimit;       //Location of upper directional limit (degrees)
        double dUpperFinalLimit;     //Location of upper final interlock limit (degrees)
} axisDefinition_t;
```

Of particular interest in the above data structure are the unsigned char definitions. These definitions describe the status of the input or output bits that must be present for a certain operation to take place. These definitions are used as follows:

ucPowerOnControl: Each bit that is on in this array must be turned on in the RTKM discrete outputs in order to power up the axis.
ucPowerOnSense: If each bit that is on in this array is on in the discrete inputs received from the RTKM, the axis has been successfully powered on.
ucEnableControl: Each bit that is on in this array must be turned on in the RTKM discrete outputs in order to enable the axis.
ucEnableSense: If each bit that is on in this array is on in the discrete inputs received from the RTKM, the axis has been successfully enabled.
ucBrakeControl: Each bit that is on in this array must be turned on in the RTKM discrete outputs in order to release the brakes for the axis.
ucBrakeSense: If each bit that is on in this array is on in the discrete inputs received from the RTKM, the brakes for the axis have been successfully released.
ucInterlockl: If any bit that is on in this array is also on in the inputs received from the RTKM, the axis is in an interlock condition and may not be powered up.
ucBrakeSense: If any bit that is on in this array is also on in the pseudo inputs received from the RTKM, the axis is in an interlock condition and may not be powered up.

The ***axisConfiguration*** structure holds all data describing the axis that may be changed at any time. This data may change due to design, implementation or test modifications such as compensators and maximum performance characteristics. The ***axisConfiguration*** structure is loaded with initial data on startup and may be modified at any time via a call to the ***configureAxis*** method. The ***axisConfiguration*** structure is defined below.

```
typedef struct axisConfiguration {
        double dPositionOffset;        //Axis position offset (deg)
        double dMinPosition;           //Location of lower software limit (deg)
        double dMaxPosition;           //Location of upper software limit (deg)
        double dStowPosition[3];       //Stow positions (deg)
        double dMaxVelocity;           //Maximum allowed velocity (deg/sec)
        double dMaxAcceleration;       //Maximum allowed acceleration (degr/sec^2)
        double dMaxTotalTorque;        //Max allowed torque (at motor shaft) (ft-lb-sec^2)
        double dBrakeOnDelay;          //Delay from brake release until torque applied (sec)
        double dBrakeOffDelay;         //Delay from torque off to break set (seconds)
        double dBrakeSetVelocity;      //Max axis vel at which brakes can be set (deg/sec)
        double dStowPositionTolerance;       //Tolerance for attaining stow position (deg)
        double dTorqueBias;            //Torque bias amount (percent of max total torque)
        eTorqueBias_t cTorqueBias;     //Torque bias method
        double dTachFilter;             //Bandwidth of filter for tachometer inputs (Hz)
        double dPositionFilter;         //Low pass filter on position differentiator position
        string strVelocityCompensator;          //Velocity compensator definition
        string strVelLimitCompensator;          //Velocity limit definition
        string strType1PosCompensator;          //Type 1 position loop compensator
definition
        string strLoGainPosCompensator;         //Lo Gain position loop compensator
definition
        string strHiGainPosCompensator;         //Hi Gain position loop compensator
definition
        string strRFFPosCompensator;            //Rate feed forward compensator definition
} axisConfiguration_t;
```

The ***axisStatus*** structure holds the current state of any variables associated with the axis that may change with time. These variables include pedestal orientation parameters to allow for the future addition of sensors that may be incorporated into a mount model located within the ACU itself. Additionally, the raw data received from the RTKM and the raw data sent to the RTKM is stored here. The raw data received from the RTKM is utilized by the RTUM to determine the current pedestal state and to compute the desired state and torques for each drive. This data is sent to the RTKM and also stored in the ***axisStatus*** structure. The ***rawIn*** and ***rawOut*** variables are available to be streamed to any storage device to capture the complete state of the pedestal during each computation period. The ***axisStatus*** structure is defined below.

```
typedef struct axisStatus {
        eAxisState_t eAxisState;
        eAxisCommand_t eAxisCommand;
        eCompensator_t eCurrentCompensator;
        eCompensator_t eSelectedCompensator;
        double dRawPosition;
        double dCurrentPosition;                //Current axis position (degrees)
        double dDesiredPosition;                //Current desired axis position (degrees)
        double dPositionError;                  //Position error (degrees)
        double dCurrentShaftVelocity;           //Motor shaft velocity (from Tachs)
(deg/sec)
        double dCurrentVelocity;                //Axis vel (from Tachs/Gear Ratio) (deg/sec)
        double dComputedVelocity;               //Axis vel (from differentiating pos)
(deg/sec)
        double dDesiredVelocity;                //Output of pos compensator, vel cmd
(deg/sec)
        double dDrivenVelocity;                 //Velocity cmd after all limits applied
(deg/sec)
```

```
        double dVelocityError;              //Velocity error (degrees/second)
        double dDesiredAcceleration;        //Output of vel loop compensator
(deg/sec^2)
        double dDesiredTorque;              //Desired axis torque (ft-lbs)
        double dSharedTorque;               //Desired torque for each motor pair (ft-lbs)
        double dTorque1;                    //Torque applied to first motor of each pair
        double dTorque2;                    //Torque applied to second motor of each
pair
} axisStatus_t;
```

Axis Object Configuration File

The axis is instantiated with a configuration object that has been created from a configuration file (axisname.cfg). A sample axis configuration file is shown below. Note that the configuration file is divided into two sections: DEFINITION and CONFIGURATION. Those items in the definition section cannot be modified during ACU runtime. The items in the CONFIGURATION section may be modified at any time. If any configuration items are modified, the ACU application shall update the axis configuration file so that the modifications persist through to the next ACU startup. Note that the compensators are defined by strings. Compensator design is discussed in paragraph.

```
//Azimuth axis definition
[AXIS 0 DEFINITION]
NAME                    = ELEVATION         //Name of axis (31 characters or less)
GEAR_RATIO              = 1200              //Total gear ratio of azimuth axis
INERTIA                 = 50.0              //ft-lb sec^2
LRF                     = 15.0              //Hz
GEARBOX_INERTIA         = 0.01              //ft-lb sec^2
SHAFT_INERTIA           = 0.000034722       //ft-lb sec^2
VISCOUS_FRICTION              = 2.5         //ft-lb / rad / sec
COULUMB_FRICTION              = 0.0         //ft-lb
MOTOR_FRICTION                = 0.0         //ft-lb / rad / sec
DAMPING_RATIO                 = 0.1         //Azimuth axis damping ratio (for
simulation)
POSITION_SENSOR_GEARING       = 0.0         //Gear ratio of two sensors (0 means
average)
POSITION_SENSORS              = 2           //Position sensor inputs (Up to two)
POSITION_SENSOR_OFFSETS       = 0.0, 0.0    //Physical offsets of position sensors
MOTOR_KV                      = 0.0         //Deg/Sec per volt if non-zero (i.e. no vel
loop)
MOTOR_DRIVES                  = 0, 1, 2, 3  //Channels for motor drive
MOTOR_KT                      = 0.167, 0.167, 0.167, 0.167  //ft-lb per amp
```

```
MOTOR_CV                    = 1.0, 1.0, 1.0 ,1.0          //amps per volt
MOTOR_TACHS                 = 0, 1, 2, 3     //Channels for motor tachs
TACH_KV                     = 2.5, 2.5, 2.5, 2.5     //Tach scaling in V per KRPM
POWER_ON_CONTROL            =       //Discrete outputs to power on axis
POWER_ON_SENSE              =       //Discrete input bits to sense power on
ENABLE_CONTROL              =       //Discrete outputs to enable axis
ENABLE_SENSE                =       //Discrete input bits to sense enable
BRAKE_CONTROL               =       //Discrete outputs to release brakes
BRAKE_SENSE                 =       //Discrete inputs to sense brake release
INTERLOCK                   =       //Discrete input bits which create interlock
PSEUDO_INTERLOCK            =       //Pseudo Input bits which create interlock
LOWER_PRELIMIT_BIT          = 1     //Discrete input for lower prelimit switch
LOWER_FINAL_LIMIT_BIT       = 2     //Discrete input for lower final limit switch
UPPER_PRELIMIT_BIT          = 3     //Discrete input for upper prelimit switch
UPPER_FINAL_LIMIT_BIT       = 4     //Discrete input for upper final limit switch
SECTOR_BIT                  = 5     //Discrete input for sector switch

//Simulation Parameters
LOWER_PRELIMIT              = -182.5     //CCW directional limit switch (degrees)
LOWER_FINAL_LIMIT           = -185.0     //CCW final limit switch position (degrees)
UPPER_PRELIMIT              = 182.5      //CW directional limit switch (degrees)
UPPER_FINAL_LIMIT           = 185.0      //CW final limit switch position (degrees)
TACH_NOISE                  = 0.01       //RMS value of tach noise (in volts)
TACH_BIAS                   = 0.00       //Bias in volts of Tach A/D input
DRIVE_NOISE                 = 0.00       //RMS value of drive D/A noise (volts)
DRIVE_BIAS                  = 0.00       //Bias in volts of Drive A/D output
POSITION_SENSOR_NOISE       = 0.00       //RMS value of position sensor noise
(degrees)

//Azimuth axis configuration
[AXIS 0 CONFIGURATION]
POSITION_OFFSET             = 0.0        //Axis position offset
LOW_LIMIT                   = -180.0     //Minimum azimuth position (degrees)
HIGH_LIMIT                  = 180.0      //Maximum azimuth position (degrees)
STOW_POSITION_1             = 0.0        //Azimuth stow position #1
STOW_POSITION_2             = 0.0        //Azimuth stow position #2
STOW_POSITION_3             = 0.0        //Azimuth stow position #3
MAX_VELOCITY                = 20.0       //Maximum azimuth velocity
(degrees/second)
MAX_ACCELERATION            = 50.0       //Maximum azimuth acceleration
(deg/sec^2)
```

```
MAX_TORQUE                  = 2.0        //ft-lb
BRAKE_ON_DELAY              = 0.010      //seconds
BRAKE_OFF_DELAY             = 0.010      //seconds
BRAKE_SET_VELOCITY          = 0.05       //degrees/second
STOW_POSITION_TOLERANCE     = 0.01       //degrees
TORQUE_BIAS                 = 15         //Torque Bias (percent of max axis torque)
TACH_FILTER                 = 0.0        //Tach Input Filter Frequency (Hz)
POSITION_FILTER             = 0.0        //Low pass filter on pos differentiator for
computed vel
TORQUE_BIAS_METHOD          = 1          //0: No Crossover; 1: Crossover
VELOCITY_COMPENSATOR        = Velocity; 0.0; 0.0; 0.0; 0.0; 20.0
VELLIMIT_COMPENSATOR        = Velocity Limit; 0.0; 0.0; 20.0; 0.0; 1.0
TYPE1_POS_COMPENSATOR       = Type 1 Position; 0.0; 0.0; 20.0; 0.0; 1.0; LG, 1.33
LO_GAIN_POS_COMPENSATOR = Lo Gn Pos; 0.0; 0.0; 20.0; 0.0; 2.0; LD, 0.179; LG, 1.79;
LI
HI_GAIN_POS_COMPENSATOR     = Hi Gn Pos; 0.0; 0.0; 20.0; 20.0; 4.0; LD, 0.433; LG, 4.33;
LI
RFF_POS_COMPENSATOR         = RFF; 0.0; 0.0; 20.0; 20.0; 4.0; LD, 0.433; LG, 4.33; LI
```

Axis Object Methods

The following methods are provided by the Axis Object.

CONFIGUREAXIS

The ***configureAxis*** method can be invoked asynchronously with respect to the servo loop calculations. Thus it can be called from the RTUM or the NRTUM. Normally it will be called from the NRTUM via the Pedestal object as the result of receiving new configuration from the maintenance computer. It is called with an ***axisConfiguration*** structure which holds the new information. All this method does is copy the ***axisConfiguration*** structure to an internal memory area for later used by the ***updateServo*** method. In this way there are no conflicts due to the asynchronous nature of the updates.

UPDATESERVO

The updateServo method of the axis object is where the bulk of the servo loop processing occurs. The following processing occurs in the updateServo method of the axis object.

- Check to see if there is new configuration data. If so, the data is used to update the internal configuration of the axis. This allows configuration data to be sent to the object in an asynchronous manner similar to the pedestal object.
- Convert the raw data received from the RTKM via the pedestal object to real-world coordinates. This data includes the current position of the axis, the current velocity of the axis and the torques reported by each drive.

- Determine the correct mode command for the axis. The possible mode commands and their associated processing are as follows:
    - STOW: Set the desired position to the axis stow position and set the axis mode command to STOW_AT.
    - STOW_AT: If the position error is less than the stow tolerance and the axis velocity is less than the brake set velocity, change the axis mode to STANDBY.
    - CLOSED_POSITION: No special action.
    - OPEN_POSITION: No special action.
    - CLOSED_VELOCITY: No special action.
    - OPEN_VELOCITY: No special action.
- Check and update the control bits associated with the state of the axis. There are four possible states for an axis.
    - INACTIVE: Drives disabled and brakes set.
    - ENABLED: Brakes releasing, waiting on delay to enable drives.
    - ACTIVE: Brakes released, drives enabled.
    - DISABLED: Drives disabled, waiting on delay to set brakes.
- The processing of these modes is as follows:
    - If the bits that control drive power on are not set, clear the drive enable bits and the brake release bits. The current state of the axis is INACTIVE.
    - Processing continues based on the current state of the drive.
    - INACTIVE
        - IF the commanded state is STANDBY then the axis remains INACTIVE.
        - IF the commanded state is other than STANDBY then set a timer for the brake release and set the bits to release the brakes. Set the current axis state to ENABLED.
    - ACTIVE
        - If the command state is not STANDBY then the axis remains state remains ACTIVE.
        - If the command state is STANDBY then the desired velocity is set to zero. If the current velocity is less than the maximum brake set velocity the drives are disabled, a timer is set to the brake set delay time and the axis state is set to DISABLED. If the current velocity is not less than the maximum brake set velocity, the axis state remains ACTIVE.
    - ENABLED
        - If the commanded state is other than STANDBY then if the brake timer has expired enable the drives. If the bits that show that the drives have been enabled are set, set the current axis state to ACTIVE, else the current axis state remains ENABLED.
        - If the commanded state is standby clear the bits that release the brakes and set the drive state to INACTIVE.

- DISABLED
- If the command state is STANDBY and the brake timer has expired. Clear the bits that release the brakes and set the axis mode to INACTIVE.
- If the command state is other than STANDBY, enable the drives and set the axis mode to ACTIVE.

- If the current state of the axis is ACTIVE, perform the following processing based upon the current command mode for the axis.
- STOW_AT, CLOSED_POSITION
  - If the commanded position is outside of the range for the axis, limit the commanded position to a legal value.
  - Compute the position error by subtracting the current axis position from the desired axis position.
  - Pass the position error and the desired velocity (if available for RFF) to the selected position loop compensator .
  - Pass the resulting desired velocity to the velocity loop compensator instructing it to perform a closed (with feedback) computation.
  - Pass the desired acceleration to the Torque bias algorithm. This algorithm will fill in the necessary data to be sent to the RTKM.
- OPEN_POSITION
  - Use the desired position as the position error.
  - Pass the position error and the desired velocity (if available for RFF) to the selected position loop compensator.
  - Pass the resulting desired velocity to the velocity loop compensator instructing it to perform a closed (with feedback) computation.
  - Pass the desired acceleration to the Torque bias algorithm. This algorithm will fill in the necessary data to be sent to the RTKM.
- CLOSED_VELOCITY
  - Set the desired position to the current position.
  - Set the position error to zero.
  - Pass the desired velocity to the velocity loop compensator instructing it to perform a closed (with feedback) computation.
  - Pass the desired acceleration to the Torque bias algorithm. This algorithm will fill in the necessary data to be sent to the RTKM.
- CLOSED_VELOCITY
  - Set the desired position to the current position.
  - Set the position error to zero.
  - Pass the desired velocity to the velocity loop compensator instructing it to perform an open (without feedback) computation.
  - Pass the desired acceleration to the Torque bias algorithm. This algorithm will fill in the necessary data to be sent to the RTKM.

POSITION LOOP COMPENSATORS

The ACU supports four position loop compensators called Type1, LoGain, HiGain and RFF. These compensators may be defined to be any legal string of filters and limits via the axis configuration file, however, the purpose of the four compensators is as follows:

- Type 1: Non-Integrating Compensator used for prepositioning and large errors.
- LoGain: A low gain compensator used for low dynamic applications.
- HiGain: A high gain compensator used for high dynamics applications.
- RFF: A high gain compensator with rate feed forward applied.

The maintenance computer and pointing computer may switch between any of these compensators on a per axis basis at any time. The only time a compensator other than the selected compensator is used is when the LoGain, HiGain or RFF compensator is selected and the error exhibits a discontinuity which indicates a large move. In this case the Type 1 compensator is automatically selected until the error gets below TBD degrees. Whenever the compensator is switched, the new compensator is precharged with the old compensators output value to prevent any discontinuity in compensator output.

VELOCITY LOOP

The velocity loop is presented with the following inputs:

Desired Velocity
Desired Acceleration (if RFF is on)
RFF on/off
Loop Open/Closed

The processing of the velocity loop is as follows:

- Limit the desired velocity to the maximum allowed velocity for the axis.
- Limit the change in desired velocity to maintain acceleration within axis limits. This may lower the desired velocity.
- Perform the Variable Velocity Limit Algorithm (Paragraph Variable Velocity Limit).
- If calculation is to be closed loop, compute the velocity error which is equal to the limited desired velocity minus the current velocity. If the calculation is to be open loop, set the velocity error to the limited desired velocity.
- Compute the desired acceleration by putting the velocity error through the velocity compensator.

VARIABLE VELOCITY LIMIT

The ACU implements a variable velocity limit which guarantees that the antenna is moving at zero velocity when it reaches a position limit. This allows for controlled deceleleration of the axis by to the limit without velocity control steps. The geometry of the variable velocity limit is shown in Figure 13 and the algorithm is discussed below. Note that the region of variable velocity limit starts at MaxPosition – DecelDistance in the positive direction and at MinPosition + DecelDistance in the negative direction.

The variable velocity limit is implemented as follows:

```
 DecelTime = MaxVelocity / MaxAccel;
 dDecelDistance = MaxAccel * DecelTime * DecelTime / 2.0;
UpperLimit = MaxPosition - DecelDistance;
 LowerLimit = MinPosition + DecelDistance;
 if (CurrentPosition > UpperLimit) {
         Distance = MaxPosition - CurrentPosition;
         if (Distance <= 0.0) VelocityLimit = 0.0;
         else VelocityLimit =dMaxAccel * sqrt(2.0 * Distance / MaxAccel);
 }
 if (CurrentPosition < LowerLimit) {
         Distance = CurrentPosition - MinPosition;
         if (Distance <= 0.0) VelocityLimit = 0.0;
         else VelocityLimit = -dMaxAccel * sqrt(2.0 * Distance / MaxAccel);
 }
```

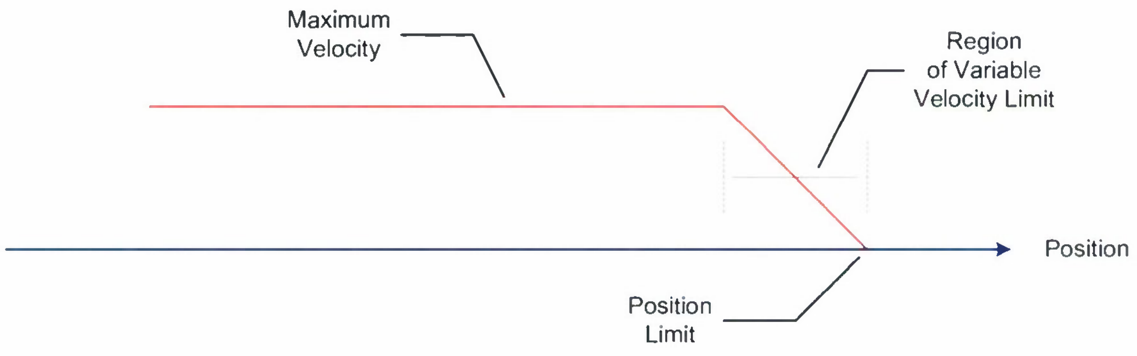


Figure 13. Variable velocity limit geometry.

TORQUE BIAS/TORQUE SHARING

The torque bias/torque sharing algorithm is called after the velocity loop determines the desired acceleration that is to be applied to the axis. This acceleration is multiplied by the axis inertia and divided by the axis gear ratio to provide the desired torque. This torque is then limited to the maximum allowed torque for the axis. The baseline torque sharing algorithm is to divide the total desired torque by the number of drives per axis. This number is then multiplied by two to give the total torque desired from each motor pair. This baseline algorithm shall be implemented in a modular fashion to allow for simple incorporation of more complex algorithms.

The desired torque per motor pair is passed to the torque bias part of the algorithm. The ACU supports two different torque bias schemes, each with its own unique advantages and disadvantages. The torque bias scheme can be selected at run time as a configuration parameter. The first torque bias scheme is shown in Figure 14 (No Crossover). In this scheme, the drives are always biased so that there is never a backlash region. The drawback to this scheme is that the maximum torque that is available to deliver to the drive is equal to the maximum torque available from one motor minus half the torque bias. This scheme is not suited to high acceleration situations but is ideal for low acceleration applications and during testing when positive contact of both gears must be assured.

The second torque bias scheme is shown in Figure 15 (Crossover). In this scheme, the motors are opposed only in the region of low torque requirements. This is the case for most tracking applications. When more torque is required, the opposing motor crosses over to help thus providing the total torque available from each motor to the axis. The advantage of this algorithm is that all of both motors torque is available to accelerate the axis. The down side is that there is an instant of crossover that can add instability to the tracking loop.

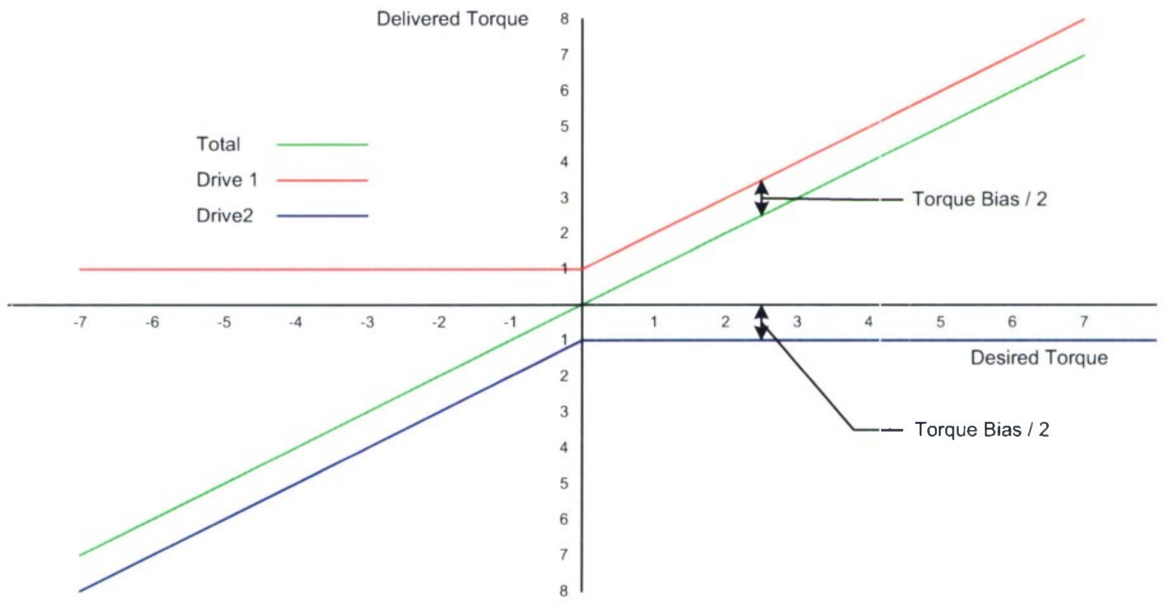


Figure 14. Torque bias with no crossover.

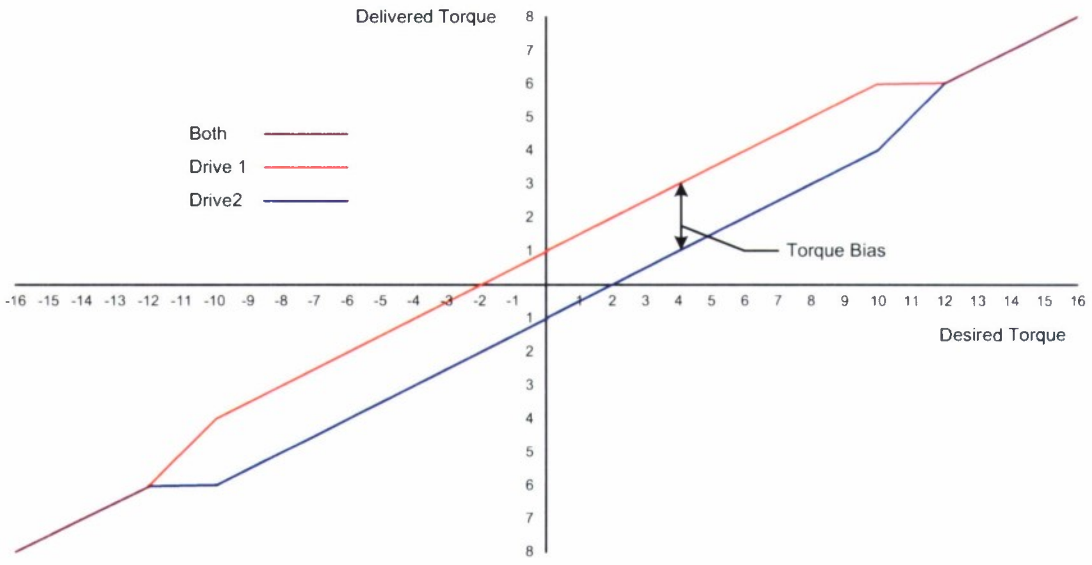


Figure 15. Torque bias with crossover.

REPORTAXISSTATUS

The reportAxisStatus method may be called at any time to determine the status of an axis. This method returns the status of all devices and internal computations associated with the requested axis. The units reported by this method are compatible with the maintenance computer and pointing computer Interface Control Documents, they are not in units that are necessarily compatible with the SERCOS drives. This method is implemented by the Pedestal Object by calling the requested axis object's reportAxisStatus method.

Compensator Object

The generic compensator object is shown in Figure 16 below. It consists of an input limit, an input rate limit, a gain, a string of filters which may be in series or parallel or both, an output rate limit and output limit. Provision is also made for the target position input which is differentiated and filtered to provide a rate feed forward input. A sample filter string is shown in Figure 17. Not that the filter string can consist of any number of series and parallel filter elements. At a minimum, the following elements shall be supported:

- Derivative
- Limiting Integrator (no windup)
- Phase Lead
- Phase lag

- Gain
- Rate Limit
- Value Limit

The implementation of the compensator is such that other filter elements may be easily programmed and added. The configuration of the compensator may be changed at any time by passing in a string or filename which contains the definition of the limits and filter string.

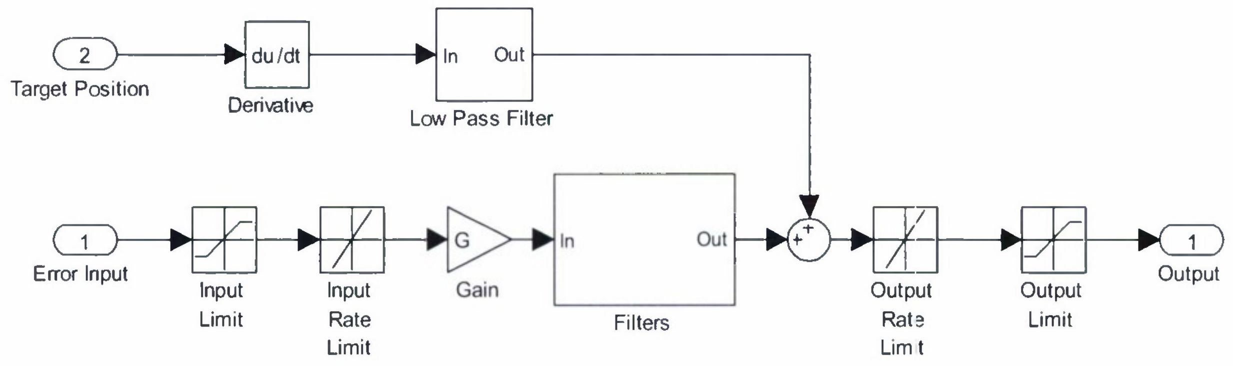


Figure 16. Generic compensator.

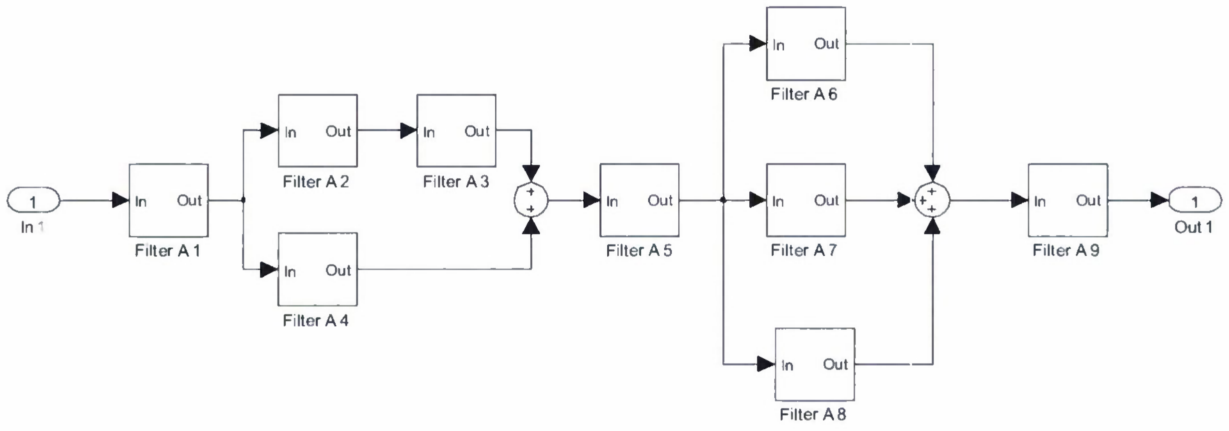


Figure 17. Sample filter string.

Configuration Object

The C onfiguration O bject s implifies t he f unctions of r eading a nd w riting configuration files. A Configuration Object is instantiated with a filenamc which is read to sct thc initial values for all the configuration items. Configuration files may contain lincs of the following forms

Blank Lines: Blank lines are ignored.

Section Lines: Section lines start with a lcft brackct and cnd with a right brackct. Thc name of the section is containcd within thc brackcts (cxample: [AZIMUTH AXIS]. A section line denotes that the following definitions arc within the named section until another section line is encountered. Leading and trailing white spacc is ignorcd. Section names are case sensitive.

Definition Lines: Definition lines are of the form XXXX=YYYY. Thcy pair thc valuc YYYY with the name XXXX. YYYY may be a string, integer, float or doublc. Strings are not enclosed in quote marks. Leading and trailing whitespace is ignorcd as is whitespace surrounding the equals sign.

Comments: Comments start with the // character sequence. This sequencc may start at any point in any line. All characters following a // arc ignored. Notc therc is no cscapc sequence to ignore //.

Once instantiatcd, the Configuration Object supports the mcthods shown bclow. Note that the g etXXXValue an d s etValue m ethods m ay b e cal led at an y t imc. Any d ata m odified b y a setValue call (including new section/entry pairs) is held internal to the objcct until a call is madc to the writeConfiguration method.

```
      int getIntValue(const string section,           //Section of entry
                      const string entry,             //Entry name
                      const int defaultValue = 1      //Default value
                     );

      long getLongValue(const string section,               //Section of
entry
                        const string entry,                 //Entry name
                        const long defaultValue = 1         //Default value
                       );

      double getDoubleValue(const string section,           //Section of
entry
                            const string entry,             //Entry name
                            const double defaultValue = 1   //Default value
                       );
      bool getBoolValue(const string section,               //Section of
entry
                        const string entry,                 //Entry name
                        const bool defaultValue = true      //Default value
                       );

      string getStringValue(const string section,     //Section of entry
                            const string entry,       //Entry name
                            const string defaultValue = "Default" //Default
                           );

      void setValue(const string section,       //Section of entry
                    const string entry,         //Entry name
                    const int value             //value to set
                   );

      void setValue(const string section,       //Section of entry
                    const string entry,         //Entry name
                    const long value            //value to set
                         );

      void setValue(const string section,       //Section of entry
                    const string entry,         //Entry name
                    const double value          //value to set
                    );

      void setValue(const string section,       //Section of entry
                    const string entry,         //Entry name
                    const string value          //value to set
                         );
```

void writeConfiguration();

RTUM Initialization

Upon s tartup, t he A ntenna C ontrol U nit a pplication i nstantiates a P edestal obj ect a nd spawns a thread called servoLoop with a pointer to the newly created Pedestal object. This thread creates the mailbox for communication to the Non-Real Time User Space Module (NRTUM), connects to the RTKM Real-Time Input and Output FIFOs, calls RTAI support routines to place itself in real time mode for the fastest possible response times, and then blocks waiting for input from the RTKM via the Real-Time Input FIFO. Normal continuous operation of the RTUM occurs each time a message is received on the Real-Time FIFO.

RTUM Servo Loop Processing

The bulk of the processing performed by the ACU occurs in the RTUM servo loop. This processing starts upon reception of a message from the RTKM on the Real Time Input FIFO. Each time a message is received on this FIFO (nominally once every millisecond) the message is pulled from the FIFO in its entirety. The format of this message is discussed earlier in this section and repeated below for convenience.

```
typedef struct RTKMOutputData {
        int nanosecond;                         //Time of validity
        int second;
        int minute;
        int hour;
        int jday;
        int year;
        int AzEncoderCounts;                    //Binary az encoder value
        int ElEncoder Counts[2];                //Two binary el encoder values
        int MotorVelocity[12];                  //Velocity off all motors
        int motorTorque[12];                    //Reported torque from all motors
        int motorVolts[12];                     //Bus Voltage at each motor
        int motorWatts[12];                     //Power consumption of each motor
        int motorTemp[12];                      //Temperature of each motor
        int ampTemp[12];                        //Temperature of each amplifier
        int shaftPosition[6];                   //Position of each hexapod shaft
        unsigned char inputs[DIGITAL_INPUT_CNT];        //State of discrete ins
        int     unsigned char outputs[DIGITAL_OUTPUT_CNT]; //State of
discrete outs
        unsigned char Pseudo[PSEUDO_INPUT_BYTES]; //State of generated
bits
} RTKMOutputData_t;
```

The next step is to check the mail box connected to the NRTUM for any maintenance or pointing computer messages. Messages will be received from the pointing computer at a rate of 100 times per second. Messages will be received from the maintenance computer asynchronously. If a message is present it is processed based upon its contained command mode as follows:

Standby Mode: Desired position need not be entered—it will be ignored.
Set desired velocity for each axis to zero.
Set current mode for each axis to STANDBY.
Continue with processing as if no message was received.

Slew Mode: Desired position need not be entered—it will be ignored.
Set desired velocity for each axis to commanded velocity.
Set current mode to SLEW.
Continue with processing as if no message was received.

Stow 1:	Desired position need not be entered—it will be ignored. Desired velocity need not be entered—it will be ignored. Set current mode to STOW1. Continue with processing as if no message was received.
Stow 2:	Desired position need not be entered—it will be ignored. Desired velocity need not be entered—it will be ignored. Set current mode to STOW2. Continue with processing as if no message was received.
Point:	Pick up the state vector from the message. Use the desired acceleration for each axis from this message, the desired acceleration for each axis from the last message, and the difference between the two messages' State Vector Time of Validity to compute a desired jerk for each axis. Place the entire state vector in a circular buffer. This circular buffer will be capable of holding at least 10 state vector entries to allow for predictive pointing. The state vector contains the following information: Desired Azimuth Position (***dAZp***) Desired Azimuth Velocity (***dAZv***) Desired Azimuth Acceleration (***dAZa***) Desired Azimuth Jerk (***dAZj***) Desired Elevation Position (***dELp***) Desired Elevation Velocity (***dELv***) Desired Elevation Acceleration (***dELa***) Desired Elevation Velocity (***dELv***) Desired Elevation Acceleration (***dELa***) Desired Elevation Jerk (***dELj***) State vector time of validity (***svTov***) Set current mode to POINT. Continue with processing as if no message was received.

If no message is received or after the message that was received has been processed as discussed above, the following processing occurs.

- If the current mode is POINT calculate the current desired position and velocity of each axis utilizing a cubic spline interpolation or extrapolation. The cubic spline algorithm is described in paragraph XXXX. If the current mode is not POINT, the desired position and the desired velocity were determined when the command message was received.
- Pass the following data to the ***updateServo*** method of the Pedestal Object for processing.

Desired Position for azimuth and elevation
Desired Velocity for azimuth and elevation
Desired Acceleration for azimuth and elevation
Desired mode of operation for azimuth and elevation
The message received from the RTKM which contains the status of all I/O
A pointer to the structure that will hold the data destined for the RTKM (RTKMInputData)

- The updateServo of the pedestal object will update the data in the RTKMInputData structure (with help from each axis object). The data in this structure is in the units used by the SERCOS interface card and the SERCOS drives. The RTKM data structure is described in the section Real-Time User-Space Module and repeated below for convenience.

```
typedef struct RTKMInputData {
        int pseudoDigitalOutputs;
        int desiredVelocity[12];
        int desiredTorque[12];
        int desiredPosition[6];
        unsigned char outputs[DIGITAL_OUTPUT_CNT]; //State of discrete outs
RTKMInputData_t;
```

- The RTUM will the request the current pedestal status and status of each axis from the pedestal object. This data will be in standard units (rad, etc). This data will be formatted into a status message that is compatible with the pointing computer and the maintenance computer and place on the NRTUM via the output mailbox. The NRTKM will pass the status on to the pointing computer and/or maintenance computer if necessary.

Once the above processing is complete, the RTUM has finished all functions necessary to handle the RTKM FIFO input. The RTUM then again blocks on waiting for a message from the RTKM on the Real Time Input FIFO.

Cubic Spline Algorithm

The data on the circular buffer containing the state vectors received from the pointing computer and augmented by the jerk calculation is processed each time through the servo loop to compute the desired position, velocity and acceleration for each input to the ***updateServo*** method of the Pedestal Object. Use of the cubic spline guarantees continuous derivatives and smooth interpolation and extrapolation of desired position. The cubic spline algorithm is detailed below. Note that the algorithm must be performed for each axis (azimuth and elevation) individually.

- Get the current seconds since midnight from the RTKM structure, call it ***ssm***.
- Search through the circular buffer of received state vectors for until a state vector is found with ***svTov > ssm*** or the last state vector is found.
- Call the state vector which was found in the previous step ***sv1***.
- Call ***sv0*** the state vector that was received immediately previous to ***sv1***.
- If ***ssm - sv1.svTov > 0*** we have to extrapolate. Compute the following.
    - ***deltaT = ssm - sv1.svTov***
    - ***acc = sv1.accel + deltaT * sv1.jerk***
    - ***V1 = sv1.vel + deltaT * acc***
    - ***P1 = sv1.pos + deltaT * V1***
    - ***t = 1.0***
- If ***ssm - sv1.svTov ≤ 0*** we have to interpolate. Compute the following.
    - ***deltaT = sv1.svTov - sv0.svTov***
    - ***P1 = sv1.pos***
    - ***V1 = sv1.vel***
    - ***t = ssm - sv0.svTov/DeltaT***
- Complete the computation as follows.
    - ***P0 = sv0.pos***
    - ***V0 = sv0.vel * deltaT***
    - ***d = P0***
    - ***c = V0***
    - ***b = 3.0 * P1 - V1 - 2.0 * V0 - 3.0 * P0***
    - ***a = P1 - b - B0 - P0***
    - ***DesiredPosition = a * t * t * t + b * t * t _ c * t + d***
    - ***DesiredVelocity = V1/ deltaT***

Non-Real Time User Space Module (NRTUM) The Non-Real Time User Space Module is responsible for providing the interface from the ACU to the Pointing Computer (both radar and astronomy) and the Maintenance Computer, Hence it provides the bridge between the Real Time User Space Module (RTUM) and the sockets interface which implements the TC/IP interface the Maintenance computer and the UDP interface to the Pointing Computer.

NRTUM Initialization

After instantiating the Pedestal object and spawning the RTUM thread, the ACU continues with initialization of the NRTUM. This initialization consists of using ACU configuration file (acu.cfg) to define the port for communication to the Maintenance computer (nominally 4001) and the port for communication to the pointing computer (nominally 4003). At this point a mailbox is set up for communication with the RTUM. This mailbox utilizes code provided by RTAI to insure that communication with the RTUM does not interfere with its real-time nature. Finally, the Maintenance computer and Pointing Computer ports are initialized and the NRTUM waits on an input event from the MC, PC or RTUM utilizing the select function provided by LINUX.

NRTUM Pointing Computer Message Handling

Pointing Computer messages are simply stripped of the communications wrapper, time tagged and placed on the RTUM mailbox for pickup by the servo loop during its next processing cycle. The only other processing performed by the NRTUM is to note if a specific status message is requested by the message. If so, the NRTUM builds the requested status message and forwards it to the pointing computer.

NRTUM Maintenance Computer Message Handling

Messages from the Maintenance Computer fall into three categories. First, the Maintenance Computer can send messages to the ACU which request pedestal motion. These messages are formatted to emulate a pointing computer message and sent to the RTUM via the communications mailbox. The second category of message consist of configuration messages. These messages are stripped of their communications wrapper and sent to the RTUM for parsing and implementation within the pedestal and axis objects. The last form of message is an instrumentation message. This message is also forwarded to the RTUM for processing.

NRTUM Mailbox (from RTUM) Message Handling

Status messages arrive at the RTUM mailbox at the servo loop rate. The NRTUM decimates these messages to the status message rate specified in the ACU configuration file. At the requested rate, the NRTUM packages a status message and forwards it to the Pointing Computer or the Maintenance computer.

5. MAINTENANCE COMPUTER

5.1 MAINTENANCE COMPUTER HARDWARE

5.1.1 Maintenance Computer

The Maintenance Computer chassis is a standard 19" rack mount commercial-off-the-shelf (COTS) PC. The model chosen is the Superlogics SL-1U-D31PR-DB with a 2.8 GHz Intel Core 2 Quad and 4 GB of RAM. It also has an Nvidia GeForce 9400 GT video card with 512 MB of RAM and is capable of dual headed display. Two 20" Dell LCD flat panel monitors were procured for each maintenance computer.

The maintenance computers are running Fedora Core 9 as the operating system and are configured as DHCP servers and are used to host the ACU operating system and control system software.

5.2 MAINTENANCE COMPUTER SOFTWARE

The Maintenance Computer application is loosely based on the software that was delivered to support the Water Vapor Radiometer (WVR) and the XTR-1 antenna pedestals. The purpose of the Maintenance Computer software is to provide the following functions:

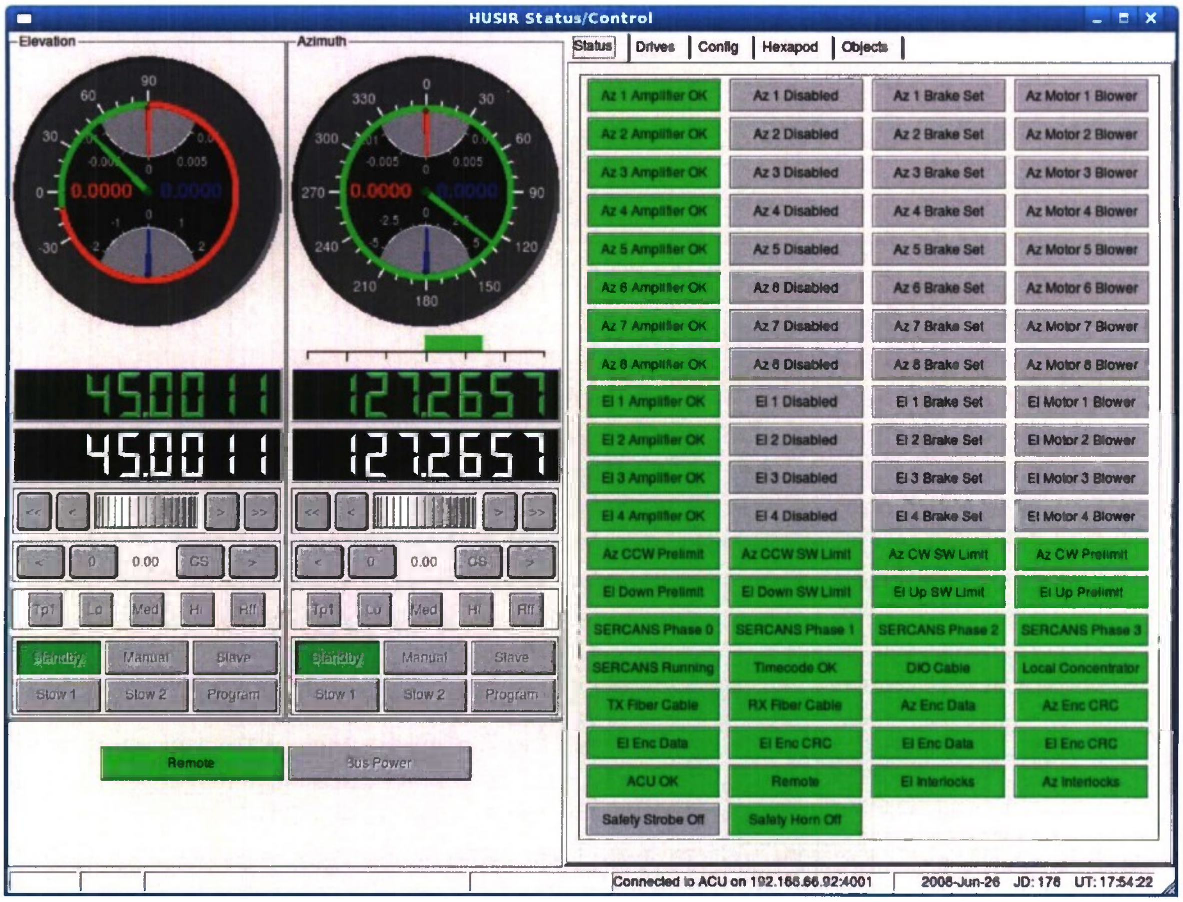
- Allow for local control of the pedestal via an easy to use Graphical User Interface (GUI). This allows the antenna to be moved without the ACS or RTP equipment during periods of testing or maintenance.
- Provide graphical display of all pertinent ACU internal functions allowing users to evaluate ACU and pedestal performance.
- Provide a method to save data describing antenna operation. This data may be examined offline to uncover any performance issues or inconsistencies
- Provide a method to modify configuration parameters associated with the position loop, velocity loop, torque sharing and torque bias algorithms. This will facilitate initial commissioning of the antenna system as will as the ability to experiment with different performance parameters.

The Maintenance Computer software will be developed in C++ utilizing tools available under the GNU Public License (GPL). Specifically the starting Linux distribution will be Fedora 9 and the gcc version shall be Version 4.1.2. The graphical user interface shall utilize the widget set supplied by QT version 3 and supplemented with QWT Version 4.2.0. Newer visions of the abovementioned development tools may be used if desired.

5.2.1 Functional Overview

The Maintenance Computer software presents two modeless display windows to the user of the application. One of these display windows (Status/Control Window) will show pedestal status and control information, hexapod status and control information, configuration options and the ability to track celestial objects. The other display window (Instrumentation Window) allows the operator to graphically view internal calculations as they are performed in the ACU, record this data to media for later processing, and stimulate the antenna pedestal with various waveforms for testing, fault isolation and performance verification. Multiple independent instrumentation windows may be created.

The Maintenance Computer application interacts with the ACU via a sockets interface (TCP/IP) in accordance with the *MCS-ACU Interface Control Document*.



Connected to ACU on 192.168.66.92:4001 | 2008-Jun-26 JD: 178 UT: 17:54:22

Figure 18. Status/Control window–remote mode.

5.2.2 Status/Control Window

The Status/Control window is the portal which provides complete information concerning the status of the antenna pedestal and hexapod as will as the mechanisms necessary to control the motion of the pedestal and hexapod. Figure 18 shows an instance of the status/control window. The left side of the window displays the current antenna position and provides the control widgets for the pedestal. Each dial shows the current position (green needle and digits), desired position (white needle and digits), current velocity (blue needle and digits) and current error (red needle and digits). The azimuth axis display also provides a cable wrap indicator (the green bar graph below the dial) which indicates where the azimuth axis is in relationship to the total allowed travel.

The right side of the display is a tabbed dialog which can show one of the following displays:

Status Display: State of each of the discrete input points monitored by the ACU

Drives Tab: Detailed information concerning each of the pedestal drives

Config Tab: Displays and allows modification of the pedestal configuration

Hexapod Tab: Displays the status of the Hexapod and allows for manual control of same

Objects Tab: Provides a mechanism to command the pedestal to track a celestial object

Remote Mode In Figure 18, the ACU is in remote mode as designated by the green button in the lower left hand side of the display. This is the normal mode of the ACU. While in remote mode, the pointing computer is the controlling entity, not the Maintenance computer. Note that all the controls associated with the pedestal are grayed out signifying that they are disabled. It is not possible to change the state of the pedestal from the Maintenance computer when the remote indicator is green.

Local Mode Clicking on the remote button will turn it blue and indicate that the ACU is now in Local control mode as shown in Figure 19. Note that the controls for the Bus Power control button, antenna mode control box (Manual, Stow, etc.) and servo bandwidth controls (Lo, Med, etc.) are now enabled and may be used to control the antenna. When in local mode, the ACU does not respond to any commands from the pointing computer. This ensures that the antenna pedestal can only be controlled from one physical location at a time. The provides a measure of safety by not allowing surprise motion from a remote location. Status messages are streamed to both the maintenance computer and the pointing computer regardless of which machine is the controlling entity. This allows for monitoring of pedestal motion from either interface at any time.

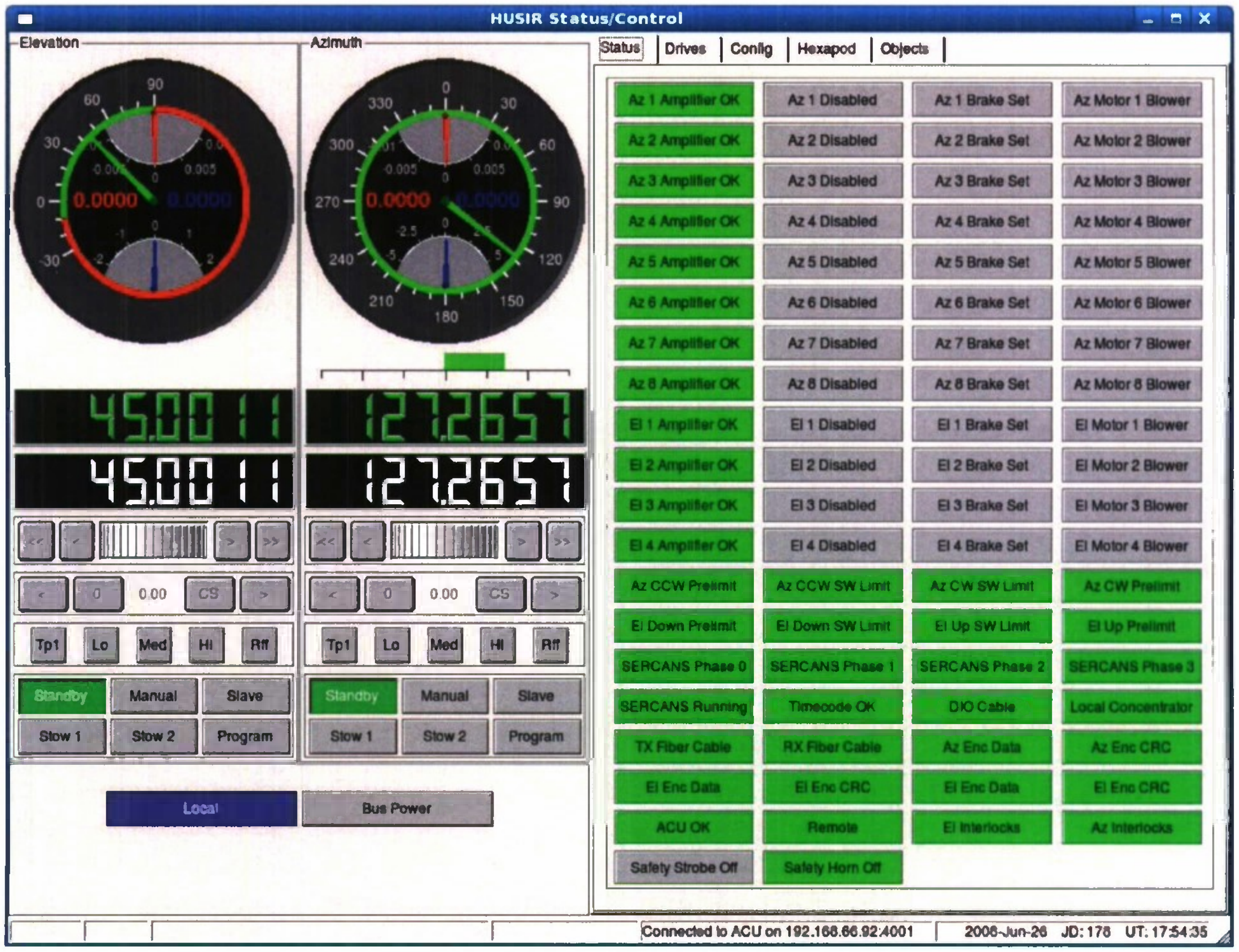
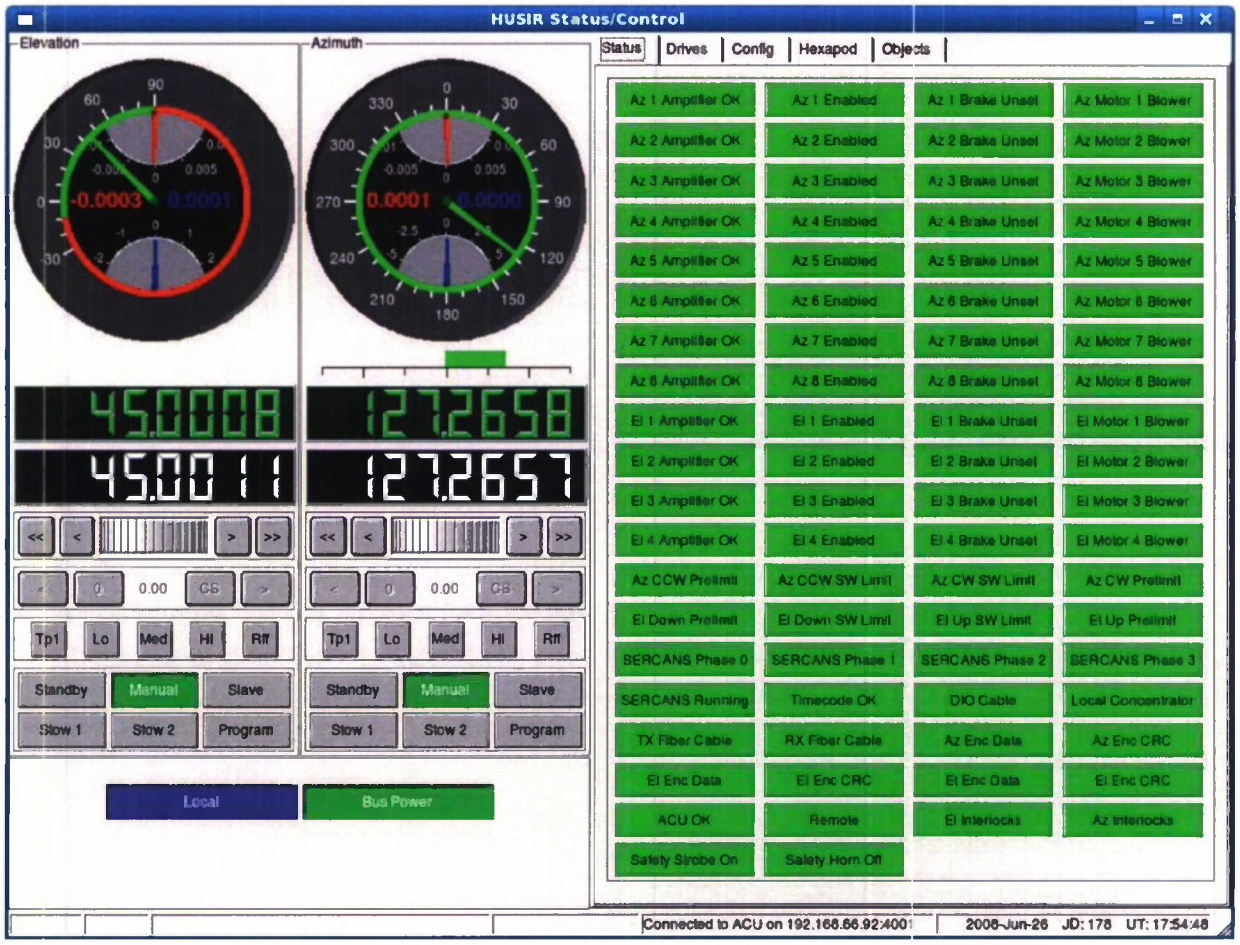


Figure 19. Status/Control window–local mode.

Manual Mode Figure 20 shows the control/status after the local operator has clicked on the Manual mode buttons for each axis. Note that the Bus Power is now on, all of the drives are enabled, all brakes are released and the motor blowers have been turned on. Note also that the safety strobe, which indicates that the pedestal can move at any minute is energized. When the antenna i s co mmanded from a s tandby s tate t o an en ergized s tate, t he A CU p erforms t he following actions:

1. Illuminate the Safety Strobe.
2. Sound the Safety Horn.
3. Wait 10 seconds (this time is configurable).
4. Turn off the Safety Horn, leave the Safety Strobe illuminated.
5. Enable the drives.

Note also that the manual control buttons (immediately below the white desired position digits) are now enabled. Although it cannot be seen in the figure, the axis dial and the desired position digits are also now enabled. The operator will realize this when the cursor changes shape when moved over either of these widgets.



Connected to ACU on 192.168.66.92:4001 | 2008-Jun-26 JD: 178 UT: 17:54:48

Figure 20. Status/Control window–manual mode.

When in Manual mode, the operator may change the antenna position in any one of the following ways:

- Clicking on one of the <<, <, >, >> buttons. This will cause the selected axis to slew at the slow rate of 0.1°/second in the case of the < and > button or at the faster rate of

2^{o}/second in the case of the >> or << buttons. The direction of the motion is indicated by the direction of the arrows on the button.

- Right clicking on the white desired position digits. This will allow the operator to type in a new desired position.
- Moving the cursor over the white desired position digits. While the cursor is over the upper half of a digit an up arrow will be displayed which will cause that digit to increase (with appropriate carry to next digit) each time the left mouse button is pressed. While the cursor is over the lower half of a digit a down arrow will be displayed which will cause that digit to decrease (with appropriate borrow from the next digit) each time the left mouse button is pressed.
- Dragging the white needle in the axis dial display to the desired position.

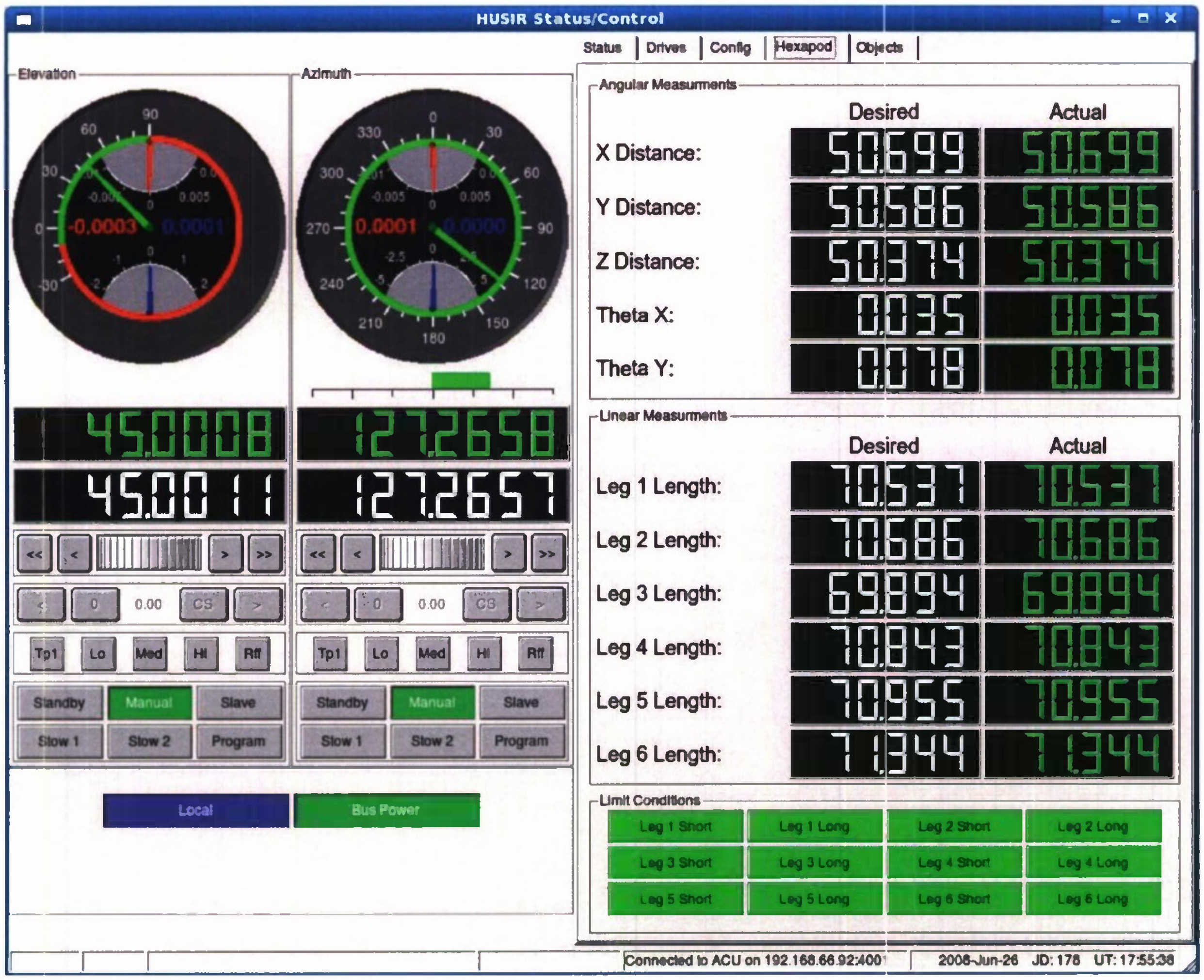
Hexapod Control Figure 21 shows the Hexapod Status/Control display. This display is obtained by clicking on the hexapod tab in the upper left hand side of the Status/Control Display. Note that when this tab is revealed, 6 new control buttons appear in the lower left-hand portion of the display. These buttons allow the operator to individually enable and disable each leg of the Hexapod assembly. In Figure 21, the drives for legs 2, 4 and 5 are enabled while the drives for legs 1, 3 and 6 are disabled.

The left right hand side of the display shows the current and desired positions of the Hexapod assembly in two different coordinate systems. The top coordinate system (Angular Measurement) shows the position of the Hexapod as ***X***, ***Y***, ***Z***, ***θX*** and ***θY***. This coordinate system represents the position and orientation of the Hexapod with respect to the reflector. The bottom coordinate system (Linear Measurements) simply shows the length of each leg that supports the Hexapod. The very bottom of the right hand display shows the status of the limits of motion for each of the legs. Each leg of the Hexapod has a limit to prevent motion past the shortest and longest possible throw.

Control of the Hexapod is similar to the control of the pedestal itself. Any of the 11 white desired position fields associated with the Hexapod can be modified in two different ways as follows:

- Right clicking on the white desired position digits. This will allow the operator to type in a new desired position.
- Moving the cursor over the white desired position digits. While the cursor is over the upper half of a digit an up arrow will be displayed which will cause that digit to increase (with appropriate carry to next digit) each time the left mouse button is pressed. While the cursor is over the lower half of a digit a down arrow will be displayed which will cause that digit to decrease (with appropriate borrow from the next digit) each time the left mouse button is pressed.

Motion effected in one eoordinate system will update the neeessary desired positions in the other eoordinate system. All hexapod motion eommands will be error eheeked for validity before execution.



	Elevation	Azimuth
	45.0008	127.2658
	45.0011	127.2657

Angular Measurments

	Desired	Actual
X Distance:	50.699	50.699
Y Distance:	50.586	50.586
Z Distance:	50.374	50.374
Theta X:	0.035	0.035
Theta Y:	0.078	0.078

Linear Measurments

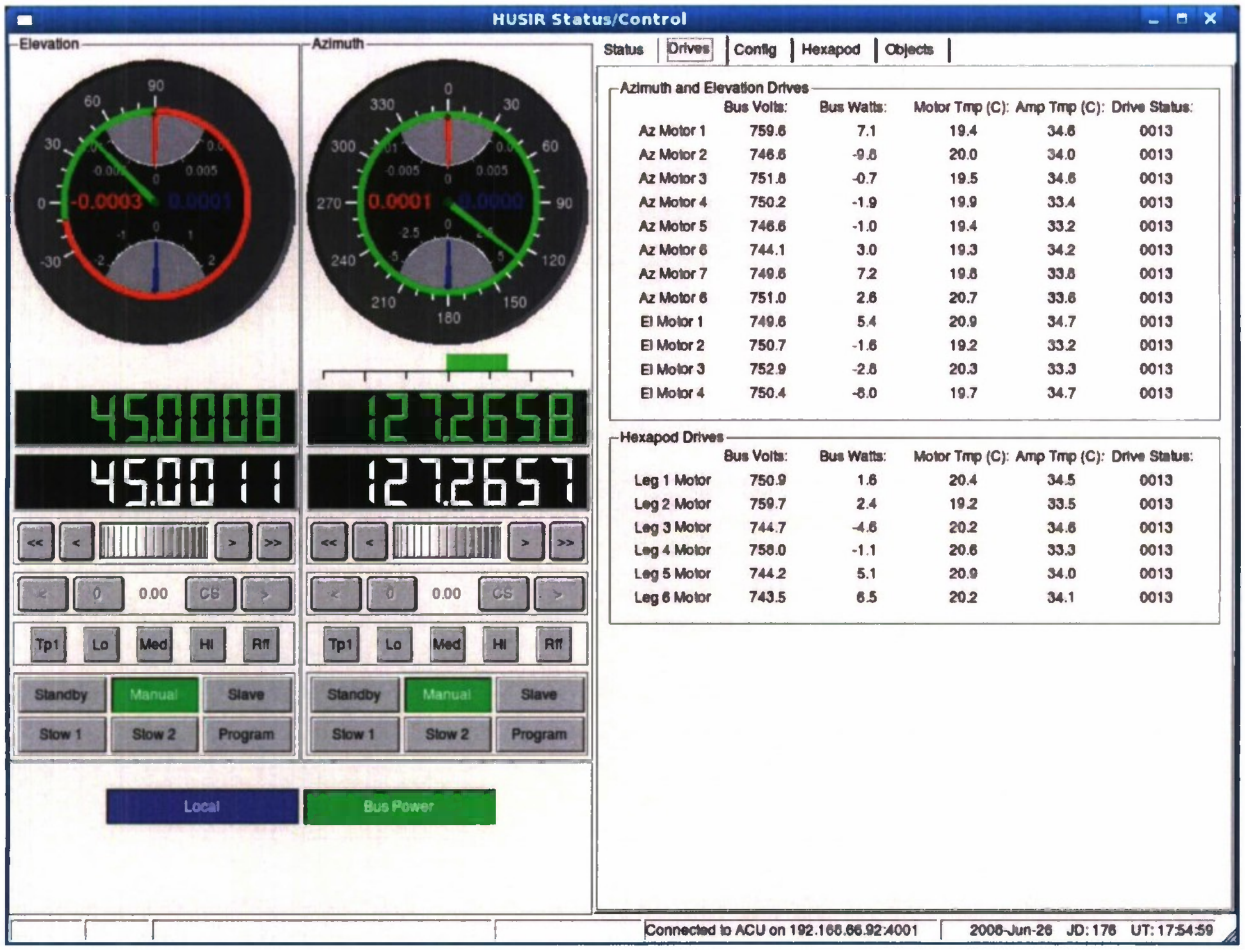
	Desired	Actual
Leg 1 Length:	70.537	70.537
Leg 2 Length:	70.686	70.686
Leg 3 Length:	69.894	69.894
Leg 4 Length:	70.843	70.843
Leg 5 Length:	70.955	70.955
Leg 6 Length:	71.344	71.344

Connected to ACU on 192.168.66.92:4001 2008-Jun-26 JD: 178 UT: 17:55:38

Figure 21. Status/Control window–hexapod control.

Drive Status The D rive S tatus di splay i s obt ained b y clieking on t he D rives t ab i n t he upper right hand eorner of the Status/Control display. This display, depieted in Figure 22, shows detailed s tatus i nformation f or each d rive as sociated w ith t he p edestal an d h exapod. This information is plaeed on its own tab due to the fact that the information eonsumes a significant amount o f s ereen r eal-estate a nd t he i nformation ne ed onl y be refereneed oe easionally f or possible fault isolation.

Configuration Tab Figure 23 shows the Status/Control display with the Configuration Tab selected. This tab reveals three further sub-tabs: one for the pedestal, one for the azimuth axis, and one for the elevation axis. Each of these sub-tabs contain configuration information that may be changed at any time. Figure 23 shows the pedestal sub-tab. The information on this tab affects the entire pedestal and will contain information including the location of the pedestal (to allow for propagation of ephemeris) and other parameters that may be used in a mount model. The ACU shall contain provisions to implement a mount model based upon parameters contained on this sub-tab of the Configuration Tab.



Azimuth and Elevation Drives

	Bus Volts:	Bus Watts:	Motor Tmp (C):	Amp Tmp (C):	Drive Status:
Az Motor 1	759.6	7.1	19.4	34.6	0013
Az Motor 2	746.6	-9.8	20.0	34.0	0013
Az Motor 3	751.8	-0.7	19.5	34.6	0013
Az Motor 4	750.2	-1.9	19.9	33.4	0013
Az Motor 5	746.6	-1.0	19.4	33.2	0013
Az Motor 6	744.1	3.0	19.3	34.2	0013
Az Motor 7	749.6	7.2	19.8	33.8	0013
Az Motor 6	751.0	2.6	20.7	33.6	0013
El Motor 1	749.6	5.4	20.9	34.7	0013
El Motor 2	750.7	-1.6	19.2	33.2	0013
El Motor 3	752.9	-2.8	20.3	33.3	0013
El Motor 4	750.4	-6.0	19.7	34.7	0013

Hexapod Drives

	Bus Volts:	Bus Watts:	Motor Tmp (C):	Amp Tmp (C):	Drive Status:
Leg 1 Motor	750.9	1.6	20.4	34.5	0013
Leg 2 Motor	759.7	2.4	19.2	33.5	0013
Leg 3 Motor	744.7	-4.6	20.2	34.6	0013
Leg 4 Motor	756.0	-1.1	20.6	33.3	0013
Leg 5 Motor	744.2	5.1	20.9	34.0	0013
Leg 6 Motor	743.5	6.5	20.2	34.1	0013

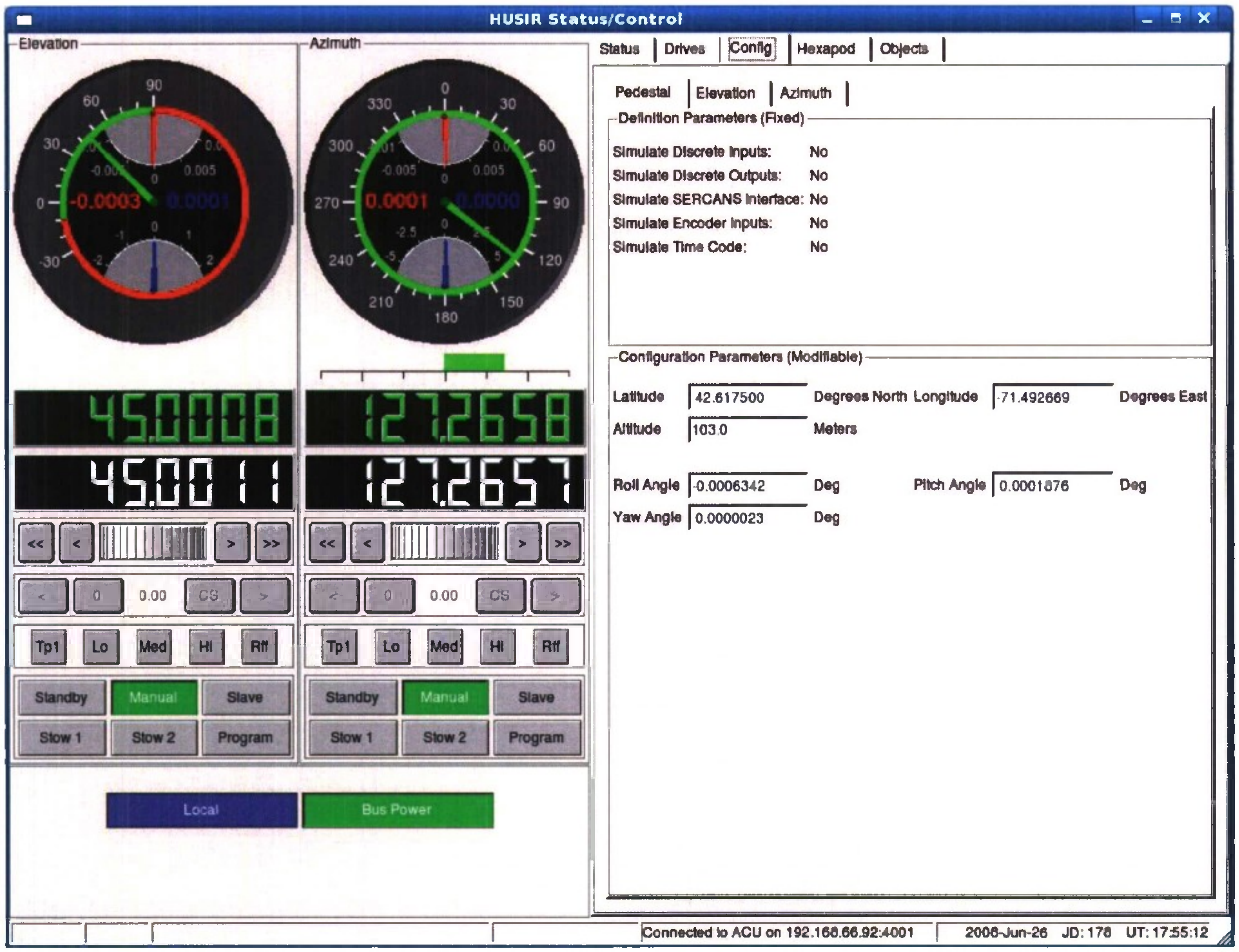
Connected to ACU on 192.168.66.92:4001 | 2008-Jun-26 JD: 178 UT: 17:54:59

Figure 22. Status/Control window–drive status display.

Figure 24 shows the sub-tab associated with the configuration of the Azimuth axis. This sub-tab is identical in format to the sub-tab for the Elevation Axis. Each of these tabs allow for the modification of axis specific parameters including the following:

- Upper and Lower Limits
- Maximum Velocity
- Maximum Acceleration
- Maximum Torque
- Torque Bias Parameters
- Compensator Configuration

Configuration parameters can be modified on these sub-tabs and sent down to the ACU at any time. The ACU is designed to allow for configuration on the fly. Any configuration parameters sent to the ACU shall be utilized immediately as well as retained and automatically loaded the time the ACU boots up.



Definition Parameters (Fixed)

Simulate Discrete Inputs:	No
Simulate Discrete Outputs:	No
Simulate SERCANS Interface:	No
Simulate Encoder Inputs:	No
Simulate Time Code:	No

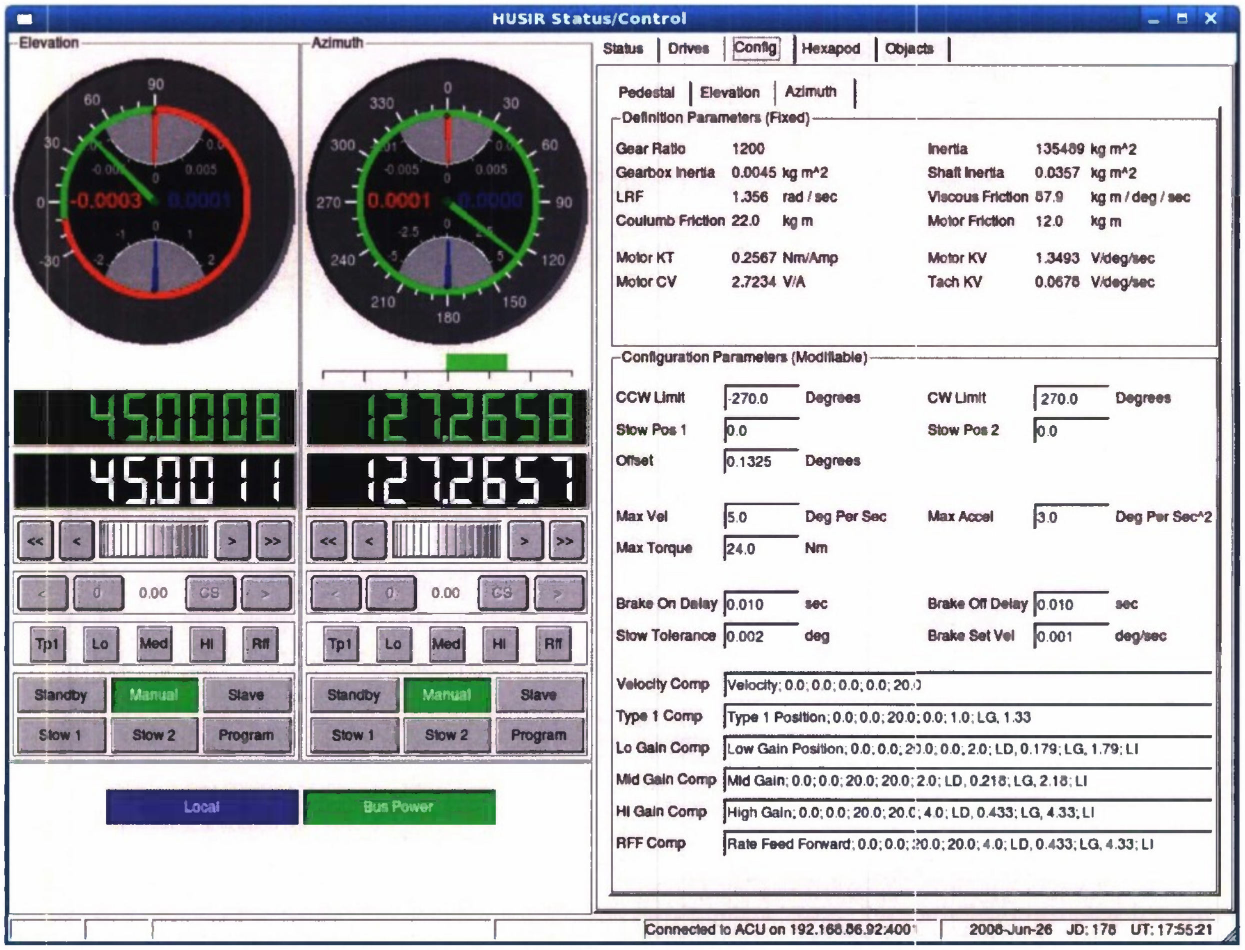
Configuration Parameters (Modifiable)

Latitude 42.617500 Degrees North Longitude -71.492669 Degrees East
Altitude 103.0 Meters

Roll Angle -0.0006342 Deg Pitch Angle 0.0001876 Deg
Yaw Angle 0.0000023 Deg

Connected to ACU on 192.168.66.92:4001 2006-Jun-26 JD: 178 UT: 17:55:12

Figure 23. Status/Control window–configuration tab (pedestal).



Status | Drives | Config | Hexapod | Objects

Pedestal | Elevation | Azimuth

Definition Parameters (Fixed)

Gear Ratio	1200		Inertia	135489	kg m^2
Gearbox Inertia	0.0045	kg m^2	Shaft Inertia	0.0357	kg m^2
LRF	1.356	rad / sec	Viscous Friction	57.9	kg m / deg / sec
Coulumb Friction	22.0	kg m	Motor Friction	12.0	kg m
Motor KT	0.2567	Nm/Amp	Motor KV	1.3493	V/deg/sec
Motor CV	2.7234	V/A	Tach KV	0.0678	V/deg/sec

Configuration Parameters (Modifiable)

CCW Limit	-270.0	Degrees	CW Limit	270.0	Degrees
Stow Pos 1	0.0		Stow Pos 2	0.0	
Offset	0.1325	Degrees			
Max Vel	5.0	Deg Per Sec	Max Accel	3.0	Deg Per Sec^2
Max Torque	24.0	Nm			
Brake On Delay	0.010	sec	Brake Off Delay	0.010	sec
Stow Tolerance	0.002	deg	Brake Set Vel	0.001	deg/sec

Velocity Comp	Velocity; 0.0; 0.0; 0.0; 0.0; 20.0
Type 1 Comp	Type 1 Position; 0.0; 0.0; 20.0; 0.0; 1.0; LG, 1.33
Lo Gain Comp	Low Gain Position; 0.0; 0.0; 20.0; 0.0; 2.0; LD, 0.179; LG, 1.79; LI
Mid Gain Comp	Mid Gain; 0.0; 0.0; 20.0; 20.0; 2.0; LD, 0.218; LG, 2.18; LI
Hi Gain Comp	High Gain; 0.0; 0.0; 20.0; 20.0; 4.0; LD, 0.433; LG, 4.33; LI
RFF Comp	Rate Feed Forward; 0.0; 0.0; 20.0; 20.0; 4.0; LD, 0.433; LG, 4.33; LI

Connected to ACU on 192.168.86.92:4001 | 2008-Jun-26 JD: 178 UT: 17:55:21

Figure 24. Status/Control window–configuration tab (azimuth axis).

Celestial Object Tabs The ACU has the ability to propagate ephemeris for the following objects based upon the position of the antenna pedestal and information specific to the type of ephemeris being propagated. The information on each of these objects is maintained in a MYSQL database local to the Maintenance Computer. Hence the Radar and Astronomy groups can maintain their own unique lists of test objects of interest.

- Satellites based upon Norad two line element sets via the SGP/SDP4 ephemeris propagation algorithm
- Stars based upon right ascension, declination and epoch time
- The sun and the moon (no other information necessary)
- Any designated position (specified azimuth and elevation)

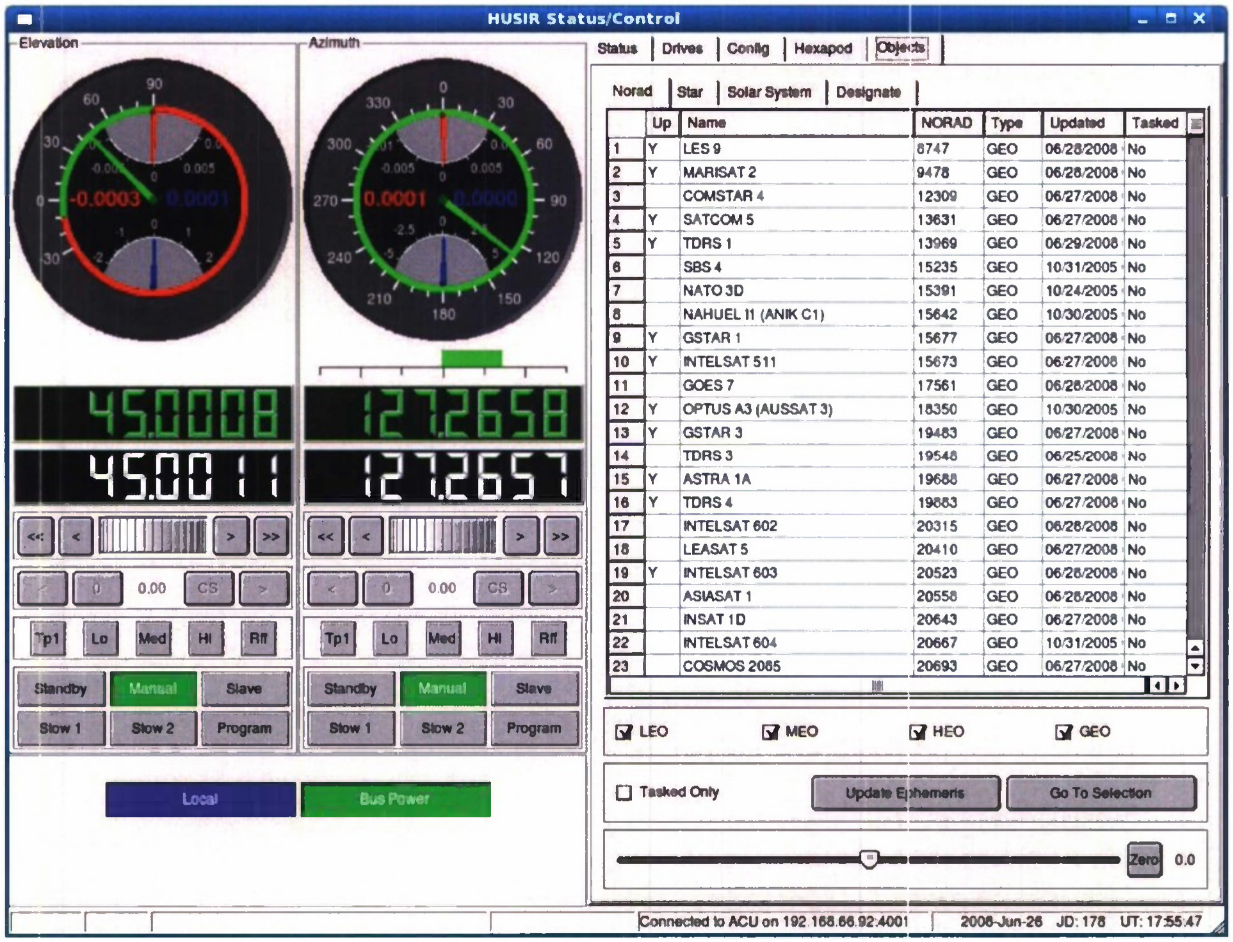
Figure 25 shows the Status/Control display with the Norad sub-tab of the Object Tab displayed. This display displays trackable satellites by Name, NORAD number and type (Low Earth Orbiter (LEO), Highly Elliptical Orbiter (HEO), Medium Earth Orbiter (MEO) and Geostationary (GEO). The list may be sorted by any one of these criteria. Additionally, each satellite can be designated as tasked which means that it is of special interest. This way all satellites that are not of interest can be maintained in the database for future use but eliminated from the display to reduce clutter by clicking on the Tasked Only checkbox as in Figure 26. The displayed list my be further culled by clicking on the LEO, MEO, HEO or GEO checkboxes at the bottom of the screen.

NORAD ephemeris may be updated from a text file of two line element sets on any media that is accessible by the Maintenance computer. The operator need only click on the Update Ephemeris button to start this process.

Figure 27 shows the Star Database tab, this tab allows for the operator to designate a star as the object for the antenna pedestal to point to. Figure 28 shows the Status/Control display when a star is selected for Program Track. Not that the Program Track button on the left hand side of the display is green and that the offset buttons (above the bandwidth select buttons) are now enabled. The operator may control the offset applied to the propagated ephemeris of any object as follows

- The wheel widget (immediately below the white desired position digits) may be used to dial in an offset in 0.001 degree steps for either axis.
- The < or > offset buttons may be clicked to decrement or increment the current offset by 0.001 degrees.
- The CS key be clicked to move the axis to a configurable cold sky offset (nominally 5 degrees for azimuth and 0 degrees for elevation to facilitate G/T measurements).
- The 0 key my be clicked in order to zero the offset in either axis.
- The time offset may be modified via the slider bar at the bottom of the Objects tab on the right hand side of the display.

Figure 29 and Figure 30 show the solar system and designate sub-tabs respectively. These solar system sub-tab allows the operator to select the sun or moon for ephemeris tracking. The designate sub-tab allows the operator to specify angles of interest for automatic pointing operations.



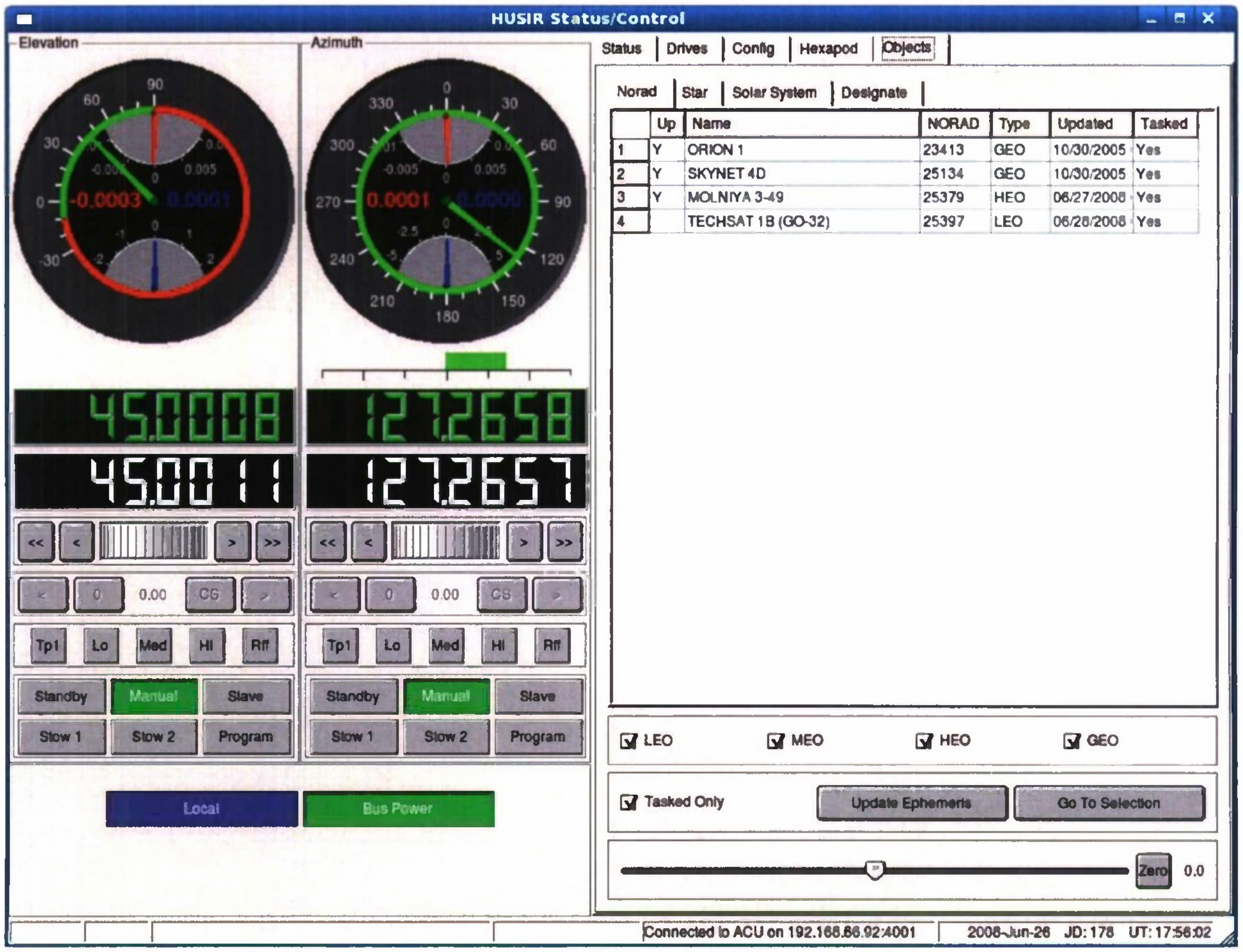
	Up	Name	NORAD	Type	Updated	Tasked
1	Y	LES 9	8747	GEO	06/28/2008	No
2	Y	MARISAT 2	9478	GEO	06/28/2008	No
3		COMSTAR 4	12309	GEO	06/27/2008	No
4	Y	SATCOM 5	13631	GEO	06/27/2008	No
5	Y	TDRS 1	13969	GEO	06/29/2008	No
6		SBS 4	15235	GEO	10/31/2005	No
7		NATO 3D	15391	GEO	10/24/2005	No
8		NAHUEL I1 (ANIK C1)	15642	GEO	10/30/2005	No
9	Y	GSTAR 1	15677	GEO	06/27/2008	No
10	Y	INTELSAT 511	15673	GEO	06/27/2008	No
11		GOES 7	17561	GEO	06/28/2008	No
12	Y	OPTUS A3 (AUSSAT 3)	18350	GEO	10/30/2005	No
13	Y	GSTAR 3	19483	GEO	06/27/2008	No
14		TDRS 3	19548	GEO	06/25/2008	No
15	Y	ASTRA 1A	19688	GEO	06/27/2008	No
16	Y	TDRS 4	19883	GEO	06/27/2008	No
17		INTELSAT 602	20315	GEO	06/28/2008	No
18		LEASAT 5	20410	GEO	06/27/2008	No
19	Y	INTELSAT 603	20523	GEO	06/28/2008	No
20		ASIASAT 1	20558	GEO	06/28/2008	No
21		INSAT 1D	20643	GEO	06/27/2008	No
22		INTELSAT 604	20667	GEO	10/31/2005	No
23		COSMOS 2085	20693	GEO	06/27/2008	No

☑ LEO ☑ MEO ☑ HEO ☑ GEO

☐ Tasked Only — Update Ephemeris — Go To Selection

Connected to ACU on 192.168.66.92:4001 | 2008-Jun-26 JD: 178 UT: 17:55:47

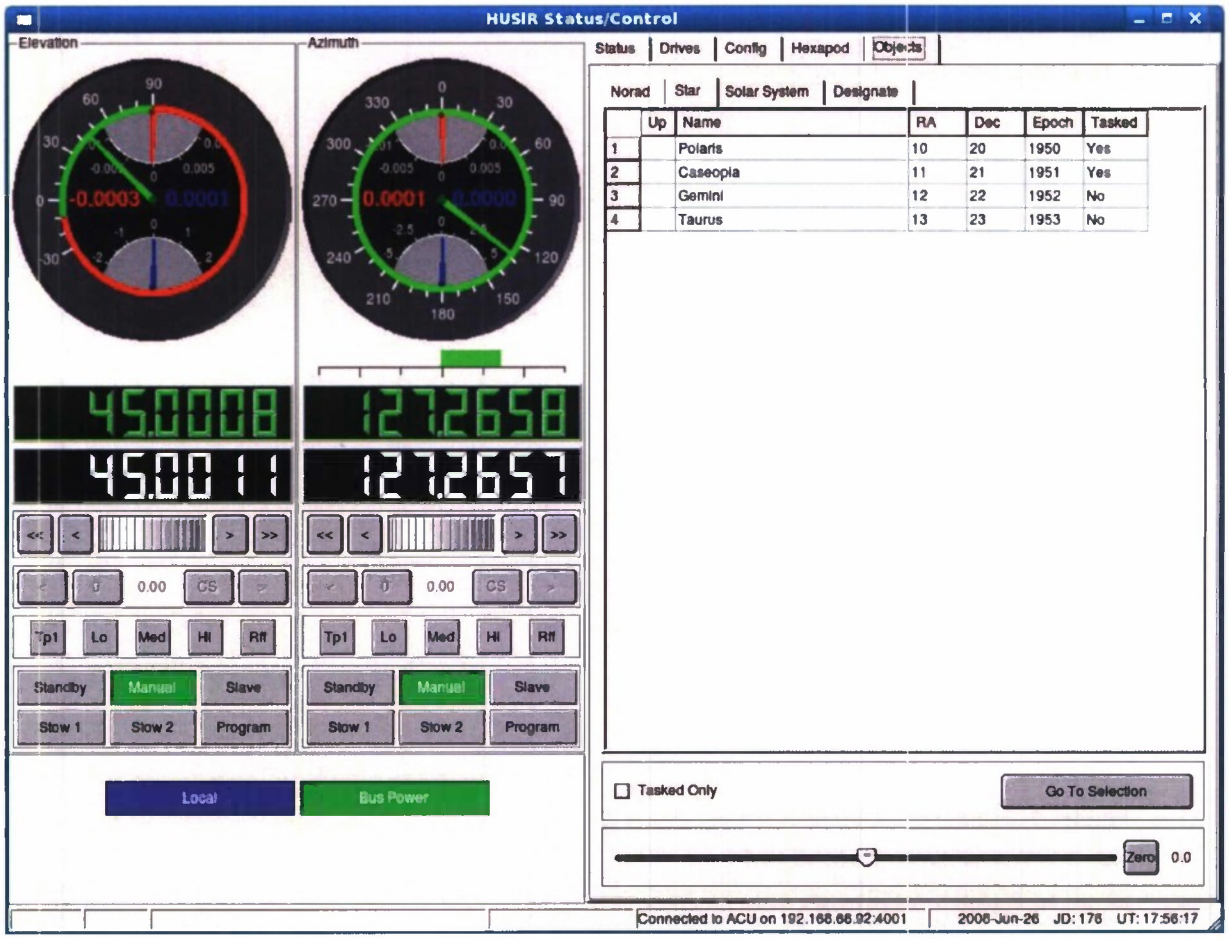
Figure 25. Status/Control window–object tab (NORAD).



	Up	Name	NORAD	Type	Updated	Tasked
1	Y	ORION 1	23413	GEO	10/30/2005	Yes
2	Y	SKYNET 4D	25134	GEO	10/30/2005	Yes
3	Y	MOLNIYA 3-49	25379	HEO	06/27/2008	Yes
4		TECHSAT 1B (GO-32)	25397	LEO	06/26/2008	Yes

Connected to ACU on 192.168.86.92:4001 | 2008-Jun-26 JD: 178 UT: 17:56:02

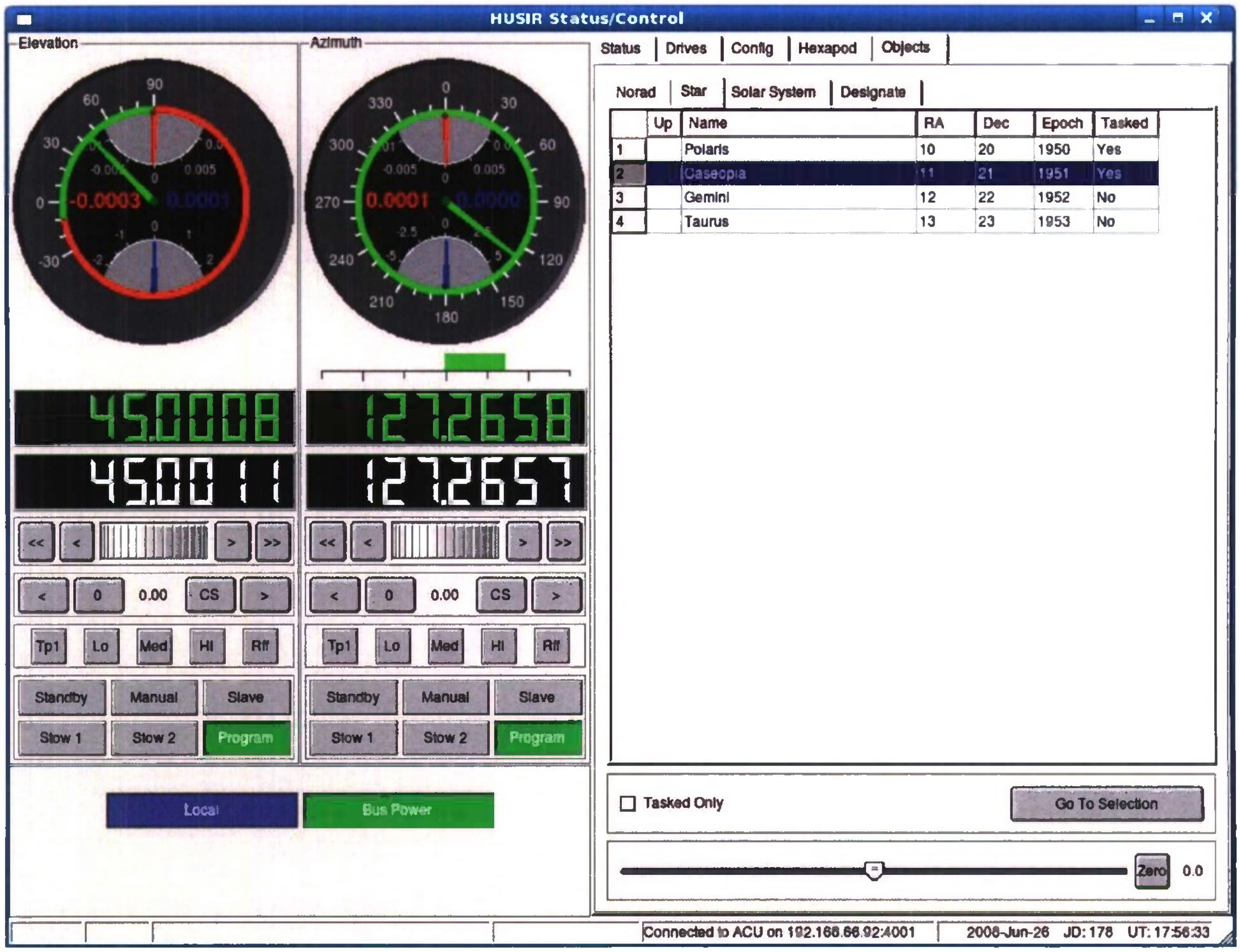
Figure 26. Status/Control window–object tab (NORAD, tasked only).



	Up	Name	RA	Dec	Epoch	Tasked
1		Polaris	10	20	1950	Yes
2		Caseopia	11	21	1951	Yes
3		Gemini	12	22	1952	No
4		Taurus	13	23	1953	No

Connected to ACU on 192.168.68.92:4001 | 2008-Jun-26 JD: 178 UT: 17:56:17

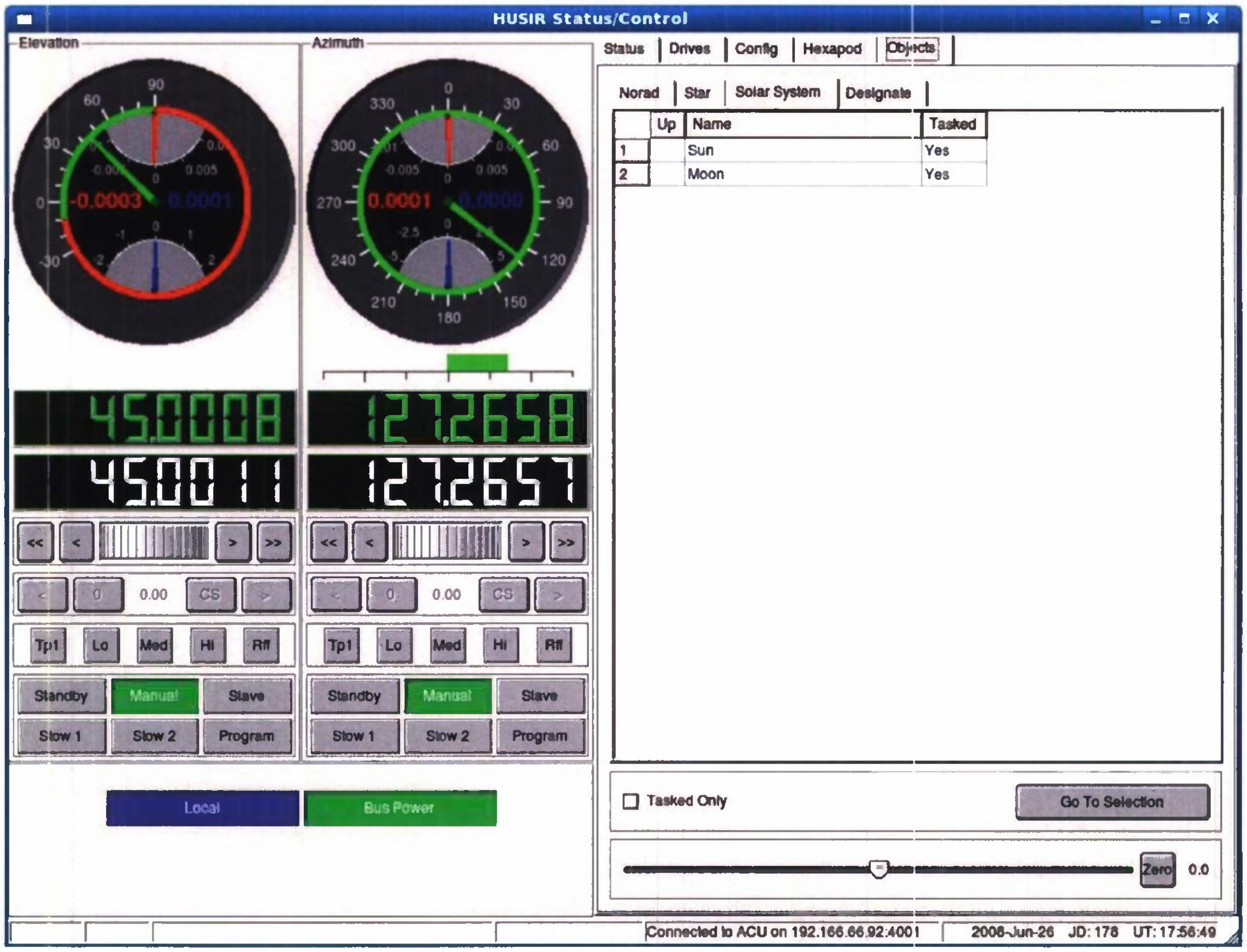
Figure 27. Status/Control window–object tab (star).



	Up	Name	RA	Dec	Epoch	Tasked
1		Polaris	10	20	1950	Yes
2		Caseopia	11	21	1951	Yes
3		Gemini	12	22	1952	No
4		Taurus	13	23	1953	No

Connected to ACU on 192.168.66.92:4001 | 2008-Jun-26 JD: 178 UT: 17:56:33

Figure 28. Status/Control window–object tab (program tracking).

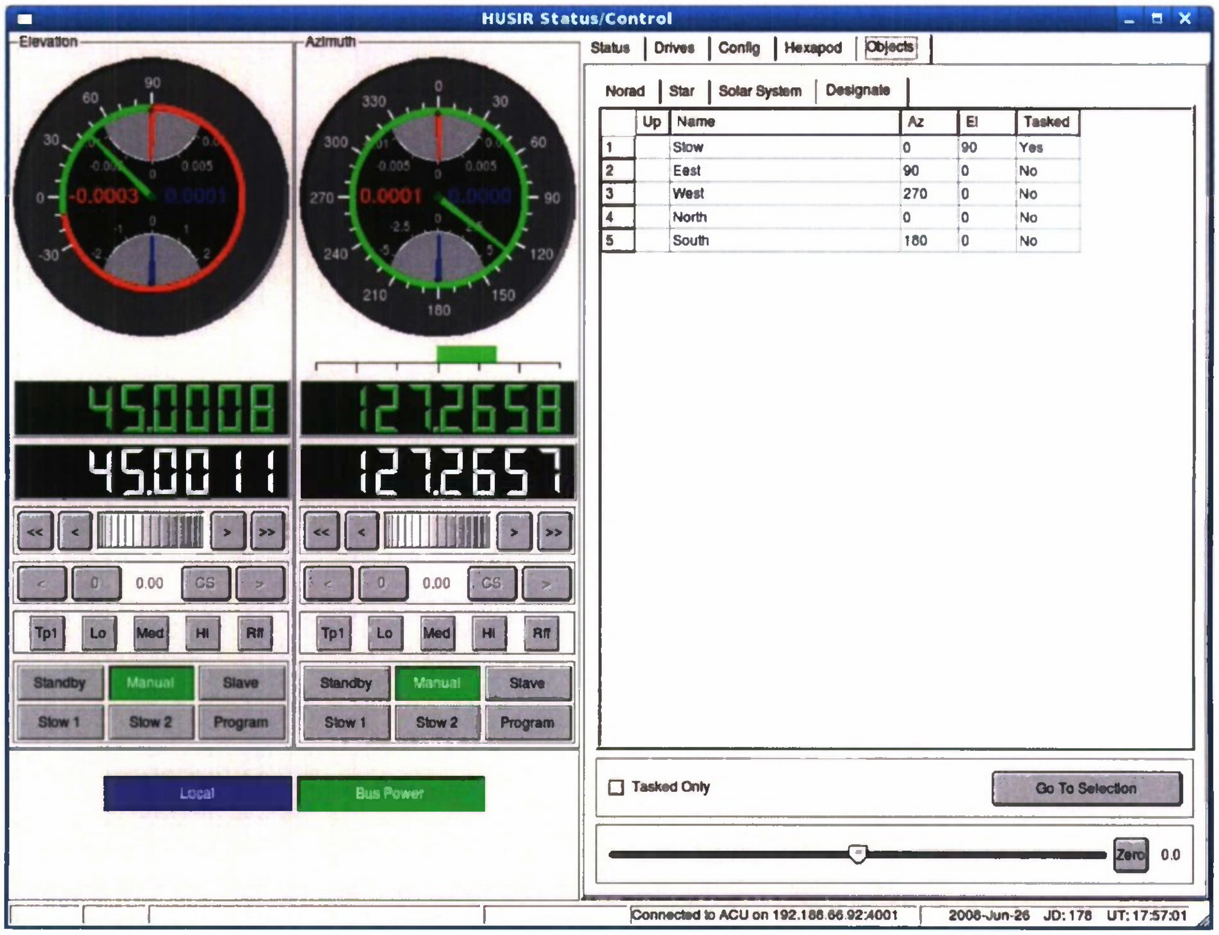


45.0008
45.0011
127.2658
127.2657

	Up	Name	Tasked
1		Sun	Yes
2		Moon	Yes

Connected to ACU on 192.166.66.92:4001 | 2006-Jun-26 JD: 178 UT: 17:56:49

Figure 29. Status/Control window–object tab (solar system).



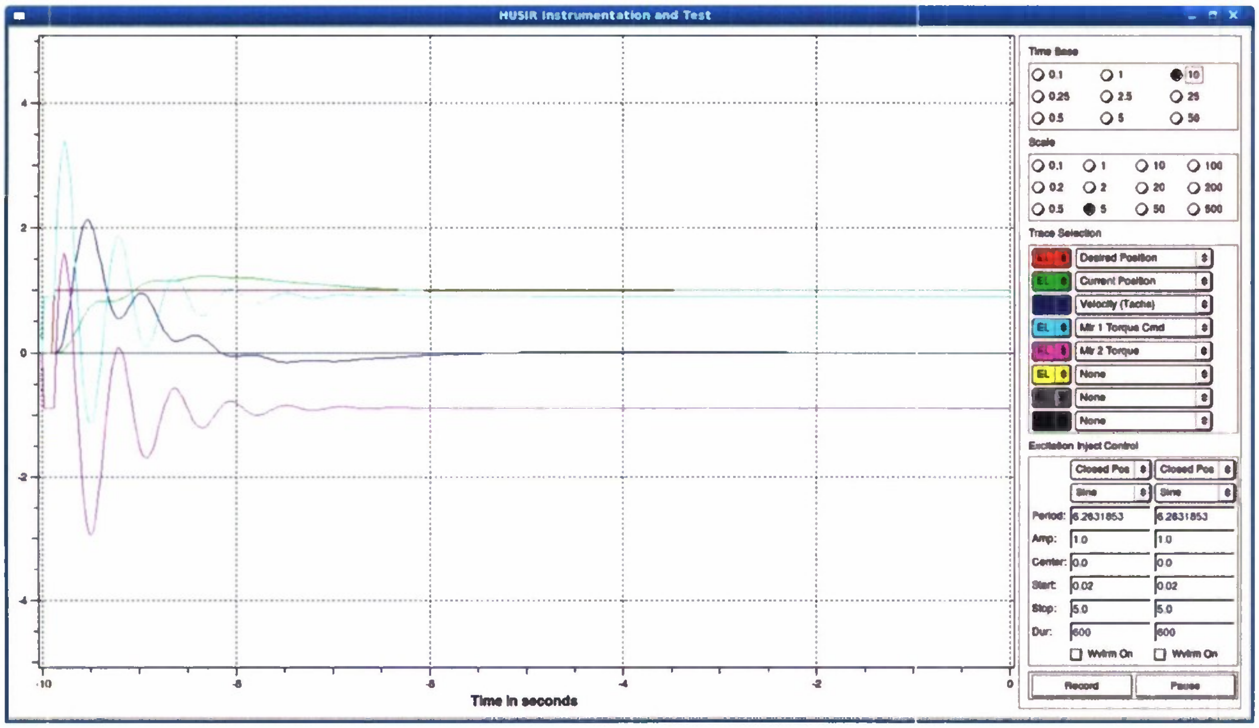
	Up	Name	Az	El	Tasked
1		Stow	0	90	Yes
2		East	90	0	No
3		West	270	0	No
4		North	0	0	No
5		South	180	0	No

Connected to ACU on 192.168.66.92:4001 | 2006-Jun-26 JD: 178 UT: 17:57:01

Figure 30. Status/Control window–object tab (designate).

5.2.3 Instrumentation Window

The instrumentation window is completely separate from the Status/Control window. The Maintenance computer shall be fitted with dual SVGA adapters and monitors to provide double the normal screen real-estate. Hence, the instrumentation window and Status/Control window can be displayed simultaneously. Additional independent instrumentation windows may be created.



Time Base: ○ 0.1 ○ 1 [✓ 10] ○ 0.25 ○ 2.5 ○ 25 ○ 0.5 ○ 5 ○ 50

Scale: ○ 0.1 ○ 1 ○ 10 ○ 100 ○ 0.2 ○ 2 ○ 20 ○ 200 ○ 0.5 [✓ 5] ○ 50 ○ 500

Trace Selection:
- 🟥 Desired Position
- 🟩 EL Current Position
- 🟦 Velocity (Tach)
- 🟦 EL Mtr 1 Torque Cmd
- 🟪 Mtr 2 Torque
- 🟨 EL None
- ⬛ None
- ⬛ None

Excitation Inject Control

	Closed Pos	Closed Pos
	Sine	Sine
Period:	6.2831853	6.2831853
Amp:	1.0	1.0
Center:	0.0	0.0
Start:	0.02	0.02
Stop:	5.0	5.0
Dur:	600	600
	☐ Wvfrm On	☐ Wvfrm On

Figure 31. Instrumentation window.

The Instrumentation window is shown in Figure 31. Its purpose is to allow for the monitoring and recording of all calculations internal to the ACU. Additionally, it provides a mechanism to inject test waveforms into each axis for performance verification and fault-isolation.

The main feature of this display is the strip chart widget which occupies 80% of the display from the left hand side. This strip chart is a horizontally scrolling display which updates in real time based upon instrumentation status messages from the ACU. In the above figure a rather poor step response to a position input is depicted. The time base and scale are selectable by the radio buttons on the right hand side of the display and the information to be plotted is selected via the list boxes in the center of the right hand side of the display. At a minimum, the following information can be plotted for each axis:

- Current position
- Desired position
- Position error
- Current velocity
- Desired velocity

- Veloeity error
- Current Aeeeleration
- Desired Aeeeleration
- Desired Torque for eaeh motor
- Current torque for eaeh motor
- Outputs of all eompensators

The design of this display is sueh that other data items may be added very easily in order to faeilitate testing and performanee monitoring.

Regardless of what is displayed on the strip ehart, all data shall be streamed from the ACU to t he M aintenanee e omputer f or pot ential reeording. Clieking t he Reeord but ton on t he Instrumentation display shall eause a prompt for a filename. Onee the filename is speeified, all data streamed from the ACU shall be recorded to the file in a CSV format for ease of proeessing by M atlab, E xeel o r a nother ma thematieal ma nipulation u sability. This C SV f ile s hall a t a minimum eontain all the data listed above.

In o rder to faeilitate performanee evaluation, the Instrumentation D isplay s hall a llow the operator to speeify waveforms to be input to the position and veloeity loops. These waveforms may b e s tarted an d s topped at w ill an d t he r esults o f t he ex eitation m ay be r eeorded b y t he mechanism d eseribed ab ove. The generated waveforms m ay b e i njeeted i nto the s ystem at the following points:

- Open Position Loop
- Closed Position Loop
- Open Veloeity Loop
- Closed Veloeity Loop

The following waveforms may be generated and injeeted into the above points:

- Sine Wave
- Square Wave
- Triangle Wave
- Saw Tooth Wave
- Swept Sine Wave

The l ower l eft hand portion of the instrumentation di splay a llows the op erator to speeify the waveform to be generated and the axis and injeetion point for the test.

6. PLC SYSTEM

The PLC system maintains overall control of the antenna. It determines whether the antenna is safe to move from a personnel and equipment safety perspective. If all of the systems it monitors indicate safe to operate, it will energize the prime power to the antenna drives. The design philosophy is "There are 1001 ways to turn of the antenna drives, and only 1 to turn them on." The PLC also controls the behavior of the antenna when it enters directional limits, as well as the motor brakes. This is done so that all 8 of the azimuth amplifiers (or 4 in elevation) get the same limit or brake command at the same time. There are situations where a single amplifier can request brakes on due to an internal error, and turning a single brake on can damage the system. Therefore, the amplifier requests brakes on and the PLC system engages all of the brakes on that axis, preventing any potential damage. A description of each of the subsystems controlled and monitored by the PLC is given below.

The current status of all systems controlled by the PLC, and any faults detected are displayed on an annunciator panel in the astronomy control room. The system logs the time and details of any change in status of the system.

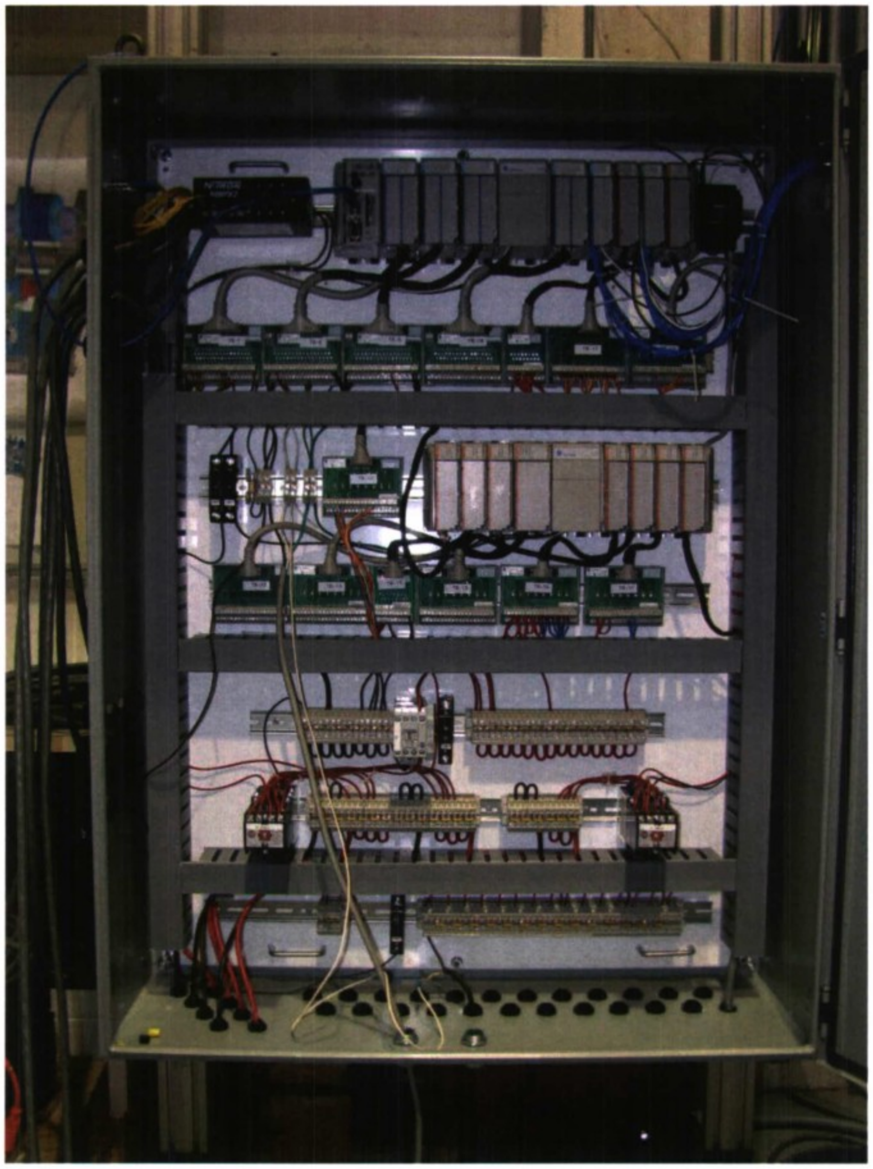


Figure 32. PLC cabinet with temporary wiring for simulator testing.

6.1 HYDROSTATIC BEARING

A fault by any component in the Hydrostatic bearing system will stop or inhibit motion in the a zimuth di rection vi a t he d rive s hutdown pr ocedure. 48 p ressure t ransducers m onitor t he thrust and radial pressure. 4 E ddy current s ensors m onitor the ride he ight of the be aring. A NI compact field point is used to monitor these and the oil temperature and pressure at the bearing. The overall status is then relayed to the PLC in a go/no-go fashion. Other inputs to the PLC include a reservoir float switch, water detection circuit and status indicators from the filters (for maintenance purposes only).

6.2 BOX INSERTION SYSTEM

The PLC system monitors a set of contact closures for each of the track pins, dolly stow pins, and hoist frame stow pins used in the RFBIS. This is to ensure that the RFBIS is properly stowed prior to moving the antenna.

6.3 BOX BOLTS

The PLC system will monitor a set of contact closures on each RF box bolt that are configured to be closed when the bolt is completely inserted. This ensures that the RF box is properly nested in the antenna prior to motion. There are 4 bolts used to mount the RF box.

6.4 RFBIS

The antenna safety PLC system communicates with the RFBIS PLC system via several relay contacts. The antenna safety PLC closes a set of contacts when it is safe for the RFBIS to be operated. The drives must be disabled, the antenna at face-side and the brake engaged before the RFBIS system can operate.

6.5 LUBE SYSTEM

The lube system provides oil to the gearboxes that drive the antenna. The PLC system controls and monitors the lube system. Should the oil level in the reservoir drop below the minimum level for operation or the flow switches indicate no flow, the PLC shuts down the drives and disables the lube system.

6.6 DRIVE CABINETS

The PLC interfaces directly with the AZ and EL Drive Cabinets. It connects to the motor control units (CSH01) to relay whenever there is contact with a limit in order to ensure proper direction limiting of the motors. The PLC monitors each amplifiers brake request line. If any amplifier requests brake on, the PLC commands all of the brake on that axis to come on. In order to ensure proper powering up and down the internal mains contactor is controlled via the PLC over these lines.

6.7 WATER DETECTION SYSTEM

The water detection circuits are used to prevent contamination of the hydrostatic bearing oil and lubrication oil should a hose break occur. There is one circuit in the base of the tower and two circuit in the azimuth transition area which provide discrete inputs to the PLC. If water is detected, the antenna is stopped via the brakes, prime power removed from the drive system, and the lubrication system and hydrostatic bearing pump systems are shut down.

6.8 ELEVATION STOW PINS

The stow pins will be monitored and controlled via the PLC. There will be 2 or 4 control lines and 4 monitoring. The PLC will command which state each pin needs to be in and verify that the pins are in the proper set-up. If there is conflicting information of any sort (e.g., the status indicates that stow pin 1 is both in and out or is in when it should be out) the PLC will inhibit elevation drive prime power via the drive cabinet shutdown procedure.

6.9 E-STOP CIRCUIT (PERSONAL SAFETY LOOP)

The E-stop circuit is a series loop passing through every E-stop switch, as well as the final limit switches for the antenna. If any of the contacts are broken, prime power is removed immediately. A second set of contacts on each switch is used to indicate which of the switches was tripped.

6.10 SECTOR BLANKING

A set of switches contained in the limits switch assemblies will be available for the transmitter to use for RF sector blanking. This is required to prevent RF radiation being directed at any of the other antennas at site. The antenna safety PLC system does not interact with the switches.

6.11 BRAKES

The PLC system is in control of the brakes at all times. The ACU, drive amps, or the PLC itself may request that the brakes be engaged. All brakes on a given axis are actuated simultaneously by way of a contactor.

When a given antenna axis must be hand cranked, the PLC brake control can be bypassed using the personnel safety Kirk Key system. A key block near the hand crank location for each axis allows the brake to be disengaged for that axis.

6.12 PLC → ACU COMMUNICATIONS

The PLC will communicate with the ACU via a one way link that provides for sufficient bandwidth to allow all status information to be transferred into the ACU. The ACU will use discrete lines to communicate with the PLC to allow for a high-to-low security transition (The ACU is on a classified network for radar operations, and the PLC is unclassified). This link and the information passed on it is defined in the *ACU to safety PLC system Interface Control Document.*

6.13 LIMIT PACKAGES

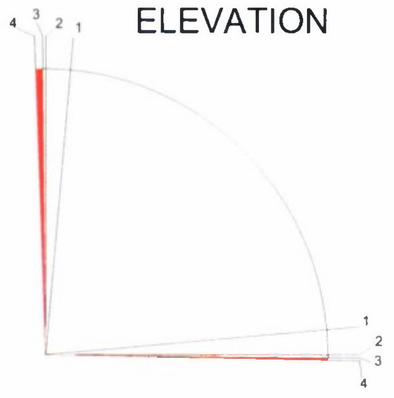
There will be a first and seeond limit switeh in eaeh direetion, on eaeh axis. The final limit switeh i s l oeated on t he e opper l oop t hat di sables pr ime pow er t o t he dr ives. A t e aeh l imit loeation there will be two eontaets one that tells the PLC whieh loeation it is and the other that performs the aetion. Two bypass keys will be used whenever overriding the final limit switehes is needed to baek out the antenna. One of the key eireuits bypasses both of the elevation limits while the other bypasses the azimuth limits. The keys are spring loaded so that they can not be left in the bypass state.

See Seetion 14 for a eomplete deseription of the limit switeh assemblies. The preliminary limit pos itions a re s hown i n Figure 33. N ote th at th e e xaet p ositions w ill b e d etermined a t commissioning.

HAYSTACK ANTENNA LIMITS

September 1994

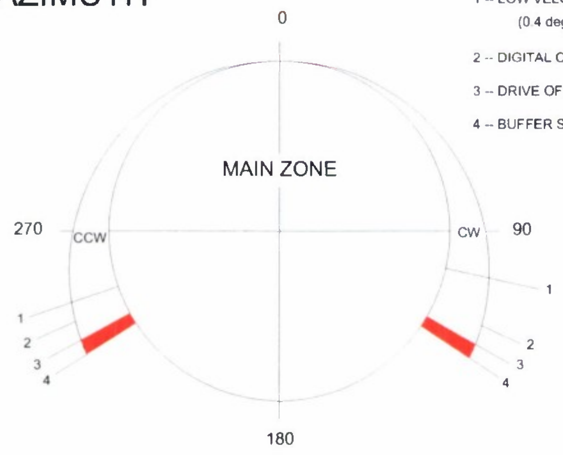
ELEVATION



LIMITS:	HIGH	LOW
1 -- LOW VELOCITY (0.3 deg/sec)	85.0	5.0
2 -- DIGITAL COMMAND	90.0	0.0
3 -- DRIVE OFF	90.6	359.4
4 -- BUFFER STOP	91.2	358.9

MAX VELOCITY = 1 deg/sec

AZIMUTH



OVERLAP ZONE LIMITS:	CCW	CW
1 -- LOW VELOCITY (0.4 deg/sec)	251.0	102.6
2 -- DIGITAL COMMAND	246.0	115.0
3 -- DRIVE OFF	241.0	120.0
4 -- BUFFER STOP	237.2	123.8

MAX VELOCITY = 2 deg/sec

{antlimit.dwg}

Figure 33. Antenna limit positions.

7. POWER DISTRIBUTION SYSTEM

The servo control system requires three-phase 480 VAC at 200A. A breakdown of the power requirements is provided in Appendix B. This power is provided by the HUSIR transmitter shelter. Power from the shelter is routed into the base of the tower and distributed to various circuits by panelboard HDHY1J. The servo control system is wired into this panelboard. From here power is fed through an isolation transformer, up the cable wrap, and distributed to all servo control systems through the power distribution cabinet A12 located in the azimuth transition area. A block diagram of the power distribution portion of the control system starting at the isolation transformer is shown in Figure 34.

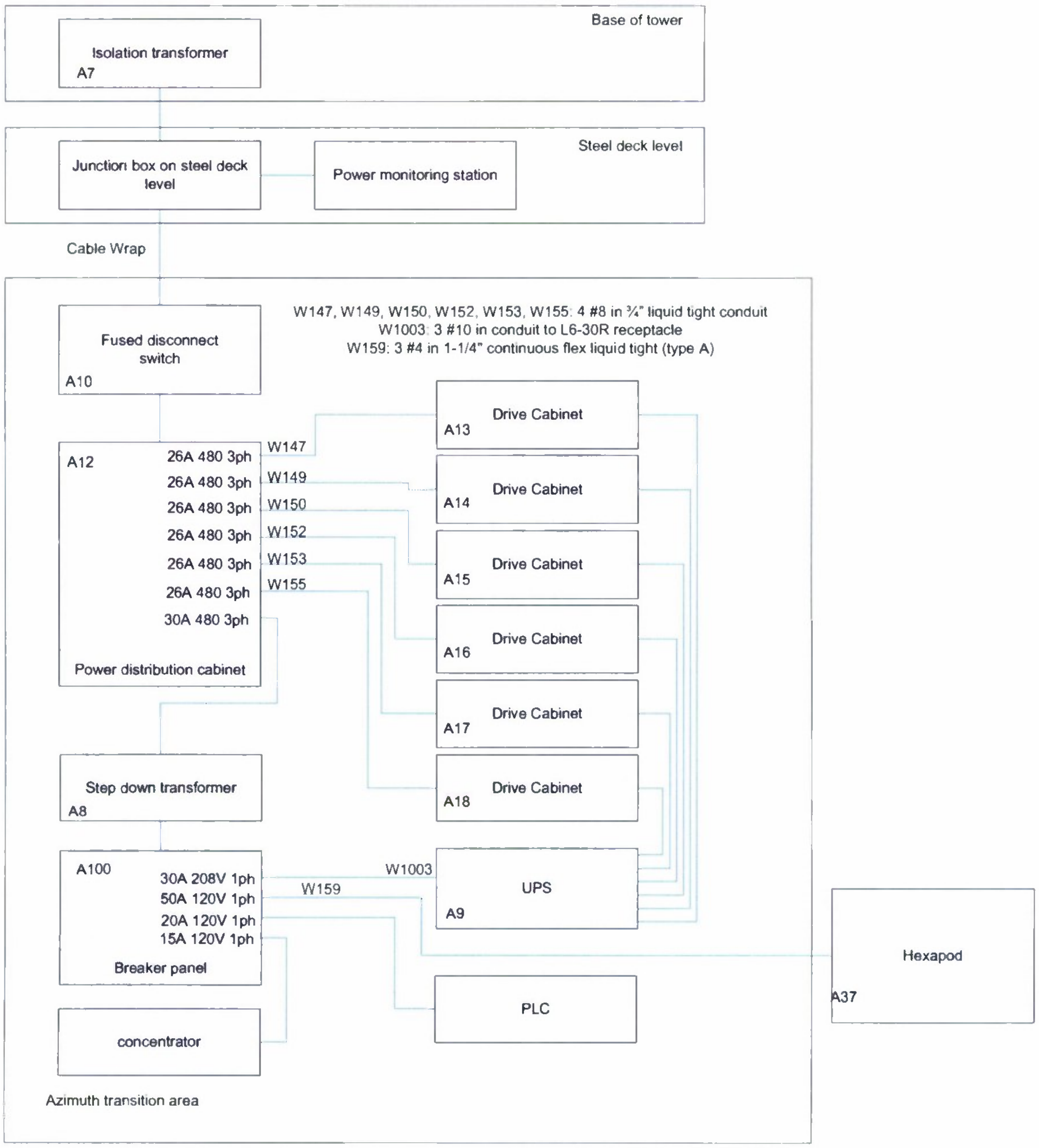


Figure 34. Control system power distribution block diagram.

7.1 MAINS DISCONNECT PANELBOARD HDHY1J

Three-phase 480 VAC at 270 A is provided via a 270 A breaker in panelboard HDHY1J. This is the main breaker disconnect for the entire drive system. This panelboard is already installed in the base of the tower.

7.2 ISOLATION TRANSFORMER A7

Three-phase 480 VAC is fed through conduit W1002 to isolation transformer A7. A7 is a three-phase 480 VAC delta to three-phase 480 VAC Y isolation transformer. The purpose of this transformer is to isolate the drive amplifier noise induced on the lines by the solid state Bosch Rexroth drive amplifiers. The output of A7 is fed through conduit W187 to the below wrap junction box where the power feed transitions to flexible cable.

7.3 POWER MONITORING JUNCTION BOX

This junction box serves two purposes. It converts from rigid conduit to flexible cable to go through the wrap, and contains the current sensing coils and voltage taps for the power monitoring system. The current and voltage lines route through conduit to the Carlo Gavazzi WM22 power monitor in a separate enclosure.

7.4 AZIMUTH CABLE WRAP AND THE ABOVE AXIS DISCONNECT A10

The output of the power monitoring junction box is fed via one flexible power cable (General Cable 82393 Type GG mining cable) up through the azimuth cable wrap to the above axis disconnect box A10. The azimuth cable wrap will carry three-phases of 480 VAC at 200 A. A10 allows drive system power to be shut down in the azimuth transition area.

7.5 AZIMUTH TRANSITION AREA POWER DISTRIBUTION CABINET A12

Three-phase 480 VAC is fed from A10 through conduit W167 to the power distribution cabinet A12. The power distribution cabinet contains the main system contactor that is controlled by the PLC, and six breakers for the drive cabinets. The is an additional 50 amp breaker that is fed from the supply side of the contactor that provides power for the step down transformer.

7.6 STEP DOWN TRANSFORMER

The step down transformer supplies power for all control system components that require voltages other than 480v 3 phase power. This feeds the low voltage breaker panel.

7.7 LOW VOLTAGE BREAKER PANEL

The breaker panel supplies power for the hexapod, brake solenoids, encoder concentrator, PLC, and the UPS that powers all of the drive cabinet control sections. The layout of the breakers in the panel is shown in Figure 35.

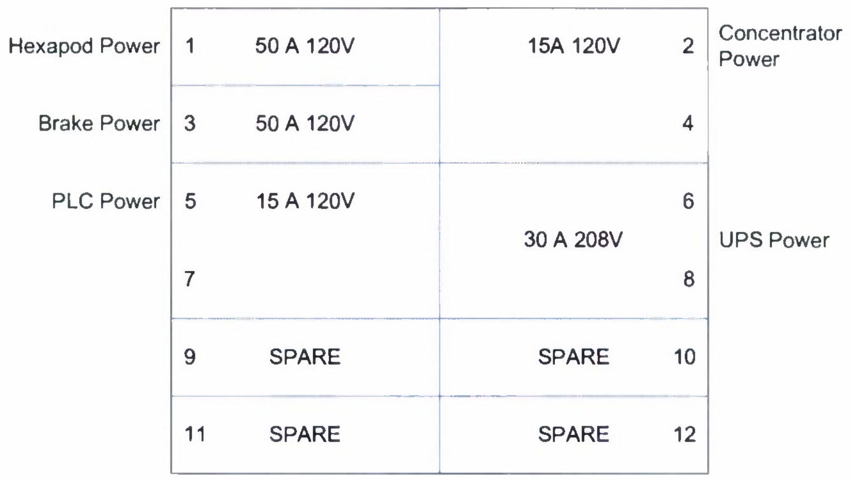


Figure 35. 120/208 V breaker panel.

8. AZIMUTH AND ELEVATION DRIVE CABINETS A13, A14, A15, A16, A17, AND A18

The HUSIR servo drive system is based on the Bosch Rexroth IndraDrive Family. HUSIR requires a total of eight servo motors and geared power trains to move the antenna in azimuth at the specified rates. HUSIR requires a total of four servo motors and geared power trains to move the antenna in elevation at the specified rates. Each servo motor requires a Bosch Rexroth CSH01.1C-SE-EN1-NNN-NNN-NN-S-NN-FW drive amp controller, a Bosch Rexroth HMS01.1N-W0036-A-07-NNNN drive amplifier power section, and a Bosch Rexroth HMV01.1R-W0018-A-07-NNNN power supply unit. However one HMV01.1R-W0018-A-07-NNNN power supply unit is capable of powering two CSH01.1C-SE-EN1-NNN-NNN-NN-S-NN-FW drive amp controllers and two HMS01.1N-W0036-A-07-NNNN drive amplifier power sections. It is for this reason that the HUSIR servo drive system is divided up into only six drive cabinets, where each drive cabinet controls two servo motors. The Firmware used by the amplifiers is version 04V34.

There are a total of four azimuth drive cabinets, denoted in the system block diagram as A13, A14, A15, and A16. There are a total of two elevation drive cabinets, denoted in the system block diagram as A17 and A 16. Each drive cabinet contains one HMV01.1R-W0018-A-07-NNNN power supply unit, two CSH01.1C-SE-EN1-NNN-NNN-NN-S-NN-FW drive amp controllers, and two HMS01.1N-W0036-A-07-NNNN drive amplifier power sections. The drive cabinet wiring is in support of these Bosch Rexroth units so that proper power-up, power-down, and monitoring will occur.

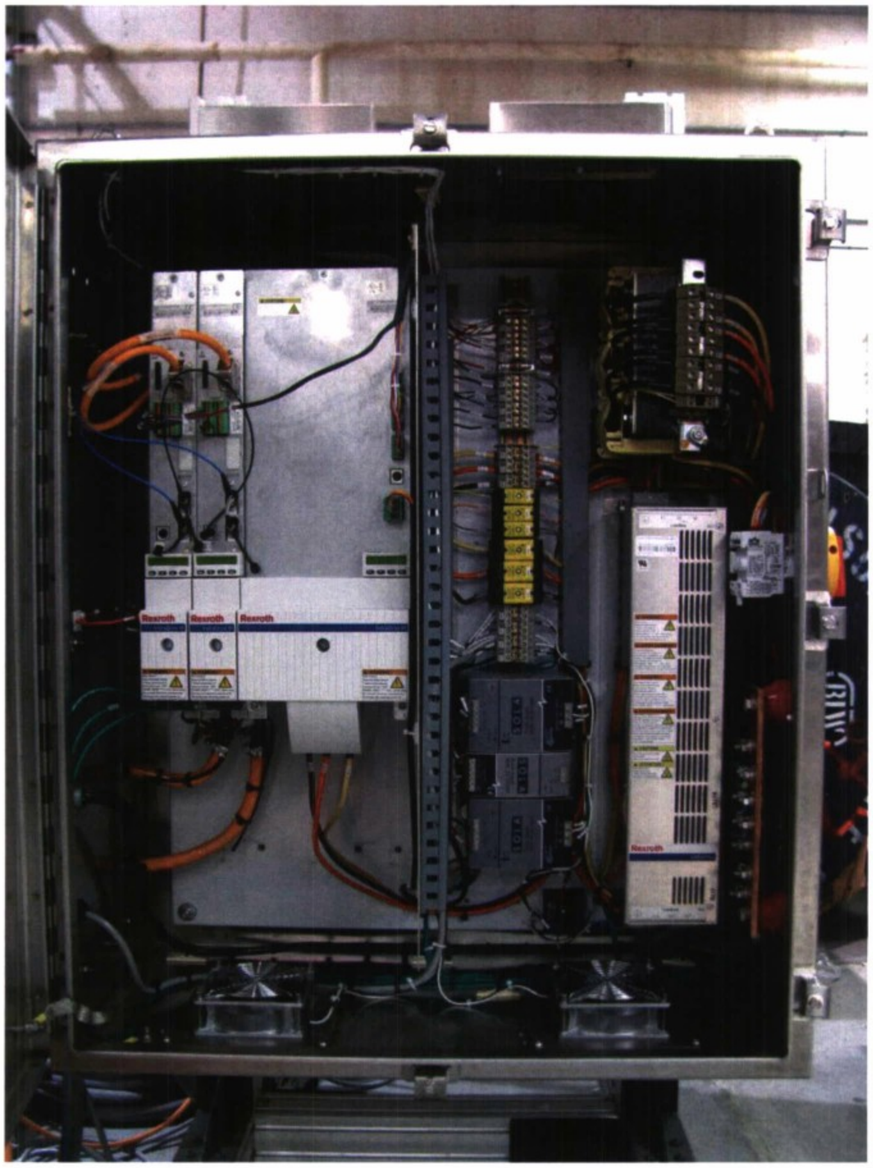


Figure 36. Photograph of inside of completed drive cabinet.

8.1 POWER DISTRIBUTION

Three-phase 480 VAC and single phase 120 VAC are fed to the drive cabinets A13, A14, A15, A16, A17, and A18 from the power distribution cabinet A12 via conduits W147, W149, W150, W152, W153, and W155 respectively. The three-phase 480 VAC is used to provide power for the drive motors. The single phase 120 VAC is used to power the control circuitry inside of the HMV01.1R-W0018-A-07-NNNN power supply unit, two CSH01.1C-SE-EN1-NNN-NNN-NN-S-NN-FW drive amp controllers, and two HMS01.1N-W0036-A-07-NNNN drive amplifier power sections.

8.1.1 Three-Phase 480 VAC Power to the Bosch Rexroth Power Supply Unit HMV01.1R-W0018-A-07-NNNN

Three-phase 480 VAC enters a drive cabinet and is immediately fed through a local power contactor relay K5. K5 is a special contactor relay that has incorporated into it a time delay between the main contactor and a status feedback switch (which indicates what state the relay contactor is in, closed or open). This delay is mechanically implemented so that the status feedback switch changes states at least 10 mS before the contacts open on the contactor relay. This provides the proper power-down sequence to protect the drive amplifier system in the event of a loss of power at site or an emergency stop.

Three-phase 480 VAC is fed through contactor K5 to line filter LF1. LF1 filters much of the RFI caused by the drive amplifier before it can be leaked into the main supply lines. The load lines of LF1 are fed through three 35 A fuses F1, F2, and F3. From here the three-phase power is fed through inductor L1 then wired into port X3 on the Bosch Rexroth 18KW regenerative power supply unit HMV01.1R-W0018-A-07-NNNN.

Taps are spliced into the three-phase power before L1 and fed through 6 A fuses F4, F5, and F6. This spliced three-phase reference power is fed into the mains synchronization port X14 on the power supply unit HMV01.1R-W0018-A-07-NNNN.

8.1.2 Single-Phase 120 VAC Power to the 24 VDC Redundant Power Supply Circuitry

Single-phase 120 VAC power enters a drive cabinet and is fed into two 24 VDC DIN rail mounted switching power supplies PS1 and PS2. PS1 and PS2 are connected to a redundant power module PS3. In the event of a single power supply failure, in either PS1 or PS2, PS3 will rout power to the drive cabinet 24 VDC bus from the working supply.

Single-phase 120 VAC power for PS1 and PS2 is provided by the UPS A9 so that in the event of a power failure the DC supplies will continue to supply power to the 24 VDC bus.

8.1.3 Single-Phase 120 VAC Power for Optional Blower Fans

Single-phase 120 VAC is provided for blower fans to cool the drive motors. Blower fans are not currently installed on the drive motor assemblies, however, it will remain an option in case over-heating is observed during testing. Blowers can be connected to J3 and J6 on each drive cabinet.

8.1.4 The 24 VDC Bus

The 24 V DC is supplied by PS3, which runs between all Bosch Rexroth drive amplifier components starting at the power supply unit HMV01.1R-W0018-A-07-NNNN. The 24 VDC bus also powers the control circuitry in the two control sections SCH01.1C-SE-EN1-NNN-NNN-NN-S-NN-FW and the two power units HMS01.1N-W0036-A-07-NNNN.

8.1.5 The 700 VDC Bus

The output of the (approximately) 700 V DC solid-state power supply unit HMV01.1R-W0018-A-07-NNNN is bussed out to the two power units HMS01.1N-W0036-A-07-NNNN.

8.2 POWER-UP SEQUENCE

In order to prevent damage to the motor amplifiers the proper power-up sequence must be followed. According to the Bosch Rexroth IntraDrive Drive System Project Planning Manual, R911309636 Edition 4, the following power-up sequence must be implemented in the HUS1R servo control system PLC programming for turning on power to the drive amplifier systems:

1. Turn on 120 VAC to PS1 and PS2 by powering the UPS.
2. Wait 6 seconds for all of the drive systems to boot-up.
3. Short mains-off on the amplifier port X32.
4. Wait 100 ms.
5. Short mains-on on the amplifier port X32.
6. Wait until UD contact closes (pins 3 and 4 on port X31 on the power supply unit HMV01.1R-W0018-A-07-NNNN). This is accomplished in the drive cabinet circuitry by wiring the pins for Bb1, UD, and WARN in series on across port X31. The output of this circuit is called CHECK ENGINE. CHECK ENGINE is high (+24 VDC) when all indicators; Bb1, UD, and WARN are closed. CHECK ENGINE is an open circuit when one of those three indicators is open. If any one of the three indicators Bb1, UD, or WARN is open then the local contactor K5 should not be closed. Effectively the circuit CHECK ENGINE indicates a problem, or a 'not-ready,' state when it is an open-circuit. Everything is ok when CHECK ENGINE is high.
7. Wait 500 mS before starting the next drive cabinet power up procedure.

PLEASE NOTE: Only one drive cabinet can be powered-up at a time. This sequence must be repeated for each individual drive cabinet. According to the Bosch Rexroth manual the three-phase 480 VAC main power supply lines are loaded with a test current which is part of the Bosch internal calibration procedure. Only one HMV01.1R-W0018-A-07-NNNN power supply can be powered-up at a time or else this current measurement will load down the power distribution system too much casing inaccurate calibration, or possible blown fuses or breakers.

8.3 POWER-DOWN SEQUENCE

In order to prevent damage to the motor amplifiers the proper power-down sequence must be followed. According to the *Bosch Rexroth IndraDrive Drive System Project Planning Manual, R911309636 Edition 4,* the following power-down sequence must be implemented in the HUSIR servo control system PLC programming for turning off power to the drive amplifier systems:

1. Open mains-off pins 6 and 7 on port X32 on the power supply unit HMV01.1R-W0018-A-07-NNNN.
2. Wait for UD pins on the power supply unit HMV01.1R-W0018-A-07-NNNN to open by waiting for the CHECK ENGINE line to go low.

PLEASE NOTE: The three-phase 480 VAC can be shut down by shutting down the prime power contactor K1 in the power distribution cabinet A12. This must not happen until the mains-off pins 6 and 7 on port X32 on the power supply unit HMV01.1R-W0018-A-07-NNNN are open at least 10 mS before the three-phase 480 VAC power is shut down or else damage to the drive amplifier system will occur.

8.4 STATUS INDICATORS AND CONTROL FOR THE PLC INDUSTRIAL CONTROL SYSTEM

Status indicators and some control of the drive cabinets is achieved by a PLC industrial control system. The inputs and outputs of the drive cabinet that are controlled by the PLC are discussed in this section. All status indicators and PLC control lines are fed back to the PLC remote I/O cabinet A102 through various pins on the MIL connector J9 on each drive cabinet. Drive cabinets A13, A14, A15, A16, A17, and A18 connect to PLC remote I/O cabinet A102 via cables W243, W242, W241, W240, W239, and W238 respectively.

8.4.1 Fault Indicator Pins Bb1, UD, and WARN from Power Supply Unit HMV01.1R-W0018-A-07-NNNN

The fault indicator pins Bb1, UD, and WARN, on port X32 in power supply unit HMV01.1R-W0018-A-07-NNNN are monitored by the PLC. These pins are normally open relay contacts. In the drive cabinet design all three of these switch contacts are wired in series. Therefore, the status of Bb1, UD, and WARN are placed in series and fed back to the PLC as the +24 VDC output CHECK ENGINE. If CHECK ENGINE is high (+24 VDC) then the system is ok. If CHECK ENGINE is open circuit then there is a problem. Exact status of the problem can be found by interrogation of the drive cabinet via the SERCOS loop.

8.4.2 Brake Control from the Two Power Sections HMS01.1N-W0036-A-07-NNNN

Each power section, HMS01.1N-W0036-A-07-NNNN, of the drive amplifier assembly has a brake control output. These brake outputs are fed back to the PLC to alert the PLC that one or more of the drive amplifiers is requesting to engage the brakes. If the PLC reads a low (open circuit) from the brake output then the PLC will immediately turn off the brake solenoids on all 12 brakes on the servo control system. The brakes are held open using a 24 VDC solenoid. When power is cut to the brakes the brakes will close, stopping all antenna motion.

8.4.3 Mains-on Request to the Power Supply Unit HMV01.1R-W0018-A-07-NNNN

The PLC controls the mains-on request pins 4 and 5 on port X32 on the HMV01.1R-W0018-A-07-NNNN power supply unit. These are controlled by the PLC remote I/O cabinet A102 by use of PLC relay I/O.

8.4.4 Mains-off Request to the Power Supply Unit HMV01.1R-W0018-A-07-NNNN

The PLC controls the mains-off request pins 6 and 7 on port X32 on the HMV01.1R-W0018-A-07-NNNN power supply unit. These are controlled inside of the PLC remote I/O cabinet A102 by use of PLC relay I/O. Pins 6 and 7 are also controlled by a time delayed status indicator switch that is part of the local contactor K5. This allows pins 6 and 7 to be open circuited at least 10 mS before the contactor K5 is lifted, protecting the drive amplifier circuitry from damage.

8.4.5 LIMIT(+) Input to the Two Drive Control Units CSH01.1C-SE-EN1-NNN-NNN-NN-S-NN-FW

The LIMIT(+) input pin 5 on port X31 on the two drive control units, CSH01.1C-SE-EN1-NNN-NNN-NN-S-NN-FW, are wired in parallel inside of the drive cabinet. The PLC controls the LIMIT(+) drive control unit input.

8.4.6 LIMIT(-) Input to the Two Drive Control Units CSH01.1C-SE-EN1-NNN-NNN-NN-S-NN-FW

The LIMIT(-) input pin 6 on port X31 on the two drive control units, CSH01.1C-SE-EN1-NNN-NNN-NN-S-NN-FW, are wired in parallel inside of the drive cabinet. The PLC controls the LIMIT(-) drive control unit input.

8.4.7 +24 VDC Reference for Ground Loop Avoidance

+24 VDC and a return line from each drive cabinet is provided on the MIL connector J9. This is used to prevent ground loops between the drive cabinet and the PLC remote I/O cabinet A102. All input and output wiring from the drive cabinets must be isolated from the local ground in A102. Therefore, optically isolated or relay isolated modules are required when operating the control input and output lines from the drive cabinets.

8.5 SERVO MOTOR CURRENT FEEDBACK AND TEMPERATURE SENSING

Drive cabinet connectors J5 and J8 connect the power section of the drive amplifier assembly to each of the servo motor phase connections and temperature sensors. The current loop in the control system is closed by the drive amplifier system. Current and torque commands can be issued to the drive amp controller CSH01.1C-SE-EN1-NNN-NNN-NN-S-NN-FW via the SERCOS loop.

Connectors J5 and J8 on each of the drive cabinets connect the two drive amplifier power sections, HMS01.1N-W0036-A-07-NNNN, to two of the servo motors. This is done by custom length Bosch Rexroth cables. For drive cabinet A13 the motor phases and temperature connection is achieved by cable W100 to servo motor A17 and W103 to drive motor A18. For drive cabinet A14 the motor phases and temperature connection is achieved by cable W106 to drive motor A19 and W109 to drive motor A20. For drive cabinet A15 the motor phases and temperature connection is achieved by cable W112 to drive motor A21 and W115 to drive motor A22. For drive cabinet A16 the motor phases and temperature connection is achieved by cable W117 to drive motor A23 and W120 to drive motor A24. For drive cabinet A17 the motor phases and temperature connection is achieved by cable W123 to drive motor A25 and W126 to drive motor A26. For drive cabinet A18 the motor phases and temperature connection is achieved by cable W129 to drive motor A27 and W132 to drive motor A28.

8.6 SERVO MOTOR VELOCITY FEEDBACK

The velocity loop in the control system is closed by an encoder mounted on the motor shaft, where encoder data is fed back into the CSH01.1C-SE-EN1-NNN-NNN-NN-S-NN-FW drive amp controller. Velocity commands are issued to the drive amplifier assembly via the SERCOS loop.

Connectors J4 and J7 on each of the drive cabinets connect the two CSH01.1C-SE-EN1-NNN-NNN-NN-S-NN-FW drive control units to the motor encoders. This is done by the use custom length Bosch Rexroth cables. For drive cabinet A13 the drive motor encoder connection is achieved by cable W101 to drive motor A17 and W104 to drive motor A18. For drive cabinet A14 the drive motor encoder connection is achieved by cable W107 to drive motor A19 and W110 to drive motor A20. For drive cabinet A15 the drive motor encoder connection is achieved by cable W113 to drive motor A21 and W116 to drive motor A22. For drive cabinet A16 the drive motor encoder connection is achieved by cable W118 to drive motor A23 and W121 to drive motor A24. For drive cabinet A17 the drive motor encoder connection is achieved by cable W124 to drive motor A25 and W127 to drive motor A26. For drive cabinet A18 the drive motor encoder connection is achieved by cable W130 to drive motor A27 and W133 to drive motor A28.

8.7 SERCOS LOOP

SERCOS is an industry standard, high-speed, fiber optic link between industrial controls. SERCOS is used throughout the HUSIR servo system for high-speed communication between the ACU and the drive amplifiers in the drive cabinets. Rate, position, torque, and other commands are issued from the ACU to the drive amplifiers via the SERCOS loop. All six drive cabinets form one SERCOS loop.

There are two CSH01.1C-SE-EN1-NNN-NNN-NN-S-NN-FW drive amplifier controllers in each azimuth or elevation drive cabinet. The SERCOS is routed in to the drive cabinet through fiber optic through-jack J1. From J1 it is fed via a fiber optic cable to the X20 port on one of the CSH01.1C-SE-EN1-NNN-NNN-NN-S-NN-FW drive amplifier controllers. The X21 port on that drive amplifier controller is connected to the X20 port on the other drive controller. The X21 port on the 2nd drive amplifier controller is fed out of the drive cabinet via fiber optic through-jack J2.

9. ENCODER POSITION REPORTING SYSTEM

There are a total of three encoders which are used to report the position of the antenna. One encoder reports the azimuth angle. Two encoders report the elevation angle, measuring the angle of elevation of the antenna on each of the two yoke arms. Due to the long cable run between the antenna and the ACU there must be an encoder concentrator cabinet, A11, which multiplexes these encoder readings back to the ACU over a long fiber optic cable.

9.1 AZIMUTH ENCODER A41

The azimuth encoder A41 is a Heidenhain RCN829 absolute angle encoder. This encoder communicates angle position back to the encoder concentrator cabinet A11 by using the Heidenhain EnDat RS-485 communications protocol. The EnDat data is fed to the encoder concentrator cabinet via cable W1004. Cable W1004 is a Heidenhain custom length encoder extension cable.

RCN 829 Test Data

Heidenhain Encoder Catalog

9.2 ELEVATION ENCODERS A43 AND A42

The two elevation encoders A42 and A43 are Heidenhain RCN727 absolute angle encoders. Each encoder communicates angle position back to the encoder concentrator cabinet A11 by using the Heidenhain EnDat RS-485 communications protocol. The EnDat data from A42 and A43 is fed to the encoder concentrator cabinet via cables W1005 and W1006 respectively. Cables W1005 and W1006 are Heidenhain custom length encoder extension cables.

RCN 727 #1 Test Data

RCN 727 #2 Test Data

9.3 ENCODER CONCENTRATOR CABINET A11

RS-485 from the azimuth and elevation encoders is fed in through J11, J12, and J13 on the encoder concentrator cabinet A11. J11, J12, and J13 are each a Heidenhain cable assembly which includes an encoder socket on one end, a 1 M run of cable, and bare hook-up wires on the other end. The wires are wired into TB1 inside of A11. The pair of DATA lines and pair of CLOCK lines for each cable are fed from TB1 directly into the encoder concentrator board inside of A11. The encoder concentrator board provides high-speed multiplexed communication between the encoders on the antenna and the ACU in the control room via a long run of fiber optic cable. 120 VAC single-phase power is provided to the encoder concentrator cabinet via conduit W1007. This 120 VAC line voltage power is used to run 5 VDC power supplies. The concentrator and each encoder have independent power supplies. The encoders use the sense outputs on the power supplies to ensure that 5 V is available at the encoder after a long cable run (particularly for the elevation encoders).

10. ENCODER CONCENTRATOR BOARD

An Encoder Concentrator (EC) is required to transfer encoder data from the above the azimuth wrap in the antenna down to the ACU in the control room. The EC must support synchronously triggered read out of encoders. The position sense encoder data must be synchronously read then transferred from the antenna back to the ACU.

10.1 HARDWARE

The encoder concentrator cabinet A11 contains an Encoder Concentrator (EC) printed circuit board that collects position data from the azimuth and elevation encoders and transmits it to the ACU, and the power supplies for each encoder. The concentrator has three EnDat v2.1 interfaces for the elevation encoders and azimuth encoder. The collected data is transmitted to the ACU over a fiber optic interface.

10.2 FORM FACTOR

There are two identical boards with different firmware loads used in the system. One is located above the azimuth wrap in the Encoder Concentrator Cabinet, and interfaces to the PLC and encoders. The other is located in a 1U rack mount chassis in the radar control room and interfaces to the ACU.

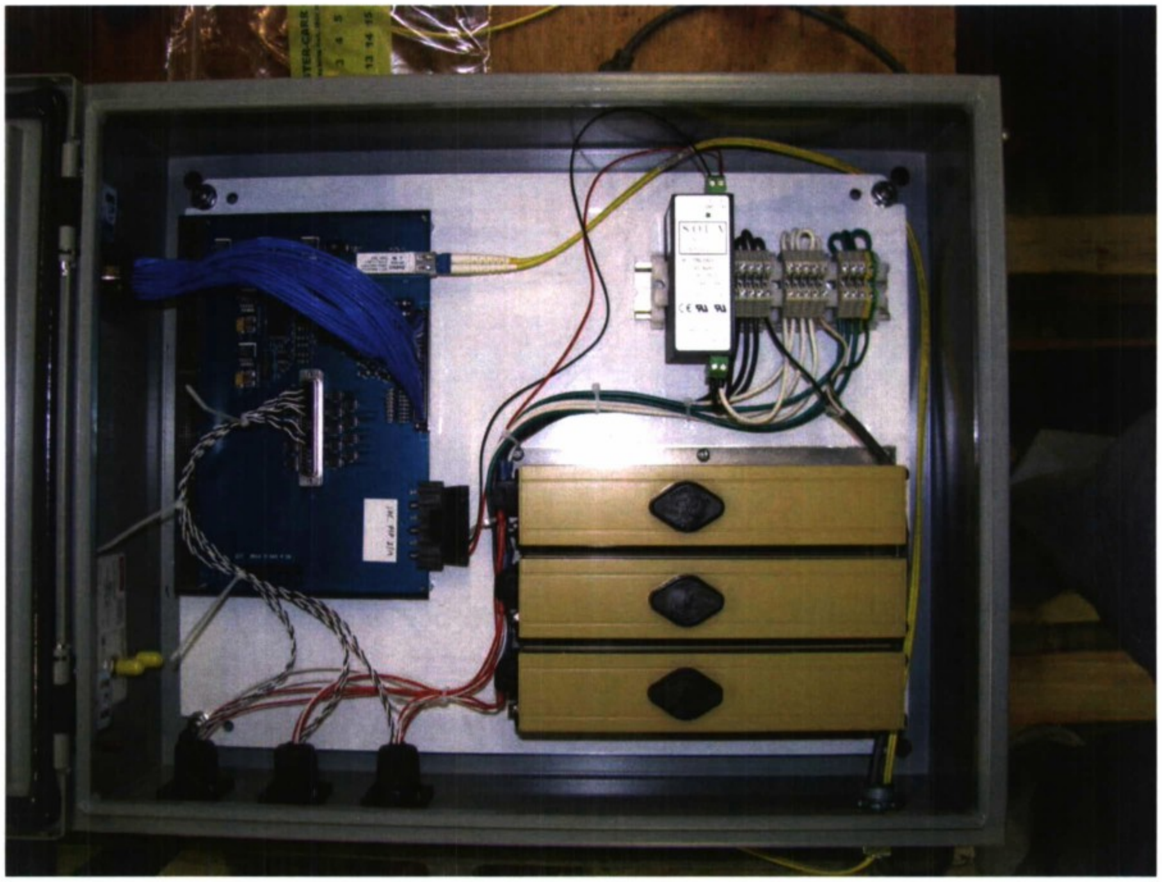


Figure 37. Encoder concentrator cabinet.

10.3 POWER

The Encoder Concentrator board requires 5 Volts with less then 1 Ampere. Power is provided by the rack chassis power supply for the control room concentrator. The antenna side concentrator is powered by a 5 Volt DIN rail mounted power supply.

10.4 CONTROL ROOM TO ANTENNA HARDWARE INTERFACE

The Encoder Concentrator board incorporates a single-mode fiber optic transceiver with integrated clock recovery. Each transceiver has an LC duplex receptacle to connect the fiber optic cable. The transceiver will feature an un-cooled 1300 nm laser transmitter with automatic output power control, and a wide dynamic range receiver with InGaAs PIN photodetector. The raw data rate is 125 M bits/second with biphase encoding, resulting in a final data rate of 25 Mbit/second.

10.5 ENCODER HARDWARE INTERFACES

For the antenna side of the EC PC board pair, the EC design features fourteen RS422/485 channels to interface with the encoders. The RS422/485 channels are accessible via one DB-37 connector. A cable harness connects the DB37 to the IO Concentrator Cabinet's encoder feed thru connecters (Heidenhain bulkhead connectors), which are then cabled to the individual encoders. Each channel is half-duplex and bidirectional. The drivers are slew-rate-limited to minimize EMI and reduce reflections caused by improperly terminated cables, allowing error-free data transmission at data rates up to 2.5 Mbps. The line length limitation for each RS422/485 driver is specified to 4000 feet with 26 AWG twisted-pair wire into 120 ohm load. The required line length for Haystack is 200 feet max.

10.5.1 Azimuth Encoder Hardware Interface

Two of the RS485 channels are dedicated to the azimuth encoder. One channel will be a dedicated output clock signal. And the other channel will be bidirectional data signal. The data transfer rate from the azimuth encoder to the EC will be 1 Mb/s.

10.5.2 Elevation Encoder Hardware Interfaces

There are two elevation encoders. Each elevation encoder will use two RS485 channels. One channel will be a dedicated output clock signal. And the other channel will be bidirectional data signal. The data transfer rate from the elevation encoders to the EC will be 1 Mb/s.

10.6 ACU HARDWARE INTERFACE

For the control room side of the EC PC board pair, the EC design features 32 general IO channels to interface with the ACU. A 68 pin SCSI-3 cable will be used to interface from the control room concentrator chassis (a 1U rack mount chassis) to the ACU. Inside the concentrator chassis there is a transition board that converts from the 68 pin SCSI-3 connector to a 50 pin ribbon cable.

Eagle CAM files for transition board

10.7 PLC INTERFACE

The concentrator in the azimuth transition area communicates with the PLC via the 32 IO channels available. They are configured 16 bits for input and 16 bits for output. Details of the signals on the interface are in the PLC to ACU ICD.

10.8 CONFIGURABLE HARDWARE GLUE LOGIC

A Xilinx Spartan2E FPGA is employed on the EC. This FPGA has the optical transceiver's required PECL interface built in. The FT256 package was selected, as it comes in a range of gate count to ensure that the resources will be available for the encoder serial and optical transceiver interfaces glue logic.

10.9 HARDWARE PERSONALITY SELECTOR

To enable the same circuit board to work both at the antenna and at the ACU, different firmware loads are programmed into the EEPROM on the board.

10.10 FIRMWARE

A full description of the firmware, is provided in the document *HUSIR Encoder Concentrator FPGA Design Document*. Manufacturing of additional spares requires:

PCB Schematic

PCD Gerber Files

PCB Bill-of-Materials

PCB Photos

The *Firmware Source Code* is maintained in the source code version control system for HUSIR.

11. APEX MAINTENANCE STATION A104

The apex maintenance station A104 is a basic enclosure mounted on the apex maintenance platform which contains an E-stop button, telephone, 120 VAC outlets, a trouble light, and an Ethernet connection for the portable maintenance computer. The apex maintenance station allows service personnel to control the motion of the antenna using a portable maintenance computer. It also provides personnel the ability to telephone the control room or other areas of the laboratory. In addition to this, there are outlets provided to power a trouble light, the portable maintenance computer, or any additional power tools such as a drill or a heat gun.

There are a total of four 120 VAC outlets in A104. Power is routed to these outlets via conduit W1200 from a 10 A breaker inside of the hexapod drive cabinet A37.

The emergency stop switch is mounted on top of A104 so that access to the E-stop button is available if the cover to A104 is closed. The E-stop button is a dual-pole, single-throw, normally closed switch. When depressed it opens up both of its contacts which causes the copper interlock loop to be broken and alerts the PLC to an E-stop condition. The E-stop wires are routed from the apex through the elevation cable wrap to the PLC remote I/O A102 via cable W1201. Cable W1201 also contains two phone wires for the standard lab telephone M8004, which is mounted inside of A104. Cable W1201 should be shielded to protect the phone line and the PLC I/O lines from EMI.

Ethernet is provided for the portable maintenance computer via a fiber media converter mounted inside of A104. Power for the fiber media converter is drawn from one of the four 120 VAC outlets inside of A104. The Ethernet is converted to fiber then it is sent down the elevation cable wrap to the fiber termination panel in the azimuth transition area. From this panel the fiber is routed to the radar control room where it is fed into another fiber media converter A106. The Ethernet output of A106 is fed into the pointing computer switch mechanism A105 where control of the ACU can be ported directly to the apex maintenance station and locked out of all other stations.

In addition to these features there is a trouble light to aid in servicing operations. This light is simply plugged into one of the four 120 VAC outlets inside of A104.

This page intentionally left blank.

12. FIBER OPTIC BACKBONE

The a ntenna will ha ve a f iber ba ekbone i nstalled t o s upport t he va rious e ontrol s ystem requirements. The drive system uses SERCOS for eontrol, and that is a fiber interfaee using hard elad siliea (HCS) fiber with a 200 mieron core. The eneoder eoneentrator uses single mode fiber, and the PLC system uses 50 mieron multimode fiber to eommunicate with the remote IO chassis in t he a stronomy e ontrol r oom a nd on t he de ek l evel of t he a ntenna. There a re f iber opt ie requirements for other parts of the radar and astronomy systems that will be met with this fiber baekbone. The baekbone will eontain three different kinds of fiber and is eonfigured as shown in Figure 38.

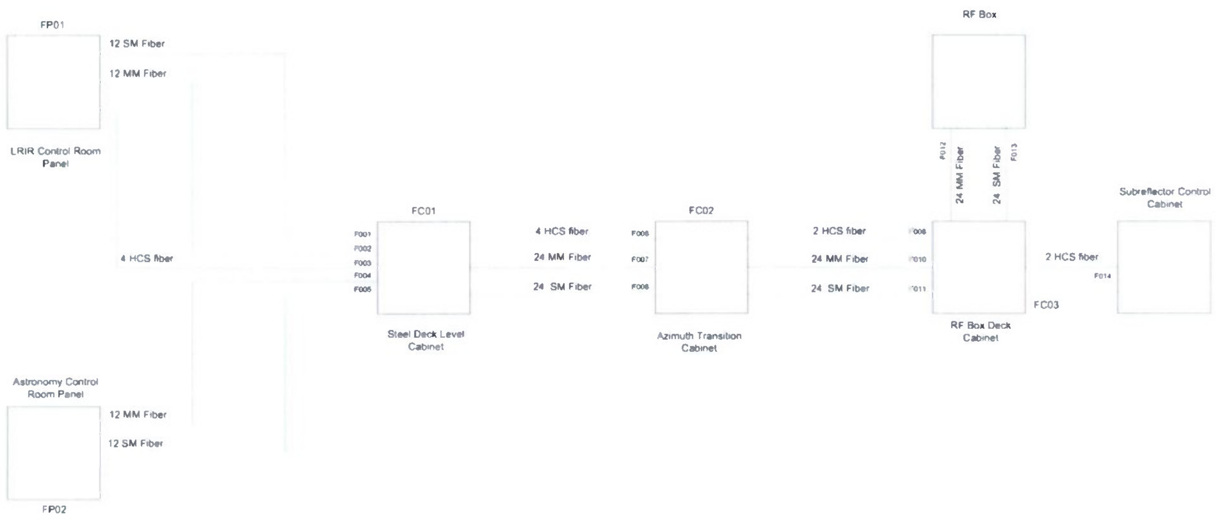


Figure 38. Antenna fiber optic backbone.

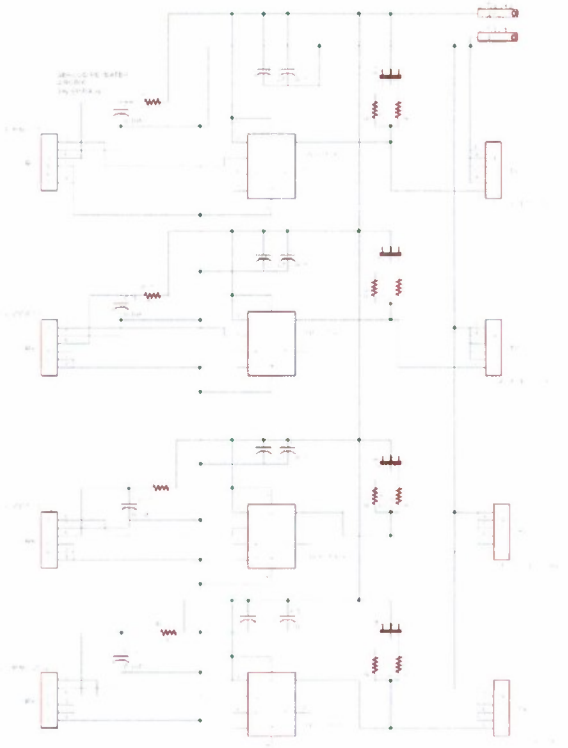


Figure 39. SERCOS repeater schematic.

There are fiber termination chassis in each of the control room and fiber patch panels on the steel deck level, azimuth transition area, and the RF Box deck. The termination chassis and patch panels are standard components used by the telecommunications industry. The termination chassis in the control room allow for easy reconfiguration of the signals on the fiber backbone and the patch panels at each of the three antenna levels are NEMA4X outdoor rated enclosures. These enclosures are slightly modified to allow for three separate fiber cable, rather than the one that they were originally designed for. Inside each of the termination chassis and patch panels are a set of termination panels, and a set of splice trays. This will allow for the mechanical splicing of the bulk fiber to the pig-tail terminations on the panels. In cases where low loss is required (like RF over fiber), fusion splicing is accommodated in the splice trays as well. The azimuth transition fiber patch panel also contains a SERCOS repeater board that boosts the signal going to the hexapod. The schematic for the board is shown in Figure 39. Each bulkhead connection results in approximately 1.5 dB of signal loss, and the link margin is such that five bulkhead results in loss of signal. The hexapod loop has four more bulkheads than the az/el drive loop.

The three fiber cables are contained inside a single 1.5" diameter water hose where it passes through the azimuth cable wrap. This is to allow for simple fiber change-outs should a cable need replacement, while at the same time providing the correct outside diameter for the cable wrap assembly.

The multimode and single mode fibers will be terminated with ST style connectors in all termination chassis and patch panels. Differentiation between single mode and multi mode fiber is done by color code defined by international standard TIA/EIA 598-A (yellow jackets are single-mode and orange jackets are multi-mode). The fiber numbering scheme inside a single buffer tube (and buffer tubes within a cable) are also defined by international standard and presented below:

Fiber/Tube	Color
1	Blue
2	Orange
3	Green
4	Brown
5	Gray
6	White
7	Red
8	Black
9	Yellow
10	Purple
11	Rose
12	Aqua

The SERCOS cable is not typically used in the Ethernet/telecommunications industry, but is very common in the industrial control world. The connectors to be used for the SERCOS fibers are SMA. This will allow easy visual differentiation from the telecommunications type fibers, as the color scheme for jacketing on these fibers is not standard. The SERCOS cable is a step-index multi-mode cable, it is NOT compatible with the multi-mode fiber used in networking. It is nice from a termination stand-point in that it is a cleave-and-crimp termination requiring no polishing.

Data Sheet for Sercos Cable

Termination Instructions

Sercos Repeater Board Gerber Files

This page intentionally left blank.

13. SUBREFLECTOR CONTROL

13.1 HEXAPOD DRIVE CABINET

A photograph of the Hexapod drive cabinet is shown in Figure 40.

13.1.1 Power Distribution

Main power to the hexapod comes in via 1-1/4" conduit from the step down transformer in the azimuth transition area. 120 V ac comes in and then is distributed to the six amplifiers, 90 VDC brake power supply, 24 V DC power supply, and the apex maintenance station. The 24 V power supply distributes power to a terminal bus which supplies 24 V wherever needed.

13.1.2 Amplifier

Inside the cabinet are six amplifiers, one for each actuator. The amplifier is a Kollmorgen Servostar CD series 5 amplifier. Part number CE03561 corresponds to encoder feedback(E), 3 A(03), 24 V logic power (6) with SERCOS control interface(1). It communicates with a Kollmorgen AKM series motor via J3(to the motor and brake commands) and J6 which reads position data back from the motor encoder. The amplifier reads data back from both the motor encoder (via C2 the feedback connector) and the Heidenhain encoder (via C8 the remote encoder input). It utilizes a user I/O port (C3) to communicate with the limit sensors, and for the index pulse from the Heidenhain encoder.

Servostar Installation Manual

Servostar Hardware Manual

Servostar IDN Manual (SERCOS Manual)

KMTG Motion Suite (serial control for Servostar)

Servostar amplifier parameter file

Goldline XT motor data sheet

Heidenhain encoder catalog

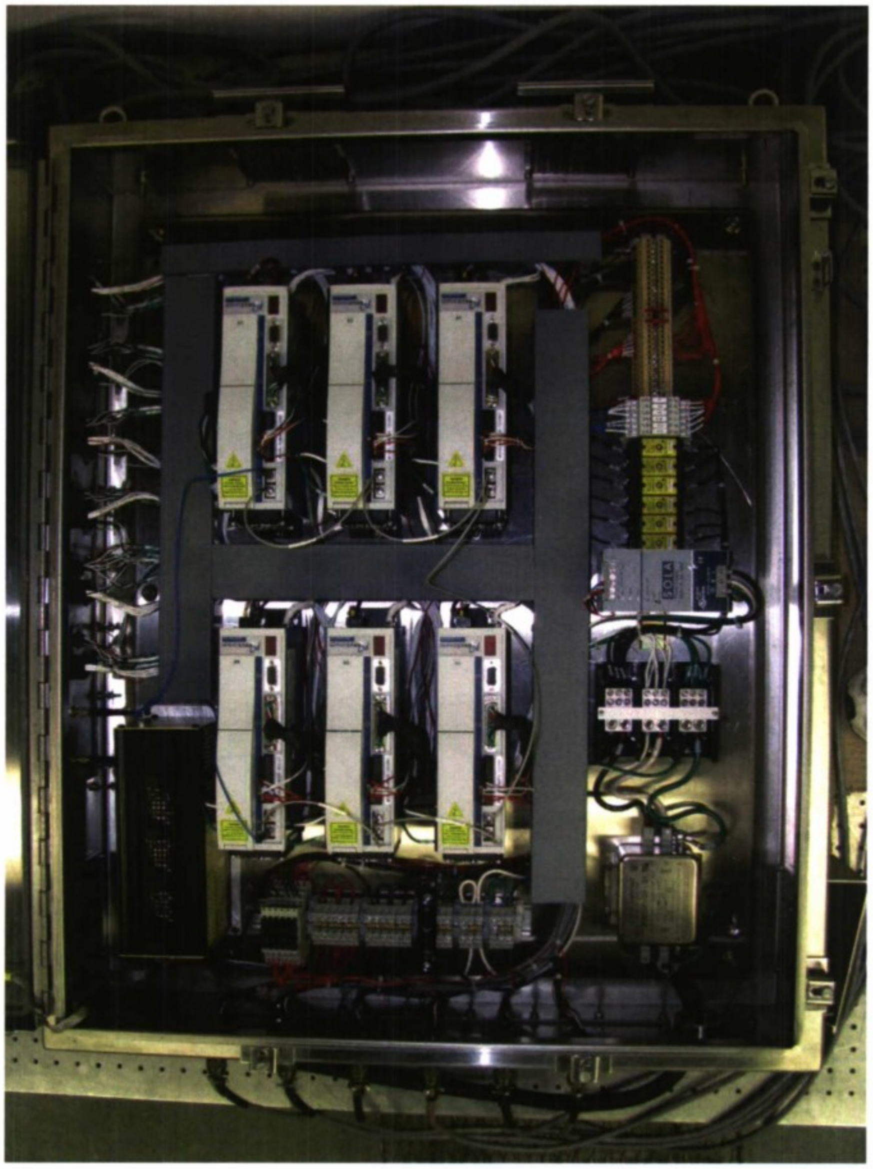


Figure 40. Hexapod drive cabinet internals.

13.1.3 Communication

All communication with the hexapod is done via the SERCOS fiber loop.

13.2 HEXAPOD ACTUATORS

The hexapod actuators consist of a ball screw linear actuator with a 200 mm stroke. The actuator is an Industrial Devices Corporation EC2-BK23-200-05B-100. Each end of the actuator has a ball joint coupling to connect it to the sub-reflector and the apex structure. A photograph of the complete hexapod strut with the cover over the linear encoder removed is shown in Figure 41. There are pre and final limit switches at either end of the stroke. The linear encoder will be used to close the position loop on each of the struts, and the motor encoder will be used to close the velocity loop.

It should be noted that the maximum speed of the actuator is 414 mm/second to avoid exciting a mechanical resonance in the lead screw that can result in damage to the actuator. Under normal operation our maximum expected velocity is <1.5 mm/sec.

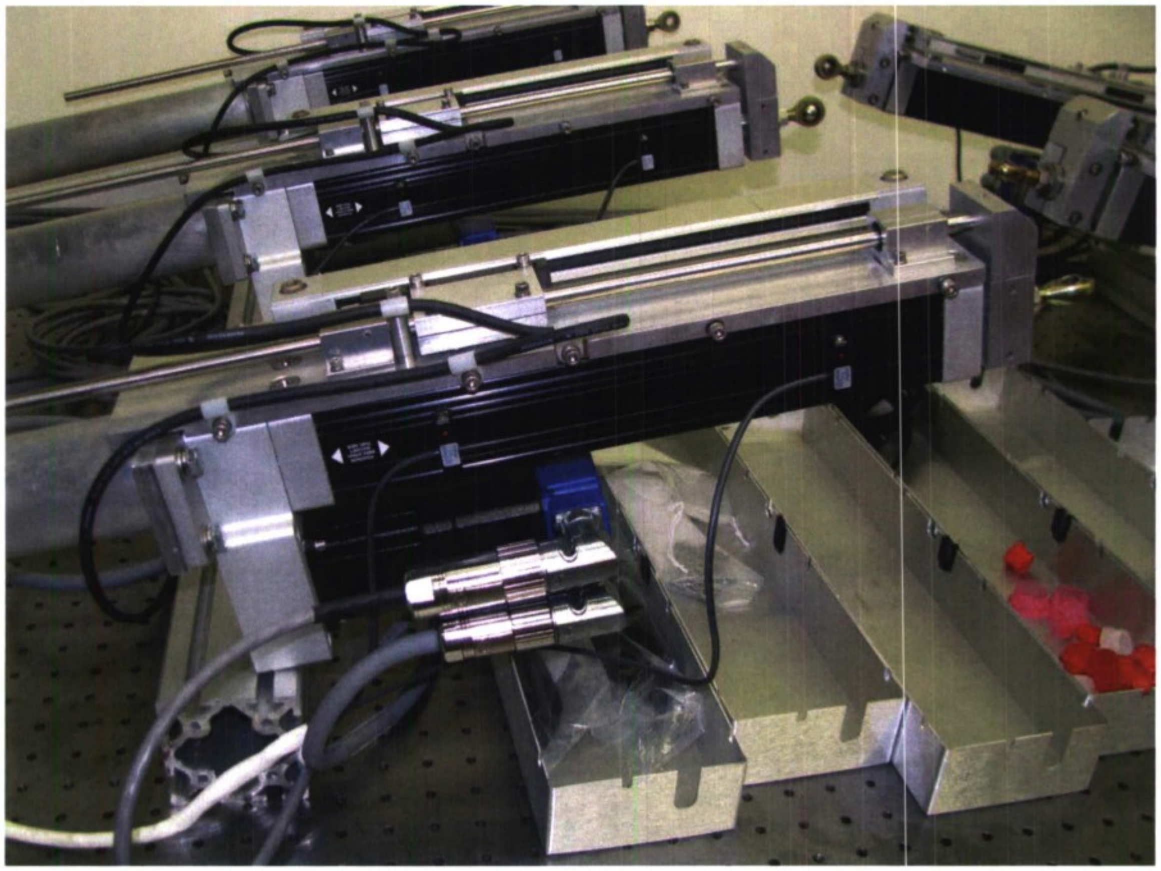


Figure 41. Hexapod actuator.

When the six actuators are installed the hexapod is as shown in Figure 42.

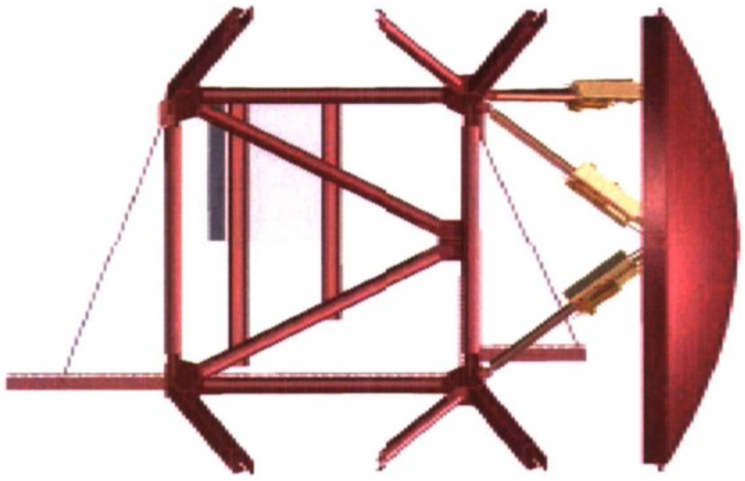


Figure 42. Hexapod configuration.

13.3 HEXAPOD MOTION

The hexapod is a set of actuators used to precisely position the subreflector. The hexapod is a construct known as a Stewart platform. It consists of six linear actuators that have the ability to translate the subreflector in all three dimensions, and rotate around all three axes.

Calculation of the linear actuator lengths given the attitude of the subreflector is straight-forward. Calculating the attitude given the leg length is a one-to-many problem. There are at least 40 different solutions to the problem. Fortunately, this should not be a problem because of the mode of operation planned for the subreflector.

To generate the equations for determining leg length, the geometry of the hexapod must be known. Specifically, the nominal length of the legs, the positions of the connection points on the apex of the quadrapod, and the position of the connection points on the subreflector must be known.

A local coordinate system is defined at the surface of the apex as shown below:

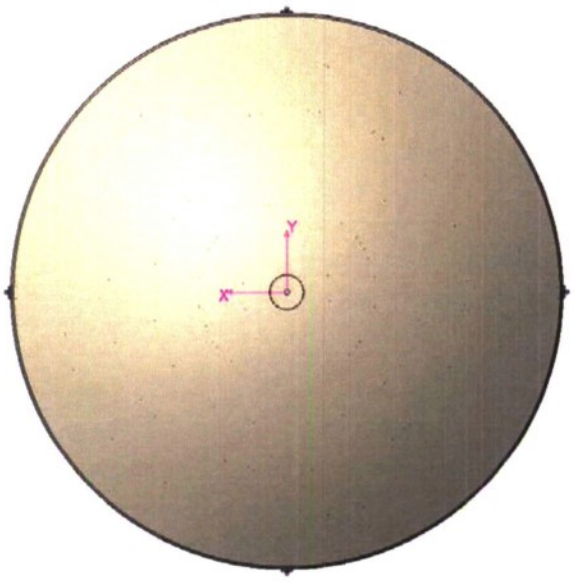


Figure 43. Subreflector coordinate system definition. Note the Z axis is positive into the page with the origin of the system at the vertex of the subreflector.

On the back side of the subreflector there are three clevis blocks that have one end of each of the six legs attached as shown in Figure 44.

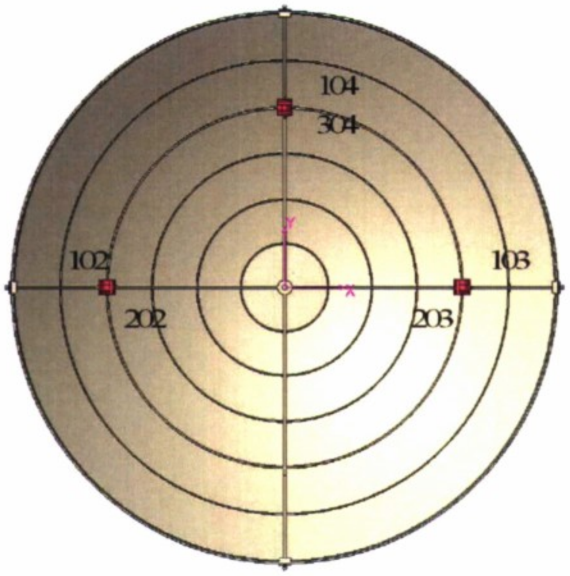


Figure 44. Subreflector clevis locations on the back of the subreflector.

The theoretical location of the mounting points for the subreflector end of each of the legs is shown in the following table.

		Subreflector local coordinates			**Primary reflector coordinates**		
Leg # end point	**Point**	**X (inches)**	**Y (inches)**	**Z (inches)**	**X (inches)**	**Y (inches)**	**Z (inches)**
L5	102	–37.590	0.000	16.326	–37.590	0.000	549.165
L6	202	–35.790	0.000	16.326	–35.790	0.000	549.165
L4	104	0.000	37.590	16.326	0.000	37.590	549.165
L3	304	0.000	35.790	16.326	0.000	35.790	549.165
L1	103	37.590	0.000	16.326	37.590	0.000	549.165
L2	203	35.790	0.000	16.326	35.790	0.000	549.165

The apex has a set of six clevis points that are defined in an apex local coordinate system as shown in Figure 45.

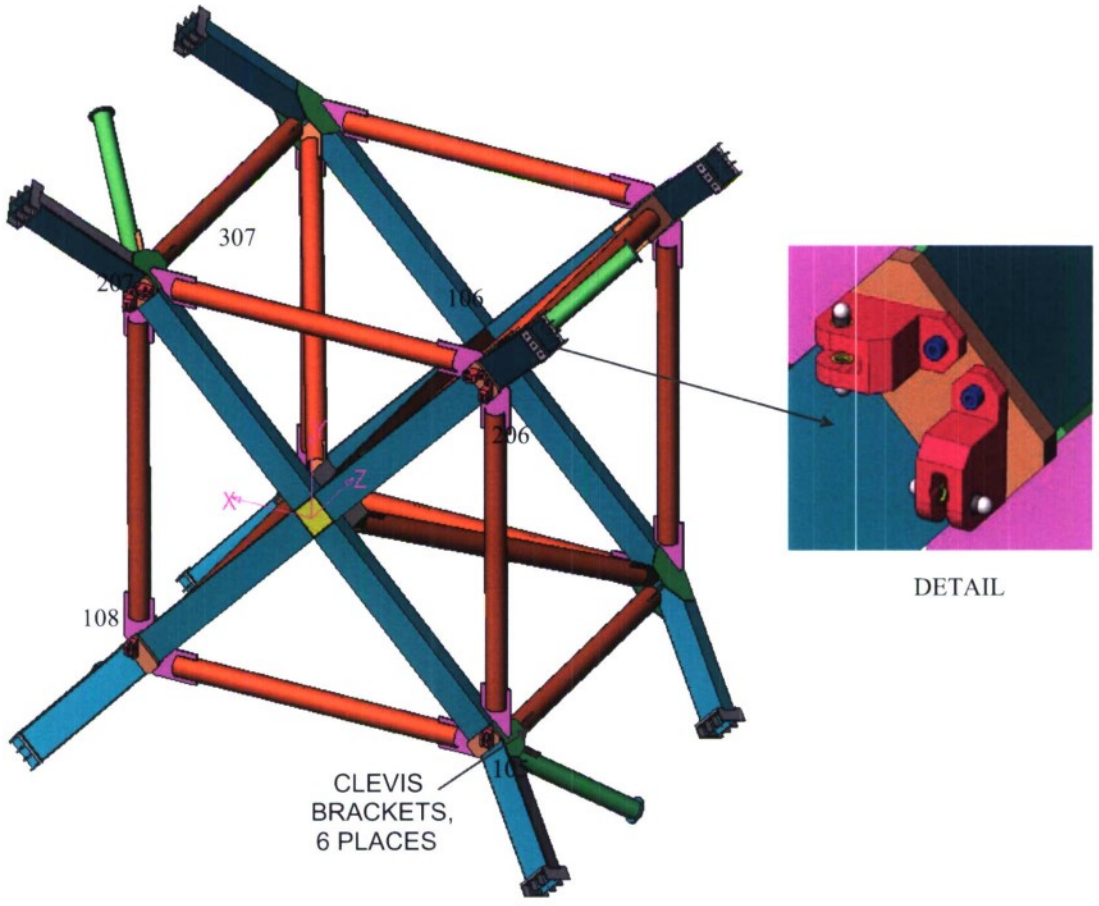


Figure 45. Apex clevis point locations.

The clevis positions in the apex coordinate frame are:

			Apex local coordinates			Primary reflector coordinates		
	Leg	point	XX (inches)	YY (inches)	ZZ (inches)	XX (inches)	YY (inches)	ZZ (inches)
Clevis Work Points	L5	105	–36.779	–33.076	–1.872	–36.779	–33.076	586.892
	L4	106	–33.076	36.779	–1.872	–33.076	36.779	586.892
	L1	108	36.779	–33.076	–1.872	36.779	–33.076	586.892
	L6	206	–36.601	33.076	–1.872	–36.601	33.076	586.892
	L2	207	36.601	33.076	–1.872	36.601	33.076	586.892
	L3	307	33.076	36.601	–1.872	33.076	36.601	586.892

Using those locations and the nominal leg lengths of 50.180", it is possible to develop the equations to be used for controlling the subreflector.

If the subreflector is to be translated by dX, dY, dZ, and oriented at an angle θx, θy, θz, then the desired X,Y,Z position of each of the subreflector leg end points is given by:

$X_{L1}=X_{ov}+dX+R_1+Z_0\theta_y$
$Y_{L1}=Y_{ov}+dY-Z_0\theta_x+R_1\theta_z$
$Z_{L1}=Z_{ov}+dZ+Z_0-R_1\theta_y$
$X_{L2}=X_{ov}+dX+R_2+Z_0\theta_y$
$Y_{L2}=Y_{ov}+dY-Z_0\theta_x+R_2\theta_z$
$Z_{L2}=Z_{ov}+dZ+Z_0-R_2\theta_y$
$X_{L3}=X_{ov}+dX-R_3\theta_z+Z_0\theta_y$
$Y_{L3}=Y_{ov}+dY+R_3-Z_0\theta_x$
$Z_{L3}=Z_{ov}+dZ+Z_0+R_3\theta_x$
$X_{L4}=X_{ov}+dX-R_4\theta_z+Z_0\theta_y$
$Y_{L4}=Y_{ov}+dY+R_4-Z_0\theta_x$
$Z_{L4}=Z_{ov}+dZ+Z_0+R_4\theta_x$
$X_{L5}=X_{ov}+dX-R_5+Z_0\theta_y$
$Y_{L5}=Y_{ov}+dY-Z_0\theta_x-R_5\theta_z$
$Z_{L5}=Z_{ov}+dZ+Z_0+R_5\theta_y$
$X_{L6}=X_{ov}+dX-R_6+Z_0\theta_y$
$Y_{L6}=Y_{ov}+dY-Z_0\theta_x-R_6\theta_z$
$Z_{L6}=Z_{ov}+dZ+Z_0+R_6\theta_y$

where $X_{ov} = 0"$, $Y_{ov} = 0"$, $Z_{ov} = 532.839"$ is the subreflector vertex position in the primary coordinate system.

Given those values, and the known apex mounting points of the hexapod legs, the length of each of the legs can be found using:

$L_i = \sqrt{(XX_{Li} - X_{Li})^2 + (YY_{Li} - Y_{Li})^2 + (ZZ_{Li} - Z_{Li})^2}$, where i is the number of the leg as shown in Figure 46, below.

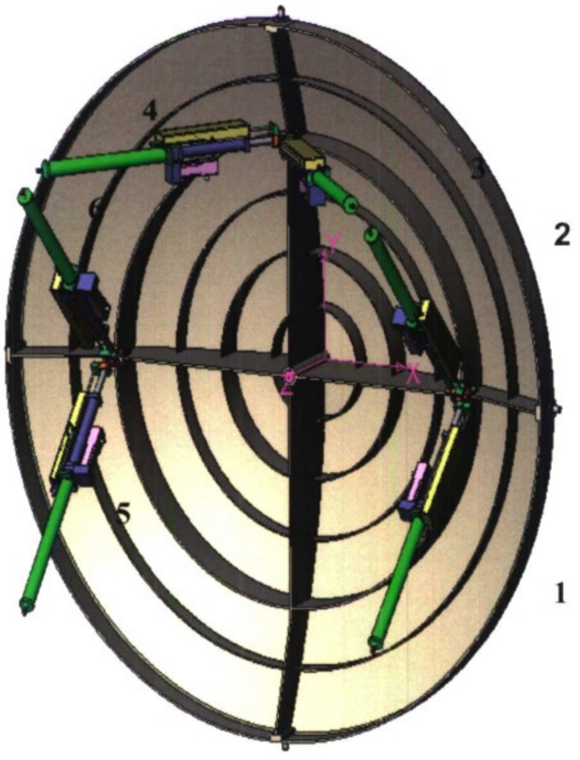


Figure 46. Hexapod actuator numbering.

13.4 CALCULATION OF HEXAPOD MOTION VERSUS ELEVATION

At the rigging angle, the hexapod actuators are set to their nominal length of 50.180". The antenna is a homologous design, meaning that the primary will change shape as the elevation changes, but it will always be a parabola. To correctly focus the antenna, the subreflector is required to move versus elevation to compensate for the changing shape of the primary. There are three adjustments that need to be made to the subreflector. The dominant change is in the Z direction (i.e., axial or piston motion). Additionally, the Y direction and θ_x tilt are adjusted to compensate for the varying sag of the quadrapod and subreflector structure. Each of those corrections is of the form:

$A(\cos E - \cos E_0) + B(\sin E - \sin E_0)$, where E_0 is the rigging angle and E is the current elevation.

Expected Z travel from 0 to 90 is ~97 mm, passing through the nominal length, and 0 Y and θ_x positions at the rigging angle. The A and B coefficients for each correction will be determined empirically during commissioning, first with the laser radar and then using observations of stars.

Typical operation of the subreflector will use a lookup table. Each of the optimum leg lengths will be stored as a function of elevation, and the hexapod servo loop will close the loop on those leg positions. When offset feeds are required, a static adjustment to the legs changing tilt and position of the subreflector will be added to the lookup table lengths. This will allow each of the offset feeds to be placed at the focus of the antenna. The required translation and tilts of the subreflector will also be determined empirically during commissioning.

13.5 HEXAPOD HOMING PROCEDURE

In the case of initial power up, and after any power failures, the hexapod will have to be homed. Each individual actuator has an external incremental encoder with index position. The index position is used to define the home location. The hexapod homing procedure is shown in Figure 47. The sub-procedures used in the homing process are given in Figure 48 and Figure 49. The ACU implements these procedures, and are accessible via a button on the maintenance computer display.

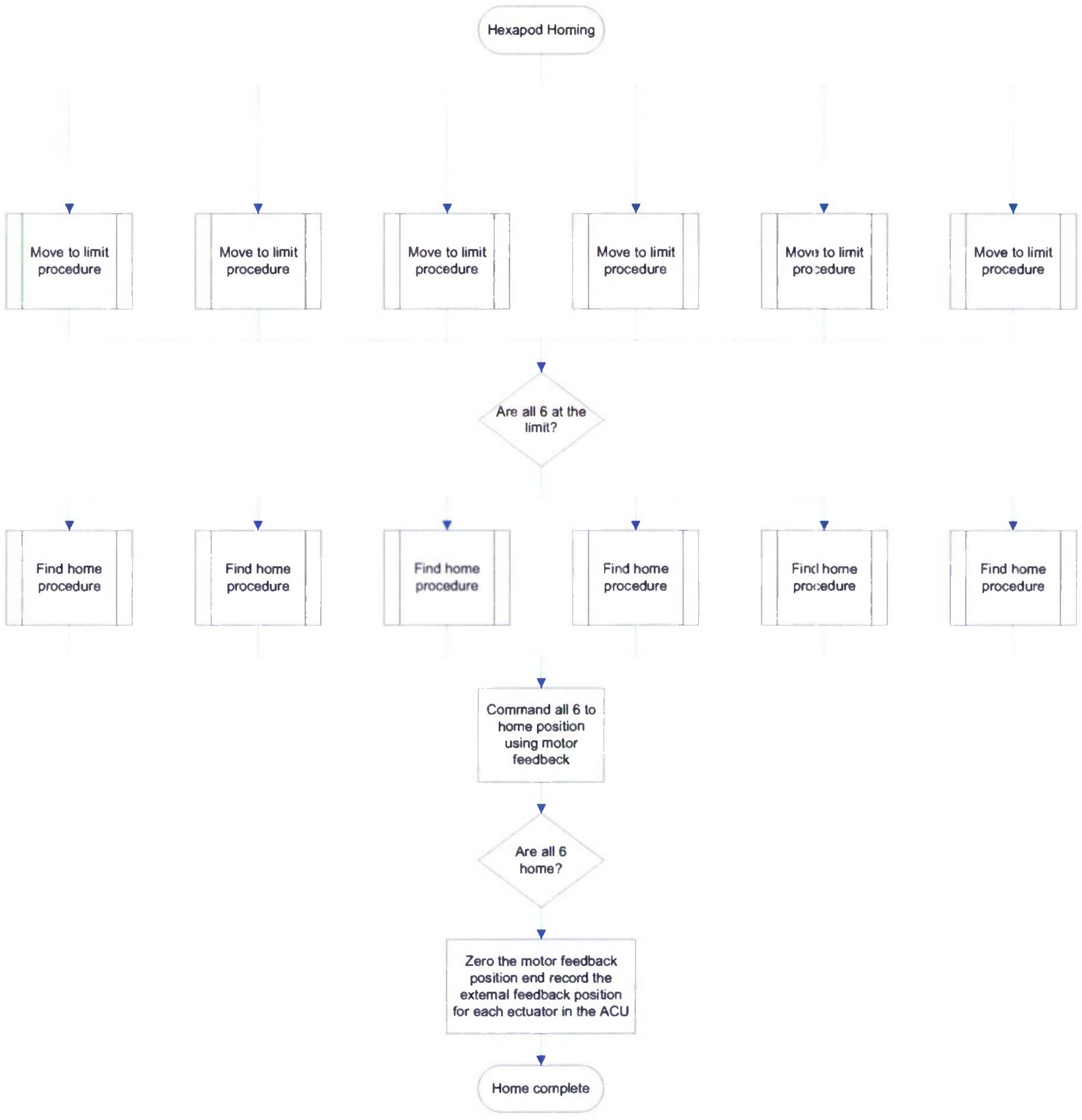


Figure 47. Hexapod homing flowchart.

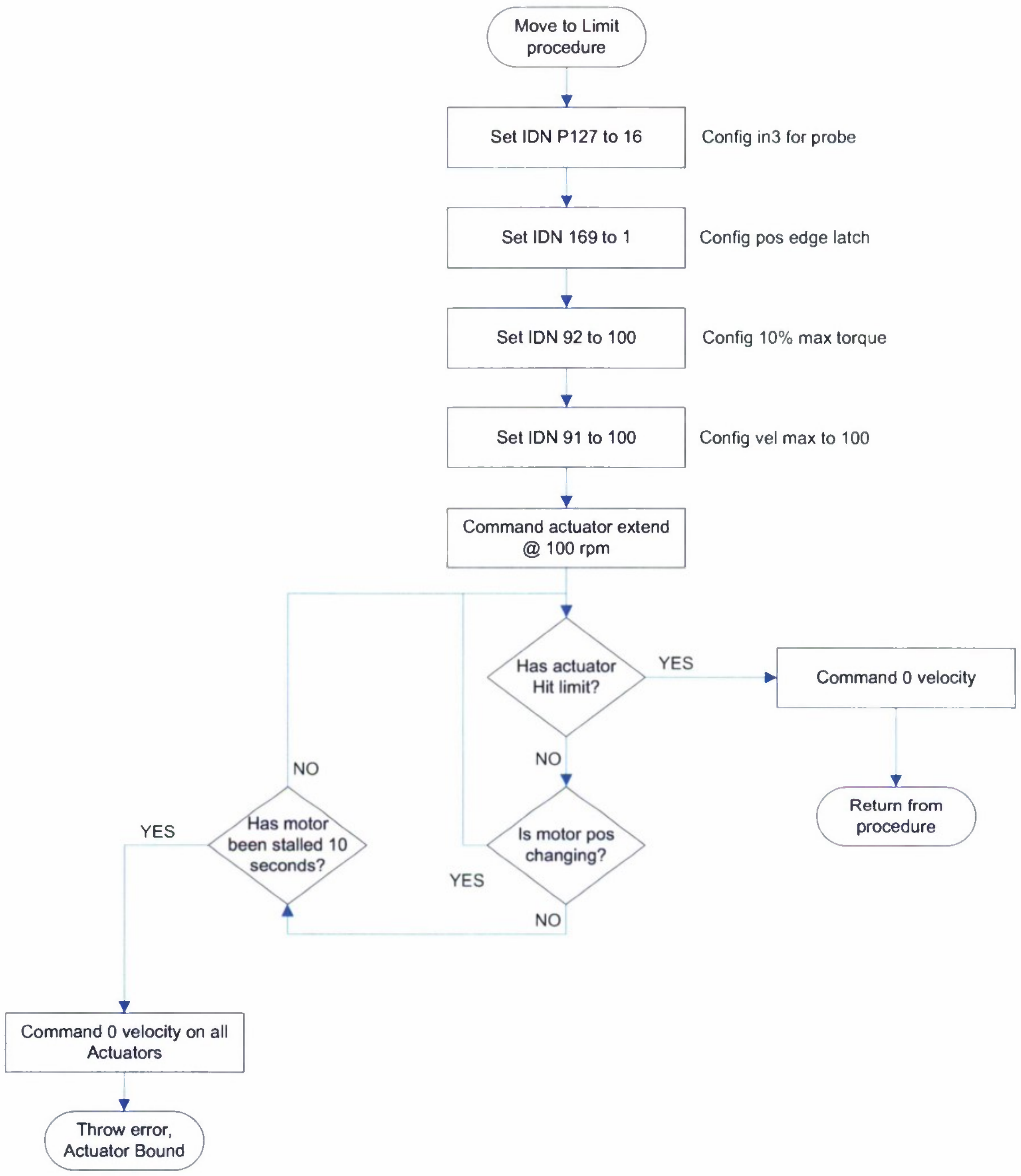


Figure 48. Move-to-limit procedure flowchart.

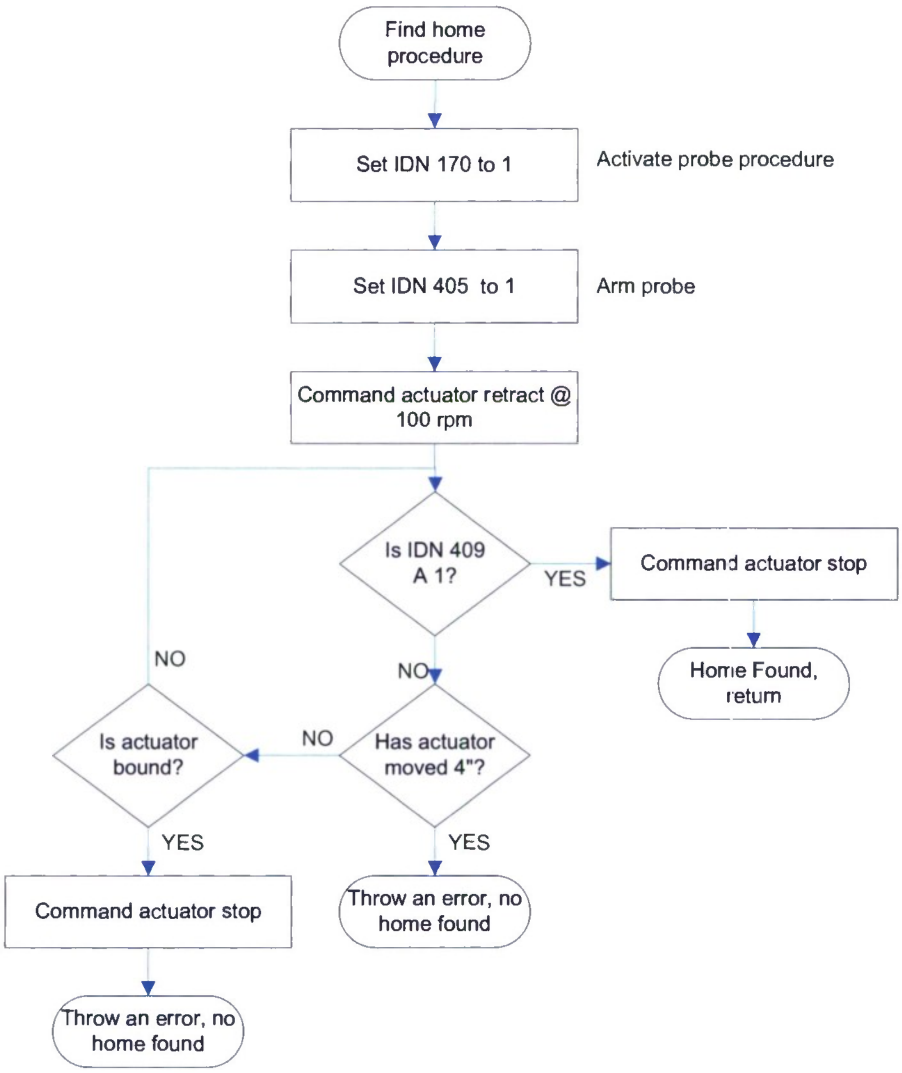


Figure 49. Actuator homing procedure flowchart.

This page intentionally left blank.

14. LIMIT SWITCH ASSEMBLIES

One limit switch package is required for each axis. The switches are implemented differently for the two axes. These assemblies provide for pre-limits, final limits, and RF sector switches, as well as providing for any future needs.

14.1 AZIMUTH AXIS

In azimuth, a pinion is driven off of the bull gear which in turn drives a gear arrangement that provides one full revolution of the cams that actuate the switches. The antenna is capable of rotating ~ 700 degree from stop to stop. This means that for roughly two revolutions of the antenna, the cams should rotate approximately one revolution. Note that the gear ratio does not have to be selected to be exactly one rev of the cams, rather as close to but less than one rev as possible while using only stock gears and rotary limit switches. The reason for maximum cam travel is to provide for adequate resolution when setting the cam positions.

The existing azimuth limit assembly uses a pinion that should be reused, as THE AZIMUTH BULL GEAR IS NOT A STANDARD GEAR PROFILE. The pinion is 6.5" in diameter. The azimuth bull gear diameter is 141". Azimuth requires twelve (12) contacts. Four switches will be used by the PLC for limits, four will be used for RF sectors, and there are four for future use.

The rotary limit switches used in the azimuth assembly shall be GEMCO rotary limit switches. They have a variety of gear reduction ratios to choose from from 5:1 to 1000:1. Each rotary limit switch shall contain four SPDT contacts with the standard 25 degree cams. Ten additional blank 360 cams should be procured with the switches to allow for custom cams to be made later if required.

Physical mounting of this assembly to the antenna structure shall be bolted and provide adjustment to obtain proper gear meshes. This shall be placed in locations that are accessible by personnel, as the switch cams are adjusted periodically. The mounting should allow for cabling to exit the rotary switch housings and be routed to the antenna control system without interfering with other components on the antenna.

14.2 ELEVATION AXIS

The elevation limit package shall consist of Allen Bradley 802 T-APD limit switches stacked side-by side mounted to the yoke near the elevation drives. The cams in this case are blocks that are bolted to the side of the elevation sector gear. Each sector gear shall have four (4) switches. To adjust a limit position, the cam block is removed and modified. The switches should be able to function under oil and water splashing conditions (i.e., hose burst on lube system or water cooling system).

This page intentionally left blank.

15. DRIVE TRAIN

The drive train of the system will not be detailed here, but the pertinent information for the control system will be repeated. The parts of the drive train that are important include: gear ratio, torque available, brake control and timing.

There are three different gearbox designs for the system. Azimuth requires two different gearbox types depending on the mount location (old position vs. new position). The three gearbox types are shown in Figure 50, and the azimuth locations are shown in Figure 51. Elevation gearbox mounting locations are shown in Figure 52.

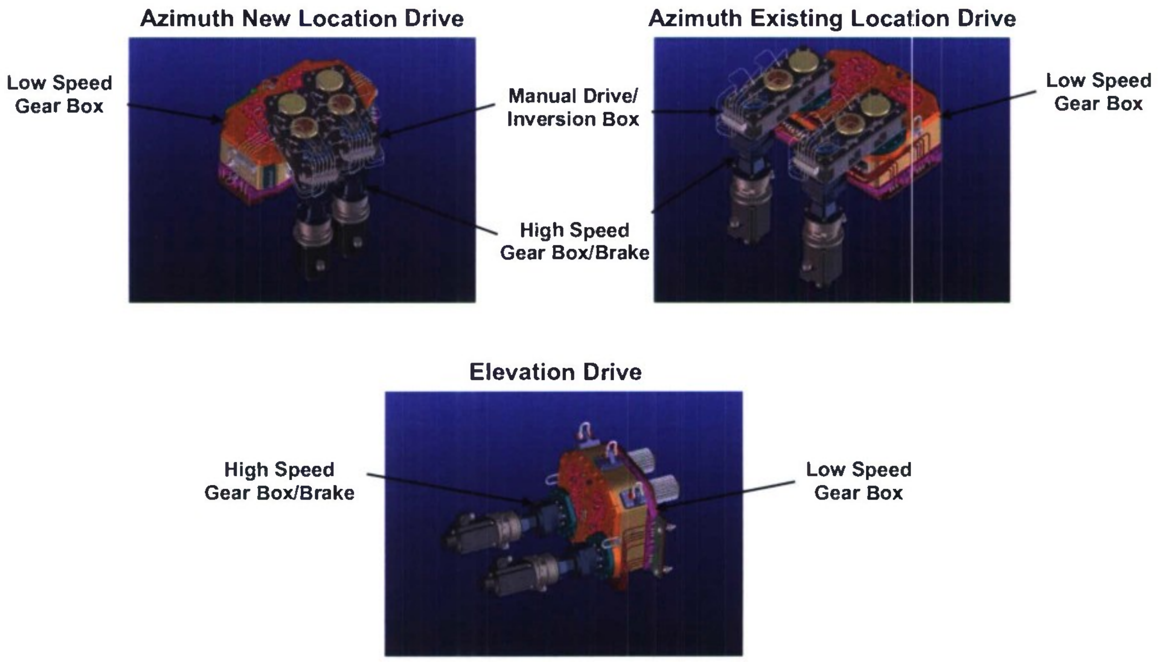

Azimuth New Location Drive
Azimuth Existing Location Drive
Low Speed Gear Box
Manual Drive/ Inversion Box
Low Speed Gear Box
High Speed Gear Box/Brake
Elevation Drive
High Speed Gear Box/Brake
Low Speed Gear Box

Figure 50. Azimuth and elevation gearbox CAD models.

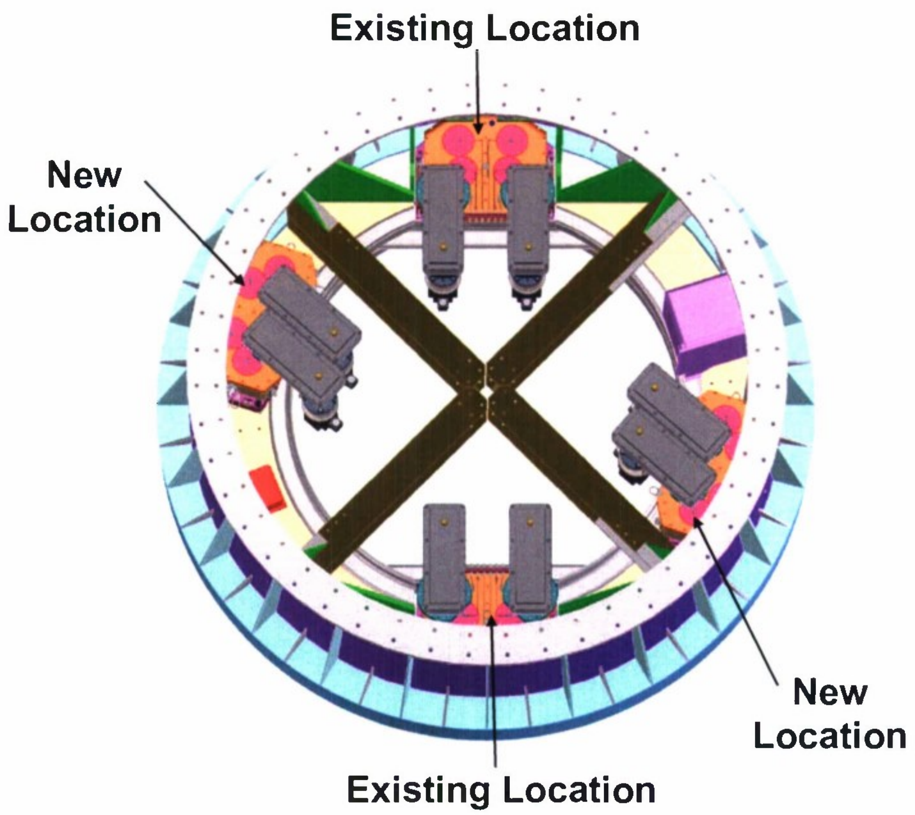


Figure 51. Azimuth gearbox locations.

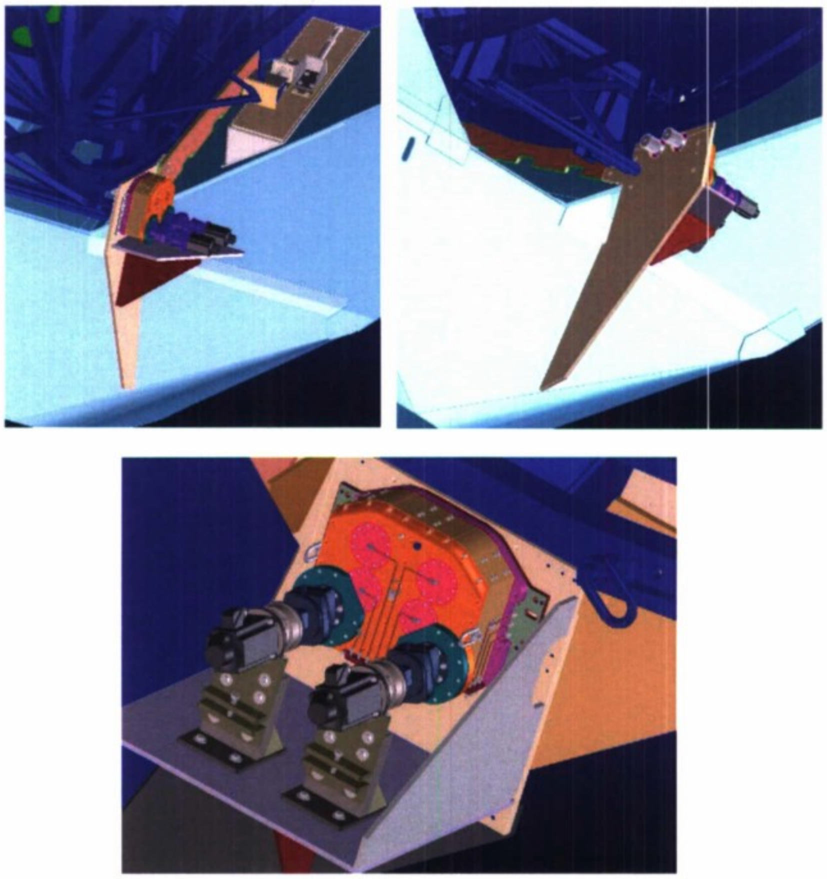


Figure 52. Elevation gearbox locations.

Each pinion on the gearboxes has a shear key that is used to protect the bull gear from damage. The pinon is designed to shear at loads higher than the control system can impart, but less than the bull gear can withstand. Details on the drive pinion shear keys are shown in Figure 53.

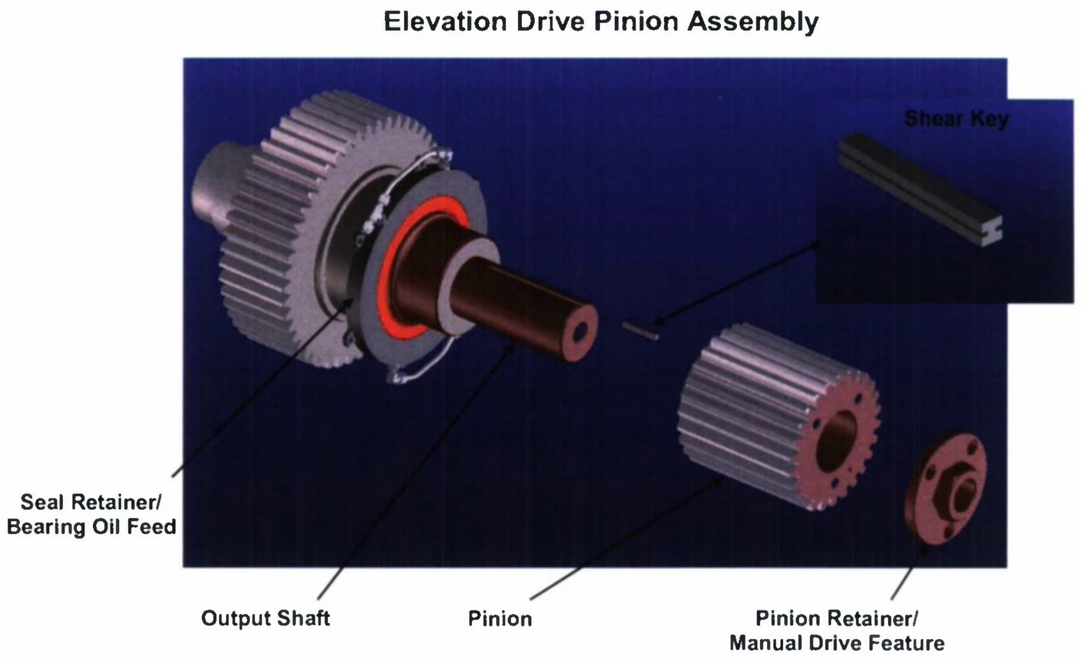

Elevation Drive Pinion Assembly
Shear Key
Seal Retainer/ Bearing Oil Feed
Output Shaft
Pinion
Pinion Retainer/ Manual Drive Feature

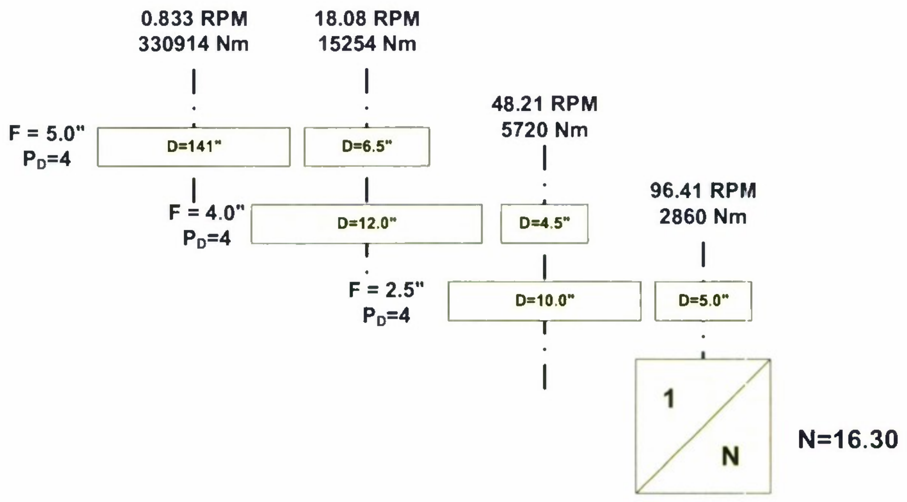
Figure 53. Drive pinion assembly.

15.1 GEAR RATIO

15.1.1 Azimuth Axis

The azimuth gear ratio is 1888:1, with a 141" bull gear. This implies that the high-speed and low speed gearboxes have an aggregate gear-ratio of 87:1 to reduce the motor spindle speed down to the proper main pinion speed. At full rated speed of the antenna (5 degrees per second), the motor spindles will be rotating at 1573 revolutions per minute.

A schematic of the azimuth axis gear train is shown in Figure 54.



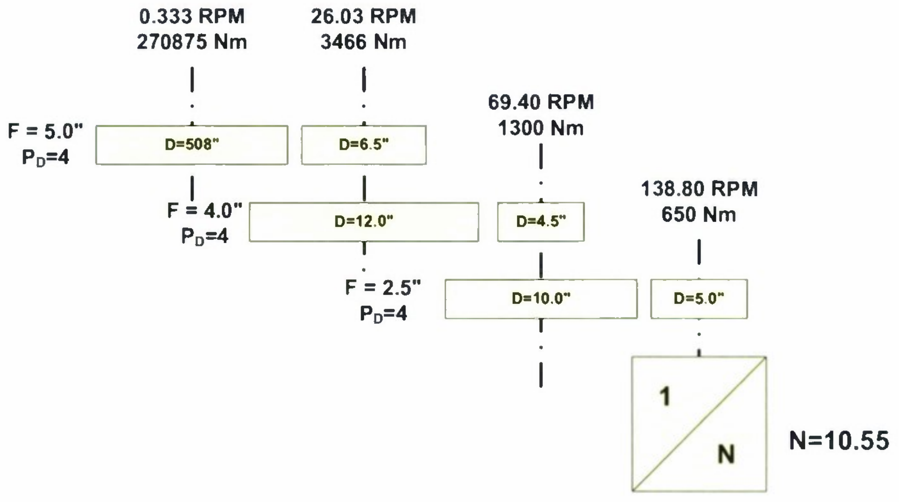
The overall ratio is $\left(\frac{141}{6.5}\right)\left(\frac{12.0}{4.5}\right)\left(\frac{10}{5}\right)(16.30) = 1888:1$

Figure 54. Azimuth gear train schematic.

15.1.2 Elevation Axis

The elevation gear ratio is 4101.5:1, with a 508" bull gear. The high-speed and low speed gearboxes will have an aggregate gear-ratio of 52.48:1 to obtain the proper pinion speed. At 2 degrees per second, the motor spindles will be rotating at 1367 revolutions per minute.

The elevation axis gear train is shown in Figure 55.



The overall ratio is $\left(\frac{508}{6.5}\right)\left(\frac{12.0}{4.5}\right)\left(\frac{10}{5}\right)(10.55) = 4395:1$

Figure 55. Elevation gear train.

15.2 MOTOR TORQUES

The MHD115A-024 motors have a rated torque of 24.9 N m, with a continuous torque at standstill of 36.5 Nm.

15.3 BRAKES

Each high speed gearbox (KEB G42-M NEMA180 16.3:1 or 10.55:1) has a dynamic brake installed (KEB 06.17.67X) that is capable of dissipating the antenna inertia and bringing the antenna to a stop (as a group, each individual brake is not capable of this). Plots of the timing of the brake engagement and release are shown in Figure 56 and Figure 57, respectively. At 24 VDC, each brake will draw 2.7A of current to hold the solenoid in (i.e., brakes OFF). When power is lost, the brakes are automaticly applied using springs. The braking torque is 204 inch-lbs, resulting in 6 degrees of antenna rotation to stop from full speed in azimuth, and 0.5 degrees of antenna rotation to stop from full speed in azimuth.

Brake Instruction Manual

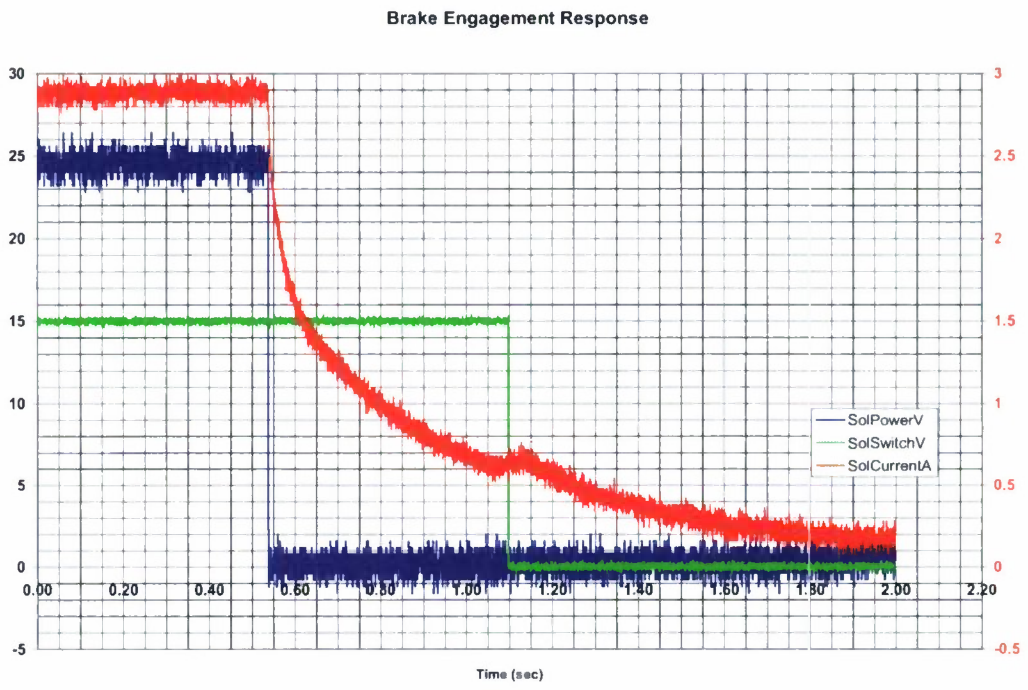


Figure 56. Brake engagement timing.

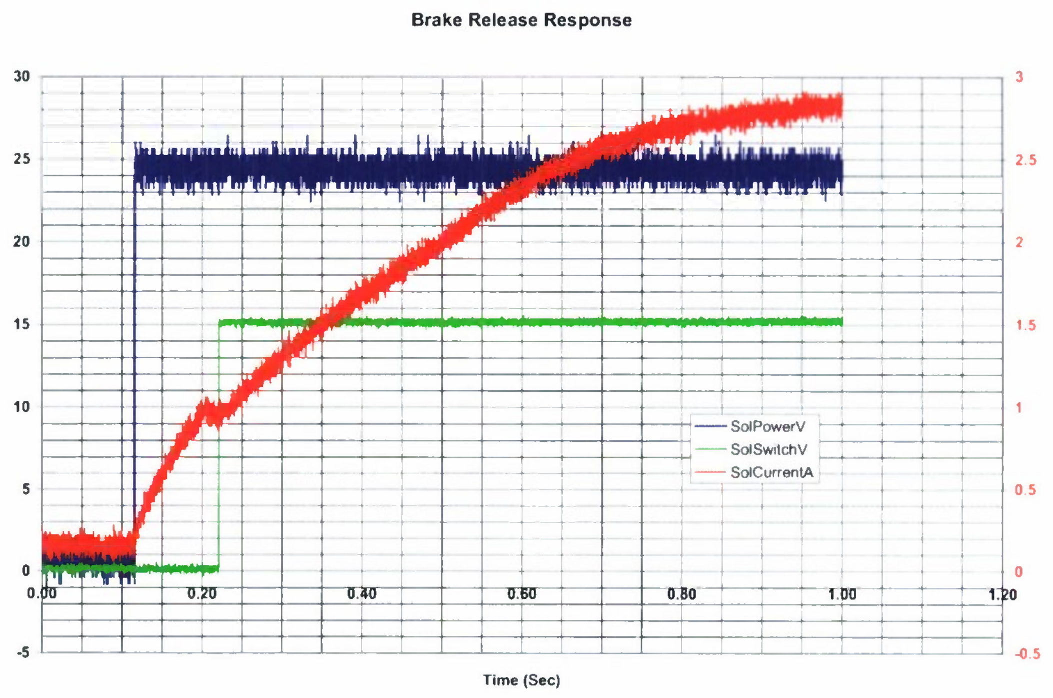


Figure 57. Brake release timing.

Each brake has a limit switch that the PLC will utilize to determine the state of the brake (off or on).

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE 7 December 2010	**2. REPORT TYPE** Project Report	**3. DATES COVERED** *(From - To)*
4. TITLE AND SUBTITLE Haystack Antenna Control System Design Document		**5a. CONTRACT NUMBER** FA8721-05-C-0002
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) James V. Eshbaugh and Micheal T. Clarke, Group 92; Greg W. Maglathlin, Maglathlin Consulting Services, LLC		**5d. PROJECT NUMBER** 1247
		5e. TASK NUMBER 111
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AND ADDRESS(ES) MIT Lincoln Laboratory, 244 Wood Street, Lexington, MA 02420-9108		**8. PERFORMING ORGANIZATION REPORT NUMBER** HUSIR-8
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) 850 ELSG/NS, 11 Barksdale Street, Building 1614, Hanscom AFB, 01731-1700		**10. SPONSOR/MONITOR'S ACRONYM(S)**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S) ESC-TR-2010-071

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

MIT Lincoln Laboratory is carrying out a program to upgrade the existing Haystack radar by adding a W-band (96 GHz) radar capability. The existing X-band radar (10 GHz) will remain intact and the resulting radar will be capable of either simultaneous dual-band or single-band operation. This document describes the design of the control system for the HUSIR antenna. A new higher precision antenna is being installed, allowing the antenna to be used efficiently at W-Band. At W-Band, the system will have a very narrow beam that needs to be accurately pointed by a new control system while operating at higher velocities than the old hydraulic system.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	Same as report	135	**19b. TELEPHONE NUMBER** *(include area code)*

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

This page intentionally left blank.