MULTI-ACCESS LASER COMMUNICATIONS TERMINAL

FINAL REPORT

CONTRACT NO. NAS5-31170

Submitted to:

NASA Goddard Space Flight Center
Greenbelt, Maryland 20771

November 3, 1992

Laser Data Technology, Inc.
Saint Louis, Missouri

(NASA-CR-191251) MULTI-ACCESS
LASER COMMUNICATIONS TERMINAL Final Report (Laser Data Technology)

305 p

N94-23830

Unclas

G3/32 0204535
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 EXECUTIVE SUMMARY</td>
<td>1</td>
</tr>
<tr>
<td>2.0 DESIGN OVERVIEW</td>
<td>6</td>
</tr>
<tr>
<td>2.1 Optical</td>
<td>6</td>
</tr>
<tr>
<td>2.1.1 Telescope</td>
<td>6</td>
</tr>
<tr>
<td>2.1.2 Focal-plane Optics</td>
<td>6</td>
</tr>
<tr>
<td>2.2 Mechanical</td>
<td>6</td>
</tr>
<tr>
<td>2.2.1 Disk and Arm Assemblies</td>
<td>6</td>
</tr>
<tr>
<td>2.3 Acquisition and Tracking Design Constraints</td>
<td>14</td>
</tr>
<tr>
<td>2.4 Electronics</td>
<td>22</td>
</tr>
<tr>
<td>2.4.1 Motion Control Circuitry</td>
<td>22</td>
</tr>
<tr>
<td>2.4.2 Communications Circuits</td>
<td>31</td>
</tr>
<tr>
<td>2.5 Software</td>
<td>31</td>
</tr>
<tr>
<td>2.5.1 Executive</td>
<td>35</td>
</tr>
<tr>
<td>2.5.2 Angle Processor Software</td>
<td>40</td>
</tr>
<tr>
<td>2.5.3 Motion Processor Software</td>
<td>40</td>
</tr>
<tr>
<td>3.0 EVALUATION TEST DATA</td>
<td>42</td>
</tr>
<tr>
<td>3.1 Optical Tests</td>
<td>42</td>
</tr>
<tr>
<td>3.1.1 Telescope Performance</td>
<td>42</td>
</tr>
<tr>
<td>3.1.2 Arm Performance</td>
<td>43</td>
</tr>
<tr>
<td>3.2 Mechanical Tests</td>
<td>43</td>
</tr>
<tr>
<td>3.2.1 Resolution/backlash/hysteresis</td>
<td>43</td>
</tr>
<tr>
<td>3.2.2 Slewing Rate</td>
<td>49</td>
</tr>
<tr>
<td>3.3 Acquisition Time</td>
<td>54</td>
</tr>
<tr>
<td>3.4 Tracking Error</td>
<td>54</td>
</tr>
<tr>
<td>3.5 Weight Analysis</td>
<td>54</td>
</tr>
<tr>
<td>4.0 AREAS FOR FURTHER DEVELOPMENT</td>
<td>58</td>
</tr>
<tr>
<td>5.0 CONCLUSIONS</td>
<td>59</td>
</tr>
<tr>
<td>APPENDIX A TELESCOPE OPTICAL DESIGN SPECIFICATIONS</td>
<td>A1-A??</td>
</tr>
<tr>
<td>APPENDIX B COMPLETE ELECTRONIC SCHEMATICS</td>
<td>B1-B??</td>
</tr>
<tr>
<td>APPENDIX C COMPLETE SOFTWARE LISTINGS</td>
<td>C1-C??</td>
</tr>
</tbody>
</table>
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>Multi-access Laser Communication System</td>
</tr>
<tr>
<td>1-2</td>
<td>Prototype Terminal</td>
</tr>
<tr>
<td>1-3</td>
<td>Movable Image Pickup System (MIPS) Concept</td>
</tr>
<tr>
<td>1-4</td>
<td>Disk and Arm Rotation System</td>
</tr>
<tr>
<td>1-5</td>
<td>Goals of Multi-access Terminal</td>
</tr>
<tr>
<td>2.1-1</td>
<td>Glass Telescope Elements for Space Design</td>
</tr>
<tr>
<td>2.1-2</td>
<td>Glass Telescope Spot Diagrams - Positions 1 &amp; 2</td>
</tr>
<tr>
<td>2.1-3</td>
<td>Glass Telescope Spot Diagrams - Positions 3 &amp; 4</td>
</tr>
<tr>
<td>2.1-4</td>
<td>Plastic Telescope Elements for Prototype Design</td>
</tr>
<tr>
<td>2.1-5</td>
<td>Use of Plastic Elements for Prototype System</td>
</tr>
<tr>
<td>2.1-6</td>
<td>Movable Image Pickup Optics</td>
</tr>
<tr>
<td>2.1-7</td>
<td>Completed Arm Assembly</td>
</tr>
<tr>
<td>2.2-1</td>
<td>Arm and Disk Coordinates in the Telescope Field of View</td>
</tr>
<tr>
<td>2.2-2</td>
<td>Worm Drive Assembly</td>
</tr>
<tr>
<td>2.2-3</td>
<td>Disk Support Structure</td>
</tr>
<tr>
<td>2.2-4</td>
<td>Disk Assembly End Plate</td>
</tr>
<tr>
<td>2.2-5</td>
<td>Disk/Arm Resolution</td>
</tr>
<tr>
<td>2.3-1</td>
<td>Acquisition Approach</td>
</tr>
<tr>
<td>2.4-1</td>
<td>System Electronics Layout</td>
</tr>
<tr>
<td>2.4-2</td>
<td>Motor Controller Chip Block Diagram</td>
</tr>
<tr>
<td>2.4-3</td>
<td>Total Control Loop</td>
</tr>
<tr>
<td>2.4-4</td>
<td>Motor and Total Loop Responses</td>
</tr>
<tr>
<td>2.4-5</td>
<td>Motor Switching Schematic for Channel 2</td>
</tr>
<tr>
<td>2.4-6</td>
<td>Quadrant Detector Electronics Schematic</td>
</tr>
<tr>
<td>2.4-7</td>
<td>Receiver Waveform Showing Data and Tone</td>
</tr>
<tr>
<td>2.4-8</td>
<td>Angle Processor Electronics Schematic</td>
</tr>
<tr>
<td>2.4-9</td>
<td>Wideband Amplifier Schematic</td>
</tr>
<tr>
<td>2.4-10</td>
<td>Laser Driver Circuit Schematic</td>
</tr>
<tr>
<td>2.5-1</td>
<td>Control Software Top-level Architecture</td>
</tr>
<tr>
<td>2.5-2</td>
<td>Error Signal Amplitude Detection</td>
</tr>
<tr>
<td>3.1-1</td>
<td>Elevation Quadrant Scan</td>
</tr>
<tr>
<td>3.1-1</td>
<td>Azimuth Quadrant Scan</td>
</tr>
<tr>
<td>3.1-3</td>
<td>Spot Shape Measurement</td>
</tr>
<tr>
<td>3.2-1</td>
<td>Hysteresis Plot - Before Adjustment</td>
</tr>
<tr>
<td>3.2-2</td>
<td>Hysteresis Plot - After Adjustment</td>
</tr>
<tr>
<td>3.2-3</td>
<td>Backlash Test</td>
</tr>
<tr>
<td>3.2-4</td>
<td>Spot Shape Measurement</td>
</tr>
<tr>
<td>3.2-5</td>
<td>Motor Step Response - Before Cleaning</td>
</tr>
<tr>
<td>3.2-6</td>
<td>Motor Step Response - After Cleaning</td>
</tr>
<tr>
<td>3.4-1</td>
<td>Tracking Error Parallel to Arm</td>
</tr>
<tr>
<td>3.4-2</td>
<td>Tracking Error Perpendicular to Arm</td>
</tr>
<tr>
<td>3.5-1</td>
<td>Multiple-Access Terminal Weight Analysis</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>2.3-1</td>
<td>Acquisition Error Budget</td>
</tr>
<tr>
<td>2.5-1</td>
<td>Manual Operation Software Organization</td>
</tr>
<tr>
<td>2.5-2</td>
<td>Manual Control Menu</td>
</tr>
<tr>
<td>2.5-3</td>
<td>Operational Satellite Software Organization</td>
</tr>
<tr>
<td>2.5-4</td>
<td>Automatic Real-time Operation Software Organization</td>
</tr>
<tr>
<td>2.5-5</td>
<td>Angle Processing Routines</td>
</tr>
</tbody>
</table>
1.0 EXECUTIVE SUMMARY

The Optical Multi-Access (OMA) Terminal is capable of establishing up to six simultaneous high-data-rate communication links between low-Earth-orbit satellites and a host satellite at synchronous orbit with only one 16-inch-diameter antenna on the synchronous satellite. The advantage over equivalent RF systems in space weight, power, and swept volume is great when applied to NASA satellite communications networks (Figure 1-1).

Figure 1-2 is a photo of the 3-channel prototype constructed under the present contract to demonstrate the feasibility of the concept. The telescope has a 10-inch clear aperture and a 22° full field of view. It consists of 4 refractive elements to achieve a telecentric focus, i.e., the focused beam is normal to the focal plane at all field angles. This feature permits image pick-up optics in the focal plane to track satellite images without tilting their optic axes to accommodate field angle. Figure 1-3 shows the geometry of the image-pick-up concept, and Figure 1-4 shows the coordinate system of the swinging arm and disk mechanism for image pick-up. Optics in the arm relay the telescope focus to a communications and tracking receiver and introduce the transmitted beacon beam on a path collinear with the receive path. The electronic circuits for the communications and tracking receivers are contained on the arm and disk assemblies and relay signals to an associated PC-based operator's console for control of the arm and disk motor drive through a flexible cable which permits ±240° travel for each arm and disk assembly. Power supplies and laser transmitters are mounted in the cradle for the telescope. A single-mode fiber in the cable is used to carry the laser transmitter signal to the arm optics. Figure 1-5 shows the promise of the optical multi-access terminal towards which the prototype effort worked. The emphasis in the prototype development was the demonstration of the unique aspects of the concept, and where possible, cost avoidance compromises were implemented in areas already proven on other programs.

The design details are described in Section 2, the prototype test results in Section 3, additional development required in Section 4, and conclusions in Section 5.
MULTI-ACCESS LASER COMMUNICATIONS SYSTEM

- 6 Simultaneous Duplex Channels -
- 3 Mbps Each Downlink Channel -
- Command Data Rate on Each Uplink Channel -

FIGURE 1-1

PROTOTYPE MULTI-ACCESS TERMINAL

FIGURE 1-2
MOVABLE IMAGE PICKUP SYSTEM (MIPS) CONCEPT

EACH ANGULAR POINT IN THE FIELD OF VIEW IS REPRESENTED BY A POINT IN THE IMAGE PLANE

FIXED TELESCOPE
(22 DEGREE FOV)

IMAGE PLANE

TYPICAL SATELLITE MOVEMENT ACROSS IMAGE PLANE

EACH SATELLITE IS TRACED BY A MOVABLE IMAGE PICKUP UNIT

SIDE

FRONT

FIGURE 1-3
GOALS OF BASELINE MULTI-ACCESS TERMINAL

- **SIMULTANEOUS COMMUNICATION WITH SIX INDEPENDENT ASYNCHRONOUS LINKS**
- **FULL COVERAGE OF EARTH AND LOW ORBITs**
- **LIGHT WEIGHT (≈150 POUNDS)**
- **SMALL SIZE (10-INCH-DIAMETER CLEAR APERTURE)**
- **LOW POWER (<100 WATTS)**
- **LOW SWEPT VOLUME (NO GIMBALS)**
- **3 MBPS PER COMM LINK (CONCEPT SUPPORTS HIGHER RATE)**
- **10 KBPS FORWARD RATE PER LINK**
2.0 DESIGN OVERVIEW

2.1 Optical

2.1.1 Telescope - The telescope elements are shown in Figure 2.1-1 for the space design. Radiation hardened glass is specified for long life in space. Figure 2.1-2 and -3 show the spot diagram at the focal plane for various off-axis positions with a 300-µm circle for reference. This diameter represents 480 µrad in the far field of the system. Figure 2.1-4 depicts the plastic design used in the prototype to save cost. This material is unsuitable for space due to darkening effects of radiation but provides comparable optical performance for demonstration in the laboratory. Figure 2.1-5 compares the space and prototype designs. Appendix A contains the detailed lens specifications for both the glass and plastic designs.

2.1.2 Focal-plane optics - Figure 2.1-6 shows the arrangement of optics for the space design. The figure is arranged with the arms aligned one over the other in a vertical plane and show the locations of the channels with respect to the telescope focal plane. The receive channels are designed to relay the focus to the quadrant detector through a narrow-band optical filter for transmitter rejection. The transmitter signals are introduced by means of a single-mode fiber whose image is formed at the telescope image plane and coaligned with the arm receiver optical axis. The beamwidth is 150 µrad FWHM, so the image is 94 µm, FWHM. Because the telescope forms a telecentric image, the arms need only be located at the correct spot to receive the light, and no tilt of the optic axis as a function of field angle is required. In the prototype we included Channel 1, a channel between 2 and 3, and Channel 4, renamed Channels 1, 2, and 3, respectively. Figure 2.1-7 is a photograph of a complete arm for Channel 3 fully assembled with optics and electronics.

2.2 Mechanical

2.2.1 Disk and arm assemblies - The arm is mounted on a 354-tooth worm-gear ring and attached to a second worm-gear ring. Moving the two rings together moves the arm pivot point through more than one complete revolution, while differential movement of the rings with respect to each other swings the arm on its pivot. Therefore, by driving the worm on each ring (or "disk", as we have referred to them previously) to the proper location, any point in the field of the telescope can be
Circles are centered at transmit beam centers.

Circle diameter = .3mm
FIGURE 2.1-3
# USE OF PLASTIC ELEMENTS FOR PROTOTYPE SYSTEM

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>SIMILARITY TO SPACE DESIGN</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUNCTIONAL PERFORMANCE</td>
<td>✓</td>
</tr>
<tr>
<td>SIZE</td>
<td>SLIGHTLY LARGER FOR EQUIVALENT PERFORMANCE</td>
</tr>
<tr>
<td>FIELD OF VIEW</td>
<td>✓</td>
</tr>
<tr>
<td>BACKSCATTER</td>
<td>✓</td>
</tr>
<tr>
<td>ALIGNMENT SENSITIVITY</td>
<td>✓</td>
</tr>
<tr>
<td>MINIMUM SPOT SIZE</td>
<td>CLOSE</td>
</tr>
<tr>
<td>POLARIZATION PRESERVATION</td>
<td>NOT SIMILAR</td>
</tr>
<tr>
<td>WEIGHT</td>
<td>LIGHTER</td>
</tr>
<tr>
<td>THERMAL MISALIGNMENT</td>
<td>NOT SIMILAR</td>
</tr>
</tbody>
</table>

*FIGURE 2.1-5*
addressed. Figure 2.2-1 shows the geometry of the field and the arm and disk angles. For each location in the field, there are two possible orientations of the arm and disk. If a low-Earth-orbit satellite passes directly through or very near the center of the field, there will be a short outage while the disks swing 180° to pick up the signal on the other side of center. These outages will be rare and are readily predicted well in advance.

The worm assembly is shown in Figure 2.2-2. The worm is captive between preloaded bearings with no contribution to backlash. The assembly is spring-loaded into the worm gear to minimize backlash in the disk drive. The motors are of two types. Channel 1 is driven by two MicroMo 3557CR motors with up to 8.5 in-oz of continuous torque. The other two channels are driven by MicroMo 2842 motors with 3 in-oz of torque. The two different types of motors were used to ensure adequate torque margin on at least one channel, while exploring the possible savings in size and weight of the smaller motor. Position feedback for each motor is provided by a Hewlett-Packard HEDS 5010 2000-count-per-revolution incremental optical shaft encoder.

The 3-channel structure is shown in cross-section in Figure 2.2-3 showing the "sandwich" construction of the disk assembly. A photo showing the baseplate with the motor and worm interfaces is presented in Figure 2.2-4. Figure 2.2-5 gives the mechanical constants for the arm and disk mechanical design.

### 2.3 Acquisition and Tracking Design Constraints

The approach for acquisition is simple and straightforward. Spacecraft attitude reference, ephemeris, and mounting uncertainties require that a square 3.5 milliradians on a side be searched at the GEO. Table 2.3-1 presents the analysis of this requirement. The LEO will stare at its uncertainty region and respond as soon as it is illuminated by slewing its narrow transmit beam to reduce the error measured by means of the received signal from the GEO. Consequently, if the GEO will dwell at a given location for a time sufficient for the signal to reach the LEO (1 transit time), the LEO to slew to boresight (say, 30 milliseconds), and return its optical signal (1 transit time), then before it moved to the next scan location, it would know to stop where it is and track. Figure 2.3-1 shows the resulting worst case acquisition times given 100% probability of detection when illumination occurs.
FIGURE 2.2-1 ARM AND DISK COORDINATES IN THE TELESCOPE FIELD OF VIEW
DISK/ARM RESOLUTION

CONTINUOUS MOTOR/WORM DRIVE IMPLEMENTATION

GEAR TEETH PER REVOLUTION
24 PITCH ON 14.75-INCH DIAMETER 354

WORM DRIVE ENCODER RESOLUTION 2000

DISK STEP RESOLUTION $2\pi/354/2000 = 8.87\ \mu\text{RAD}$

CORRESPONDING MOTION AT EDGE OF 116-MM FOCAL PLANE
$8.87\ \mu\text{RAD} \times 116\text{mm}$ 1.03 $\mu\text{m}$

CORRESPONDING FAR-FIELD ANGLE
$1.03\ \mu\text{m}/.625\text{m (FOCAL LENGTH)}$ 1.64 $\mu\text{RAD}$

FOR $D_4/D_3 = 2.56$

ARM STEP ANGULAR RESOLUTION
$8.87\ \mu\text{RAD} \times 2.56$ 22.7 $\mu\text{RAD}$

LINEAR MOTION AT TIP OF ARM
$22.7\ \mu\text{RAD} \times \text{ARM LENGTH (167mm)}$ 3.79 $\mu\text{m}$

CORRESPONDING FAR-FIELD ANGLE
$3.79\ \mu\text{m}/.625\text{m}$ 6.07 $\mu\text{RAD}$

FIGURE 2.2-5
### ACQUISITION ERROR BUDGET

<table>
<thead>
<tr>
<th>GEO ATTITUDE KNOWLEDGE</th>
<th>STEL ATTITUDE MODEL</th>
<th>AZIMUTH</th>
<th>ELEVATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pitch</td>
<td>± 1 mrad</td>
<td>± 1 mrad</td>
<td>0</td>
</tr>
<tr>
<td>Roll</td>
<td>± 1 mrad</td>
<td>0</td>
<td>± 1 mrad</td>
</tr>
<tr>
<td>Yaw</td>
<td>± 3 mrad</td>
<td>± 0.5 mrad</td>
<td>± 0.5 mrad</td>
</tr>
<tr>
<td>OTHER ERRORS</td>
<td>± 0.25 mrad</td>
<td>± 0.2 mrad</td>
<td>± 0.2 mrad</td>
</tr>
<tr>
<td>WORST CASE</td>
<td></td>
<td>± 1.7 mrad</td>
<td>± 1.7 mrad</td>
</tr>
</tbody>
</table>

**TABLE 2.3-1**
ACQUISITION APPROACH

- LEO STARES WITH RECEIVE IFOV GREATER THAN UNCERTAINTY WINDOW
- GEO SCANS WITH 200 µRAD TRANSMIT BEAM AND 500 µRAD IFOV COVERING UNCERTAINTY WINDOW WITH 50% OVERLAPS
- GEO DWELLS LONG ENOUGH AT EACH SPOT FOR LEO TO ACHIEVE FINE TRACKING IF WITHIN TRANSMIT BEAM

ROUND-TRIP DELAY
- CONVERGENCE TIME
TOTAL DWELL TIME
0.27 SECONDS
0.03 SECONDS
0.30 SECONDS

INITIAL ACQUISITION TIME

UNCERTAINTY REGION
3.4 X 3.4 MRAD
# DWELLS
1089
MAXIMUM ACQUISITION TIME
5.4 MINUTES

SUBSEQUENT ACQUISITION TIME

UNCERTAINTY REGION
1.2 X 1.2 MRAD
# DWELLS
136
MAXIMUM ACQUISITION TIME
41 SECONDS

RE-ACQUISITION TIME

UNCERTAINTY REGION
<1 X 1 MRAD
# DWELLS
<100
MAXIMUM ACQUISITION TIME
<30 SECONDS
2.4 Electronics

Figure 2.4-1 presents an overall block diagram of the electronics showing where the various circuits are housed. The computer contains the motor control electronics on a special internal expansion card that plugs into the computer bus. The power supplies, limit-switch relay circuits, and computer interface connectors are mounted in the cradle that supports the telescope and its focal-plane optics assemblies. The quadrant-detector receiver circuits are mounted in the arm assemblies themselves and connect to angle-processing circuitry mounted on the disk assemblies. Connection to the computer from each arm is made via the cradle-mounted circuitry through a flexible cable which permits ±240° travel of each arm.

Summary descriptions of the electronic assemblies are presented below. Detailed schematics of all circuits are contained in Appendix B.

2.4.1 Motion control circuitry - Figure 2.4-2 is a block diagram of the motion-control chip used for control of the motors driving the worm-gear assemblies. The performance of the compensation loop is controlled by downloading the desired gain, cycle time, and digital transfer function parameters.

The overall control loop is shown in Figure 2.4-3. This diagram incorporates the functions of the HCTL-1000 motor-control chip, the motor drive amplifier, the quadrant detector and its angle processor, and the computer control algorithms for open- and closed-loop control of the arm position for each channel. The design of the control loop sought to achieve a 4-Hertz control bandwidth, supported by a 40-Hertz update rate for each channel by the computer. Figure 2.4-4 shows the design Bode plots for the open- and closed-loop motor control transfer functions for a 4-Hertz control loop.

Motion control circuit board - This board contains six Hewlett-Packard HCTL-1000 chips and the associated interface circuitry. This chip accepts input position commands, incremental encoder outputs, and mode-control and loop-compensation parameters and develops the appropriate motor drive word for D/A conversion and amplification. Other circuitry performs multiplexing of the computer bus among the various motor-control chips and output word conversion.
MOTOR CONTROLLER CHIP BLOCK DIAGRAM
HEWLETT-PACKARD HCTL-1100

FIGURE 2.4-2
Figure 2.4-3

TOTAL CONTROL LOOP

LOCATION OF FOCUSED IMAGE
LOCATION OF DETECTOR CENTER

K/s

f_4

f_3

f_2

MOTOR CONTROLLER

DISK POSITIONS

f_x f_2 f_3

AZIMUTH, ELEVATION ANGLES

POINT AHEAD BIAS

-24-
Limit-switch relay circuit board - Each of the three channels has four limit switches to sense the positive and negative limits of motion for the disk pair and for the arm. These switches operate relays which prevent current flow in the appropriate motor in the appropriate direction to prevent further motion in the offending direction but which permit motion in the opposite direction. Encoder outputs and motor drive signals are routed through this board, which is mounted in the cradle assembly. Figure 2.4-5 shows the arrangement for Channel 2. Channels 1 and 3 are similar except that the power amplifiers are not required for the smaller motors used on those channels.

Quadrant detector preamplifier - Two stages of preamplification are utilized to detect a 5 kHz tone on the received optical signal incident on the four segments of the quadrant detector. The stages are AC coupled and rolled off above 6 kHz to eliminate DC drifts and high frequency components in of the communications data signal. A low-noise operational preamplifier, OP-470, and post amplifier, OP-471, from PMI are used for this purpose. See Figure 2.4-6.

Digital angle processor - The Analog Devices ADSP-2101 is a digital signal processor utilized to process samples of the four quadrants to produce a measure of the position of the centroid of the optical signal focused on the quadrant detector. The familiar horizontal and vertical difference-over-sum functions of the four quadrant signals, A, B, C, and D, are given by

\[ \delta_v = \frac{(A+B-C-D)}{(A+B+C+D)} \]

\[ \delta_h = \frac{(A+C-B-D)}{(A+B+C+D)} \]

The processor computes the numerators of the above expressions in addition to the more demanding task of extracting the amplitude of the 5-kHz tone from the composite data and tone signal. Figure 2.4-7 illustrates the signal waveform. The amplitude of the basic Manchester pulse train is modulated with the 5-kHz tone to a depth of 10%. The preamps described above perform some initial processing by attenuating the components at the data frequency. The resulting 5-kHz signal is sampled at 27.18 kHz by a 4-channel A-to-D converter which simultaneously samples and holds the 4 quadrant signals then sequentially converts them for use by the processor. The processor passes 100 such samples for each quadrant through a 40-tap bandpass
The receiver waveform showing data and tone includes:

- A 5 kHz tone envelope
- Data levels labeled P1, P2, and P3

Key parameters:

- Maximum extinction ratio: $\frac{P_3}{P_2} = 0.114$
- Tone modulation depth: $1 - \frac{P_2}{P_1} = 0.125$
- Peak data power: $P_2 - P_3 = 1.5 \times$ measured average power

**Figure 2.4-7**
digital filter, a rectifier, and a 40-tap low-pass to extract the amplitude of the tone. These four values are then used in the above equations to produce the vertical and horizontal position components. The entire process requires 4.2 milliseconds and has a latency of about 7.8 milliseconds, i.e., the data used to control the motors is about 7.8 milliseconds old when the resulting control is applied. The circuit configuration is presented in Figure 2.4-8. The software for this processor is described in the software section.

2.4.2 Communications circuits - The quadrant detector bias current is modulated by the total optical signal incident on the detector and contains the wideband data signal. Figure 2.4-9 shows the wideband amplifier used to detect the data component of the optical signal. The TIEF151 amplifiers have 40-MHz bandwidth and a transimpedance gain of 4 kilohm. The arrangement utilize here achieves a closed-loop bandwidth 3 MHz to support the 3 Mbps receive-channel design data rate. Higher bandwidth is achievable with the detector used given the required amplifier design and packaging developments.

The transmitted optical signal serves as a tracking beacon to the low-Earth-orbit (LEO) satellite and provides low-rate data capability for command, control, and housekeeping functions. Figure 2.4-10 presents the circuit to drive an SDL-5301 laser diode to impose a 5-kHz. tone and 100-kHz. data signal on the optical beam. Adjustments for the diode slope efficiency and threshold characteristics to achieve the desired depth of modulation for the tone are provided. These circuits are mounted in the cradle area with the laser diodes they drive, and the optical signal is fed to the arm optics in each case by means of a single-mode fiber.

2.5 Software

Executive control of the terminal is provided by a CompuAdd 325 personal computer with a math co-processor. The motor control processor described above receives its operating parameters and mode control instructions from the executive while the angle processors, triggered by a sync-pulse generator, acquire the latest tracking errors. Overall synchronization of the executive, motor-control, and angle processors is accomplished by the sync-pulse generator. This software is described in this section. Detailed listings with comments are contained in Appendix C.
2.5.1 Executive - There are three main executive programs for operation of the terminal: MANUAL.C, TRACK.C, and OMA.FOR. The first two programs, written using Microsoft Quick C are used for real-time control of the terminal when operated in the manual or automatic mode, respectively. In addition, there is a Fortran executive, OMA.FOR, which calls the other Fortran programs to exercise a simulation of the system, and which represents an initial version of satellite operational executive software. The Fortran and C programs are compiled into MS-DOS-executable files to perform the complete job of operating the terminal in the modes required.

Manual Operation Software - Table 2.5-1 illustrates the make-up of the manual MS-DOS-executable file, MANUAL.EXE. As seen from the table, these programs are written entirely in C. Their functions are accessed from the main menu, presented in Table 2.5-2. These functions permit open-loop exercising of the disk and arm actuators for the three channels. For example, initial calibration of the position references for all channels can be accomplished with Option 6, Channel Alignment. This action is required only if a disk has been moved by hand either with the power off or with neither MANUAL.EXE nor TRACK.EXE running. Arm and disk motion to specific locations can be accomplished with Options 1 and 2. Exercising channels to troubleshoot or validate mechanical operation can be accomplished with Options 3 and 4. Various tests of the motion-control electronics can be performed with Options 7-14.

Operational Satellite Software - The operational satellite software is illustrated by Figure 2.5-1, which diagrams the interaction of the various Fortran modules of the control software for an operational system. Where arrows indicate hardware interfaces, this software simulates the hardware when necessary to complete an action in the simulation mode. This organization was developed during the design phase of this project in order to understand the constraints imposed by the satellite and the mission. The modifications required to interface with and operate the prototype terminal preserve this organization to the maximum extent possible. They are discussed in Paragraph 2.5.4.

Table 2.5-3 illustrates the make-up of OMA.EXE from source code files written in Fortran. Each terminal interface module is sensitive to a REALTIME discrete to determine whether it simulates the inputs it needs or gets them from a hardware source. Therefore the programs are useful for simulation studies and also contain the switches to permit interfacing with actual hardware.
TABLE 2.5-1 MANUAL OPERATION SOFTWARE ORGANIZATION

<table>
<thead>
<tr>
<th>MODULE/FILE</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPUTE.C</td>
<td>Subroutines to convert among az/el, alpha/delta, and delta A/B coordinates.</td>
</tr>
<tr>
<td>DISCRETE.C*</td>
<td>Subroutines to monitor the status of all the discretes in the system.</td>
</tr>
<tr>
<td>DISPLAY.C</td>
<td>Subroutines to generate the menu for the manual operation.</td>
</tr>
<tr>
<td>FILE.C*</td>
<td>Subroutines to open and close all files, read and write system and test data.</td>
</tr>
<tr>
<td>INITMOT.C</td>
<td>Initializes motion control digital parameters.</td>
</tr>
<tr>
<td>MOTCNT.C*</td>
<td>Subroutines for mechanical alignment of the system.</td>
</tr>
<tr>
<td>MOTOR.C*</td>
<td>Subroutines for direct interfacing with the motor-control circuit board in the computer.</td>
</tr>
<tr>
<td>NEWMOT.C</td>
<td>Subroutines for controlling the motion of the disks.</td>
</tr>
<tr>
<td>SERL.C*</td>
<td>Subroutines for communication over RS-232 links from the PC to the channel angle processors or to another PC.</td>
</tr>
</tbody>
</table>

* Also used in the automatic mode presented in Table 2.5-3.

TABLE 2.5-2 MANUAL CONTROL MENU

* Move Channel Commands *
1. Move optics arm delta & alpha increment
2. Move arm to designated Az and El
3. Exercise single channel through limits
4. Exercise all channels through limits
5. Compute channel Az & El

* Alignment Command *
6. Channel alignment

* System Test Commands *
7. Motor safety switch test
8. DSP to PC serial link test
9. Host PC to Monitoring PC serial link test
10. Power monitoring discrete test
11. Motor drive backlash test
12. Motor step response test
13. Channel eccentricity test
14. Reset Motor Command
CONTROL SOFTWARE
TOP-LEVEL ARCHITECTURE

FIGURE 2.5-1
### TABLE 2.5-3 OPERATIONAL SATELLITE SOFTWARE ORGANIZATION

Modules called by OMA.FOR or by each other to make up OMA.EXE

<table>
<thead>
<tr>
<th>MODULE FILE</th>
<th>CALLED BY</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALDLDLAB.FOR</td>
<td>CNTRLFN.FOR</td>
<td>Converts alpha/delta to disk A/B coord.</td>
</tr>
<tr>
<td>ANGLPR.FOR</td>
<td>CNTRLFN.FOR</td>
<td>Simulates angle processor in sim. mode.</td>
</tr>
<tr>
<td>ARKTNS.FOR</td>
<td>ANGLEPR.FOR</td>
<td>Takes arctan of arguments in 4 quadrants.</td>
</tr>
<tr>
<td>AZELALDL.FOR</td>
<td>ASGNMFN.FOR</td>
<td>Converts az/el to alpha/delta coordinates.</td>
</tr>
<tr>
<td>CNTRLFN.FOR</td>
<td>ASGNMFN.FOR</td>
<td>Provides the arm/disk position commands.</td>
</tr>
<tr>
<td>COPYVEC.FOR</td>
<td>TRUTHGEN.FOR</td>
<td>Copies a vector to a new name.</td>
</tr>
<tr>
<td>CROSS.FOR</td>
<td>XYAZEL.FOR</td>
<td>Performs vector cross product mathematics.</td>
</tr>
<tr>
<td>DFQ.FOR</td>
<td>RUK.FOR</td>
<td>Satellite state differential equations</td>
</tr>
<tr>
<td>DOT.FOR</td>
<td>ANGLEPR.FOR</td>
<td>Performs vector dot product mathematics.</td>
</tr>
<tr>
<td>ELSTAT.FOR</td>
<td>OMA.FOR</td>
<td>Converts orbital to Cartesian coordinates</td>
</tr>
<tr>
<td>GAUSCL.FOR</td>
<td>ANGLEPR.FOR</td>
<td>Generates random numbers from clipped Gaussian distribution.</td>
</tr>
<tr>
<td>GAUSS.FOR</td>
<td>GAUSCL.FOR</td>
<td>Generates random numbers from Gaussian distribution.</td>
</tr>
<tr>
<td>INV3X3.FOR</td>
<td>QFITVAL.FOR</td>
<td>Inverts a 3x3 matrix</td>
</tr>
<tr>
<td>NAVIGFN.FOR</td>
<td>OMA.FOR</td>
<td>Performs the navigation function</td>
</tr>
<tr>
<td>NAVIGPR.FOR</td>
<td>NAVIGFN.FOR</td>
<td>Adds ephemeris errors to true pos. and vel.</td>
</tr>
<tr>
<td>QFITVAL</td>
<td>TRUTHGEN.FOR</td>
<td>Quadratic time fit for pos. and vel.</td>
</tr>
<tr>
<td>RNDM.FOR</td>
<td>SCENEPR.FOR</td>
<td>Generates random numbers for noise sim.</td>
</tr>
<tr>
<td>RUK.FOR</td>
<td>TRUTHFN.FOR</td>
<td>Fourth order Runge-Kutta integrator.</td>
</tr>
<tr>
<td>SCANGN.FOR</td>
<td>CNTRLFN.FOR</td>
<td>Scan generator for acquisition/reacquisition functions</td>
</tr>
<tr>
<td>SCENEPR.FOR</td>
<td>TRUTHGEN.FOR</td>
<td>Computes az/el for each LEO beam.</td>
</tr>
<tr>
<td>TRUTHGN.FOR</td>
<td>NAVIGFN.FOR</td>
<td>Simulates satellite orbital dynamics.</td>
</tr>
<tr>
<td>UNIT.FOR</td>
<td>XYAZEL.FOR</td>
<td>Converts vector to unit vector.</td>
</tr>
<tr>
<td>VLEN.FOR</td>
<td>UNIT.FOR</td>
<td>Computes the length of a vector.</td>
</tr>
<tr>
<td>XKEP.FOR</td>
<td>ELSTAT.FOR</td>
<td>Solves Kepler's orbital equation.</td>
</tr>
<tr>
<td>XYAZEL.FOR</td>
<td>SCENEPR.FOR</td>
<td>Converts from Cartesian to OMA FOV coord.</td>
</tr>
<tr>
<td></td>
<td>ASGNMFN.FOR</td>
<td></td>
</tr>
</tbody>
</table>
Real-time Automatic Operation Software - The software of the previous section was modified to permit operation with the prototype hardware in a laboratory environment. The satellite trajectory information is replaced with an initial point for acquisition of a signal to be tracked. No effort was made to produce a trajectory in the test equipment that could be acquired by open-loop pointing at any point in the trajectory similar to acquiring a satellite. The sophistication, complexity, and cost of such elaborate test equipment was beyond the scope of this effort. Instead, the acquisition program scans a region known to intersect the trajectory of the test source so that when the source approaches it can be acquired. Table 2.5-4 illustrates the make-up of TRACK.EXE from source code files written in both C and Fortran. The Fortran portion controls the automatic acquisition and tracking functions while the C portion sets up the initial conditions and handles all system-level interfaces.

### TABLE 2.5-4 AUTOMATIC REAL-TIME OPERATION SOFTWARE ORGANIZATION

<table>
<thead>
<tr>
<th>MODULE FILE</th>
<th>CALLED BY</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>MENU.C</td>
<td>Subroutines to generate the menu for the automatic operation.</td>
<td>See Table 2.5-1</td>
</tr>
<tr>
<td>Other C files</td>
<td>See Table 2.5-1</td>
<td></td>
</tr>
<tr>
<td>ARKTNS.FOR</td>
<td>ANGLEPR.FOR</td>
<td></td>
</tr>
<tr>
<td>AZELALDL.FOR</td>
<td>ASGNMFNR</td>
<td>Takes Arctangent of arguments in 4 quadrants.</td>
</tr>
<tr>
<td>ANGLPR.FOR</td>
<td>CNTRLFN.FOR</td>
<td>Converts az/el to alpha/delta coordinates.</td>
</tr>
<tr>
<td>ASGNMFN.FOR</td>
<td>INIT</td>
<td></td>
</tr>
<tr>
<td>ASGNMFNR (sub)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLKDATA.FOR</td>
<td>Compiler</td>
<td></td>
</tr>
<tr>
<td>CNTRLFN.FOR</td>
<td>ASGNMFN.FOR</td>
<td></td>
</tr>
<tr>
<td>DISCPOS.FOR</td>
<td>CNTRLFN.FOR</td>
<td></td>
</tr>
<tr>
<td>DOT.FOR</td>
<td>ANGLEPR.FOR</td>
<td></td>
</tr>
<tr>
<td>GAUSCL.FOR</td>
<td>ANGLEPR.FOR</td>
<td></td>
</tr>
<tr>
<td>GAUSS.FOR</td>
<td>GAUSCL.FOR</td>
<td></td>
</tr>
<tr>
<td>INTRFACE.FOR</td>
<td>TRACK.C</td>
<td></td>
</tr>
<tr>
<td>INIT (sub)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>POSITION (sub)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCANGN.FOR</td>
<td>CNTRLFN.FOR</td>
<td></td>
</tr>
</tbody>
</table>
2.5.2 Angle Processor Software - The Analog Devices ADSP-2101 digital signal processor was programmed to extract the 5-kHz components from each of the 4 quadrant signals, then produce the horizontal and vertical coordinates of the centroid of the spot by sums and differences as described in Paragraph 2.4.1. The system was set up to sample each quadrant at 27.18 kHz for 100 samples and process these samples in an algorithm described by Figure 2.5-2. The files which make up this set of algorithms are listed in Table 2.5-5.

<table>
<thead>
<tr>
<th>MODULE FILE</th>
<th>CALLED BY</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADASYS.DSP</td>
<td>MAIN.DSP</td>
<td>Contains all routines for sampling data.</td>
</tr>
<tr>
<td>ALGORITHM.DSP</td>
<td>MAIN.DSP</td>
<td>Computes the un-normalized tracking error.</td>
</tr>
<tr>
<td>BANDFILT.DSP</td>
<td>ALGORITHM.DSP</td>
<td>Code for bandpass filter in Figure 2.5-2.</td>
</tr>
<tr>
<td>LOWPASS.DSP</td>
<td>ALGORITHM.DSP</td>
<td>Code for low-pass filter in Figure 2.5-2.</td>
</tr>
<tr>
<td>MAIN.DSP</td>
<td>On power-up</td>
<td>Executive software for angle processor.</td>
</tr>
<tr>
<td>UART.DSP</td>
<td>MAIN.DSP</td>
<td>Routine for formatting and sending data to host PC.</td>
</tr>
</tbody>
</table>

2.5.3 Motion Processor Software - The HP HCTL-1000 has built-in software to implement the functions illustrated in Figure 2.4-2. The executive is programmed to load the appropriate parameter values into the processor to effect the desired digital filtering characteristic and controls the modes of the device in the real-time operation of the system.
ERROR SIGNAL AMPLITUDE DETECTION

DIGITIZED DATA FROM A/D CONVERTER

DATA
SAMPLED
AT 29 KHZ

BANDPASS
FIR
FILTER
CENTERED AT 5 KHZ.

FULLWAVE
RECTIFIED
SIGNAL

LOWPASS
FIR
FILTER

ERROR SIGNAL AMPLITUDE
FROM QUADRANT A,B,C,D

FIGURE 2.5-2
3.0 EVALUATION TEST DATA

Prior to delivery of the system to NASA Goddard, as part of the integration process we performed some evaluation tests on the system and its components. More extensive testing could guide the continuing development of an operational terminal by providing concrete results to design improvements economically before applying them to the next-generation hardware.

3.1 Optical Tests

3.1.1 Telescope Performance - The telescope transmission was evaluated using a HeNe laser at 6328 Å to be 67%. There are 4 lenses with two surfaces each for a total of 8 surfaces each apparently transmitting about 95% of the incident energy. The performance at the operational wavelength should have been better since the AR coating would be superior nearer the operational wavelength. The space system should be capable of significantly better performance with superior space coatings.

It was noted that near but not at the focus, the spot shape to the naked eye had a triangular shape that suggested distortion of the lens shape by the 6-point mounting hardware. We determined which lens of the set of four was misshapen by rotating each lens with respect to the remaining three and observing the orientation of the triangle before and after each such rotation. The spot rotated with the rotation of the frontmost element, i.e., the lens furthest from the focal plane, suggesting that it alone was distorted. We conclude from this test that (1) the mounting rings for both the front pair and the back pair should have more fasteners around each ring to equalize the clamping around the periphery of the lens, and (2) the interface between the two lenses in each pair should be a part machined to the correct shape rather than a flexible O-ring. This modification prevents the pair to be clamped in a distorted manner. It is possible but unlikely that the part was distorted in manufacturing. The nature of the distortion is not characteristic of manufacturing errors, which would be axisymmetric rather than triangular. The spot shape suggests a lens bent by externally applied force either from mishandling or asymmetrical clamping in the mounting bracketry. Because the ring had 6 equally spaced clamping bolts, it is reasonable that it could cause a distortion with three-fold symmetry, so as to produce a triangular spot. This aberration did not severely affect the focussed spot size based on the observations of spot shape on the detector given in
3.1.2 Arm Performance - The arm optics were tested for transmission and for quadrant detector position transfer function. The transmissions were measured prior to assembly into the disk assembly at 65%, 61%, and 55% for Channels 1, 2, and 3, respectively. Figure 3.1-1 & -2 shows the quadrant processing characteristic for motion along one axis for Channel 3. These data were obtained prior to installation of the arm in the system. Note the S-shaped characteristic in the horizontal axis and the slight tilt in the vertical axis. This result shows a slight rotation of the detector about the optic axis. This error was corrected prior to installation and is presented to show the value of these tests.

The spot shape on the detector was inferred from scans of the arm past the test sources. Figure 3.1-3 shows a typical such scan using Channel 1 and diode #2 on a three-source test fixture. The derivative of the power on the detector versus position is a measure of the shape of the spot in the direction of the scan. Note that the focused spot is twin-lobed due to the laser's characteristics but that the resolving power of the optical system is on the order of 100 microns.

3.2 Mechanical Tests

The mechanisms for the disks were tested and run in during assembly to remove roughness and minor imperfections in order to ensure smooth operation. Upon completion of assembly, tests for resolution, backlash, hysteresis and slew rate were performed. Tests to determine that the lighter motors are adequate to drive the worm assemblies when sufficiently preloaded to eliminate backlash were performed.

3.2.1 Resolution/backlash/hysteresis - Figures 3.2-1 and -2 show the role of the worm assembly adjustment mechanism of Figure 2.2-2. The "Displacement" and "Command" refer to the rotation of a single disk. Initial tests showed that the path traversed by the mechanism was very different depending on the direction of travel. After adjusting the worm into the worm gear, the hysteresis was significantly reduced and the current in the motors increased from .25 amps to .5 amps - still well within capacity. In an operational design, improved design and machining approaches would be considered to reduce friction and backlash, including
FIGURE 3.1-1

ELEVATION QUADRANT SCAN

SUM SIGNAL (RELATIVE)

ELEVATION ERROR (MIRCORADNNS)

DETECTOR OUTPUT (DIMENSIONLESS)

SUM SIGNAL

AZIMUTH

SLOPE AT CENTER:

245.3 MICRORADIAN UNIT
SPOT SHAPE MEASUREMENT
- Channel 1 with Test Aid Diode 2 -

FIGURE 3.1-3
HYSTERESIS PLOT
- Before Adjustment -

FIGURE 3.2.1

DISPLACEMENT (MICORAD)

COMMAND (MICORAD)
the use of a split, spring-loaded worm which could act as an anti-backlash gear. Additional development of this assembly is recommended in order to improve the automatic tracking performance. The steps required appear to be straightforward extensions of the present work - not great departures from the current design.

Backlash tests were performed on each arm in each axis. Figure 3.2-3 shows a scan which swung the arm across a spot from a test source, then back again. This test was performed with the complete assembly, and, therefore, includes the effects of both the telescope and arm optics. Note that the retrace is displaced about 120 microns from the first trace, indicating a similar amount of backlash in the mechanism. The quadrant data displays some amount of cross-coupling, but the motion is predominantly in the cross-arm axis of the detector. The amount of backlash observed is more than is tolerable and could have been significantly reduced on disassembly and readjustment for optimum performance. Figure 3.2-4 shows another spot shape scan derived from the same data as for the backlash test. The spot shape is different because of the difference in channel optics and in location in the field.

3.2.2 Slewing rate - The motor control chip accepts maximum values for velocity and acceleration which result in what are referred to as "trapezoidal" moves. The name derives from a plot of speed versus time for a point-to-point move. The speed increases linearly at the maximum acceleration until the maximum velocity is reached, then stays constant until close to the terminus of the move, when the speed decreases linearly at the maximum deceleration, and the motor position arrives at the terminus as the speed reaches zero. The computations for this move are accomplished by the motor control chip, which also issues the commands to accomplish the move. Figure 3.2-5 shows a step response of the motors which compares the response of the large and small motors under various conditions after initial assembly. The small motors were having trouble reacting fast enough when required to work against the other disk or move the arm. We disassembled the set-up and carefully cleaned out any residual machining debris caught in the lubricant with the initial run in and re-ran the test. The result in Figure 3.2-6 revealed that (1) the large motor was not hindered by the additional friction due to its higher torque capability, and (2) the small motor can provide adequate response under all circumstances when properly prepared.
BACKLASH TEST
- Channel 2.5 with Test Aid Diode 2 -

FIGURE 3.2.3
SPOT SHAPE MEASUREMENT
- Channel 2 with Test Aid Diode 2 -

FIGURE 3.2-4
MOTOR STEP RESPONSE
- After Cleaning -

POSITION (ENCODER COUNTS)

0  0.01  0.02  0.03  0.04  0.05  0.06  0.07  0.08  0.09  0.1
0.0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1.0
-1500 -1000 -500  0  500  1000  1500

LARGE MOTOR ON DISK "B"

SMALL MOTOR ON DISK "A" MOVING AGAINST DISK "B" WITH AND WITHOUT ARM ATTACHED

SMALL MOTOR ON DISK "A" MOVING WITH DISK "B", WITH AND WITHOUT ARM ATTACHED

FIGURE 3.2-6
The slewing rate in the far field is a complex function of the trajectory across the field. The most significant occurrence of maximum velocity is when the satellite being tracked passes through the center of the field of view. In the limiting case an outage occurs until the disks can swing through 180° and pick up the signal on the other side of center. We designed for 10° per second and tested 11.4° per second and 7.7° per second, depending on what hexadecimal number was loaded into the chip for maximum velocity. For the prototype we utilized the slower rate to avoid a potential problem overrunning the limit switches on initial calibration. In the space system, the maximum slew rate can be set up to at least 30° per second, resulting in a maximum outage time of 6 seconds for those extreme cases passing directly through the center of the field of view. When another channel is available, hand-off could occur to avoid any appreciable outage. Additional testing would reveal the maximum reasonable value for the slew rate so that we could determine under what circumstances the complexity of the hand-off solution would be superior to the brute-force technique of operating at maximum slew rate.

3.3 Acquisition Time - The primary parameters which make up acquisition time - slew rate, scan pattern, the delay inserted to allow for round-trip optical signal propagation - were either verified or are self-evident. The statistical characteristics of the process having to do with signal fluctuations at the threshold of detectability were beyond the scope of this exercise, but have been explored extensively elsewhere. We feel that the estimates of acquisition time provided in the design rational are conservative.

3.4 Tracking Error - Figure 3.4-1 & -2 shows the tracking error for a stationary target taken on both the across-arm and the along-arm components. The backlash caused some rumble in the motion which is illustrated by the cross-arm component. Further experimentation with the mechanical adjustments and worm/worm gear interface should reduce this effect and result in a smoother, quieter operation.

3.5 Weight Analysis - Table 3.5-1 presents a weight analysis of the space design as compared to the implementation of the prototype. We conclude that the 150-pound goal weight is feasible when lightweighting is utilized in the structural design.
FIGURE 3.42

SIGNAL LEVEL (RELATIVE)

TIME (SECONDS)

ERROR (MICRORADIAN)

TRACKING ERROR PERPENDICULAR TO ARM

11.6 MICRORAD RMS
# MULTIPLE-ACCESS TERMINAL WEIGHT ANALYSIS

<table>
<thead>
<tr>
<th>SUBASSEMBLY</th>
<th>WEIGHT (LBS)</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISKS</td>
<td>12</td>
<td>42 Be REPLACES BRASS AND DISK HEIGHT IS OPTIMIZED IN SPACE DESIGN</td>
</tr>
<tr>
<td>ARMS</td>
<td>2</td>
<td>2 Be NOT JUSTIFIED</td>
</tr>
<tr>
<td>MOTORS/ENCODERS</td>
<td>6</td>
<td>10 ASSUMES PROTOTYPE DESIGN HAS EXCESS TORQUE MARGIN</td>
</tr>
<tr>
<td>TRANSMITTER MODULES</td>
<td>3</td>
<td>9 MATERIALS AND CONFIGURATION OPTIMIZED FOR SPACE</td>
</tr>
<tr>
<td>STRUCTURE/HARDWARE</td>
<td>23</td>
<td>66 ASSUMES A 70% WEIGHT SAVINGS ON MAIN STRUCTURE DUE TO USE OF LIGHT-WEIGHTING TECHNIQUES AND MATERIALS, PLUS 6 LBS FOR HARDWARE</td>
</tr>
<tr>
<td>TELESCOPE</td>
<td>90</td>
<td>115 Be REPLACES AI IN SPACE DESIGN OF STRUCTURE; GLASS REPLACES PLASTIC FOR SPACE OPTICAL ELEMENTS.</td>
</tr>
<tr>
<td>ELECTRONICS AND CABLES</td>
<td>12</td>
<td>25 SPACE DESIGN OPTIMIZED</td>
</tr>
<tr>
<td>TOTALS</td>
<td>148</td>
<td>269* ACTUAL PROTOTYPE WITH 3 CHANNELS IS ESTIMATED TO WEIGH APPROXIMATELY 200 LBS.</td>
</tr>
</tbody>
</table>

*FIGURE 3.5-1*
4.0 AREAS FOR FURTHER DEVELOPMENT

The design, development, assembly and testing of the OMA prototype model demonstrated the feasibility of the basic concept and its advantages. A number of issues became clear as to what is needed to enable the OMA to reach its full potential and to be useful space hardware. The following paragraphs discuss these issues.

First, the telescope used in the OMA was necessarily composed of plastic lenses due to cost considerations. This approach adequately demonstrated the concept, but in space, special glass must be used to avoid radiation darkening and to avoid cold-flow shape changes when in space for long periods. We accomplished a glass design to ensure that the same wide field of view can be achieved. In fact, the telescope performance in terms of spot diameters in the focal plane will be better in glass, and this will help tracking performance. Any next step in development must include glass lenses in order to properly evaluate the terminal performance capability.

Second, custom arm optics are required to achieve the vertical spacing required of the six-channel space design. The three-channel prototype design was accomplished with standard optics. Small custom optics allow less depth, less arm width to reduce channel interference possibilities, and overall lighter weight. The next development should include at least one custom optics arm to illustrate the space design capability.

Third, more effort is needed to improve the backlash from the gearing which will smooth out the tracking from some peak errors that accrue. These errors are still within the error budget but cause noise and make the ratio of peak tracking error to rms tracking much greater than it needs to be. The trade-off between less backlash and more friction torque can be efficiently made using the prototype as a tool. Time and money limits prevented further attention in this effort.

Fourth, the tracking bandwidth probably should be raised from 40Hz to about 60Hz to further improve the tracking performance.

Effort should be given to ensuring the optical transmission efficiency and needed performance is obtained on both transmit and receive. On transmit, the proper beamwidth of the transmitter should be established, and any minor optical design
changes or hardware adjustments to achieve required performance should be accomplished. The receiver optics in the arm would benefit from careful measurement for transmission efficiency, image quality, and field of view and adjustment to optimize the output.

The above discussion covers the major areas for further development. The potential and demonstrated advantages of the OMA terminal can be fully realized with a development model based on the achievements of this program and incorporation of the improvements discussed here.

5.0 CONCLUSIONS

The OMA prototype represents a first step towards operational multi-access optical communications terminal development. The concept was shown to be feasible in a low-cost demonstration program which points the way to improvements for operational implementation. The basic design is sound and would benefit from a second program which would provide a thoroughgoing evaluation of the prototype implementation from an optical, mechanical, and electronic viewpoint as a lead-in to a six-channel engineering model. This evaluation program would quantify the performance capability of the present implementation and identify where design improvements are necessary to meet space system requirements and where the present design is adequate. This information would guide the next phase toward activities which are most cost-effective in achieving the space design.

The Phase II program was successful in that a prototype was delivered that demonstrated the key aspects and special advantages of this unique approach to simultaneous multi-access operation. The program showed that the basic advantages of size, weight, power, and Earth-viewing swept volume are realizable. They are substantial when compared to RF implementations or to separate optical transceivers. A patent has been granted for the concept.

The ability of user satellites to have small-size optical transceivers to work with the multi-access terminal is attractive because of the reduced burden on the user satellite. It is recommended that communications applications involving wide bandwidth simultaneously from various satellites to a relay satellite seriously consider this approach as a significantly cost-effective technical solution.
### TELESCOPE OPTICAL DESIGN SPECIFICATIONS

<table>
<thead>
<tr>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Telescope Elements with Ray Trace</td>
<td>A3</td>
</tr>
<tr>
<td>Ray Trace and Surface Curvature Specifications</td>
<td>A5</td>
</tr>
<tr>
<td>Sag Calculations for Individual Elements</td>
<td>A24</td>
</tr>
</tbody>
</table>
THIS PAGE WAS INTENTIONALLY LEFT BLANK
THIS PAGE WAS INTENTIONALLY LEFT BLANK
MULTI-ACCESS FREE SPACE LASER COMMUNICATIONS TERMINAL - FINAL REPORT

LDT NASA TELESCOPE AT 1060 NM WAVELENGTH

File ldtplas 11:08:00 02-14-1991

ZOOM POSITION 1
EFL = 627.483

<table>
<thead>
<tr>
<th>#</th>
<th>TYPE</th>
<th>CURVE</th>
<th>SEPN</th>
<th>INDEX1</th>
<th>INDEX2</th>
<th>INDEX3</th>
<th>DISPN</th>
<th>CLRAD</th>
<th>GLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S</td>
<td>0.000000</td>
<td>0.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>125.61</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>0.002135</td>
<td>400.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>195.34</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>S</td>
<td>0.003544</td>
<td>12.700</td>
<td>1.57182</td>
<td>1.57597</td>
<td>1.57182-0.00415</td>
<td>190.58</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>A</td>
<td>0.004324</td>
<td>1.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>193.87</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>S</td>
<td>0.006633</td>
<td>81.524</td>
<td>1.48222</td>
<td>1.48466</td>
<td>1.48222-0.00245</td>
<td>193.08</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>A</td>
<td>0.006593</td>
<td>543.944</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>170.50</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>S</td>
<td>0.002784</td>
<td>102.098</td>
<td>1.48222</td>
<td>1.48466</td>
<td>1.48222-0.00245</td>
<td>164.06</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>0.004486</td>
<td>1.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>193.87</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>S</td>
<td>0.007514</td>
<td>12.700</td>
<td>1.57182</td>
<td>1.57597</td>
<td>1.57182-0.00415</td>
<td>123.88</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>S</td>
<td>0.000000</td>
<td>130.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>117.52</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>S</td>
<td>0.000000</td>
<td>318.999</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>116.66</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>S</td>
<td>0.000000</td>
<td>1.949</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>116.66</td>
<td></td>
</tr>
</tbody>
</table>

ASPERIC SURFACE 2 CC = 2.477985

ASPERIC SURFACE 4 CC = -1.247590
A4 = 1.901898E-11 A6 = 7.800049E-14 A8 = -1.635492E-18 A10 = 2.510175E-23

ASPERIC SURFACE 6 CC = -0.354793 (ELLIPSE)
A4 = -3.357945E-09 A6 = 1.277421E-14 A8 = -2.20762E-18 A10 = 3.734123E-23

Defocus = -0.750000 U' = -0.200000

CHIEF RAY

<table>
<thead>
<tr>
<th>L</th>
<th>M</th>
<th>X</th>
<th>Y</th>
<th>OPD</th>
<th>COLOR</th>
<th>S</th>
<th>T</th>
<th>DIST(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.19969</td>
<td>0.04986</td>
<td>-0.01416</td>
<td>-0.03584</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.09998</td>
<td>0.07627</td>
<td>-0.00368</td>
<td>-0.00851</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.19907</td>
<td>0.07775</td>
<td>-0.00609</td>
<td>-0.01198</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.17949</td>
<td>0.00651</td>
<td>-0.00685</td>
<td>-0.00756</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.15946</td>
<td>-0.03360</td>
<td>-0.00563</td>
<td>-0.00384</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.13912</td>
<td>-0.05378</td>
<td>-0.00562</td>
<td>-0.00086</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.09930</td>
<td>-0.05975</td>
<td>-0.00324</td>
<td>0.00278</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.05917</td>
<td>-0.04020</td>
<td>-0.00118</td>
<td>0.00367</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.06125</td>
<td>0.04603</td>
<td>-0.00120</td>
<td>-0.00976</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.10137</td>
<td>0.11702</td>
<td>-0.00433</td>
<td>-0.01992</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.14071</td>
<td>0.20302</td>
<td>-0.01070</td>
<td>-0.03304</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.16149</td>
<td>0.21961</td>
<td>-0.01516</td>
<td>-0.04132</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.18155</td>
<td>0.19138</td>
<td>-0.01936</td>
<td>-0.05025</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.20055</td>
<td>0.11954</td>
<td>-0.02238</td>
<td>-0.05961</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.19978</td>
<td>-0.00134</td>
<td>-0.00402</td>
<td>0.04571</td>
<td>-0.00899</td>
<td>-0.03587</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.17600</td>
<td>-0.00127</td>
<td>0.07360</td>
<td>0.03750</td>
<td>-0.00800</td>
<td>-0.02743</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.13995</td>
<td>-0.00119</td>
<td>0.08343</td>
<td>0.02058</td>
<td>-0.00492</td>
<td>-0.01699</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.10010</td>
<td>-0.00112</td>
<td>0.05048</td>
<td>0.00561</td>
<td>-0.00223</td>
<td>-0.00853</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ZOOM POSITION 2
EFL = 627.483

<table>
<thead>
<tr>
<th>#</th>
<th>TYPE</th>
<th>CURVE</th>
<th>SEPN</th>
<th>INDEX1</th>
<th>INDEX2</th>
<th>INDEX3</th>
<th>DISPN</th>
<th>CLRAD</th>
<th>GLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S</td>
<td>0.000000</td>
<td>0.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>125.61</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>0.002135</td>
<td>400.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>195.34</td>
<td></td>
</tr>
</tbody>
</table>

---

PRECEDING PAGE (MARKS CUT OFF)
MULTI-ACCESS FREE SPACE LASER COMMUNICATIONS TERMINAL - FINAL REPORT

<table>
<thead>
<tr>
<th>#</th>
<th>TYPE</th>
<th>CURVE</th>
<th>SEPN</th>
<th>INDEX1</th>
<th>INDEX2</th>
<th>INDEX3</th>
<th>DISPZ</th>
<th>CLRAD</th>
<th>GLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 S</td>
<td></td>
<td>0.000000</td>
<td>0.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>125.61</td>
<td></td>
</tr>
<tr>
<td>2 A</td>
<td></td>
<td>0.002135</td>
<td>400.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>195.34</td>
<td></td>
</tr>
<tr>
<td>3 S</td>
<td></td>
<td>0.003544</td>
<td>12.700</td>
<td>1.57182</td>
<td>1.57597</td>
<td>1.57182</td>
<td>0.00415</td>
<td>190.58</td>
<td></td>
</tr>
<tr>
<td>4 A</td>
<td></td>
<td>0.004324</td>
<td>1.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>193.87</td>
<td></td>
</tr>
<tr>
<td>5 S</td>
<td></td>
<td>0.000633</td>
<td>81.524</td>
<td>1.48222</td>
<td>1.48466</td>
<td>1.48222</td>
<td>-0.00245</td>
<td>193.08</td>
<td></td>
</tr>
<tr>
<td>6 A</td>
<td></td>
<td>0.006593</td>
<td>543.944</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>170.50</td>
<td></td>
</tr>
<tr>
<td>7 S</td>
<td></td>
<td>0.002784</td>
<td>102.098</td>
<td>1.48222</td>
<td>1.48466</td>
<td>1.48222</td>
<td>-0.00245</td>
<td>164.06</td>
<td></td>
</tr>
<tr>
<td>8 A</td>
<td></td>
<td>0.004486</td>
<td>1.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>151.30</td>
<td></td>
</tr>
<tr>
<td>9 S</td>
<td></td>
<td>0.007514</td>
<td>12.700</td>
<td>1.57182</td>
<td>1.57597</td>
<td>1.57182</td>
<td>0.00415</td>
<td>123.88</td>
<td></td>
</tr>
<tr>
<td>10 S</td>
<td></td>
<td>0.000000</td>
<td>130.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>117.52</td>
<td></td>
</tr>
<tr>
<td>11 S</td>
<td></td>
<td>0.000000</td>
<td>31899.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>115.63</td>
<td></td>
</tr>
<tr>
<td>12 S</td>
<td></td>
<td>0.000000</td>
<td>-31904.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>116.66</td>
<td></td>
</tr>
</tbody>
</table>

ASPERIC SURFACE 2 CC = 2.477985
A4 = 4.479591E-09 A6 = -8.743832E-14 A8 = 1.936009E-18

ASPERIC SURFACE 4 CC = -1.247590
A4 = 1.901898E-11 A6 = 7.800049E-14 A8 = -1.635492E-18

ASPERIC SURFACE 6 CC = -0.354793 (ELLIPSE)
A4 = -3.357945E-09 A6 = -3.357945E-09 A8 = -3.357945E-09

Defocus = -0.750000 U'

CHIEF RAY -0.00205
0.19826 0.12553 -0.00176 0.01159
0.18082 0.04902 -0.00323 0.01322
0.16049 -0.00662 -0.00361 0.01443
0.13813 -0.03681 -0.00307 0.01496
0.09953 -0.03873 -0.00146 0.01392
0.05891 -0.01573 -0.00035 0.01030
0.06303 0.05263 -0.00106 -0.01655
0.10367 0.14769 -0.00504 -0.03150
0.14219 0.20683 -0.01216 -0.04893
0.16474 0.19081 -0.01671 -0.06073
0.18511 0.16483 -0.02030 -0.07251
0.20232 0.18265 -0.02320 -0.08339
0.19982 0.00249 -0.14106 0.07209 0.00431 -0.03583
0.17672 0.00239 -0.04290 0.06486 0.00233 -0.02764
0.13995 0.00227 -0.00309 0.04179 0.00180 -0.01698
-0.10048 -0.00217 -0.01555 0.01664 0.00147 -0.00859

ZOOM POSITION 3
EFL = 627.483

<table>
<thead>
<tr>
<th>#</th>
<th>TYPE</th>
<th>CURVE</th>
<th>SEPN</th>
<th>INDEX1</th>
<th>INDEX2</th>
<th>INDEX3</th>
<th>DISPZ</th>
<th>CLRAD</th>
<th>GLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 S</td>
<td></td>
<td>0.000000</td>
<td>0.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>125.61</td>
<td></td>
</tr>
<tr>
<td>2 A</td>
<td></td>
<td>0.002135</td>
<td>400.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>195.34</td>
<td></td>
</tr>
<tr>
<td>3 S</td>
<td></td>
<td>0.003544</td>
<td>12.700</td>
<td>1.57182</td>
<td>1.57597</td>
<td>1.57182</td>
<td>0.00415</td>
<td>190.58</td>
<td></td>
</tr>
<tr>
<td>4 A</td>
<td></td>
<td>0.004324</td>
<td>1.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>193.87</td>
<td></td>
</tr>
<tr>
<td>5 S</td>
<td></td>
<td>0.000633</td>
<td>81.524</td>
<td>1.48222</td>
<td>1.48466</td>
<td>1.48222</td>
<td>-0.00245</td>
<td>193.08</td>
<td></td>
</tr>
<tr>
<td>6 A</td>
<td></td>
<td>0.006593</td>
<td>543.944</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>170.50</td>
<td></td>
</tr>
<tr>
<td>7 S</td>
<td></td>
<td>0.002784</td>
<td>102.098</td>
<td>1.48222</td>
<td>1.48466</td>
<td>1.48222</td>
<td>-0.00245</td>
<td>164.06</td>
<td></td>
</tr>
<tr>
<td>8 A</td>
<td></td>
<td>0.004486</td>
<td>1.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>151.30</td>
<td></td>
</tr>
<tr>
<td>9 S</td>
<td></td>
<td>0.007514</td>
<td>12.700</td>
<td>1.57182</td>
<td>1.57597</td>
<td>1.57182</td>
<td>0.00415</td>
<td>123.88</td>
<td></td>
</tr>
<tr>
<td>10 S</td>
<td></td>
<td>0.000000</td>
<td>130.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>117.52</td>
<td></td>
</tr>
<tr>
<td>11 S</td>
<td></td>
<td>0.000000</td>
<td>31899.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>115.63</td>
<td></td>
</tr>
<tr>
<td>12 S</td>
<td></td>
<td>0.000000</td>
<td>-31904.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>116.66</td>
<td></td>
</tr>
</tbody>
</table>
### Multi-Access Free Space Laser Communications Terminal - Final Report

**Aspheric Surface 2 CC = 2.477985**

<table>
<thead>
<tr>
<th>A4</th>
<th>A6</th>
<th>A8</th>
<th>A10</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.479591E-09</td>
<td>-8.743832E-14</td>
<td>1.936009E-18</td>
<td>-3.557055E-23</td>
</tr>
</tbody>
</table>

**Aspheric Surface 4 CC = -1.247590**

<table>
<thead>
<tr>
<th>A4</th>
<th>A6</th>
<th>A8</th>
<th>A10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.901898E-11</td>
<td>7.800049E-14</td>
<td>-1.635492E-18</td>
<td>2.510175E-23</td>
</tr>
</tbody>
</table>

**Aspheric Surface 6 CC = -0.354793 (Ellipse)**

<table>
<thead>
<tr>
<th>A4</th>
<th>A6</th>
<th>A8</th>
<th>A10</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3.357945E-09</td>
<td>1.277421E-14</td>
<td>-2.20762E-18</td>
<td>3.734123E-23</td>
</tr>
</tbody>
</table>

**Defocus = -0.750000 U' = -0.200000**

<table>
<thead>
<tr>
<th>L</th>
<th>M</th>
<th>X</th>
<th>Y</th>
<th>OPD</th>
<th>COLOR</th>
<th>S</th>
<th>T</th>
<th>DIST(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.19969</td>
<td>0.04986</td>
<td>-0.1416</td>
<td>-0.03584</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.15984</td>
<td>0.12738</td>
<td>-0.01000</td>
<td>-0.02241</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Chief Ray**

| -0.00304 | 96.92833 | 0.03167 | -0.00231 | 0.03478 |
| 0.19779 | -0.02492 | 0.00231 | 0.03394 |
| 0.17875 | -0.04897 | 0.00148 | 0.03241 |
| 0.15781 | -0.04607 | -0.00048 | 0.03030 |
| 0.13743 | -0.00947 | 0.00068 | 0.02446 |
| 0.09723 | 0.01291 | 0.00053 | 0.01699 |
| 0.05966 | 0.03325 | -0.00060 | -0.02355 |
| -0.10371 | 0.05731 | -0.00246 | -0.04142 |
| -0.14185 | 0.02371 | -0.00413 | -0.06264 |
| -0.16487 | 0.03433 | -0.00464 | -0.07727 |
| -0.18587 | 0.12763 | -0.00625 | -0.09200 |
| -0.20156 | 0.08258 | -0.00832 | -0.10363 |
| -0.20020 | 0.03963 | 0.01175 | -0.03573 |
| -0.17814 | 0.04820 | 0.00803 | -0.02793 |
| -0.14014 | 0.04450 | 0.00523 | -0.01695 |
| -0.10124 | 0.02520 | 0.00344 | -0.00869 |

**Zoom Position 4**

| EFL = 627.483 |

<table>
<thead>
<tr>
<th>#</th>
<th>TYPE</th>
<th>CURVE</th>
<th>SEPN</th>
<th>INDEX1</th>
<th>INDEX2</th>
<th>INDEX3</th>
<th>DISPNAV</th>
<th>CLRAD</th>
<th>GLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S</td>
<td>0.000000</td>
<td>0.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>125.61</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>0.002135</td>
<td>400.0</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>195.34</td>
<td>POLYST</td>
</tr>
<tr>
<td>3</td>
<td>S</td>
<td>0.003544</td>
<td>12.700</td>
<td>1.57182</td>
<td>1.57597</td>
<td>1.57182</td>
<td>0.00415</td>
<td>190.58</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>A</td>
<td>0.004324</td>
<td>1.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>193.87</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>S</td>
<td>0.006333</td>
<td>81.524</td>
<td>1.48222</td>
<td>1.48466</td>
<td>1.48222</td>
<td>0.00245</td>
<td>193.08</td>
<td>ACRYLC</td>
</tr>
<tr>
<td>6</td>
<td>A</td>
<td>0.006593</td>
<td>543.944</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>170.50</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>S</td>
<td>0.002784</td>
<td>120.098</td>
<td>1.48222</td>
<td>1.48466</td>
<td>1.48222</td>
<td>0.00245</td>
<td>164.06</td>
<td>ACRYLC</td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>0.004486</td>
<td>1.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>151.30</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>S</td>
<td>0.007514</td>
<td>12.700</td>
<td>1.57182</td>
<td>1.57597</td>
<td>1.57182</td>
<td>0.00415</td>
<td>123.88</td>
<td>POLYST</td>
</tr>
<tr>
<td>10</td>
<td>S</td>
<td>0.000000</td>
<td>31904.0</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>118.25</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>S</td>
<td>0.000000</td>
<td>0.000000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>118.25</td>
<td></td>
</tr>
</tbody>
</table>

**Aspheric Surface 2 CC = 2.477985**

<table>
<thead>
<tr>
<th>A4</th>
<th>A6</th>
<th>A8</th>
<th>A10</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.479591E-09</td>
<td>-8.743832E-14</td>
<td>1.936009E-18</td>
<td>-3.557055E-23</td>
</tr>
</tbody>
</table>

**Aspheric Surface 4 CC = -1.247590**

<table>
<thead>
<tr>
<th>A4</th>
<th>A6</th>
<th>A8</th>
<th>A10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.901898E-11</td>
<td>7.800049E-14</td>
<td>-1.635492E-18</td>
<td>2.510175E-23</td>
</tr>
</tbody>
</table>

**Aspheric Surface 6 CC = -0.354793 (Ellipse)**

<table>
<thead>
<tr>
<th>A4</th>
<th>A6</th>
<th>A8</th>
<th>A10</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3.357945E-09</td>
<td>1.277421E-14</td>
<td>-2.20762E-18</td>
<td>3.734123E-23</td>
</tr>
</tbody>
</table>

---

A7
<table>
<thead>
<tr>
<th>Defocus</th>
<th>U'</th>
<th>L</th>
<th>M</th>
<th>X</th>
<th>Y</th>
<th>OPD</th>
<th>COLOR</th>
<th>S</th>
<th>T</th>
<th>DIST(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.75000</td>
<td>-0.20000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.19969</td>
<td>0.04986</td>
<td>-0.01416</td>
<td>-0.03584</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.17977</td>
<td>0.11205</td>
<td>-0.01245</td>
<td>-0.02869</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHIEF RAY</td>
<td>-0.00367</td>
<td>117.07032</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.17954</td>
<td>0.04506</td>
<td>0.00699</td>
<td>0.04752</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.15564</td>
<td>0.04058</td>
<td>0.00601</td>
<td>0.04344</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.14243</td>
<td>0.04538</td>
<td>0.00545</td>
<td>0.04092</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.12483</td>
<td>0.05329</td>
<td>0.00458</td>
<td>0.03725</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.09603</td>
<td>0.05833</td>
<td>0.00294</td>
<td>0.03056</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.05610</td>
<td>0.03814</td>
<td>0.00092</td>
<td>0.01980</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.06281</td>
<td>0.03820</td>
<td>-0.00079</td>
<td>-0.02515</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.10180</td>
<td>0.05982</td>
<td>-0.00283</td>
<td>-0.04553</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.13616</td>
<td>0.07907</td>
<td>-0.00501</td>
<td>-0.06658</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.14681</td>
<td>0.11211</td>
<td>-0.00601</td>
<td>-0.07383</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.15956</td>
<td>0.16762</td>
<td>-0.00780</td>
<td>-0.08299</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.18130</td>
<td>-0.03959</td>
<td>-0.01074</td>
<td>-0.09906</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.19916</td>
<td>-0.00069</td>
<td>-0.00202</td>
<td>-0.03419</td>
<td>-0.01153</td>
<td>-0.03449</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.18002</td>
<td>-0.00122</td>
<td>0.07080</td>
<td>-0.01468</td>
<td>-0.01080</td>
<td>-0.02788</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.13936</td>
<td>-0.00217</td>
<td>0.11434</td>
<td>0.01284</td>
<td>-0.00658</td>
<td>-0.01638</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.10226</td>
<td>-0.00285</td>
<td>0.07775</td>
<td>0.01406</td>
<td>-0.00293</td>
<td>-0.00868</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MULTI-ACCESS FREE SPACE LASER COMMUNICATIONS TERMINAL - FINAL REPORT

LDT NASA TELESCOPE AT 860NM WAVELENGTH File ldtplus 10:49:05 02-14-1991

ZOOM POSITION 1
EFL = 625.402

# TYPE CURVE SEPN INDEX1 INDEX2 INDEX3 DISP'N CLRAD GLASS
1 S 0.000000 0.000 1.00000 1.00000 1.00000 0.00000 125.61
2 A 0.002135 400.000 1.00000 1.00000 1.00000 0.00000 195.34
3 S 0.003544 12.700 1.57597 1.57720 1.57597-0.00123 190.58
4 A 0.004324 1.000 1.00000 1.00000 1.00000 0.00000 193.87
5 S 0.000633 81.524 1.48466 1.48527 1.48466-0.00061 193.08
6 A 0.006593 543.944 1.00000 1.00000 1.00000 0.00000 170.50
7 S 0.002784 102.098 1.48466 1.48527 1.48466-0.00061 164.06
8 S 0.004486 1.000 1.00000 1.00000 1.00000 0.00000 151.30
9 S 0.007514 12.700 1.57597 1.57720 1.57597-0.00123 123.88
10 S 0.000000 130.000 1.00000 1.00000 1.00000 0.00000 117.52
11 S 0.000000 31899.000 1.00000 1.00000 1.00000 0.00000 116.66
12 S 0.000000 0.291 1.00000 1.00000 1.00000 0.00000 116.66

ASPHERIC SURFACE 2 CC = 2.477985

ASPHERIC SURFACE 4 CC = -1.247590
A4 = 1.901898E-11 A6 = 7.800049E-14 A8 = -1.635492E-18 A10 = 2.510175E-23

ASPHERIC SURFACE 6 CC = -0.357945 (ELLIPSE)
A4 = -3.357945E-09 A6 = 1.277421E-14 A8 = -2.20762E-18

Defocus = -0.750000 U' = -0.200000

Y OPD COLOR S T DIST(%) L M X
-0.19974 0.000782 -0.01189 -0.00674 0.000000
-0.09999 0.07005 -0.00352 -0.00153 0.000000

CHIEF RAY -0.00102 32.69479 0.418 0.593 -0.250%
0.19913 0.09423 -0.00566 -0.00205
0.17953 0.01079 -0.00668 -0.00122
0.15949 -0.02777 -0.00653 -0.00053
0.13914 -0.05146 -0.00570 0.00000
0.09931 -0.06087 -0.00333 0.00062
0.05917 -0.04151 -0.00121 0.00073
-0.06125 0.04188 -0.00112 -0.00182
-0.10137 0.10328 -0.00391 -0.00373
-0.14073 0.17362 -0.00946 -0.00624
-0.16151 0.17955 -0.01320 -0.00785
-0.18157 0.13922 -0.01648 -0.00962
-0.20059 0.05339 -0.01838 -0.01149
-0.19983 -0.00134 -0.04542 0.03665 -0.00676 -0.00675
-0.17603 -0.00127 0.04374 0.03022 -0.00661 -0.00511
-0.13997 -0.00118 0.06736 0.01578 -0.00434 -0.00311
-0.10010 -0.00111 0.04438 0.00310 -0.00208 -0.00154

ZOOM POSITION 2
EFL = 625.402

# TYPE CURVE SEPN INDEX1 INDEX2 INDEX3 DISP'N CLRAD GLASS
1 S 0.000000 0.000 1.00000 1.00000 1.00000 0.00000 125.61
2 A 0.002135 400.000 1.00000 1.00000 1.00000 0.00000 195.34
3 S 0.003544 12.700 1.57597 1.57720 1.57597-0.00123 190.58

-9-
MULTI-ACCESS FREE SPACE LASER COMMUNICATIONS TERMINAL - FINAL REPORT

<table>
<thead>
<tr>
<th>CURVE</th>
<th>SEPN</th>
<th>INDEX1</th>
<th>INDEX2</th>
<th>INDEX3</th>
<th>DISPEN</th>
<th>CLRAD</th>
<th>GLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 S</td>
<td>0.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>0.000000</td>
<td>125.61</td>
</tr>
<tr>
<td>2 A</td>
<td>0.002135</td>
<td>40.0000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>190.58</td>
</tr>
<tr>
<td>3 S</td>
<td>0.003544</td>
<td>12.7000</td>
<td>1.57597</td>
<td>1.57730</td>
<td>1.57597</td>
<td>0.00123</td>
<td>193.87</td>
</tr>
<tr>
<td>4 A</td>
<td>0.004324</td>
<td>12.7000</td>
<td>1.57597</td>
<td>1.57730</td>
<td>1.57597</td>
<td>0.00123</td>
<td>193.08</td>
</tr>
<tr>
<td>5 S</td>
<td>0.000633</td>
<td>12.7000</td>
<td>1.57597</td>
<td>1.57730</td>
<td>1.57597</td>
<td>0.00123</td>
<td>193.08</td>
</tr>
<tr>
<td>6 A</td>
<td>0.006593</td>
<td>12.7000</td>
<td>1.57597</td>
<td>1.57730</td>
<td>1.57597</td>
<td>0.00123</td>
<td>193.08</td>
</tr>
<tr>
<td>7 S</td>
<td>0.002784</td>
<td>12.7000</td>
<td>1.57597</td>
<td>1.57730</td>
<td>1.57597</td>
<td>0.00123</td>
<td>193.08</td>
</tr>
<tr>
<td>8 A</td>
<td>0.004486</td>
<td>12.7000</td>
<td>1.57597</td>
<td>1.57730</td>
<td>1.57597</td>
<td>0.00123</td>
<td>193.08</td>
</tr>
<tr>
<td>9 S</td>
<td>0.007514</td>
<td>12.7000</td>
<td>1.57597</td>
<td>1.57730</td>
<td>1.57597</td>
<td>0.00123</td>
<td>193.08</td>
</tr>
<tr>
<td>10 S</td>
<td>0.000000</td>
<td>12.7000</td>
<td>1.57597</td>
<td>1.57730</td>
<td>1.57597</td>
<td>0.00123</td>
<td>193.08</td>
</tr>
<tr>
<td>11 S</td>
<td>0.000000</td>
<td>12.7000</td>
<td>1.57597</td>
<td>1.57730</td>
<td>1.57597</td>
<td>0.00123</td>
<td>193.08</td>
</tr>
<tr>
<td>12 S</td>
<td>0.000000</td>
<td>12.7000</td>
<td>1.57597</td>
<td>1.57730</td>
<td>1.57597</td>
<td>0.00123</td>
<td>193.08</td>
</tr>
</tbody>
</table>

ZOOM POSITION 3
EFL = 625.402
MULTI-ACCESS FREE SPACE LASER COMMUNICATIONS TERMINAL - FINAL REPORT

13 S 0.000000 0.291 1.000000 1.000000 1.000000 0.000000 116.66

ASPHERIC SURFACE 2 CC = 2.477985

ASPHERIC SURFACE 4 CC = -1.247590
A4 = 1.901898E-11 A6 = 7.800049E-14 A8 = -1.635492E-18 A10 = 2.510175E-23

ASPHERIC SURFACE 6 CC = -0.354793 (ELLIPE)
A4 = -3.357945E-09 A6 = 1.277421E-14 A8 = -2.20762E-18 A10 = 3.734123E-23

Defocus = -0.750000 U' = -0.200000

<table>
<thead>
<tr>
<th>L</th>
<th>M</th>
<th>X</th>
<th>Y</th>
<th>OPD</th>
<th>COLOR</th>
<th>S</th>
<th>T</th>
<th>DIST(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.19974</td>
<td>0.00782</td>
<td>-0.01189</td>
<td>-0.00674</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.15987</td>
<td>0.10385</td>
<td>-0.00902</td>
<td>-0.00413</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>96.61543</td>
<td>-0.912</td>
<td>-0.135</td>
<td>-2.463%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.19787</td>
<td>-0.01894</td>
<td>-0.00682</td>
<td>0.00675</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.17881</td>
<td>-0.07098</td>
<td>-0.00590</td>
<td>0.00655</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.15785</td>
<td>-0.08929</td>
<td>-0.00416</td>
<td>0.00621</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.13745</td>
<td>-0.08022</td>
<td>-0.00240</td>
<td>0.00577</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.09723</td>
<td>-0.03064</td>
<td>-0.00012</td>
<td>0.00462</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.05966</td>
<td>0.00304</td>
<td>0.00030</td>
<td>0.00319</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.06584</td>
<td>0.02439</td>
<td>-0.00043</td>
<td>-0.00445</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.10366</td>
<td>0.03367</td>
<td>-0.00169</td>
<td>-0.00791</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.14178</td>
<td>-0.02558</td>
<td>-0.00202</td>
<td>-0.01211</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.16481</td>
<td>-0.04109</td>
<td>-0.00112</td>
<td>-0.01507</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.18581</td>
<td>0.02694</td>
<td>-0.00085</td>
<td>-0.01807</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.20146</td>
<td>-0.00352</td>
<td>-0.00138</td>
<td>-0.02048</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.20024</td>
<td>-0.00283</td>
<td>-0.26677</td>
<td>0.01110</td>
<td>0.01379</td>
<td>-0.00683</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.17817</td>
<td>-0.00286</td>
<td>-0.14624</td>
<td>0.02503</td>
<td>0.00935</td>
<td>-0.00528</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.14015</td>
<td>-0.00291</td>
<td>-0.06298</td>
<td>0.02921</td>
<td>0.00576</td>
<td>-0.00315</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.10124</td>
<td>-0.00297</td>
<td>-0.05346</td>
<td>0.01674</td>
<td>0.00359</td>
<td>-0.00160</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ZOOM POSITION 4
EFL = 625.402

<table>
<thead>
<tr>
<th>#</th>
<th>TYPE</th>
<th>CURVE</th>
<th>SEPN</th>
<th>INDEX1</th>
<th>INDEX2</th>
<th>INDEX3</th>
<th>DISPNS</th>
<th>CLRAD</th>
<th>GLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 S</td>
<td>0.000000</td>
<td>0.000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>0.000000</td>
<td>125.61</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 A</td>
<td>0.002135</td>
<td>400.000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>0.000000</td>
<td>190.58</td>
<td>POLYST</td>
<td></td>
</tr>
<tr>
<td>3 S</td>
<td>0.003544</td>
<td>12.700</td>
<td>1.57597</td>
<td>1.57720</td>
<td>1.57597</td>
<td>-0.00123</td>
<td>193.87</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 A</td>
<td>0.004324</td>
<td>1.000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>0.000000</td>
<td>193.87</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 S</td>
<td>0.000633</td>
<td>81.524</td>
<td>1.48466</td>
<td>1.48527</td>
<td>1.48466</td>
<td>-0.00061</td>
<td>164.06</td>
<td>ACRYLIC</td>
<td></td>
</tr>
<tr>
<td>6 A</td>
<td>0.006593</td>
<td>543.944</td>
<td>1.48466</td>
<td>1.48527</td>
<td>1.48466</td>
<td>-0.00061</td>
<td>164.06</td>
<td>ACRYLIC</td>
<td></td>
</tr>
<tr>
<td>7 S</td>
<td>0.002784</td>
<td>102.098</td>
<td>1.48466</td>
<td>1.48527</td>
<td>1.48466</td>
<td>-0.00061</td>
<td>164.06</td>
<td>ACRYLIC</td>
<td></td>
</tr>
<tr>
<td>8 S</td>
<td>0.004486</td>
<td>1.000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>0.000000</td>
<td>151.30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 S</td>
<td>0.007514</td>
<td>12.700</td>
<td>1.57597</td>
<td>1.57720</td>
<td>1.57597</td>
<td>-0.00123</td>
<td>123.88</td>
<td>POLYST</td>
<td></td>
</tr>
<tr>
<td>10 S</td>
<td>0.000000</td>
<td>130.000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>0.000000</td>
<td>117.52</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 S</td>
<td>0.000000031899.000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>0.000006503.48</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 S</td>
<td>0.0000000</td>
<td>0.000000</td>
<td>0.291</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>117.83</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ASPHERIC SURFACE 2 CC = 2.477985

ASPHERIC SURFACE 4 CC = -1.247590
A4 = 1.901898E-11 A6 = 7.800049E-14 A8 = -1.635492E-18 A10 = 2.510175E-23

ASPHERIC SURFACE 6 CC = -0.354793 (ELLIPE)
A4 = -3.357945E-09 A6 = 1.277421E-14 A8 = -2.20762E-18 A10 = 3.734123E-23

Defocus = -0.750000 U' = -0.200000

-AlI-
<table>
<thead>
<tr>
<th>L</th>
<th>M</th>
<th>X</th>
<th>Y</th>
<th>OPD</th>
<th>COLOR</th>
<th>S</th>
<th>T</th>
<th>DIST(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.19974</td>
<td>-0.17981</td>
<td>0.00782</td>
<td>-0.01189</td>
<td>-0.00674</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.19919</td>
<td>-0.18003</td>
<td>-0.00072</td>
<td>-0.04115</td>
<td>-0.07250</td>
<td>-0.00938</td>
<td>-0.00664</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.13935</td>
<td>-0.10225</td>
<td>0.09865</td>
<td>-0.00714</td>
<td>-0.00593</td>
<td>-0.00307</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CHIEF RAY: -0.00366

<table>
<thead>
<tr>
<th>L</th>
<th>M</th>
<th>X</th>
<th>Y</th>
<th>OPD</th>
<th>COLOR</th>
<th>S</th>
<th>T</th>
<th>DIST(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.17959</td>
<td>0.15563</td>
<td>-0.04771</td>
<td>-0.00019</td>
<td>0.00886</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.14241</td>
<td>0.12479</td>
<td>-0.03813</td>
<td>0.00089</td>
<td>0.00807</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.05606</td>
<td>0.09598</td>
<td>-0.02503</td>
<td>0.00131</td>
<td>0.00759</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.06275</td>
<td>-0.10174</td>
<td>-0.00574</td>
<td>0.00159</td>
<td>0.00690</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.13614</td>
<td>-0.14682</td>
<td>-0.01784</td>
<td>0.00137</td>
<td>0.00565</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.15960</td>
<td>-0.18124</td>
<td>0.02024</td>
<td>0.00050</td>
<td>0.00366</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.12479</td>
<td>-0.00574</td>
<td>-0.00006</td>
<td>0.00069</td>
<td>-0.00473</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.10174</td>
<td>-0.00574</td>
<td>-0.00020</td>
<td>0.00069</td>
<td>-0.00473</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.13614</td>
<td>-0.14682</td>
<td>0.01518</td>
<td>0.00028</td>
<td>0.001286</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.14682</td>
<td>-0.15960</td>
<td>0.03183</td>
<td>0.00030</td>
<td>0.001432</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.15960</td>
<td>-0.18124</td>
<td>0.07045</td>
<td>-0.00036</td>
<td>0.001618</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.18124</td>
<td>-0.15960</td>
<td>-0.07965</td>
<td>-0.00476</td>
<td>-0.01952</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

-0.13935 -0.00218

0.07019

0.00257 -0.00269 -0.00161

116.70510

0.281 0.013 -3.999
### LDT NASA Telescope at 820nm Wavelength

**File Idtplan**

#### Zoom Position 1

<table>
<thead>
<tr>
<th>EFL</th>
<th>625.007</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>#</th>
<th>TYPE</th>
<th>CURVE</th>
<th>SEPN</th>
<th>INDEX1</th>
<th>INDEX2</th>
<th>INDEX3</th>
<th>DISPNS</th>
<th>CLRAD</th>
<th>GLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S</td>
<td>0.000000</td>
<td>0.000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>0.000000</td>
<td>125.61</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>0.002135</td>
<td>400.000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>0.000000</td>
<td>195.34</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>S</td>
<td>0.003544</td>
<td>12.700</td>
<td>1.57720</td>
<td>1.577597</td>
<td>1.57720</td>
<td>0.00123</td>
<td>190.58</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>A</td>
<td>0.004324</td>
<td>1.000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>0.000000</td>
<td>193.87</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>S</td>
<td>0.000633</td>
<td>81.524</td>
<td>1.48527</td>
<td>1.48464</td>
<td>1.48527</td>
<td>0.00061</td>
<td>193.08</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>A</td>
<td>0.006593</td>
<td>543.944</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>0.000000</td>
<td>170.50</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>S</td>
<td>0.002784</td>
<td>102.098</td>
<td>1.48527</td>
<td>1.48464</td>
<td>1.48527</td>
<td>0.00061</td>
<td>164.06</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>0.004486</td>
<td>1.000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>0.000000</td>
<td>151.30</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>S</td>
<td>0.007514</td>
<td>12.700</td>
<td>1.57720</td>
<td>1.577597</td>
<td>1.57720</td>
<td>0.00123</td>
<td>123.88</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>S</td>
<td>0.000000</td>
<td>130.000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>0.000000</td>
<td>117.52</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>S</td>
<td>0.000000</td>
<td>0.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>0.000000</td>
<td>116.66</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>S</td>
<td>0.000000</td>
<td>-0.004</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>0.000000</td>
<td>116.66</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>S</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>116.66</td>
<td></td>
</tr>
</tbody>
</table>

#### Aspheric Surface 2

- **Equation:** \( A4 = 4.479591E-09 \), \( A6 = -8.743832E-14 \), \( A8 = 1.936009E-18 \)
- **Defocus:** -0.750000
- **U':** -0.200000

#### Aspheric Surface 4

- **Equation:** \( A4 = 1.901898E-11 \), \( A6 = 7.800049E-14 \), \( A8 = -1.635492E-18 \)
- **Defocus:** -0.750000
- **U':** -0.200000

#### Aspheric Surface 6 (Ellipse)

- **Equation:** \( A4 = -3.357945E-09 \), \( A6 = 1.277421E-14 \), \( A8 = -2.207620E-18 \)
- **Defocus:** -0.750000
- **U':** -0.200000

#### Chief Ray

- **Equation:** \( A0 = -3.557055E-23 \), \( A2 = 2.510175E-23 \), \( A4 = 3.734123E-23 \)

#### Zoom Position 2

<table>
<thead>
<tr>
<th>EFL</th>
<th>625.007</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>#</th>
<th>TYPE</th>
<th>CURVE</th>
<th>SEPN</th>
<th>INDEX1</th>
<th>INDEX2</th>
<th>INDEX3</th>
<th>DISPNS</th>
<th>CLRAD</th>
<th>GLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S</td>
<td>0.000000</td>
<td>0.000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>0.000000</td>
<td>125.61</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>0.002135</td>
<td>400.000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>0.000000</td>
<td>195.34</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>S</td>
<td>0.003544</td>
<td>12.700</td>
<td>1.57720</td>
<td>1.577597</td>
<td>1.57720</td>
<td>0.00123</td>
<td>190.58</td>
<td></td>
</tr>
</tbody>
</table>

### Multi-Access Free Space Laser Communications Terminal - Final Report

-
- **File Idtplan**
- **Zoom Position 1**
- **Zoom Position 2**
- **EFL:** 625.007
- **Glare:** 0.000820 0.000860 0.000820
- **Glass:** POLYST
ASPHERIC SURFACE 2 CC = 2.477985
ASPHERIC SURFACE 4 CC = -1.247590
A4 = 1.901898E-11 A6 = 7.800049E-14 A8 = -1.635492E-18 A10 = 2.510175E-23
ASPHERIC SURFACE 6 CC = -3.357945E-09 (ELLIPSE)
A4 = -3.357945E-09 A6 = 1.277421E-14 A8 = -2.20762E-18 A10 = 3.734123E-23
Defocus = -0.750000 U' = -0.200000
L M X Y OPD COLOR S T DIST(%) 65.00388
-0.19976 -0.00731 -0.01109 0.00673
-0.13994 0.09487 -0.00675 0.00310
-0.00204 65.00388
-0.19832 0.11268 -0.00361 -0.00248
0.18087 0.03426 -0.00485 -0.00275
0.16052 -0.02241 -0.00491 -0.00294
0.13815 -0.05179 -0.00403 -0.00298
0.09953 -0.04984 -0.00191 -0.00271
0.05891 -0.02093 -0.00046 -0.00197
-0.06301 0.04239 -0.00085 0.00311
-0.10366 0.11834 -0.00407 0.00597
-0.14219 0.15001 -0.00955 0.00936
-0.16475 0.11225 -0.01259 0.01171
-0.18513 0.06032 -0.01433 0.01408
-0.20236 0.05013 -0.01520 0.01630
-0.19989 -0.00248 -0.19576 0.04791 0.00725 0.00677
-0.17676 -0.00238 -0.08250 0.04536 0.00419 0.00516
-0.13997 -0.00226 -0.02417 0.02895 0.00255 0.00312
-0.10049 -0.00216 -0.02368 0.00978 0.00167 0.00156

ZOOM POSITION 3
EFL = 625.007

<table>
<thead>
<tr>
<th>#</th>
<th>TYPE</th>
<th>CURVE</th>
<th>SEPN</th>
<th>INDEX1</th>
<th>INDEX2</th>
<th>INDEX3</th>
<th>DISP</th>
<th>CLR</th>
<th>GLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S</td>
<td>0.000000</td>
<td>0.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>POLYST</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>0.002135</td>
<td>400.00</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>195.34</td>
<td>POLYST</td>
</tr>
<tr>
<td>3</td>
<td>S</td>
<td>0.003544</td>
<td>12.700</td>
<td>1.57720</td>
<td>1.57597</td>
<td>1.57720</td>
<td>0.00123</td>
<td>190.58</td>
<td>POLYST</td>
</tr>
<tr>
<td>4</td>
<td>A</td>
<td>0.004324</td>
<td>1.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>193.87</td>
<td>POLYST</td>
</tr>
<tr>
<td>5</td>
<td>S</td>
<td>0.006593</td>
<td>81.512</td>
<td>1.48527</td>
<td>1.48466</td>
<td>1.48527</td>
<td>0.00061</td>
<td>193.08</td>
<td>ACRYLC</td>
</tr>
<tr>
<td>6</td>
<td>A</td>
<td>0.002784</td>
<td>102.098</td>
<td>1.48527</td>
<td>1.48466</td>
<td>1.48527</td>
<td>0.00061</td>
<td>164.06</td>
<td>ACRYLC</td>
</tr>
<tr>
<td>7</td>
<td>S</td>
<td>0.004486</td>
<td>1.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>151.30</td>
<td>ACRYLC</td>
</tr>
<tr>
<td>8</td>
<td>A</td>
<td>0.007514</td>
<td>12.700</td>
<td>1.57720</td>
<td>1.57597</td>
<td>1.57720</td>
<td>0.00123</td>
<td>123.88</td>
<td>POLYST</td>
</tr>
<tr>
<td>9</td>
<td>S</td>
<td>0.000000</td>
<td>130.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>117.52</td>
<td>POLYST</td>
</tr>
<tr>
<td>10</td>
<td>S</td>
<td>0.000000</td>
<td>31889.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>6504.12</td>
<td>POLYST</td>
</tr>
<tr>
<td>11</td>
<td>S</td>
<td>0.000000</td>
<td>31899.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>116.66</td>
<td>POLYST</td>
</tr>
<tr>
<td>12</td>
<td>S</td>
<td>0.000000</td>
<td>-0.004</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>116.66</td>
<td>POLYST</td>
</tr>
</tbody>
</table>
ASPERIC SURFACE 2 CC = 2.477985

ASPERIC SURFACE 4 CC = -1.247590
A4 = 1.901898E-11 A6 = 7.800049E-14 A8 = -1.635492E-18 A10 = 2.510175E-23

ASPERIC SURFACE 6 CC = -0.357945E-09 A6 = 1.277421E-14 A8 = -2.20762E-18 A10 = 3.734123E-23

Defocus = -0.750000 U' = -0.200000

ZOOM POSITION 4
EFL = 625.007

# TYPE CURVE SEPN INDEX1 INDEX2 INDEX3 DISP DN CLRAD GLASS
1 S 0.000000 0.00 1.000000 1.000000 0.000000 125.61
2 A 0.002135 400.000 1.000000 1.000000 1.000000 195.34 POLYST
3 S 0.003544 12.700 1.57720 1.57597 1.57720 0.00123 190.58 ACRYL
4 A 0.004324 1.00 1.000000 1.000000 1.000000 193.87
5 S 0.006933 543.944 1.000000 1.000000 1.000000 170.50
6 A 0.009628 102.098 1.48527 1.48466 1.48527 0.00061 164.06
7 S 0.002784 12.700 1.57720 1.57597 1.57720 0.00123 123.88 POLYST
8 S 0.004486 1.00 1.000000 1.000000 1.000000 151.30
9 S 0.007514 12.700 1.57720 1.57597 1.57720 0.00123 117.52
10 S 0.000000 130.000 1.000000 1.000000 1.000000 0.000000 117.52
11 S 0.000000 31899.000 1.000000 1.000000 1.000000 0.000000 117.52
12 S 0.000000 130.000 1.000000 1.000000 1.000000 0.000000 117.52
13 S 0.000000 -0.004 1.000000 1.000000 1.000000 0.000000 117.83

ASPERIC SURFACE 2 CC = 2.477985

ASPERIC SURFACE 4 CC = -1.247590
A4 = 1.901898E-11 A6 = 7.800049E-14 A8 = -1.635492E-18 A10 = 2.510175E-23

ASPERIC SURFACE 6 CC = -0.357945E-09 A6 = 1.277421E-14 A8 = -2.20762E-18 A10 = 3.734123E-23

Defocus = -0.750000 U' = -0.200000

-A15-
<table>
<thead>
<tr>
<th>L</th>
<th>M</th>
<th>X</th>
<th>Y</th>
<th>OPD</th>
<th>COLOR</th>
<th>S</th>
<th>T</th>
<th>DIST (°)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.19976</td>
<td>-0.00731</td>
<td>-0.01109</td>
<td>0.00673</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.17982</td>
<td>0.06854</td>
<td>-0.01038</td>
<td>0.00533</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHIEF RAY</td>
<td>-0.00366</td>
<td>116.63673</td>
<td>0.268</td>
<td>-0.006</td>
<td>-3.995</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.17958</td>
<td>-0.06361</td>
<td>-0.00129</td>
<td>-0.00884</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.15562</td>
<td>-0.05132</td>
<td>0.00014</td>
<td>-0.00805</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.14239</td>
<td>-0.03663</td>
<td>0.00072</td>
<td>-0.00757</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.12477</td>
<td>-0.01517</td>
<td>0.00118</td>
<td>-0.00688</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.09596</td>
<td>0.01189</td>
<td>0.00119</td>
<td>-0.00563</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.05604</td>
<td>0.01824</td>
<td>0.00047</td>
<td>-0.00365</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.06274</td>
<td>0.02587</td>
<td>-0.00059</td>
<td>0.00472</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.10173</td>
<td>0.01991</td>
<td>-0.00167</td>
<td>0.00865</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.13613</td>
<td>-0.00883</td>
<td>-0.00175</td>
<td>0.01283</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.14682</td>
<td>0.00328</td>
<td>-0.00170</td>
<td>0.01428</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.15960</td>
<td>0.03666</td>
<td>-0.00195</td>
<td>0.01614</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.18124</td>
<td>-0.11087</td>
<td>-0.00228</td>
<td>0.01947</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.19920</td>
<td>-0.00073</td>
<td>-0.05721</td>
<td>-0.08369</td>
<td>-0.00844</td>
<td>0.00662</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.18004</td>
<td>0.02893</td>
<td>-0.05533</td>
<td>-0.00861</td>
<td>0.00531</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.13936</td>
<td>0.09193</td>
<td>-0.01287</td>
<td>-0.00564</td>
<td>0.00306</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.10225</td>
<td>0.06682</td>
<td>-0.00063</td>
<td>-0.00257</td>
<td>0.00160</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**LDT NASA TELESCOPE AT 820NM WAVELENGTH File ldtplas**  
10:07:21 02-14-1991

**ZOOM POSITION 1**

EFL = 625.007

<table>
<thead>
<tr>
<th>#</th>
<th>TYPE</th>
<th>CURVE</th>
<th>SEPN</th>
<th>INDEX1</th>
<th>INDEX2</th>
<th>INDEX3</th>
<th>DISP/N</th>
<th>CRLAD</th>
<th>GLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S</td>
<td>0.000000</td>
<td>0.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>125.61</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>0.002135</td>
<td>400.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>195.34</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>S</td>
<td>0.003544</td>
<td>12.700</td>
<td>1.57720</td>
<td>1.57597</td>
<td>1.57720</td>
<td>0.00123</td>
<td>190.58</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>A</td>
<td>0.004324</td>
<td>1.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>193.87</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>S</td>
<td>0.00633</td>
<td>81.524</td>
<td>1.48527</td>
<td>1.48466</td>
<td>1.48527</td>
<td>0.00061</td>
<td>193.08</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>A</td>
<td>0.006953</td>
<td>543.944</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>170.50</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>S</td>
<td>0.007278</td>
<td>102.098</td>
<td>1.48527</td>
<td>1.48466</td>
<td>1.48527</td>
<td>0.00061</td>
<td>164.06</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>0.004486</td>
<td>1.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>151.30</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>S</td>
<td>0.007514</td>
<td>12.700</td>
<td>1.57720</td>
<td>1.57597</td>
<td>1.57720</td>
<td>0.00123</td>
<td>123.88</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>S</td>
<td>0.000000</td>
<td>130.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>117.52</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>S</td>
<td>0.000000</td>
<td>31899.000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>116.66</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>S</td>
<td>0.000000</td>
<td>-0.003</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>116.66</td>
<td></td>
</tr>
</tbody>
</table>

**ASPHERIC SURFACE 2 CC = 2.477985**

A4 = 4.479591E-09  
A6 = -8.743832E-14  
A8 = 1.936009E-18  
A10 = -3.557055E-23

**ASPHERIC SURFACE 4 CC = -1.247959**

A4 = 1.901898E-11  
A6 = 7.800049E-14  
A8 = -1.635492E-18  
A10 = 2.510175E-23

**ASPHERIC SURFACE 6 CC = -0.354793 (ELLIPSE)**

A4 = -3.357945E-09  
A6 = -1.277421E-14  
A8 = -2.20762E-18

---

**LDT NASA TELESCOPE AT 820NM WAVELENGTH File ldtplas**  
10:08:10 02-14-1991

**ZOOM POSITION 1**

EFL = 625.007

<table>
<thead>
<tr>
<th>RADIUS</th>
<th>SEPN</th>
<th>CLR DIAM</th>
<th>MATERIAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>468.277</td>
<td>0.000000</td>
<td>400.000</td>
<td>Air</td>
</tr>
<tr>
<td>282.162</td>
<td>0.000000</td>
<td>12.700</td>
<td>POLYST</td>
</tr>
<tr>
<td>231.275</td>
<td>0.000000</td>
<td>1.000</td>
<td>Air</td>
</tr>
<tr>
<td>1579.829</td>
<td>0.000000</td>
<td>81.524</td>
<td>ACRYLC</td>
</tr>
<tr>
<td>151.670</td>
<td>0.000000</td>
<td>543.944</td>
<td>Air</td>
</tr>
<tr>
<td>359.155</td>
<td>0.000000</td>
<td>102.098</td>
<td>ACRYLC</td>
</tr>
<tr>
<td>222.913</td>
<td>0.000000</td>
<td>1.000</td>
<td>Air</td>
</tr>
<tr>
<td>133.093</td>
<td>0.000000</td>
<td>12.700</td>
<td>POLYST</td>
</tr>
<tr>
<td>31899.000</td>
<td>0.000000</td>
<td>130.000</td>
<td>Air</td>
</tr>
<tr>
<td>13147.45</td>
<td>0.000000</td>
<td>235.03</td>
<td>Air</td>
</tr>
<tr>
<td>-0.003</td>
<td>0.000000</td>
<td>233.32</td>
<td>Air</td>
</tr>
</tbody>
</table>

**ASPHERIC SURFACE 2 CC = 2.477985**
MULTI-ACCESS FREE SPACE LASER COMMUNICATIONS TERMINAL - FINAL REPORT

ASPHERIC SURFACE 4 CC = -1.247590
A4 = 1.901898E-11  A6 = 7.800049E-14  A8 =-1.635492E-18  A10 = 2.510175E-23
ASPHERIC SURFACE 6 CC = -0.354793 (ELLIPSE)
A4 =-3.357945E-09  A6 = 1.277421E-14  A8 =-2.20762E-18  A10 = 3.734123E-23

LDT NASA TELESCOPE AT 820NM WAVELENGTH File ldtplas 10:08:43 02-14-1991
ZOOM POSITION 1
EFL = 625.007

<table>
<thead>
<tr>
<th>#</th>
<th>Type</th>
<th>Xrel</th>
<th>Yrel</th>
<th>X</th>
<th>Y</th>
<th>L</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>P</td>
<td>0.000</td>
<td>1.000</td>
<td>0.00000</td>
<td>125.00238</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>2</td>
<td>M</td>
<td>0.000</td>
<td>0.500</td>
<td>0.00000</td>
<td>62.50119</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>4</td>
<td>CD</td>
<td>0.000</td>
<td>3.000</td>
<td>0.00000</td>
<td>1.26883</td>
<td>0.000000</td>
<td>0.052336</td>
</tr>
<tr>
<td>5</td>
<td>MD</td>
<td>0.000</td>
<td>-0.998</td>
<td>0.00000</td>
<td>-123.46299</td>
<td>0.000000</td>
<td>0.052336</td>
</tr>
<tr>
<td>6</td>
<td>M</td>
<td>0.000</td>
<td>-0.900</td>
<td>0.00000</td>
<td>-111.23332</td>
<td>0.000000</td>
<td>0.052336</td>
</tr>
<tr>
<td>7</td>
<td>M</td>
<td>0.000</td>
<td>-0.800</td>
<td>0.00000</td>
<td>-98.73308</td>
<td>0.000000</td>
<td>0.052336</td>
</tr>
<tr>
<td>8</td>
<td>M</td>
<td>0.000</td>
<td>-0.698</td>
<td>0.00000</td>
<td>-86.04344</td>
<td>0.000000</td>
<td>0.052336</td>
</tr>
<tr>
<td>9</td>
<td>M</td>
<td>0.000</td>
<td>-0.500</td>
<td>0.00000</td>
<td>-61.23237</td>
<td>0.000000</td>
<td>0.052336</td>
</tr>
<tr>
<td>10</td>
<td>M</td>
<td>0.000</td>
<td>-0.300</td>
<td>0.00000</td>
<td>-36.23189</td>
<td>0.000000</td>
<td>0.052336</td>
</tr>
<tr>
<td>11</td>
<td>M</td>
<td>0.000</td>
<td>0.300</td>
<td>0.00000</td>
<td>38.76954</td>
<td>0.000000</td>
<td>0.052336</td>
</tr>
<tr>
<td>12</td>
<td>M</td>
<td>0.000</td>
<td>0.500</td>
<td>0.00000</td>
<td>63.77002</td>
<td>0.000000</td>
<td>0.052336</td>
</tr>
<tr>
<td>13</td>
<td>M</td>
<td>0.000</td>
<td>0.696</td>
<td>0.00000</td>
<td>88.31071</td>
<td>0.000000</td>
<td>0.052336</td>
</tr>
<tr>
<td>14</td>
<td>M</td>
<td>0.000</td>
<td>0.800</td>
<td>0.00000</td>
<td>101.27073</td>
<td>0.000000</td>
<td>0.052336</td>
</tr>
<tr>
<td>15</td>
<td>M</td>
<td>0.000</td>
<td>0.900</td>
<td>0.00000</td>
<td>113.77097</td>
<td>0.000000</td>
<td>0.052336</td>
</tr>
<tr>
<td>16</td>
<td>M</td>
<td>0.000</td>
<td>0.995</td>
<td>0.00000</td>
<td>125.61438</td>
<td>0.000000</td>
<td>0.052336</td>
</tr>
<tr>
<td>17</td>
<td>S</td>
<td>0.999</td>
<td>0.000</td>
<td>124.89674</td>
<td>1.26883</td>
<td>0.000000</td>
<td>0.052336</td>
</tr>
<tr>
<td>18</td>
<td>S</td>
<td>0.880</td>
<td>0.000</td>
<td>110.00210</td>
<td>1.26883</td>
<td>0.000000</td>
<td>0.052336</td>
</tr>
<tr>
<td>19</td>
<td>S</td>
<td>0.699</td>
<td>0.000</td>
<td>87.42772</td>
<td>1.26883</td>
<td>0.000000</td>
<td>0.052336</td>
</tr>
<tr>
<td>20</td>
<td>S</td>
<td>0.500</td>
<td>0.000</td>
<td>62.50119</td>
<td>1.26883</td>
<td>0.000000</td>
<td>0.052336</td>
</tr>
</tbody>
</table>

ZOOM POSITION 2
EFL = 625.007

<table>
<thead>
<tr>
<th>#</th>
<th>Type</th>
<th>Xrel</th>
<th>Yrel</th>
<th>X</th>
<th>Y</th>
<th>L</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>PZ</td>
<td>0.000</td>
<td>1.000</td>
<td>0.00000</td>
<td>125.00238</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>22</td>
<td>MD</td>
<td>0.000</td>
<td>1.000</td>
<td>0.00000</td>
<td>125.00238</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>23</td>
<td>M</td>
<td>0.000</td>
<td>0.700</td>
<td>0.00000</td>
<td>87.50167</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>24</td>
<td>CD</td>
<td>0.000</td>
<td>6.000</td>
<td>0.00000</td>
<td>2.42560</td>
<td>0.000000</td>
<td>0.104529</td>
</tr>
<tr>
<td>25</td>
<td>MD</td>
<td>0.000</td>
<td>-0.986</td>
<td>0.00000</td>
<td>-120.79125</td>
<td>0.000000</td>
<td>0.104529</td>
</tr>
<tr>
<td>26</td>
<td>M</td>
<td>0.000</td>
<td>-0.900</td>
<td>0.00000</td>
<td>-110.07654</td>
<td>0.000000</td>
<td>0.104529</td>
</tr>
<tr>
<td>27</td>
<td>M</td>
<td>0.000</td>
<td>-0.800</td>
<td>0.00000</td>
<td>-97.57631</td>
<td>0.000000</td>
<td>0.104529</td>
</tr>
<tr>
<td>28</td>
<td>M</td>
<td>0.000</td>
<td>-0.690</td>
<td>0.00000</td>
<td>-83.82619</td>
<td>0.000000</td>
<td>0.104529</td>
</tr>
<tr>
<td>29</td>
<td>M</td>
<td>0.000</td>
<td>-0.500</td>
<td>0.00000</td>
<td>-70.07559</td>
<td>0.000000</td>
<td>0.104529</td>
</tr>
<tr>
<td>30</td>
<td>M</td>
<td>0.000</td>
<td>-0.300</td>
<td>0.00000</td>
<td>-35.07512</td>
<td>0.000000</td>
<td>0.104529</td>
</tr>
<tr>
<td>31</td>
<td>M</td>
<td>0.000</td>
<td>0.300</td>
<td>0.00000</td>
<td>39.92631</td>
<td>0.000000</td>
<td>0.104529</td>
</tr>
<tr>
<td>32</td>
<td>M</td>
<td>0.000</td>
<td>0.500</td>
<td>0.00000</td>
<td>64.92679</td>
<td>0.000000</td>
<td>0.104529</td>
</tr>
<tr>
<td>33</td>
<td>M</td>
<td>0.000</td>
<td>0.689</td>
<td>0.00000</td>
<td>88.59774</td>
<td>0.000000</td>
<td>0.104529</td>
</tr>
<tr>
<td>34</td>
<td>M</td>
<td>0.000</td>
<td>0.800</td>
<td>0.00000</td>
<td>102.42751</td>
<td>0.000000</td>
<td>0.104529</td>
</tr>
<tr>
<td>35</td>
<td>M</td>
<td>0.000</td>
<td>0.900</td>
<td>0.00000</td>
<td>114.92774</td>
<td>0.000000</td>
<td>0.104529</td>
</tr>
<tr>
<td>36</td>
<td>MD</td>
<td>0.000</td>
<td>0.985</td>
<td>0.00000</td>
<td>125.52866</td>
<td>0.000000</td>
<td>0.104529</td>
</tr>
<tr>
<td>37</td>
<td>S</td>
<td>0.995</td>
<td>0.000</td>
<td>124.40148</td>
<td>2.42560</td>
<td>0.000000</td>
<td>0.104529</td>
</tr>
<tr>
<td>38</td>
<td>S</td>
<td>0.880</td>
<td>0.000</td>
<td>110.00210</td>
<td>2.42560</td>
<td>0.000000</td>
<td>0.104529</td>
</tr>
<tr>
<td>39</td>
<td>S</td>
<td>0.697</td>
<td>0.000</td>
<td>87.08103</td>
<td>2.42560</td>
<td>0.000000</td>
<td>0.104529</td>
</tr>
<tr>
<td>40</td>
<td>S</td>
<td>0.500</td>
<td>0.000</td>
<td>62.50119</td>
<td>2.42560</td>
<td>0.000000</td>
<td>0.104529</td>
</tr>
</tbody>
</table>
### ZOOM POSITION 3

**EFL = 625.007**

<table>
<thead>
<tr>
<th>#</th>
<th>Type</th>
<th>Xrel</th>
<th>Yrel</th>
<th>X</th>
<th>Y</th>
<th>L</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>41</td>
<td>PZ</td>
<td>0.000</td>
<td>1.000</td>
<td>0.00000</td>
<td>125.00238</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>42</td>
<td>MD</td>
<td>0.000</td>
<td>1.000</td>
<td>0.00000</td>
<td>125.00238</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>43</td>
<td>M</td>
<td>0.000</td>
<td>0.800</td>
<td>0.00000</td>
<td>100.00191</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>44</td>
<td>CD</td>
<td>0.000</td>
<td>9.000</td>
<td>0.00000</td>
<td>5.60687</td>
<td>0.000000</td>
<td>0.156435</td>
</tr>
<tr>
<td>45</td>
<td>MD</td>
<td>0.000</td>
<td>-0.961</td>
<td>0.00000</td>
<td>-114.48861</td>
<td>0.000000</td>
<td>0.156435</td>
</tr>
<tr>
<td>46</td>
<td>M</td>
<td>0.000</td>
<td>-0.870</td>
<td>0.00000</td>
<td>-103.14520</td>
<td>0.000000</td>
<td>0.156435</td>
</tr>
<tr>
<td>47</td>
<td>M</td>
<td>0.000</td>
<td>-0.770</td>
<td>0.00000</td>
<td>-90.64496</td>
<td>0.000000</td>
<td>0.156435</td>
</tr>
<tr>
<td>48</td>
<td>M</td>
<td>0.000</td>
<td>-0.673</td>
<td>0.00000</td>
<td>-78.49396</td>
<td>0.000000</td>
<td>0.156435</td>
</tr>
<tr>
<td>49</td>
<td>M</td>
<td>0.000</td>
<td>-0.300</td>
<td>0.00000</td>
<td>-31.89384</td>
<td>0.000000</td>
<td>0.156435</td>
</tr>
<tr>
<td>50</td>
<td>M</td>
<td>0.000</td>
<td>0.300</td>
<td>0.00000</td>
<td>43.10759</td>
<td>0.000000</td>
<td>0.156435</td>
</tr>
<tr>
<td>51</td>
<td>M</td>
<td>0.000</td>
<td>0.480</td>
<td>0.00000</td>
<td>65.60801</td>
<td>0.000000</td>
<td>0.156435</td>
</tr>
<tr>
<td>52</td>
<td>M</td>
<td>0.000</td>
<td>0.661</td>
<td>0.00000</td>
<td>88.20391</td>
<td>0.000000</td>
<td>0.156435</td>
</tr>
<tr>
<td>53</td>
<td>M</td>
<td>0.000</td>
<td>0.770</td>
<td>0.00000</td>
<td>101.85870</td>
<td>0.000000</td>
<td>0.156435</td>
</tr>
<tr>
<td>54</td>
<td>M</td>
<td>0.000</td>
<td>0.870</td>
<td>0.00000</td>
<td>114.35894</td>
<td>0.000000</td>
<td>0.156435</td>
</tr>
<tr>
<td>55</td>
<td>M</td>
<td>0.000</td>
<td>0.944</td>
<td>0.00000</td>
<td>123.60264</td>
<td>0.000000</td>
<td>0.156435</td>
</tr>
<tr>
<td>56</td>
<td>S</td>
<td>0.989</td>
<td>0.000</td>
<td>123.61288</td>
<td>5.60687</td>
<td>0.000000</td>
<td>0.156435</td>
</tr>
<tr>
<td>57</td>
<td>S</td>
<td>0.880</td>
<td>0.000</td>
<td>110.00210</td>
<td>5.60687</td>
<td>0.000000</td>
<td>0.156435</td>
</tr>
<tr>
<td>58</td>
<td>S</td>
<td>0.692</td>
<td>0.000</td>
<td>86.52901</td>
<td>5.60687</td>
<td>0.000000</td>
<td>0.156435</td>
</tr>
<tr>
<td>59</td>
<td>S</td>
<td>0.500</td>
<td>0.000</td>
<td>62.50119</td>
<td>5.60687</td>
<td>0.000000</td>
<td>0.156435</td>
</tr>
</tbody>
</table>

### ZOOM POSITION 4

**EFL = 625.007**

<table>
<thead>
<tr>
<th>#</th>
<th>Type</th>
<th>Xrel</th>
<th>Yrel</th>
<th>X</th>
<th>Y</th>
<th>L</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>61</td>
<td>PZ</td>
<td>0.000</td>
<td>1.000</td>
<td>0.00000</td>
<td>125.00238</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>62</td>
<td>MD</td>
<td>0.000</td>
<td>1.000</td>
<td>0.00000</td>
<td>125.00238</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>63</td>
<td>M</td>
<td>0.000</td>
<td>0.900</td>
<td>0.00000</td>
<td>112.50214</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>64</td>
<td>CD</td>
<td>0.000</td>
<td>11.000</td>
<td>0.00000</td>
<td>4.40677</td>
<td>0.000000</td>
<td>0.190809</td>
</tr>
<tr>
<td>65</td>
<td>MD</td>
<td>0.000</td>
<td>-0.829</td>
<td>0.00000</td>
<td>-99.19891</td>
<td>0.000000</td>
<td>0.190809</td>
</tr>
<tr>
<td>66</td>
<td>M</td>
<td>0.000</td>
<td>-0.720</td>
<td>0.00000</td>
<td>-85.59495</td>
<td>0.000000</td>
<td>0.190809</td>
</tr>
<tr>
<td>67</td>
<td>M</td>
<td>0.000</td>
<td>-0.660</td>
<td>0.00000</td>
<td>-78.09481</td>
<td>0.000000</td>
<td>0.190809</td>
</tr>
<tr>
<td>68</td>
<td>M</td>
<td>0.000</td>
<td>-0.580</td>
<td>0.00000</td>
<td>-68.11722</td>
<td>0.000000</td>
<td>0.190809</td>
</tr>
<tr>
<td>69</td>
<td>M</td>
<td>0.000</td>
<td>-0.450</td>
<td>0.00000</td>
<td>-51.84431</td>
<td>0.000000</td>
<td>0.190809</td>
</tr>
<tr>
<td>70</td>
<td>M</td>
<td>0.000</td>
<td>-0.270</td>
<td>0.00000</td>
<td>-29.34388</td>
<td>0.000000</td>
<td>0.190809</td>
</tr>
<tr>
<td>71</td>
<td>M</td>
<td>0.000</td>
<td>0.270</td>
<td>0.00000</td>
<td>38.15741</td>
<td>0.000000</td>
<td>0.190809</td>
</tr>
<tr>
<td>72</td>
<td>M</td>
<td>0.000</td>
<td>0.450</td>
<td>0.00000</td>
<td>60.65784</td>
<td>0.000000</td>
<td>0.190809</td>
</tr>
<tr>
<td>73</td>
<td>M</td>
<td>0.000</td>
<td>0.610</td>
<td>0.00000</td>
<td>80.65822</td>
<td>0.000000</td>
<td>0.190809</td>
</tr>
<tr>
<td>74</td>
<td>M</td>
<td>0.000</td>
<td>0.660</td>
<td>0.00000</td>
<td>86.90834</td>
<td>0.000000</td>
<td>0.190809</td>
</tr>
<tr>
<td>75</td>
<td>M</td>
<td>0.000</td>
<td>0.720</td>
<td>0.00000</td>
<td>94.40848</td>
<td>0.000000</td>
<td>0.190809</td>
</tr>
<tr>
<td>76</td>
<td>M</td>
<td>0.000</td>
<td>0.820</td>
<td>0.00000</td>
<td>106.90872</td>
<td>0.000000</td>
<td>0.190809</td>
</tr>
<tr>
<td>77</td>
<td>S</td>
<td>0.973</td>
<td>0.000</td>
<td>121.67773</td>
<td>4.40677</td>
<td>0.000000</td>
<td>0.190809</td>
</tr>
<tr>
<td>78</td>
<td>S</td>
<td>0.880</td>
<td>0.000</td>
<td>110.00210</td>
<td>4.40677</td>
<td>0.000000</td>
<td>0.190809</td>
</tr>
<tr>
<td>79</td>
<td>S</td>
<td>0.681</td>
<td>0.000</td>
<td>85.17441</td>
<td>4.40677</td>
<td>0.000000</td>
<td>0.190809</td>
</tr>
<tr>
<td>80</td>
<td>S</td>
<td>0.500</td>
<td>0.000</td>
<td>62.50119</td>
<td>4.40677</td>
<td>0.000000</td>
<td>0.190809</td>
</tr>
</tbody>
</table>

**LDT NASA TELESCOPE AT 820NM WAVELENGTH File ldtplas**

**Zoom Position 1**

**EFL = 625.007**

**Optimization Conditions**

**Angle Solve Off**

**EFL, Back Focus, Ybar, Mbar' Track, Glass Length, Thickness**
### WEIGHTING FACTORS

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.60</td>
<td>25.00</td>
<td>0.00</td>
<td>500.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
</tr>
<tr>
<td>625.00</td>
<td>0.00</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.000</td>
<td>209.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAY NO</th>
<th>LABEL</th>
<th>WT</th>
<th>WT</th>
<th>WT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>P</td>
<td>4.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>MD</td>
<td>10.0</td>
<td>100.0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>M</td>
<td>10.0</td>
<td>100.0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>CD</td>
<td>4.0</td>
<td>4.0</td>
<td>150.0</td>
</tr>
<tr>
<td>5</td>
<td>MD</td>
<td>10.0</td>
<td>100.0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>M</td>
<td>10.0</td>
<td>100.0</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>M</td>
<td>10.0</td>
<td>100.0</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>M</td>
<td>10.0</td>
<td>100.0</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>M</td>
<td>10.0</td>
<td>100.0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>M</td>
<td>10.0</td>
<td>100.0</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>M</td>
<td>10.0</td>
<td>100.0</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>M</td>
<td>10.0</td>
<td>100.0</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>M</td>
<td>10.0</td>
<td>100.0</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>M</td>
<td>10.0</td>
<td>100.0</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>M</td>
<td>10.0</td>
<td>100.0</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>MD</td>
<td>10.0</td>
<td>100.0</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>S</td>
<td>10.0</td>
<td>10.0</td>
<td>100.0</td>
</tr>
<tr>
<td>18</td>
<td>S</td>
<td>10.0</td>
<td>10.0</td>
<td>100.0</td>
</tr>
<tr>
<td>19</td>
<td>S</td>
<td>10.0</td>
<td>10.0</td>
<td>100.0</td>
</tr>
<tr>
<td>20</td>
<td>S</td>
<td>10.0</td>
<td>10.0</td>
<td>100.0</td>
</tr>
</tbody>
</table>

### # VARIABLES

<table>
<thead>
<tr>
<th></th>
<th>CURVE</th>
<th>EDGE</th>
<th>CLR RAD LIMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.00000</td>
<td>0.000</td>
<td>428.302</td>
</tr>
<tr>
<td>2</td>
<td>CEA4A6A8AI0</td>
<td>0.00214</td>
<td>2.000</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>0.00354</td>
<td>12.700</td>
</tr>
<tr>
<td>4</td>
<td>CEA4A6A8AI0</td>
<td>0.00432</td>
<td>1.000</td>
</tr>
<tr>
<td>5</td>
<td>C</td>
<td>0.00063</td>
<td>12.700</td>
</tr>
<tr>
<td>6</td>
<td>CEA4A6A8AI0</td>
<td>0.00659</td>
<td>2.000</td>
</tr>
<tr>
<td>7</td>
<td>C</td>
<td>0.00278</td>
<td>13.000</td>
</tr>
<tr>
<td>8</td>
<td>C</td>
<td>0.00449</td>
<td>1.000</td>
</tr>
<tr>
<td>9</td>
<td>C</td>
<td>0.00751</td>
<td>12.700</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>0.00000</td>
<td>0.000</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>0.00000</td>
<td>70.000</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>0.00000</td>
<td>40000.000</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>0.00000</td>
<td>40000.000</td>
</tr>
</tbody>
</table>

### ZOOM POSITION 2

| EFL    | 625.007 |

### OPTIMIZATION CONDITIONS

- ANGLE SOLVE OFF

### WEIGHTING FACTORS

<table>
<thead>
<tr>
<th></th>
<th>1.60</th>
<th>25.00</th>
<th>0.00</th>
<th>500.00</th>
<th>0.00</th>
<th>1.00</th>
<th>0.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>625.00</td>
<td>0.00</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.000</td>
<td>209.00</td>
<td>0.000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAY NO</th>
<th>LABEL</th>
<th>WT</th>
<th>WT</th>
<th>WT</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>PZ</td>
<td>4.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>MD</td>
<td>10.0</td>
<td>100.0</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>M</td>
<td>10.0</td>
<td>100.0</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>CD</td>
<td>4.0</td>
<td>4.0</td>
<td>150.0</td>
</tr>
</tbody>
</table>
MULTI-ACCESS FREE SPACE LASER COMMUNICATIONS TERMINAL - FINAL REPORT

```
25 MD 10.0 100.0
26 M 10.0 100.0
27 M 10.0 100.0
28 M 10.0 100.0
29 M 10.0 100.0
30 M 10.0 100.0
31 M 10.0 100.0
32 M 10.0 100.0
33 M 10.0 100.0
34 M 10.0 100.0
35 M 10.0 100.0
36 MD 10.0 100.0
37 S 10.0 10.0 100.0
38 S 10.0 10.0 100.0
39 S 10.0 10.0 100.0
40 S 10.0 10.0 100.0

# VARIABLES
CURVE EDGE CLR RAD LIMIT
1 0.00000 0.000 428.302
2 0.00214 2.000 200.000
3 0.00354 12.700 200.000
4 0.00432 1.000 200.000
5 0.00663 12.700 428.302
6 0.00659 2.000 428.302
7 0.00278 13.000 200.000
8 0.00449 1.000 200.000
9 0.00751 12.700 428.302
10 0.00000 0.000 200.000
11 0.00000 70.000 18000.000
12 0.00000%-40000.000 12000.000
13 0.00000%-40000.000 428.302

ZOOM POSITION 3
EFL = 625.007
OPTIMIZATION CONDITIONS
ANGLE SOLVE OFF
EFL BACK FOCUS Ybar Mbar' Track Glass Length Th-ness
WEIGHTING FACTORS
1.60 25.00 0.00 500.00 0.00 1.00 0.00 0.00
625.000 0.000 0.0000 0.0000 0.0000 209.000 0.000 0.000

RAY NO LABEL WT WT WT
41 PZ 4.0
42 MD 10.0 100.0
43 M 10.0 100.0
44 CD 4.0 4.0 150.0
45 MD 10.0 100.0
46 M 10.0 100.0
47 M 10.0 100.0
48 M 10.0 100.0
49 M 10.0 100.0
50 M 10.0 100.0
51 M 10.0 100.0
52 M 10.0 100.0
53 M 10.0 100.0
54 M 10.0 100.0
55 M 10.0 100.0

-A21-
```
<table>
<thead>
<tr>
<th>#</th>
<th>VARIABLES</th>
<th>CURVE</th>
<th>EDGE</th>
<th>CLR</th>
<th>RAD LIMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.00000</td>
<td>0.000</td>
<td>428.302</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.00214</td>
<td>2.000</td>
<td>200.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.00354</td>
<td>12.700</td>
<td>200.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.00432</td>
<td>1.000</td>
<td>200.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.00063</td>
<td>12.700</td>
<td>428.302</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.00659</td>
<td>2.000</td>
<td>428.302</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.00278</td>
<td>13.000</td>
<td>200.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.00449</td>
<td>1.000</td>
<td>200.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0.00751</td>
<td>12.700</td>
<td>428.302</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.00000</td>
<td>0.000</td>
<td>200.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>0.00000</td>
<td>70.000</td>
<td>18000.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0.00000</td>
<td>-40000.000</td>
<td>12000.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>0.00000</td>
<td>-40000.000</td>
<td>428.302</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ZOOM POSITION 4
EFL = 625.007

OPTIMIZATION CONDITIONS
ANGLE SOLVE OFF

WEIGHTING FACTORS
1.60 25.00 0.00 500.00 0.00 1.00 0.00 0.00 TARGETS

Mbar'  Track  Glass  Length  Th-ness

RAY NO  LABEL  WT  WT  WT

61  PZ  4.0
62  MD  10.0 100.0
63  M  10.0 100.0

64  CD  4.0  4.0  150.0
65  MD  10.0 100.0
66  M  10.0 100.0
67  M  10.0 100.0
68  M  10.0 100.0
69  M  10.0 100.0
70  M  10.0 100.0
71  M  10.0 100.0
72  M  10.0 100.0
73  M  10.0 100.0
74  M  10.0 100.0
75  M  10.0 100.0
76  MD  10.0 100.0
77  S  10.0 10.0 100.0
78  S  10.0 10.0 100.0
79  S  10.0 10.0 100.0
80  S  10.0 10.0 100.0

# VARIABLES  CURVE  EDGE  CLR  RAD LIMIT
1  0.00000  0.000  428.302
2  0.00214  2.000  200.000
3  0.00354  12.700  200.000
4  0.00432  1.000  200.000

-A22-
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.00063</td>
<td>12.700</td>
<td>428.302</td>
</tr>
<tr>
<td>6</td>
<td>0.00659</td>
<td>2.000</td>
<td>428.302</td>
</tr>
<tr>
<td>7</td>
<td>0.00278</td>
<td>13.000</td>
<td>200.000</td>
</tr>
<tr>
<td>8</td>
<td>0.00449</td>
<td>1.000</td>
<td>200.000</td>
</tr>
<tr>
<td>9</td>
<td>0.00751</td>
<td>12.700</td>
<td>428.302</td>
</tr>
<tr>
<td>10</td>
<td>0.00000</td>
<td>0.000</td>
<td>200.000</td>
</tr>
<tr>
<td>11</td>
<td>0.00000</td>
<td>70.000</td>
<td>18000.000</td>
</tr>
<tr>
<td>12</td>
<td>0.00000%</td>
<td>-40000.000</td>
<td>12000.000</td>
</tr>
<tr>
<td>13</td>
<td>0.00000%</td>
<td>-40000.000</td>
<td>428.302</td>
</tr>
</tbody>
</table>
General Lens Specifications

Total Integrated Scatter < 1 % per surface
Surface Roughness < .005 microns rms
Wavelengths of operation .82 microns and .86 microns
Wedge angle < 6 arc minutes
All aspheric surfaces:
  Slope angle within .0005 radians of nominal value defined by
  surface equation for all points within clear aperture.
  Slope angle to be measured over .5 inch span.

Figure deviation +-.0075 inches max from nominal curve.
Material: Optical Grade Polystyrene

CURVATURE
See Attached Aspheric Description

Center Thickness 12.7 mm ± 0.5 mm

Curvature +0.003544 mm⁻¹ ± 0.000035 mm⁻¹

190 mm radius C. A.

345.6

303.7 mm diameter ± 1 mm
LDT NASA TELESCOPE ELEMENT 001 SURFACE 1
Sag. vs. radial distance from optical axis

\[ Z = \frac{C(R^2)}{(1+\sqrt{1-(1+K)(C^2)(R^2)})} + A_4(R^4) + A_6(R^6) + A_8(R^8) + A_{10}(R^{10}) \]

- **C** = \(2.1350000E-03\) /mm  
  Radius of Curvature
- **K** = \(2.4779850E+00\)  
  Conic constant
- **A_4** = \(4.4795910E-09\)  
  Fourth order aspheric coefficient
- **A_6** = \(-8.7438320E-14\)  
  Sixth order aspheric coefficient
- **A_8** = \(1.936090E-18\)  
  Eighth order aspheric coefficient
- **A_{10}** = \(-3.5570550E-23\)  
  Tenth order aspheric coefficient

**R** = radial distance from optical axis (mm)
**Z** = Sag. (mm)

<table>
<thead>
<tr>
<th>R (mm)</th>
<th>Z (mm)</th>
<th>R (inches)</th>
<th>Z (inches)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>1.270000</td>
<td>0.0017218</td>
<td>0.000678</td>
<td>0.000678</td>
</tr>
<tr>
<td>2.540000</td>
<td>0.0068874</td>
<td>0.002712</td>
<td>0.002712</td>
</tr>
<tr>
<td>3.810000</td>
<td>0.0154978</td>
<td>0.006101</td>
<td>0.006101</td>
</tr>
<tr>
<td>5.080000</td>
<td>0.0275541</td>
<td>0.010848</td>
<td>0.010848</td>
</tr>
<tr>
<td>6.350000</td>
<td>0.0430584</td>
<td>0.009779</td>
<td>0.009779</td>
</tr>
<tr>
<td>7.620000</td>
<td>0.0620131</td>
<td>0.008215</td>
<td>0.008215</td>
</tr>
<tr>
<td>8.890000</td>
<td>0.0844211</td>
<td>0.007773</td>
<td>0.007773</td>
</tr>
<tr>
<td>10.160000</td>
<td>0.1102861</td>
<td>0.007337</td>
<td>0.007337</td>
</tr>
<tr>
<td>11.430000</td>
<td>0.1396120</td>
<td>0.006875</td>
<td>0.006875</td>
</tr>
<tr>
<td>12.700000</td>
<td>0.1724034</td>
<td>0.006412</td>
<td>0.006412</td>
</tr>
<tr>
<td>13.970000</td>
<td>0.2086656</td>
<td>0.005949</td>
<td>0.005949</td>
</tr>
<tr>
<td>15.240000</td>
<td>0.2484042</td>
<td>0.005486</td>
<td>0.005486</td>
</tr>
<tr>
<td>16.510000</td>
<td>0.2916254</td>
<td>0.005024</td>
<td>0.005024</td>
</tr>
<tr>
<td>17.780000</td>
<td>0.3383359</td>
<td>0.004563</td>
<td>0.004563</td>
</tr>
<tr>
<td>19.050000</td>
<td>0.3885430</td>
<td>0.004101</td>
<td>0.004101</td>
</tr>
<tr>
<td>20.320000</td>
<td>0.4425246</td>
<td>0.003640</td>
<td>0.003640</td>
</tr>
<tr>
<td>21.590000</td>
<td>0.4994790</td>
<td>0.003178</td>
<td>0.003178</td>
</tr>
<tr>
<td>22.860000</td>
<td>0.5602249</td>
<td>0.002716</td>
<td>0.002716</td>
</tr>
<tr>
<td>24.130000</td>
<td>0.6245019</td>
<td>0.002254</td>
<td>0.002254</td>
</tr>
<tr>
<td>25.400000</td>
<td>0.6923198</td>
<td>0.001892</td>
<td>0.001892</td>
</tr>
<tr>
<td>26.670000</td>
<td>0.7636890</td>
<td>0.001530</td>
<td>0.001530</td>
</tr>
<tr>
<td>27.940000</td>
<td>0.8386204</td>
<td>0.001168</td>
<td>0.001168</td>
</tr>
<tr>
<td>29.210000</td>
<td>0.9171256</td>
<td>0.000806</td>
<td>0.000806</td>
</tr>
<tr>
<td>30.480000</td>
<td>0.9992164</td>
<td>0.000444</td>
<td>0.000444</td>
</tr>
<tr>
<td>31.750000</td>
<td>1.0849053</td>
<td>0.000082</td>
<td>0.000082</td>
</tr>
<tr>
<td>33.020000</td>
<td>1.1742053</td>
<td>0.001061</td>
<td>0.001061</td>
</tr>
<tr>
<td>34.290000</td>
<td>1.2671299</td>
<td>0.001695</td>
<td>0.001695</td>
</tr>
<tr>
<td>35.560000</td>
<td>1.3636930</td>
<td>0.002228</td>
<td>0.002228</td>
</tr>
<tr>
<td>36.830000</td>
<td>1.4639091</td>
<td>0.002761</td>
<td>0.002761</td>
</tr>
<tr>
<td>R (mm)</td>
<td>Z (mm)</td>
<td>R (inches)</td>
<td>Z (inches)</td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>38.100</td>
<td>1.56773</td>
<td>1.500000</td>
<td>0.0617241</td>
</tr>
<tr>
<td>39.370</td>
<td>1.67536</td>
<td>1.550000</td>
<td>0.0659591</td>
</tr>
<tr>
<td>40.640</td>
<td>1.78663</td>
<td>1.600000</td>
<td>0.0703997</td>
</tr>
<tr>
<td>41.910</td>
<td>1.90161</td>
<td>1.650000</td>
<td>0.0748665</td>
</tr>
<tr>
<td>43.180</td>
<td>2.02033</td>
<td>1.700000</td>
<td>0.0795404</td>
</tr>
<tr>
<td>44.450</td>
<td>2.14279</td>
<td>1.750000</td>
<td>0.0843618</td>
</tr>
<tr>
<td>45.720</td>
<td>2.26902</td>
<td>1.800000</td>
<td>0.0893317</td>
</tr>
<tr>
<td>46.990</td>
<td>2.39904</td>
<td>1.850000</td>
<td>0.0944506</td>
</tr>
<tr>
<td>48.260</td>
<td>2.53286</td>
<td>1.900000</td>
<td>0.0997193</td>
</tr>
<tr>
<td>49.530</td>
<td>2.67052</td>
<td>1.950000</td>
<td>1.051385</td>
</tr>
<tr>
<td>50.800</td>
<td>2.81201</td>
<td>2.000000</td>
<td>1.107092</td>
</tr>
<tr>
<td>52.070</td>
<td>2.95737</td>
<td>2.050000</td>
<td>1.164320</td>
</tr>
<tr>
<td>53.340</td>
<td>3.10661</td>
<td>2.100000</td>
<td>1.223077</td>
</tr>
<tr>
<td>54.610</td>
<td>3.25976</td>
<td>2.150000</td>
<td>1.283372</td>
</tr>
<tr>
<td>55.880</td>
<td>3.41684</td>
<td>2.200000</td>
<td>1.345214</td>
</tr>
<tr>
<td>57.150</td>
<td>3.57784</td>
<td>2.250000</td>
<td>1.408612</td>
</tr>
<tr>
<td>58.420</td>
<td>3.74287</td>
<td>2.300000</td>
<td>1.473573</td>
</tr>
<tr>
<td>59.690</td>
<td>3.91187</td>
<td>2.350000</td>
<td>1.540108</td>
</tr>
<tr>
<td>60.960</td>
<td>4.08489</td>
<td>2.400000</td>
<td>1.608225</td>
</tr>
<tr>
<td>62.230</td>
<td>4.26195</td>
<td>2.450000</td>
<td>1.677934</td>
</tr>
<tr>
<td>63.500</td>
<td>4.44308</td>
<td>2.500000</td>
<td>1.749245</td>
</tr>
<tr>
<td>64.770</td>
<td>4.62830</td>
<td>2.550000</td>
<td>1.822168</td>
</tr>
<tr>
<td>66.040</td>
<td>4.81764</td>
<td>2.600000</td>
<td>1.896712</td>
</tr>
<tr>
<td>67.310</td>
<td>5.01133</td>
<td>2.650000</td>
<td>1.972887</td>
</tr>
<tr>
<td>68.580</td>
<td>5.20879</td>
<td>2.700000</td>
<td>2.050705</td>
</tr>
<tr>
<td>69.850</td>
<td>5.41065</td>
<td>2.750000</td>
<td>2.130175</td>
</tr>
<tr>
<td>71.120</td>
<td>5.61672</td>
<td>2.800000</td>
<td>2.211309</td>
</tr>
<tr>
<td>72.390</td>
<td>5.82705</td>
<td>2.850000</td>
<td>2.294118</td>
</tr>
<tr>
<td>73.660</td>
<td>6.04167</td>
<td>2.900000</td>
<td>2.378612</td>
</tr>
<tr>
<td>74.930</td>
<td>6.26059</td>
<td>2.950000</td>
<td>2.464803</td>
</tr>
<tr>
<td>76.200</td>
<td>6.48386</td>
<td>3.000000</td>
<td>2.552702</td>
</tr>
<tr>
<td>77.470</td>
<td>6.71149</td>
<td>3.050000</td>
<td>2.642322</td>
</tr>
<tr>
<td>78.740</td>
<td>6.94353</td>
<td>3.100000</td>
<td>2.733674</td>
</tr>
<tr>
<td>80.010</td>
<td>7.17997</td>
<td>3.150000</td>
<td>2.826771</td>
</tr>
<tr>
<td>81.280</td>
<td>7.42092</td>
<td>3.200000</td>
<td>2.921624</td>
</tr>
<tr>
<td>82.550</td>
<td>7.66634</td>
<td>3.250000</td>
<td>3.018246</td>
</tr>
<tr>
<td>83.820</td>
<td>7.91629</td>
<td>3.300000</td>
<td>3.116650</td>
</tr>
<tr>
<td>85.090</td>
<td>8.17078</td>
<td>3.350000</td>
<td>3.216850</td>
</tr>
<tr>
<td>86.360</td>
<td>8.42989</td>
<td>3.400000</td>
<td>3.318857</td>
</tr>
<tr>
<td>87.630</td>
<td>8.69362</td>
<td>3.450000</td>
<td>3.422686</td>
</tr>
<tr>
<td>88.900</td>
<td>8.96201</td>
<td>3.500000</td>
<td>3.528350</td>
</tr>
<tr>
<td>90.170</td>
<td>9.23509</td>
<td>3.550000</td>
<td>3.635864</td>
</tr>
<tr>
<td>91.440</td>
<td>9.51290</td>
<td>3.600000</td>
<td>3.745240</td>
</tr>
<tr>
<td>92.710</td>
<td>9.79549</td>
<td>3.650000</td>
<td>3.856494</td>
</tr>
<tr>
<td>93.980</td>
<td>10.08288</td>
<td>3.700000</td>
<td>3.969339</td>
</tr>
<tr>
<td>95.250</td>
<td>10.37512</td>
<td>3.750000</td>
<td>4.084691</td>
</tr>
<tr>
<td>96.520</td>
<td>10.67229</td>
<td>3.800000</td>
<td>4.201665</td>
</tr>
<tr>
<td>R (mm)</td>
<td>Z (mm)</td>
<td>R (inches)</td>
<td>Z (inches)</td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>97.7900000</td>
<td>10.9742615</td>
<td>3.8500000</td>
<td>0.4320575</td>
</tr>
<tr>
<td>99.0600000</td>
<td>11.2812528</td>
<td>3.9000000</td>
<td>0.4441438</td>
</tr>
<tr>
<td>100.3300000</td>
<td>11.5932425</td>
<td>3.9500000</td>
<td>0.4564269</td>
</tr>
<tr>
<td>101.6000000</td>
<td>11.9102712</td>
<td>4.0000000</td>
<td>0.4689083</td>
</tr>
<tr>
<td>102.8700000</td>
<td>12.2323800</td>
<td>4.0500000</td>
<td>0.4815898</td>
</tr>
<tr>
<td>104.1400000</td>
<td>12.5561060</td>
<td>4.1000000</td>
<td>0.4944729</td>
</tr>
<tr>
<td>105.4100000</td>
<td>12.8920055</td>
<td>4.1500000</td>
<td>0.5075932</td>
</tr>
<tr>
<td>106.6800000</td>
<td>13.2296077</td>
<td>4.2000000</td>
<td>0.5208507</td>
</tr>
<tr>
<td>107.9500000</td>
<td>13.5724609</td>
<td>4.2500000</td>
<td>0.5343489</td>
</tr>
<tr>
<td>109.2200000</td>
<td>13.9206095</td>
<td>4.3000000</td>
<td>0.5480555</td>
</tr>
<tr>
<td>110.4900000</td>
<td>14.2749866</td>
<td>4.3500000</td>
<td>0.5619724</td>
</tr>
<tr>
<td>111.7600000</td>
<td>14.6329739</td>
<td>4.4000000</td>
<td>0.5761013</td>
</tr>
<tr>
<td>113.0300000</td>
<td>14.9972818</td>
<td>4.4500000</td>
<td>0.5904442</td>
</tr>
<tr>
<td>114.3000000</td>
<td>15.3670695</td>
<td>4.5000000</td>
<td>0.6050027</td>
</tr>
<tr>
<td>115.5700000</td>
<td>15.7423847</td>
<td>4.5500000</td>
<td>0.6197789</td>
</tr>
<tr>
<td>116.8400000</td>
<td>16.1232761</td>
<td>4.6000000</td>
<td>0.6347746</td>
</tr>
<tr>
<td>118.1100000</td>
<td>16.5097928</td>
<td>4.6500000</td>
<td>0.6499918</td>
</tr>
<tr>
<td>119.3800000</td>
<td>16.9019848</td>
<td>4.7000000</td>
<td>0.6654325</td>
</tr>
<tr>
<td>120.6500000</td>
<td>17.2999027</td>
<td>4.7500000</td>
<td>0.6810985</td>
</tr>
<tr>
<td>121.9200000</td>
<td>17.7035978</td>
<td>4.8000000</td>
<td>0.6969920</td>
</tr>
<tr>
<td>123.1900000</td>
<td>18.1131224</td>
<td>4.8500000</td>
<td>0.7131151</td>
</tr>
<tr>
<td>124.4600000</td>
<td>18.5285290</td>
<td>4.9000000</td>
<td>0.7294696</td>
</tr>
<tr>
<td>125.7300000</td>
<td>18.9498714</td>
<td>4.9500000</td>
<td>0.7460579</td>
</tr>
<tr>
<td>127.0000000</td>
<td>19.3772036</td>
<td>5.0000000</td>
<td>0.7628820</td>
</tr>
<tr>
<td>128.2700000</td>
<td>19.8105806</td>
<td>5.0500000</td>
<td>0.7799441</td>
</tr>
<tr>
<td>129.5400000</td>
<td>20.2500581</td>
<td>5.1000000</td>
<td>0.7972464</td>
</tr>
<tr>
<td>130.8100000</td>
<td>20.6956923</td>
<td>5.1500000</td>
<td>0.8147910</td>
</tr>
<tr>
<td>132.0800000</td>
<td>21.1475403</td>
<td>5.2000000</td>
<td>0.8325803</td>
</tr>
<tr>
<td>133.3500000</td>
<td>21.6056597</td>
<td>5.2500000</td>
<td>0.8506165</td>
</tr>
<tr>
<td>134.6200000</td>
<td>22.0701091</td>
<td>5.3000000</td>
<td>0.8689019</td>
</tr>
<tr>
<td>135.8900000</td>
<td>22.5404975</td>
<td>5.3500000</td>
<td>0.8874389</td>
</tr>
<tr>
<td>137.1600000</td>
<td>23.0182344</td>
<td>5.4000000</td>
<td>0.9062297</td>
</tr>
<tr>
<td>138.4300000</td>
<td>23.5020304</td>
<td>5.4500000</td>
<td>0.9252768</td>
</tr>
<tr>
<td>139.7000000</td>
<td>23.9923965</td>
<td>5.5000000</td>
<td>0.9445825</td>
</tr>
<tr>
<td>140.9700000</td>
<td>24.4893941</td>
<td>5.5500000</td>
<td>0.9641494</td>
</tr>
<tr>
<td>142.2400000</td>
<td>24.9930855</td>
<td>5.6000000</td>
<td>0.9839797</td>
</tr>
<tr>
<td>143.5100000</td>
<td>25.5035334</td>
<td>5.6500000</td>
<td>1.0040761</td>
</tr>
<tr>
<td>144.7800000</td>
<td>26.0208013</td>
<td>5.7000000</td>
<td>1.024410</td>
</tr>
<tr>
<td>146.0500000</td>
<td>26.5449529</td>
<td>5.7500000</td>
<td>1.0450769</td>
</tr>
<tr>
<td>147.3200000</td>
<td>27.0760527</td>
<td>5.8000000</td>
<td>1.0659863</td>
</tr>
<tr>
<td>148.5900000</td>
<td>27.6141656</td>
<td>5.8500000</td>
<td>1.0871719</td>
</tr>
<tr>
<td>149.8600000</td>
<td>28.1593568</td>
<td>5.9000000</td>
<td>1.1086361</td>
</tr>
<tr>
<td>151.1300000</td>
<td>28.7119922</td>
<td>5.9500000</td>
<td>1.1303816</td>
</tr>
<tr>
<td>152.4000000</td>
<td>29.2712379</td>
<td>6.0000000</td>
<td>1.1524109</td>
</tr>
<tr>
<td>153.6700000</td>
<td>29.8380606</td>
<td>6.0500000</td>
<td>1.1747268</td>
</tr>
<tr>
<td>154.9400000</td>
<td>30.4122272</td>
<td>6.1000000</td>
<td>1.1973318</td>
</tr>
<tr>
<td>156.2100000</td>
<td>30.9938049</td>
<td>6.1500000</td>
<td>1.2202285</td>
</tr>
<tr>
<td>R (mm)</td>
<td>Z (mm)</td>
<td>R (inches)</td>
<td>Z (inches)</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>157.480</td>
<td>31.582</td>
<td>6.200</td>
<td>1.243</td>
</tr>
<tr>
<td>158.750</td>
<td>32.179</td>
<td>6.250</td>
<td>1.267</td>
</tr>
<tr>
<td>160.020</td>
<td>32.783</td>
<td>6.300</td>
<td>1.291</td>
</tr>
<tr>
<td>161.290</td>
<td>33.395</td>
<td>6.350</td>
<td>1.314</td>
</tr>
<tr>
<td>162.560</td>
<td>34.015</td>
<td>6.400</td>
<td>1.339</td>
</tr>
<tr>
<td>163.830</td>
<td>34.642</td>
<td>6.450</td>
<td>1.364</td>
</tr>
<tr>
<td>165.100</td>
<td>35.278</td>
<td>6.500</td>
<td>1.389</td>
</tr>
<tr>
<td>166.370</td>
<td>35.921</td>
<td>6.550</td>
<td>1.414</td>
</tr>
<tr>
<td>167.640</td>
<td>36.572</td>
<td>6.600</td>
<td>1.439</td>
</tr>
<tr>
<td>168.910</td>
<td>37.232</td>
<td>6.650</td>
<td>1.466</td>
</tr>
<tr>
<td>170.180</td>
<td>37.899</td>
<td>6.700</td>
<td>1.492</td>
</tr>
<tr>
<td>171.450</td>
<td>38.576</td>
<td>6.750</td>
<td>1.518</td>
</tr>
<tr>
<td>172.720</td>
<td>39.258</td>
<td>6.800</td>
<td>1.545</td>
</tr>
<tr>
<td>173.990</td>
<td>39.953</td>
<td>6.850</td>
<td>1.573</td>
</tr>
<tr>
<td>175.260</td>
<td>40.653</td>
<td>6.900</td>
<td>1.600</td>
</tr>
<tr>
<td>176.530</td>
<td>41.363</td>
<td>6.950</td>
<td>1.628</td>
</tr>
<tr>
<td>177.800</td>
<td>42.081</td>
<td>7.000</td>
<td>1.657</td>
</tr>
<tr>
<td>179.070</td>
<td>42.808</td>
<td>7.050</td>
<td>1.685</td>
</tr>
<tr>
<td>180.340</td>
<td>43.543</td>
<td>7.100</td>
<td>1.714</td>
</tr>
<tr>
<td>181.610</td>
<td>44.287</td>
<td>7.150</td>
<td>1.743</td>
</tr>
<tr>
<td>182.880</td>
<td>45.040</td>
<td>7.200</td>
<td>1.773</td>
</tr>
<tr>
<td>184.150</td>
<td>45.802</td>
<td>7.250</td>
<td>1.803</td>
</tr>
<tr>
<td>185.420</td>
<td>46.573</td>
<td>7.300</td>
<td>1.833</td>
</tr>
<tr>
<td>186.690</td>
<td>47.353</td>
<td>7.350</td>
<td>1.864</td>
</tr>
<tr>
<td>187.960</td>
<td>48.143</td>
<td>7.400</td>
<td>1.895</td>
</tr>
<tr>
<td>189.230</td>
<td>48.941</td>
<td>7.450</td>
<td>1.926</td>
</tr>
<tr>
<td>190.500</td>
<td>49.749</td>
<td>7.500</td>
<td>1.958</td>
</tr>
<tr>
<td>191.770</td>
<td>50.567</td>
<td>7.550</td>
<td>1.990</td>
</tr>
<tr>
<td>193.040</td>
<td>51.393</td>
<td>7.600</td>
<td>2.023</td>
</tr>
<tr>
<td>194.310</td>
<td>52.230</td>
<td>7.650</td>
<td>2.056</td>
</tr>
<tr>
<td>195.580</td>
<td>53.076</td>
<td>7.700</td>
<td>2.089</td>
</tr>
<tr>
<td>196.850</td>
<td>53.932</td>
<td>7.750</td>
<td>2.123</td>
</tr>
</tbody>
</table>
MULTI-ACCESS FREE SPACE LASER COMMUNICATIONS TERMINAL FINAL REPORT

LDT NASA TELESCOPE ELEMENT 002

Material: Optical Grade
Methyl Methacrylate
(Acrylic)

Curvature See Attached
Aspheric Description

Center Thickness
81.5 mm
+-0.5 mm

Curvature
+0.000633 mm^-1
+-0.000006 mm^-1

Perspective Displays, Inc.

File ldtplas

Scale = 0.25
Z = \frac{(C*(R^2))/(1+\sqrt{1-(1+K)*(C^2)*(R^2)})}{+ A4*(R^4) + A6*(R^6) + A8*(R^8) + A10*(R^{10})}

C = 4.3240000E-03 /\text{mm} \quad \text{Radius of Curvature}
K = -1.2475900E+00 \quad \text{Conic constant}
A4 = 1.9018980E-11 \quad \text{Fourth order aspheric coefficient}
A6 = 7.8000490E-14 \quad \text{Sixth order aspheric coefficient}
A8 = -1.6354920E-18 \quad \text{Eighth order aspheric coefficient}
A10 = 2.5101750E-23 \quad \text{Tenth order aspheric coefficient}

R = \text{radial distance from optical axis (mm)}
Z = \text{Sag. (mm)}

<table>
<thead>
<tr>
<th>R (mm)</th>
<th>Z (mm)</th>
<th>R (inches)</th>
<th>Z (inches)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>1.270000</td>
<td>0.0034871</td>
<td>0.0050000</td>
<td>0.0001373</td>
</tr>
<tr>
<td>2.540000</td>
<td>0.0139483</td>
<td>0.1000000</td>
<td>0.0005491</td>
</tr>
<tr>
<td>3.810000</td>
<td>0.0313833</td>
<td>0.1500000</td>
<td>0.0012356</td>
</tr>
<tr>
<td>5.080000</td>
<td>0.0557918</td>
<td>0.2000000</td>
<td>0.0021965</td>
</tr>
<tr>
<td>6.350000</td>
<td>0.0871732</td>
<td>0.2500000</td>
<td>0.0034320</td>
</tr>
<tr>
<td>7.620000</td>
<td>0.1255269</td>
<td>0.3000000</td>
<td>0.0049420</td>
</tr>
<tr>
<td>8.890000</td>
<td>0.1708519</td>
<td>0.3500000</td>
<td>0.0067265</td>
</tr>
<tr>
<td>10.160000</td>
<td>0.2231474</td>
<td>0.4000000</td>
<td>0.0087853</td>
</tr>
<tr>
<td>11.430000</td>
<td>0.2824121</td>
<td>0.4500000</td>
<td>0.0111186</td>
</tr>
<tr>
<td>12.700000</td>
<td>0.3486447</td>
<td>0.5000000</td>
<td>0.0137262</td>
</tr>
<tr>
<td>13.970000</td>
<td>0.4218439</td>
<td>0.5500000</td>
<td>0.0166080</td>
</tr>
<tr>
<td>15.240000</td>
<td>0.5020080</td>
<td>0.6000000</td>
<td>0.0197641</td>
</tr>
<tr>
<td>16.510000</td>
<td>0.5891354</td>
<td>0.6500000</td>
<td>0.0231943</td>
</tr>
<tr>
<td>17.780000</td>
<td>0.6832241</td>
<td>0.7000000</td>
<td>0.0268986</td>
</tr>
<tr>
<td>19.050000</td>
<td>0.7842722</td>
<td>0.7500000</td>
<td>0.0308769</td>
</tr>
<tr>
<td>20.320000</td>
<td>0.8922775</td>
<td>0.8000000</td>
<td>0.0351290</td>
</tr>
<tr>
<td>21.590000</td>
<td>1.0072379</td>
<td>0.8500000</td>
<td>0.0396550</td>
</tr>
<tr>
<td>22.860000</td>
<td>1.1291508</td>
<td>0.9000000</td>
<td>0.0444548</td>
</tr>
<tr>
<td>24.130000</td>
<td>1.2580140</td>
<td>0.9500000</td>
<td>0.0495281</td>
</tr>
<tr>
<td>25.400000</td>
<td>1.3938246</td>
<td>1.0000000</td>
<td>0.0548750</td>
</tr>
<tr>
<td>26.670000</td>
<td>1.5365801</td>
<td>1.0500000</td>
<td>0.0604953</td>
</tr>
<tr>
<td>27.940000</td>
<td>1.6862775</td>
<td>1.1000000</td>
<td>0.0663889</td>
</tr>
<tr>
<td>29.210000</td>
<td>1.8429140</td>
<td>1.1500000</td>
<td>0.0725557</td>
</tr>
<tr>
<td>30.480000</td>
<td>2.0064866</td>
<td>1.2000000</td>
<td>0.0789955</td>
</tr>
<tr>
<td>31.750000</td>
<td>2.1769920</td>
<td>1.2500000</td>
<td>0.0857083</td>
</tr>
<tr>
<td>33.020000</td>
<td>2.3544272</td>
<td>1.3000000</td>
<td>0.0926940</td>
</tr>
<tr>
<td>34.290000</td>
<td>2.5387887</td>
<td>1.3500000</td>
<td>0.0999523</td>
</tr>
<tr>
<td>35.560000</td>
<td>2.7300733</td>
<td>1.4000000</td>
<td>0.1074832</td>
</tr>
<tr>
<td>36.830000</td>
<td>2.9282775</td>
<td>1.4500000</td>
<td>0.1152865</td>
</tr>
</tbody>
</table>
### LDT NASA Telescope Element 002 Surface 1

<table>
<thead>
<tr>
<th>R (mm)</th>
<th>Z (mm)</th>
<th>R (inches)</th>
<th>Z (inches)</th>
</tr>
</thead>
<tbody>
<tr>
<td>38.100000</td>
<td>3.133977</td>
<td>1.5000000</td>
<td>0.1233621</td>
</tr>
<tr>
<td>39.370000</td>
<td>3.3454305</td>
<td>1.5500000</td>
<td>0.1317099</td>
</tr>
<tr>
<td>40.640000</td>
<td>3.5643722</td>
<td>1.6000000</td>
<td>0.1403296</td>
</tr>
<tr>
<td>41.910000</td>
<td>3.7902190</td>
<td>1.6500000</td>
<td>0.1492212</td>
</tr>
<tr>
<td>43.180000</td>
<td>4.0229674</td>
<td>1.7000000</td>
<td>0.1583845</td>
</tr>
<tr>
<td>44.450000</td>
<td>4.2626135</td>
<td>1.7500000</td>
<td>0.1678194</td>
</tr>
<tr>
<td>45.720000</td>
<td>4.5091536</td>
<td>1.8000000</td>
<td>0.1775257</td>
</tr>
<tr>
<td>46.990000</td>
<td>4.7625838</td>
<td>1.8500000</td>
<td>0.1875033</td>
</tr>
<tr>
<td>48.260000</td>
<td>5.0229004</td>
<td>1.9000000</td>
<td>0.1977520</td>
</tr>
<tr>
<td>49.530000</td>
<td>5.2900993</td>
<td>1.9500000</td>
<td>0.2082716</td>
</tr>
<tr>
<td>50.800000</td>
<td>5.5641769</td>
<td>2.0000000</td>
<td>0.2190621</td>
</tr>
<tr>
<td>52.070000</td>
<td>5.8451293</td>
<td>2.0500000</td>
<td>0.2301232</td>
</tr>
<tr>
<td>53.340000</td>
<td>6.1329526</td>
<td>2.1000000</td>
<td>0.2414548</td>
</tr>
<tr>
<td>54.610000</td>
<td>6.4276429</td>
<td>2.1500000</td>
<td>0.2530568</td>
</tr>
<tr>
<td>55.880000</td>
<td>6.7291965</td>
<td>2.2000000</td>
<td>0.2649290</td>
</tr>
<tr>
<td>57.150000</td>
<td>7.0376095</td>
<td>2.2500000</td>
<td>0.2770712</td>
</tr>
<tr>
<td>58.420000</td>
<td>7.3528782</td>
<td>2.3000000</td>
<td>0.2894834</td>
</tr>
<tr>
<td>59.690000</td>
<td>7.6749988</td>
<td>2.3500000</td>
<td>0.3021653</td>
</tr>
<tr>
<td>60.960000</td>
<td>8.0039678</td>
<td>2.4000000</td>
<td>0.3151168</td>
</tr>
<tr>
<td>62.230000</td>
<td>8.3397813</td>
<td>2.4500000</td>
<td>0.3283378</td>
</tr>
<tr>
<td>63.500000</td>
<td>8.6824359</td>
<td>2.5000000</td>
<td>0.3418282</td>
</tr>
<tr>
<td>64.770000</td>
<td>9.0319279</td>
<td>2.5500000</td>
<td>0.3555877</td>
</tr>
<tr>
<td>66.040000</td>
<td>9.3882540</td>
<td>2.6000000</td>
<td>0.3696163</td>
</tr>
<tr>
<td>67.310000</td>
<td>9.7514108</td>
<td>2.6500000</td>
<td>0.3839138</td>
</tr>
<tr>
<td>68.580000</td>
<td>10.1213949</td>
<td>2.7000000</td>
<td>0.3984801</td>
</tr>
<tr>
<td>69.850000</td>
<td>10.4982031</td>
<td>2.7500000</td>
<td>0.4133151</td>
</tr>
<tr>
<td>71.120000</td>
<td>10.8818322</td>
<td>2.8000000</td>
<td>0.4284186</td>
</tr>
<tr>
<td>72.390000</td>
<td>11.2722793</td>
<td>2.8500000</td>
<td>0.4437905</td>
</tr>
<tr>
<td>73.660000</td>
<td>11.6695413</td>
<td>2.9000000</td>
<td>0.4594308</td>
</tr>
<tr>
<td>74.930000</td>
<td>12.0736154</td>
<td>2.9500000</td>
<td>0.4753392</td>
</tr>
<tr>
<td>76.200000</td>
<td>12.4844989</td>
<td>3.0000000</td>
<td>0.4915157</td>
</tr>
<tr>
<td>77.470000</td>
<td>12.9021892</td>
<td>3.0500000</td>
<td>0.5079602</td>
</tr>
<tr>
<td>78.740000</td>
<td>13.3268383</td>
<td>3.1000000</td>
<td>0.5246726</td>
</tr>
<tr>
<td>80.010000</td>
<td>13.7579803</td>
<td>3.1500000</td>
<td>0.5416528</td>
</tr>
<tr>
<td>81.280000</td>
<td>14.1960764</td>
<td>3.2000000</td>
<td>0.5589006</td>
</tr>
<tr>
<td>82.550000</td>
<td>14.6409701</td>
<td>3.2500000</td>
<td>0.5764161</td>
</tr>
<tr>
<td>83.820000</td>
<td>15.0926594</td>
<td>3.3000000</td>
<td>0.5941992</td>
</tr>
<tr>
<td>85.090000</td>
<td>15.5511426</td>
<td>3.3500000</td>
<td>0.6122497</td>
</tr>
<tr>
<td>86.360000</td>
<td>16.0164179</td>
<td>3.4000000</td>
<td>0.6305676</td>
</tr>
<tr>
<td>87.630000</td>
<td>16.4884839</td>
<td>3.4500000</td>
<td>0.6491529</td>
</tr>
<tr>
<td>88.900000</td>
<td>16.9673392</td>
<td>3.5000000</td>
<td>0.6680055</td>
</tr>
<tr>
<td>90.170000</td>
<td>17.4529827</td>
<td>3.5500000</td>
<td>0.6871253</td>
</tr>
<tr>
<td>91.440000</td>
<td>17.9454135</td>
<td>3.6000000</td>
<td>0.7065123</td>
</tr>
<tr>
<td>92.710000</td>
<td>18.4446306</td>
<td>3.6500000</td>
<td>0.7261666</td>
</tr>
<tr>
<td>93.980000</td>
<td>18.9506336</td>
<td>3.7000000</td>
<td>0.7460879</td>
</tr>
<tr>
<td>95.250000</td>
<td>19.4634220</td>
<td>3.7500000</td>
<td>0.7662765</td>
</tr>
<tr>
<td>96.520000</td>
<td>19.9829955</td>
<td>3.8000000</td>
<td>0.7867321</td>
</tr>
</tbody>
</table>
## LDT NASA TELESCOPE ELEMENT 002 SURFACE 1

<table>
<thead>
<tr>
<th>( \mathbf{R}(\text{mm}) )</th>
<th>( \mathbf{Z}(\text{mm}) )</th>
<th>( \mathbf{R}(\text{inches}) )</th>
<th>( \mathbf{Z}(\text{inches}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>97.7900000</td>
<td>20.5093543</td>
<td>3.8500000</td>
<td>0.8074549</td>
</tr>
<tr>
<td>99.0600000</td>
<td>21.0424985</td>
<td>3.9000000</td>
<td>0.8284448</td>
</tr>
<tr>
<td>100.3300000</td>
<td>21.5824286</td>
<td>3.9500000</td>
<td>0.8497019</td>
</tr>
<tr>
<td>101.6000000</td>
<td>22.1291451</td>
<td>4.0000000</td>
<td>0.8712262</td>
</tr>
<tr>
<td>102.8700000</td>
<td>22.6862492</td>
<td>4.0500000</td>
<td>0.8930177</td>
</tr>
<tr>
<td>104.1400000</td>
<td>23.2429418</td>
<td>4.1000000</td>
<td>0.9150764</td>
</tr>
<tr>
<td>105.4100000</td>
<td>23.8100244</td>
<td>4.1500000</td>
<td>0.9374025</td>
</tr>
<tr>
<td>106.6800000</td>
<td>24.3838978</td>
<td>4.2000000</td>
<td>0.9599960</td>
</tr>
<tr>
<td>107.9500000</td>
<td>24.9645667</td>
<td>4.2500000</td>
<td>0.9828570</td>
</tr>
<tr>
<td>109.2200000</td>
<td>25.5520306</td>
<td>4.3000000</td>
<td>1.0059855</td>
</tr>
<tr>
<td>110.4900000</td>
<td>26.1462930</td>
<td>4.3500000</td>
<td>1.0293816</td>
</tr>
<tr>
<td>111.7600000</td>
<td>26.7473567</td>
<td>4.4000000</td>
<td>1.0530455</td>
</tr>
<tr>
<td>113.0300000</td>
<td>27.3552250</td>
<td>4.4500000</td>
<td>1.0769774</td>
</tr>
<tr>
<td>114.3000000</td>
<td>27.9699013</td>
<td>4.5000000</td>
<td>1.1011772</td>
</tr>
<tr>
<td>115.5700000</td>
<td>28.5913895</td>
<td>4.5500000</td>
<td>1.1256453</td>
</tr>
<tr>
<td>116.8400000</td>
<td>29.2196940</td>
<td>4.6000000</td>
<td>1.1503817</td>
</tr>
<tr>
<td>118.1100000</td>
<td>29.8548193</td>
<td>4.6500000</td>
<td>1.1753866</td>
</tr>
<tr>
<td>119.3800000</td>
<td>30.4967706</td>
<td>4.7000000</td>
<td>1.2006603</td>
</tr>
<tr>
<td>120.6500000</td>
<td>31.1455533</td>
<td>4.7500000</td>
<td>1.2262029</td>
</tr>
<tr>
<td>121.9200000</td>
<td>31.8011733</td>
<td>4.8000000</td>
<td>1.2520147</td>
</tr>
<tr>
<td>123.1900000</td>
<td>32.4636372</td>
<td>4.8500000</td>
<td>1.2780960</td>
</tr>
<tr>
<td>124.4600000</td>
<td>33.1329158</td>
<td>4.9000000</td>
<td>1.3044469</td>
</tr>
<tr>
<td>125.7300000</td>
<td>33.8091245</td>
<td>4.9500000</td>
<td>1.3310679</td>
</tr>
<tr>
<td>127.0000000</td>
<td>34.4921634</td>
<td>5.0000000</td>
<td>1.3579592</td>
</tr>
<tr>
<td>128.2700000</td>
<td>35.1820707</td>
<td>5.0500000</td>
<td>1.3851211</td>
</tr>
<tr>
<td>129.5400000</td>
<td>35.8788747</td>
<td>5.1000000</td>
<td>1.4125541</td>
</tr>
<tr>
<td>130.8100000</td>
<td>36.5825660</td>
<td>5.1500000</td>
<td>1.4402585</td>
</tr>
<tr>
<td>132.0800000</td>
<td>37.2931617</td>
<td>5.2000000</td>
<td>1.4682347</td>
</tr>
<tr>
<td>133.3500000</td>
<td>38.0106730</td>
<td>5.2500000</td>
<td>1.4964832</td>
</tr>
<tr>
<td>134.6200000</td>
<td>38.7351120</td>
<td>5.3000000</td>
<td>1.5250044</td>
</tr>
<tr>
<td>135.8900000</td>
<td>39.4646913</td>
<td>5.3500000</td>
<td>1.5537989</td>
</tr>
<tr>
<td>137.1600000</td>
<td>40.2048249</td>
<td>5.4000000</td>
<td>1.5828671</td>
</tr>
<tr>
<td>138.4300000</td>
<td>40.9501272</td>
<td>5.4500000</td>
<td>1.6122097</td>
</tr>
<tr>
<td>139.7000000</td>
<td>41.7024138</td>
<td>5.5000000</td>
<td>1.6418273</td>
</tr>
<tr>
<td>140.9700000</td>
<td>42.4617014</td>
<td>5.5500000</td>
<td>1.6717205</td>
</tr>
<tr>
<td>142.2400000</td>
<td>43.2280075</td>
<td>5.6000000</td>
<td>1.7018901</td>
</tr>
<tr>
<td>143.5100000</td>
<td>44.0013511</td>
<td>5.6500000</td>
<td>1.7323367</td>
</tr>
<tr>
<td>144.7800000</td>
<td>44.7817520</td>
<td>5.7000000</td>
<td>1.7630611</td>
</tr>
<tr>
<td>146.0500000</td>
<td>45.5692317</td>
<td>5.7500000</td>
<td>1.7940642</td>
</tr>
<tr>
<td>147.3200000</td>
<td>46.3638127</td>
<td>5.8000000</td>
<td>1.8253470</td>
</tr>
<tr>
<td>148.5900000</td>
<td>47.1655192</td>
<td>5.8500000</td>
<td>1.8569102</td>
</tr>
<tr>
<td>149.8600000</td>
<td>47.9743767</td>
<td>5.9000000</td>
<td>1.8887550</td>
</tr>
<tr>
<td>151.1300000</td>
<td>48.7904124</td>
<td>5.9500000</td>
<td>1.9208824</td>
</tr>
<tr>
<td>152.4000000</td>
<td>49.6136553</td>
<td>6.0000000</td>
<td>1.9532935</td>
</tr>
<tr>
<td>153.6700000</td>
<td>50.4441361</td>
<td>6.0500000</td>
<td>1.9859896</td>
</tr>
<tr>
<td>154.9400000</td>
<td>51.2818873</td>
<td>6.1000000</td>
<td>2.0189719</td>
</tr>
<tr>
<td>156.2100000</td>
<td>52.1269437</td>
<td>6.1500000</td>
<td>2.0522419</td>
</tr>
<tr>
<td>R (mm)</td>
<td>Z R (mm)</td>
<td>Z (inches)</td>
<td>Z (inches)</td>
</tr>
<tr>
<td>----------------</td>
<td>------------------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>157.4800000</td>
<td>52.9793419</td>
<td>6.2000000</td>
<td>2.0858009</td>
</tr>
<tr>
<td>158.7500000</td>
<td>53.8391209</td>
<td>6.2500000</td>
<td>2.1196504</td>
</tr>
<tr>
<td>160.0200000</td>
<td>54.7063223</td>
<td>6.3000000</td>
<td>2.1537922</td>
</tr>
<tr>
<td>161.2900000</td>
<td>55.5809897</td>
<td>6.3500000</td>
<td>2.1882279</td>
</tr>
<tr>
<td>162.5600000</td>
<td>56.4631699</td>
<td>6.4000000</td>
<td>2.2229594</td>
</tr>
<tr>
<td>163.8300000</td>
<td>57.3529122</td>
<td>6.4500000</td>
<td>2.2579887</td>
</tr>
<tr>
<td>165.1000000</td>
<td>58.2502689</td>
<td>6.5000000</td>
<td>2.2933177</td>
</tr>
<tr>
<td>166.3700000</td>
<td>59.1552955</td>
<td>6.5500000</td>
<td>2.3289486</td>
</tr>
<tr>
<td>167.6400000</td>
<td>60.0680506</td>
<td>6.6000000</td>
<td>2.3648839</td>
</tr>
<tr>
<td>168.9100000</td>
<td>60.9885966</td>
<td>6.6500000</td>
<td>2.4011259</td>
</tr>
<tr>
<td>170.1800000</td>
<td>61.9169994</td>
<td>6.7000000</td>
<td>2.4376771</td>
</tr>
<tr>
<td>171.4500000</td>
<td>62.8533286</td>
<td>6.7500000</td>
<td>2.4745405</td>
</tr>
<tr>
<td>172.7200000</td>
<td>63.7976581</td>
<td>6.8000000</td>
<td>2.5117188</td>
</tr>
<tr>
<td>173.9900000</td>
<td>64.7500661</td>
<td>6.8500000</td>
<td>2.5492152</td>
</tr>
<tr>
<td>175.2600000</td>
<td>65.7106351</td>
<td>6.9000000</td>
<td>2.5870329</td>
</tr>
<tr>
<td>176.5300000</td>
<td>66.6794526</td>
<td>6.9500000</td>
<td>2.6251753</td>
</tr>
<tr>
<td>177.8000000</td>
<td>67.6566109</td>
<td>7.0000000</td>
<td>2.6636461</td>
</tr>
<tr>
<td>179.0700000</td>
<td>68.6422077</td>
<td>7.0500000</td>
<td>2.7024491</td>
</tr>
<tr>
<td>180.3400000</td>
<td>69.6363460</td>
<td>7.1000000</td>
<td>2.7415884</td>
</tr>
<tr>
<td>181.6100000</td>
<td>70.6391350</td>
<td>7.1500000</td>
<td>2.7810683</td>
</tr>
<tr>
<td>182.8800000</td>
<td>71.6506897</td>
<td>7.2000000</td>
<td>2.8208933</td>
</tr>
<tr>
<td>184.1500000</td>
<td>72.6711315</td>
<td>7.2500000</td>
<td>2.8610682</td>
</tr>
<tr>
<td>185.4200000</td>
<td>73.7005886</td>
<td>7.3000000</td>
<td>2.9015980</td>
</tr>
<tr>
<td>186.6900000</td>
<td>74.7391965</td>
<td>7.3500000</td>
<td>2.9424880</td>
</tr>
<tr>
<td>187.9600000</td>
<td>75.7870976</td>
<td>7.4000000</td>
<td>2.9837440</td>
</tr>
<tr>
<td>189.2300000</td>
<td>76.8444424</td>
<td>7.4500000</td>
<td>3.0253717</td>
</tr>
<tr>
<td>190.5000000</td>
<td>77.9113897</td>
<td>7.5000000</td>
<td>3.0673775</td>
</tr>
<tr>
<td>191.7700000</td>
<td>78.9881064</td>
<td>7.5500000</td>
<td>3.1097680</td>
</tr>
<tr>
<td>193.0400000</td>
<td>80.0747686</td>
<td>7.6000000</td>
<td>3.1525499</td>
</tr>
<tr>
<td>194.3100000</td>
<td>81.1715618</td>
<td>7.6500000</td>
<td>3.1957308</td>
</tr>
<tr>
<td>195.5800000</td>
<td>82.2786812</td>
<td>7.7000000</td>
<td>3.2393182</td>
</tr>
<tr>
<td>196.8500000</td>
<td>83.3963323</td>
<td>7.7500000</td>
<td>3.2833202</td>
</tr>
</tbody>
</table>
152.4 mm radius C. A.

317.5 mm diameter
+- 1 mm

Center Thickness
102.1 mm
+- 0.5 mm

Curvature
+0.002784 mm\(^{-1}\)
+- 0.000028 mm\(^{-1}\)

Material: Optical Grade Methyl Methacrylate (Acrylic)
Z = \frac{C(R^2)}{1+\sqrt{1-(1+K)(C^2)(R^2)}} + A_4(R^4) + A_6(R^6) + A_8(R^8) + A_{10}(R^{10})

C = \frac{6.5930000E-03}{\text{mm}} \quad \text{Radius of Curvature}

K = -3.5479300E-01 \quad \text{Conic constant}

A_4 = -3.3579450E-09 \quad \text{Fourth order aspheric coefficient}

A_6 = 1.2774210E-14 \quad \text{Sixth order aspheric coefficient}

A_8 = -2.2076200E-18 \quad \text{Eighth order aspheric coefficient}

A_{10} = 3.7341230E-23 \quad \text{Tenth order aspheric coefficient}

R = \text{radial distance from optical axis (mm)}

Z = \text{Sag. (mm)}

<table>
<thead>
<tr>
<th>R (mm)</th>
<th>Z (mm)</th>
<th>R (inches)</th>
<th>Z (inches)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>1.270000</td>
<td>0.0053170</td>
<td>0.050000</td>
<td>0.0002093</td>
</tr>
<tr>
<td>2.540000</td>
<td>0.0212685</td>
<td>0.100000</td>
<td>0.008373</td>
</tr>
<tr>
<td>3.810000</td>
<td>0.0478565</td>
<td>0.150000</td>
<td>0.018841</td>
</tr>
<tr>
<td>5.080000</td>
<td>0.0850840</td>
<td>0.200000</td>
<td>0.033498</td>
</tr>
<tr>
<td>6.350000</td>
<td>0.1329553</td>
<td>0.250000</td>
<td>0.052345</td>
</tr>
<tr>
<td>7.620000</td>
<td>0.1914760</td>
<td>0.300000</td>
<td>0.075384</td>
</tr>
<tr>
<td>8.890000</td>
<td>0.2606529</td>
<td>0.350000</td>
<td>0.102619</td>
</tr>
<tr>
<td>10.160000</td>
<td>0.3404941</td>
<td>0.400000</td>
<td>0.134053</td>
</tr>
<tr>
<td>11.430000</td>
<td>0.4310088</td>
<td>0.450000</td>
<td>0.169689</td>
</tr>
<tr>
<td>12.700000</td>
<td>0.5322078</td>
<td>0.500000</td>
<td>0.209531</td>
</tr>
<tr>
<td>13.970000</td>
<td>0.6441028</td>
<td>0.550000</td>
<td>0.253584</td>
</tr>
<tr>
<td>15.240000</td>
<td>0.7667071</td>
<td>0.600000</td>
<td>0.301853</td>
</tr>
<tr>
<td>16.510000</td>
<td>0.9000349</td>
<td>0.650000</td>
<td>0.354344</td>
</tr>
<tr>
<td>17.780000</td>
<td>1.0441022</td>
<td>0.700000</td>
<td>0.411064</td>
</tr>
<tr>
<td>19.050000</td>
<td>1.1989260</td>
<td>0.750000</td>
<td>0.472018</td>
</tr>
<tr>
<td>20.320000</td>
<td>1.3645246</td>
<td>0.800000</td>
<td>0.537214</td>
</tr>
<tr>
<td>21.590000</td>
<td>1.5409179</td>
<td>0.850000</td>
<td>0.606661</td>
</tr>
<tr>
<td>22.860000</td>
<td>1.7281269</td>
<td>0.900000</td>
<td>0.680365</td>
</tr>
<tr>
<td>24.130000</td>
<td>1.9261742</td>
<td>0.950000</td>
<td>0.758336</td>
</tr>
<tr>
<td>25.400000</td>
<td>2.1350837</td>
<td>1.000000</td>
<td>0.840584</td>
</tr>
<tr>
<td>26.670000</td>
<td>2.3548808</td>
<td>1.050000</td>
<td>0.927118</td>
</tr>
<tr>
<td>27.940000</td>
<td>2.5855921</td>
<td>1.100000</td>
<td>1.017950</td>
</tr>
<tr>
<td>29.210000</td>
<td>2.8272460</td>
<td>1.150000</td>
<td>1.113089</td>
</tr>
<tr>
<td>30.480000</td>
<td>3.0798722</td>
<td>1.200000</td>
<td>1.212548</td>
</tr>
<tr>
<td>31.750000</td>
<td>3.3435019</td>
<td>1.250000</td>
<td>1.316339</td>
</tr>
<tr>
<td>33.020000</td>
<td>3.6181679</td>
<td>1.300000</td>
<td>1.424476</td>
</tr>
<tr>
<td>34.290000</td>
<td>3.9039046</td>
<td>1.350000</td>
<td>1.536970</td>
</tr>
<tr>
<td>35.560000</td>
<td>4.2007480</td>
<td>1.400000</td>
<td>1.653838</td>
</tr>
<tr>
<td>36.830000</td>
<td>4.5087356</td>
<td>1.450000</td>
<td>1.775093</td>
</tr>
</tbody>
</table>
LDT NASA TELESCOPE ELEMENT 003 SURFACE 1

<table>
<thead>
<tr>
<th>R (mm)</th>
<th>Z (mm)</th>
<th>R (inches)</th>
<th>Z (inches)</th>
</tr>
</thead>
<tbody>
<tr>
<td>38.100000</td>
<td>4.8279067</td>
<td>1.5000000</td>
<td>0.1900751</td>
</tr>
<tr>
<td>39.370000</td>
<td>5.1583023</td>
<td>1.5500000</td>
<td>0.2030828</td>
</tr>
<tr>
<td>40.640000</td>
<td>5.4996949</td>
<td>1.6000000</td>
<td>0.2165341</td>
</tr>
<tr>
<td>41.910000</td>
<td>5.8529392</td>
<td>1.6500000</td>
<td>0.2304307</td>
</tr>
<tr>
<td>43.180000</td>
<td>6.2172712</td>
<td>1.7000000</td>
<td>0.2447745</td>
</tr>
<tr>
<td>44.450000</td>
<td>6.5910091</td>
<td>1.7500000</td>
<td>0.2596773</td>
</tr>
<tr>
<td>45.720000</td>
<td>6.9802028</td>
<td>1.8000000</td>
<td>0.2748111</td>
</tr>
<tr>
<td>46.990000</td>
<td>7.3789044</td>
<td>1.8500000</td>
<td>0.2905080</td>
</tr>
<tr>
<td>48.260000</td>
<td>7.7891676</td>
<td>1.9000000</td>
<td>0.3066601</td>
</tr>
<tr>
<td>49.530000</td>
<td>8.2110485</td>
<td>1.9500000</td>
<td>0.3232696</td>
</tr>
<tr>
<td>50.800000</td>
<td>8.6446049</td>
<td>2.0000000</td>
<td>0.3403388</td>
</tr>
<tr>
<td>52.070000</td>
<td>9.0898971</td>
<td>2.0500000</td>
<td>0.3578700</td>
</tr>
<tr>
<td>53.340000</td>
<td>9.5469873</td>
<td>2.1000000</td>
<td>0.3758656</td>
</tr>
<tr>
<td>54.610000</td>
<td>10.0159401</td>
<td>2.1500000</td>
<td>0.3943284</td>
</tr>
<tr>
<td>55.880000</td>
<td>10.4968224</td>
<td>2.2000000</td>
<td>0.4132607</td>
</tr>
<tr>
<td>57.150000</td>
<td>10.9897034</td>
<td>2.2500000</td>
<td>0.4326655</td>
</tr>
<tr>
<td>58.420000</td>
<td>11.4946548</td>
<td>2.3000000</td>
<td>0.4525455</td>
</tr>
<tr>
<td>59.690000</td>
<td>12.0117507</td>
<td>2.3500000</td>
<td>0.4729036</td>
</tr>
<tr>
<td>60.960000</td>
<td>12.5410679</td>
<td>2.4000000</td>
<td>0.4937428</td>
</tr>
<tr>
<td>62.230000</td>
<td>13.0826857</td>
<td>2.4500000</td>
<td>0.5150664</td>
</tr>
<tr>
<td>63.500000</td>
<td>13.6366865</td>
<td>2.5000000</td>
<td>0.5368774</td>
</tr>
<tr>
<td>64.770000</td>
<td>14.2031549</td>
<td>2.5500000</td>
<td>0.5591793</td>
</tr>
<tr>
<td>66.040000</td>
<td>14.7821791</td>
<td>2.6000000</td>
<td>0.5819756</td>
</tr>
<tr>
<td>67.310000</td>
<td>15.3738497</td>
<td>2.6500000</td>
<td>0.6052697</td>
</tr>
<tr>
<td>68.580000</td>
<td>15.9782607</td>
<td>2.7000000</td>
<td>0.6290654</td>
</tr>
<tr>
<td>69.850000</td>
<td>16.5955093</td>
<td>2.7500000</td>
<td>0.6533665</td>
</tr>
<tr>
<td>71.120000</td>
<td>17.2256959</td>
<td>2.8000000</td>
<td>0.6781770</td>
</tr>
<tr>
<td>72.390000</td>
<td>17.8689244</td>
<td>2.8500000</td>
<td>0.7035010</td>
</tr>
<tr>
<td>73.660000</td>
<td>18.5253021</td>
<td>2.9000000</td>
<td>0.7293426</td>
</tr>
<tr>
<td>74.930000</td>
<td>19.1949401</td>
<td>2.9500000</td>
<td>0.7557063</td>
</tr>
<tr>
<td>76.200000</td>
<td>19.8779533</td>
<td>3.0000000</td>
<td>0.7825966</td>
</tr>
<tr>
<td>77.470000</td>
<td>20.5744603</td>
<td>3.0500000</td>
<td>0.8100181</td>
</tr>
<tr>
<td>78.740000</td>
<td>21.2845841</td>
<td>3.1000000</td>
<td>0.8379758</td>
</tr>
<tr>
<td>80.010000</td>
<td>22.0084516</td>
<td>3.1500000</td>
<td>0.8664745</td>
</tr>
<tr>
<td>81.280000</td>
<td>22.7461943</td>
<td>3.2000000</td>
<td>0.8955195</td>
</tr>
<tr>
<td>82.550000</td>
<td>23.4979483</td>
<td>3.2500000</td>
<td>0.9251161</td>
</tr>
<tr>
<td>83.820000</td>
<td>24.2638541</td>
<td>3.3000000</td>
<td>0.9552698</td>
</tr>
<tr>
<td>85.090000</td>
<td>25.0440576</td>
<td>3.3500000</td>
<td>0.9859865</td>
</tr>
<tr>
<td>86.360000</td>
<td>25.8387095</td>
<td>3.4000000</td>
<td>1.0172720</td>
</tr>
<tr>
<td>87.630000</td>
<td>26.6479659</td>
<td>3.4500000</td>
<td>1.0491325</td>
</tr>
<tr>
<td>88.900000</td>
<td>27.4719886</td>
<td>3.5000000</td>
<td>1.0815744</td>
</tr>
<tr>
<td>90.170000</td>
<td>28.3109451</td>
<td>3.5500000</td>
<td>1.1146041</td>
</tr>
<tr>
<td>91.440000</td>
<td>29.1650090</td>
<td>3.6000000</td>
<td>1.1482287</td>
</tr>
<tr>
<td>92.710000</td>
<td>30.0343602</td>
<td>3.6500000</td>
<td>1.1824551</td>
</tr>
<tr>
<td>93.980000</td>
<td>30.9191854</td>
<td>3.7000000</td>
<td>1.2172908</td>
</tr>
<tr>
<td>95.250000</td>
<td>31.8196780</td>
<td>3.7500000</td>
<td>1.2527432</td>
</tr>
<tr>
<td>96.520000</td>
<td>32.7360388</td>
<td>3.8000000</td>
<td>1.2888204</td>
</tr>
<tr>
<td>R (mm)</td>
<td>Z (mm)</td>
<td>R (inches)</td>
<td>Z (inches)</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>97.7900000</td>
<td>33.6684763</td>
<td>3.8500000</td>
<td>1.3255306</td>
</tr>
<tr>
<td>99.0600000</td>
<td>34.6172067</td>
<td>3.9000000</td>
<td>1.3628822</td>
</tr>
<tr>
<td>100.3300000</td>
<td>35.5824549</td>
<td>3.9500000</td>
<td>1.4008841</td>
</tr>
<tr>
<td>101.6000000</td>
<td>36.5644543</td>
<td>4.0000000</td>
<td>1.4395454</td>
</tr>
<tr>
<td>102.8700000</td>
<td>37.5634479</td>
<td>4.0500000</td>
<td>1.4788759</td>
</tr>
<tr>
<td>104.1400000</td>
<td>38.5796880</td>
<td>4.1000000</td>
<td>1.5188854</td>
</tr>
<tr>
<td>105.4100000</td>
<td>39.6134375</td>
<td>4.1500000</td>
<td>1.5595842</td>
</tr>
<tr>
<td>106.6800000</td>
<td>40.6649700</td>
<td>4.2000000</td>
<td>1.6009831</td>
</tr>
<tr>
<td>107.9500000</td>
<td>41.7345704</td>
<td>4.2500000</td>
<td>1.6430933</td>
</tr>
<tr>
<td>109.2200000</td>
<td>42.8225357</td>
<td>4.3000000</td>
<td>1.6859266</td>
</tr>
<tr>
<td>110.4900000</td>
<td>43.9291755</td>
<td>4.3500000</td>
<td>1.7294951</td>
</tr>
<tr>
<td>111.7600000</td>
<td>45.0548131</td>
<td>4.4000000</td>
<td>1.7738115</td>
</tr>
<tr>
<td>113.0300000</td>
<td>46.1997858</td>
<td>4.4500000</td>
<td>1.8188892</td>
</tr>
<tr>
<td>114.3000000</td>
<td>47.3644459</td>
<td>4.5000000</td>
<td>1.8647420</td>
</tr>
<tr>
<td>115.5700000</td>
<td>48.5491620</td>
<td>4.5500000</td>
<td>1.9113843</td>
</tr>
<tr>
<td>116.8400000</td>
<td>49.7543193</td>
<td>4.6000000</td>
<td>1.9588315</td>
</tr>
<tr>
<td>118.1100000</td>
<td>50.9803214</td>
<td>4.6500000</td>
<td>2.0070993</td>
</tr>
<tr>
<td>119.3800000</td>
<td>52.2275909</td>
<td>4.7000000</td>
<td>2.0562044</td>
</tr>
<tr>
<td>120.6500000</td>
<td>53.4965709</td>
<td>4.7500000</td>
<td>2.1061642</td>
</tr>
<tr>
<td>121.9200000</td>
<td>54.7877263</td>
<td>4.8000000</td>
<td>2.1569971</td>
</tr>
<tr>
<td>123.1900000</td>
<td>56.1015454</td>
<td>4.8500000</td>
<td>2.2087223</td>
</tr>
<tr>
<td>124.4600000</td>
<td>57.4385411</td>
<td>4.9000000</td>
<td>2.2613599</td>
</tr>
<tr>
<td>125.7300000</td>
<td>58.7992532</td>
<td>4.9500000</td>
<td>2.3149312</td>
</tr>
<tr>
<td>127.0000000</td>
<td>60.1842501</td>
<td>5.0000000</td>
<td>2.3694587</td>
</tr>
<tr>
<td>128.2700000</td>
<td>61.5941305</td>
<td>5.0500000</td>
<td>2.4249658</td>
</tr>
<tr>
<td>129.5400000</td>
<td>63.0295265</td>
<td>5.1000000</td>
<td>2.4814774</td>
</tr>
<tr>
<td>130.8100000</td>
<td>64.4911051</td>
<td>5.1500000</td>
<td>2.5390199</td>
</tr>
<tr>
<td>132.0800000</td>
<td>65.9795719</td>
<td>5.2000000</td>
<td>2.5976209</td>
</tr>
<tr>
<td>133.3500000</td>
<td>67.4956735</td>
<td>5.2500000</td>
<td>2.6573100</td>
</tr>
<tr>
<td>134.6200000</td>
<td>69.0402010</td>
<td>5.3000000</td>
<td>2.7181181</td>
</tr>
<tr>
<td>135.8900000</td>
<td>70.6139938</td>
<td>5.3500000</td>
<td>2.7800785</td>
</tr>
<tr>
<td>137.1600000</td>
<td>72.2179436</td>
<td>5.4000000</td>
<td>2.8432261</td>
</tr>
<tr>
<td>138.4300000</td>
<td>73.8529990</td>
<td>5.4500000</td>
<td>2.9075984</td>
</tr>
<tr>
<td>139.7000000</td>
<td>75.5201704</td>
<td>5.5000000</td>
<td>2.9732351</td>
</tr>
<tr>
<td>140.9700000</td>
<td>77.2205360</td>
<td>5.5500000</td>
<td>3.0401786</td>
</tr>
<tr>
<td>142.2400000</td>
<td>78.9552475</td>
<td>5.6000000</td>
<td>3.1084743</td>
</tr>
<tr>
<td>143.5100000</td>
<td>80.7255380</td>
<td>5.6500000</td>
<td>3.1781708</td>
</tr>
<tr>
<td>144.7800000</td>
<td>82.5327290</td>
<td>5.7000000</td>
<td>3.2493200</td>
</tr>
<tr>
<td>146.0500000</td>
<td>84.3782406</td>
<td>5.7500000</td>
<td>3.3219780</td>
</tr>
<tr>
<td>147.3200000</td>
<td>86.2636005</td>
<td>5.8000000</td>
<td>3.3962047</td>
</tr>
<tr>
<td>148.5900000</td>
<td>88.1904564</td>
<td>5.8500000</td>
<td>3.4720652</td>
</tr>
<tr>
<td>149.8600000</td>
<td>90.1605892</td>
<td>5.9000000</td>
<td>3.5496295</td>
</tr>
<tr>
<td>151.1300000</td>
<td>92.1759279</td>
<td>5.9500000</td>
<td>3.6289735</td>
</tr>
<tr>
<td>152.4000000</td>
<td>94.2385673</td>
<td>6.0000000</td>
<td>3.7101798</td>
</tr>
<tr>
<td>153.6700000</td>
<td>96.3507883</td>
<td>6.0500000</td>
<td>3.7933381</td>
</tr>
<tr>
<td>154.9400000</td>
<td>98.5150815</td>
<td>6.1000000</td>
<td>3.8785465</td>
</tr>
<tr>
<td>156.2100000</td>
<td>100.7341746</td>
<td>6.1500000</td>
<td>3.9659124</td>
</tr>
</tbody>
</table>
### LDT NASA TELESCOPE ELEMENT 003 SURFACE 1

<table>
<thead>
<tr>
<th>R (mm)</th>
<th>Z (mm)</th>
<th>R (inches)</th>
<th>Z (inches)</th>
</tr>
</thead>
<tbody>
<tr>
<td>97.7900000</td>
<td>33.6684763</td>
<td>3.8500000</td>
<td>1.3255306</td>
</tr>
<tr>
<td>99.0600000</td>
<td>34.6172067</td>
<td>3.9000000</td>
<td>1.3628822</td>
</tr>
<tr>
<td>100.3300000</td>
<td>35.5824549</td>
<td>3.9500000</td>
<td>1.4008841</td>
</tr>
<tr>
<td>101.6000000</td>
<td>36.5644543</td>
<td>4.0000000</td>
<td>1.4395454</td>
</tr>
<tr>
<td>102.8700000</td>
<td>37.5634479</td>
<td>4.0500000</td>
<td>1.4788759</td>
</tr>
<tr>
<td>104.1400000</td>
<td>38.5796880</td>
<td>4.1000000</td>
<td>1.5188854</td>
</tr>
<tr>
<td>105.4100000</td>
<td>39.6134375</td>
<td>4.1500000</td>
<td>1.5595842</td>
</tr>
<tr>
<td>106.6800000</td>
<td>40.6649700</td>
<td>4.2000000</td>
<td>1.6009831</td>
</tr>
<tr>
<td>107.9500000</td>
<td>41.7345704</td>
<td>4.2500000</td>
<td>1.6430933</td>
</tr>
<tr>
<td>109.2200000</td>
<td>42.8225357</td>
<td>4.3000000</td>
<td>1.6859266</td>
</tr>
<tr>
<td>110.4900000</td>
<td>43.9291755</td>
<td>4.3500000</td>
<td>1.7294951</td>
</tr>
<tr>
<td>111.7600000</td>
<td>45.0548131</td>
<td>4.4000000</td>
<td>1.7738115</td>
</tr>
<tr>
<td>113.0300000</td>
<td>46.1997858</td>
<td>4.4500000</td>
<td>1.8188892</td>
</tr>
<tr>
<td>114.3000000</td>
<td>47.3644459</td>
<td>4.5000000</td>
<td>1.8647420</td>
</tr>
<tr>
<td>115.5700000</td>
<td>48.5491620</td>
<td>4.5500000</td>
<td>1.9113843</td>
</tr>
<tr>
<td>116.8400000</td>
<td>49.7543193</td>
<td>4.6000000</td>
<td>1.9588315</td>
</tr>
<tr>
<td>118.1100000</td>
<td>50.9803214</td>
<td>4.6500000</td>
<td>2.0070993</td>
</tr>
<tr>
<td>119.3800000</td>
<td>52.2275909</td>
<td>4.7000000</td>
<td>2.0562044</td>
</tr>
<tr>
<td>120.6500000</td>
<td>53.4965709</td>
<td>4.7500000</td>
<td>2.1061642</td>
</tr>
<tr>
<td>121.9200000</td>
<td>54.7877263</td>
<td>4.8000000</td>
<td>2.1569971</td>
</tr>
<tr>
<td>123.1900000</td>
<td>56.1015454</td>
<td>4.8500000</td>
<td>2.2087223</td>
</tr>
<tr>
<td>124.4600000</td>
<td>57.4385411</td>
<td>4.9000000</td>
<td>2.2613599</td>
</tr>
<tr>
<td>125.7300000</td>
<td>58.7992532</td>
<td>4.9500000</td>
<td>2.3149312</td>
</tr>
<tr>
<td>127.0000000</td>
<td>60.1842501</td>
<td>5.0000000</td>
<td>2.3694587</td>
</tr>
<tr>
<td>128.2700000</td>
<td>61.5941305</td>
<td>5.0500000</td>
<td>2.4249658</td>
</tr>
<tr>
<td>129.5400000</td>
<td>63.0295265</td>
<td>5.1000000</td>
<td>2.4814774</td>
</tr>
<tr>
<td>130.8100000</td>
<td>64.4911051</td>
<td>5.1500000</td>
<td>2.5390199</td>
</tr>
<tr>
<td>132.0800000</td>
<td>65.9795719</td>
<td>5.2000000</td>
<td>2.5976209</td>
</tr>
<tr>
<td>133.3500000</td>
<td>67.4956735</td>
<td>5.2500000</td>
<td>2.6573100</td>
</tr>
<tr>
<td>134.6200000</td>
<td>69.0402010</td>
<td>5.3000000</td>
<td>2.7181181</td>
</tr>
<tr>
<td>135.8900000</td>
<td>70.6139938</td>
<td>5.3500000</td>
<td>2.7800785</td>
</tr>
<tr>
<td>137.1600000</td>
<td>72.2179436</td>
<td>5.4000000</td>
<td>2.8432261</td>
</tr>
<tr>
<td>138.4300000</td>
<td>73.8529990</td>
<td>5.4500000</td>
<td>2.9075984</td>
</tr>
<tr>
<td>139.7000000</td>
<td>75.5201704</td>
<td>5.5000000</td>
<td>2.9732351</td>
</tr>
<tr>
<td>140.9700000</td>
<td>77.2205360</td>
<td>5.5500000</td>
<td>3.0401786</td>
</tr>
<tr>
<td>142.2400000</td>
<td>78.9552475</td>
<td>5.6000000</td>
<td>3.1084743</td>
</tr>
<tr>
<td>143.5100000</td>
<td>80.7255380</td>
<td>5.6500000</td>
<td>3.1781708</td>
</tr>
<tr>
<td>144.7800000</td>
<td>82.5327290</td>
<td>5.7000000</td>
<td>3.2493200</td>
</tr>
<tr>
<td>146.0500000</td>
<td>84.3782406</td>
<td>5.7500000</td>
<td>3.3219780</td>
</tr>
<tr>
<td>147.3200000</td>
<td>86.2636005</td>
<td>5.8000000</td>
<td>3.3962047</td>
</tr>
<tr>
<td>148.5900000</td>
<td>88.1904564</td>
<td>5.8500000</td>
<td>3.4720652</td>
</tr>
<tr>
<td>149.8600000</td>
<td>90.1605892</td>
<td>5.9000000</td>
<td>3.5496295</td>
</tr>
<tr>
<td>151.1300000</td>
<td>92.1759279</td>
<td>5.9500000</td>
<td>3.6289735</td>
</tr>
<tr>
<td>152.4000000</td>
<td>94.2385673</td>
<td>6.0000000</td>
<td>3.7101798</td>
</tr>
<tr>
<td>153.6700000</td>
<td>96.3507883</td>
<td>6.0500000</td>
<td>3.7933381</td>
</tr>
<tr>
<td>154.9400000</td>
<td>98.5150815</td>
<td>6.1000000</td>
<td>3.8785465</td>
</tr>
<tr>
<td>156.2100000</td>
<td>100.7341746</td>
<td>6.1500000</td>
<td>3.9659124</td>
</tr>
<tr>
<td>R (mm)</td>
<td>Z (mm)</td>
<td>R (inches)</td>
<td>Z (inches)</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>157.4800000</td>
<td>103.0110648</td>
<td>6.2000000</td>
<td>4.0555537</td>
</tr>
<tr>
<td>158.7500000</td>
<td>105.3490575</td>
<td>6.2500000</td>
<td>4.1476007</td>
</tr>
</tbody>
</table>
## COMPLETE ELECTRONIC SCHEMATICS

<table>
<thead>
<tr>
<th>Component</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Electronics Layout</td>
<td>B3</td>
</tr>
<tr>
<td>Angle Processor Electronics</td>
<td>B4</td>
</tr>
<tr>
<td>Quadrant Detector Electronics</td>
<td>B5</td>
</tr>
<tr>
<td>Wideband Amplifier</td>
<td>B6</td>
</tr>
<tr>
<td>Motor Switching (Ch 1)</td>
<td>B7</td>
</tr>
<tr>
<td>Motor Switching (Ch 2)</td>
<td>B8</td>
</tr>
<tr>
<td>Motor Switching (Ch 3)</td>
<td>B9</td>
</tr>
<tr>
<td>Relay Card Miscellaneous Circuit</td>
<td>B10</td>
</tr>
<tr>
<td>Ch 1 Cable from Relay Card to Disk Assy</td>
<td>B11</td>
</tr>
<tr>
<td>Ch 2 Cable from Relay Card to Disk Assy</td>
<td>B12</td>
</tr>
<tr>
<td>Ch 3 Cable from Relay Card to Disk Assy</td>
<td>B13</td>
</tr>
<tr>
<td>Receiver Electronics Interconnect</td>
<td>B14</td>
</tr>
<tr>
<td>Rec. Power Distribution to Custom Cable</td>
<td>B15</td>
</tr>
<tr>
<td>RS232 &amp; Sync Pulse Interconnect Card</td>
<td>B16</td>
</tr>
<tr>
<td>Single Channel Driver Circuit (Ch 1, typical)</td>
<td>B17</td>
</tr>
<tr>
<td>Demonstration Test Aid</td>
<td>B18</td>
</tr>
<tr>
<td>Laser Diode Driver Card</td>
<td>B19</td>
</tr>
<tr>
<td>OMA 6-axis Motion Control Card (Sh 1-6)</td>
<td>B20-B25</td>
</tr>
</tbody>
</table>
Note:
Pin 2 will splice into the shell of wire 7 in the custom cable.
Pin 1 will splice into wire B of the custom cable.
Channel 3

Notes:
Pin 2 will splice into the shield of wire 7 in the custom cable.
Pin 1 will splice into wire 6 of the custom cable.

Custom Cable Splice

- White
- Grey
- Green
- Violet

DB15

Disk A3 Motor
- MOTOR SERVO

Disk B3 Motor
- MOTOR SERVO

Cable 3

Laser Data Technology, Inc.
9375 Diehlman Industrial Drive
St. Louis, MO 63136

Title: Ch 3 Cable Relay Card to Disk Assembly
Size: Document Number: A
Date: August 28, 1992 Sheet of
Notes:
1. Vcc = +12V
2. U1 pin 8 is Vcc and pin 4 is Gnd

Diagram:
- VCC connected to TR trigger input.
- C1, C2, C3 capacitors with respective values.
- R2, R1, R3, R4, R5 resistors with respective values.
- Channel 1, 2, 3 Diode Drivers with R0, R7, R6 for bias.
- S1 switch for Slow and Fast modes.
- CW and CCW for motor direction.
- 12 DC Motor connected to R9 and R10.

Diagram Details:
- VCC connected to TR trigger input.
- C1, C2, C3 capacitors with respective values.
- R2, R1, R3, R4, R5 resistors with respective values.
- Channel 1, 2, 3 Diode Drivers with R0, R7, R6 for bias.
- S1 switch for Slow and Fast modes.
- CW and CCW for motor direction.
- 12 DC Motor connected to R9 and R10.

Diagram Elements:
- VCC
- TR
- R2
- C1
- C2
- C3
- R1
- R3
- R4
- R5
- R0
- R7
- R6
- S1
- M1
- 12 DC Motor

Diagram Notes:
- Vcc = +12V
- U1 pin 8 is Vcc and pin 4 is Gnd

Diagram Source:
Laser Data Technology, Inc.
9375 Dildman Industrial Drive
St. Louis, MO 63136

Diagram Description:
- This diagram represents a multi-access free space laser communications terminal final report.
- It includes a circuit diagram with various components and connections.
- Notable components include VCC, TR, R2, C1, C2, C3, R1, R3, R4, R5, R0, R7, R6, S1, M1, and 12 DC Motor.
- The diagram provides details on the connections and values for components R2, R1, R3, R4, R5, R0, R7, R6, S1, and M1.
- Notes specify Vcc = +12V and U1 pin 8 is Vcc and pin 4 is Gnd.
APPENDIX C

COMPLETE SOFTWARE LISTINGS

<table>
<thead>
<tr>
<th>Program</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPUTE.C</td>
<td>C3</td>
</tr>
<tr>
<td>DISCRETE.C</td>
<td>C5</td>
</tr>
<tr>
<td>DISPLAY.C</td>
<td>C9</td>
</tr>
<tr>
<td>FILE.C</td>
<td>C25</td>
</tr>
<tr>
<td>INITMOT.C</td>
<td>C30</td>
</tr>
<tr>
<td>MANUAL.C</td>
<td>C34</td>
</tr>
<tr>
<td>MENU.C</td>
<td>C36</td>
</tr>
<tr>
<td>MONITOR.C</td>
<td>C42</td>
</tr>
<tr>
<td>MOTCNT.C</td>
<td>C47</td>
</tr>
<tr>
<td>MOTOR.C</td>
<td>C61</td>
</tr>
<tr>
<td>NEWMOT.C</td>
<td>C70</td>
</tr>
<tr>
<td>SERL.C</td>
<td>C78</td>
</tr>
<tr>
<td>TRACK.C</td>
<td>C82</td>
</tr>
<tr>
<td>AZEL.DAT</td>
<td>C91</td>
</tr>
<tr>
<td>INPUT.DAT</td>
<td>C92</td>
</tr>
<tr>
<td>ADSYS.DSP</td>
<td>C93</td>
</tr>
<tr>
<td>ALGORITHMDSP</td>
<td>C95</td>
</tr>
<tr>
<td>BANDFILT.DSP</td>
<td>C99</td>
</tr>
<tr>
<td>LOWPASS.DSP</td>
<td>C102</td>
</tr>
<tr>
<td>MAIN.DSP</td>
<td>C104</td>
</tr>
<tr>
<td>UART.DSP</td>
<td>C112</td>
</tr>
<tr>
<td>ATTBLK.FCB</td>
<td>C117</td>
</tr>
<tr>
<td>CIRCBLK.FCB</td>
<td>C118</td>
</tr>
<tr>
<td>DISCBLK.FCB</td>
<td>C119</td>
</tr>
<tr>
<td>FOVBK.FCB</td>
<td>C120</td>
</tr>
<tr>
<td>MISCBK.FCB</td>
<td>C121</td>
</tr>
<tr>
<td>ORBITBLK.FCB</td>
<td>C122</td>
</tr>
<tr>
<td>SAVEBLK.FCB</td>
<td>C123</td>
</tr>
<tr>
<td>ALDLDDLAB.FOR</td>
<td>C124</td>
</tr>
<tr>
<td>ANGLEPR.FOR</td>
<td>C126</td>
</tr>
<tr>
<td>ARKTN.FOR</td>
<td>C129</td>
</tr>
<tr>
<td>ASGMNFN.FOR</td>
<td>C130</td>
</tr>
<tr>
<td>AZELALDL.FOR</td>
<td>C134</td>
</tr>
<tr>
<td>BLKDATA.FOR</td>
<td>C135</td>
</tr>
<tr>
<td>CNTRLFN.FOR</td>
<td>C136</td>
</tr>
<tr>
<td>COPYVEC.FOR</td>
<td>C143</td>
</tr>
<tr>
<td>CROSS.FOR</td>
<td>C144</td>
</tr>
<tr>
<td>DFQ.FOR</td>
<td>C145</td>
</tr>
<tr>
<td>DISCPOS.FOR</td>
<td>C146</td>
</tr>
<tr>
<td>DOT.FOR</td>
<td>C149</td>
</tr>
<tr>
<td>ELSTAT.FOR</td>
<td>C150</td>
</tr>
<tr>
<td>GAUSCL.FOR</td>
<td>C151</td>
</tr>
<tr>
<td>GAUSS.FOR</td>
<td>C152</td>
</tr>
<tr>
<td>INTRFACE.FOR</td>
<td>C153</td>
</tr>
<tr>
<td>Program</td>
<td>Page</td>
</tr>
<tr>
<td>------------------</td>
<td>------</td>
</tr>
<tr>
<td>INV3X3.FOR</td>
<td>C159</td>
</tr>
<tr>
<td>NAVIGFN.FOR</td>
<td>C160</td>
</tr>
<tr>
<td>NAVIGPR.FOR</td>
<td>C162</td>
</tr>
<tr>
<td>OMA.FOR</td>
<td>C164</td>
</tr>
<tr>
<td>QFITVAL.FOR</td>
<td>C165</td>
</tr>
<tr>
<td>RNDM.FOR</td>
<td>C167</td>
</tr>
<tr>
<td>RUK.FOR</td>
<td>C168</td>
</tr>
<tr>
<td>SCANGN.FOR</td>
<td>C169</td>
</tr>
<tr>
<td>SCENEPR.FOR</td>
<td>C171</td>
</tr>
<tr>
<td>TRUTHGN.FOR</td>
<td>C174</td>
</tr>
<tr>
<td>UNIT.FOR</td>
<td>C176</td>
</tr>
<tr>
<td>VLEN.FOR</td>
<td>C177</td>
</tr>
<tr>
<td>XKEP.FOR</td>
<td>C178</td>
</tr>
<tr>
<td>XYAZEL.FOR</td>
<td>C179</td>
</tr>
</tbody>
</table>
/* Compute.C: Contains all programs for computing Azimuth and Elevation numbers*/

#include <stdio.h>
#include <conio.h>
#include <ctype.h>
#include <time.h>
#include <graph.h>
#include <dos.h>
#include <time.h>
#include <math.h>
#include <float.h>

#define PI 3.141592654

/* Compute_Az_El computes azimuth and elevation parameters based on motor encoder positions StepA and StepB. Reflect the positions of the of DiskA and DiskB*/

Compute_AZ_EL (long int StepA, long int StepB, int Print_to_screen, double *Az_El, double *Alpha_Delta) {
  double Gear_ratio, Counts_per_degree;
  double Tangle, alpha, delta, Theta;
  double Pie_180 = PI / 180;

  /* 354 teeth per gear with 2000 motor encoder counts per one 360 degree motor shaft rotation. 4.640 (in.) is the radius of the drive drum and 1.70(in) is the radius of the pivot point */
  Counts_per_degree = (354.0 / 360.0) * 2000.0;
  Gear_ratio = Counts_per_degree / (4.640 / 1.70);
  Tangle = 0.0;
  alpha = 90.0 + ((double)StepA - (double)StepB) / Gear_ratio;
  Alpha_Delta[0] = alpha;
  delta = (double)StepA / Counts_per_degree;
  Alpha_Delta[1] = delta;

  if (alpha >= 90.0) {
    Theta = (3.141592654 / 2.0) + acos(Tangle) - (delta * Pie_180);
  } else {
    Theta = (3.141592654 / 2.0) - acos(Tangle) - (delta * Pie_180);
  }

  Az_El[0] = atanl(0.5427 * Tangle * cos(Theta)); /* Compute AZ error */
  Az_El[1] = atanl(0.5427 * Tangle * sin(Theta)); /* Compute Elevation error */

  if (Print_to_screen == 1) {
    printf("IstepA = %9li IstepB = %9li", StepA, StepB);
    printf(" Delta = %7.4f Deg. Alpha = %7.4f Deg.", delta, alpha);
    printf("AZ = %10.8f Rad. Elevation = %10.8f Rad.",
            Az_El[0], Az_El[1]);
    printf("AZ = %10.7f Deg. Elevation = %10.7f Deg.",
            Az_El[0] * 57.29577951, Az_El[1] * 57.29577951);
    printf("\n");
    Wait_for_key_Press();
  }
}

Compute_Az_El From_XY (double x, double y, int Print_results, double *AzEl) {
  AzEl[0] = atan((x / 116.0) * 0.194380309);
  AzEl[1] = atan((y / 116.0) * 0.194380309);
}

/* Compute_Az_El computes azimuth and elevation parameters based on X and Y position in millimeters */

Compute_Az_El From_XY (double x, double y, int Print_results, double *AzEl) {
if (Print_results == 1)
{
  printf("n X = %7.4f mm, Y = %7.4f mm ",x,y);
  printf("n Azimuth = %10.8f, Elevation = %10.8f", AzEl[0],AzEl[1]);
  printf("n\n\n");
  Wait_for_key_Press();
}

/* Function AzEl2AB computes disk positions from azimuth and elevation */
void AzEL2AB(double Azimuth, double Elevation,
              long int *DiskA, long int *DiskB,
              double *Delta, double *Alpha,
              int *Number_of_Solutions)
{
  double theta;
  double Ratio;
  double theta;
  double Ratio;
  int Array, Alt_element;
  theta=tan2(tan(Elevation),tan(Azimuth));
  Ratio=sqrt((pow(tan(Azimuth),2) + pow(tan(Elevation),2))/(2.0*1.396*tan(11.0*Pl/180.0));
  Delta[0]=90.0+180.0/Pl * (-theta + acos(Ratio));
  Alpha[0]=90.0+360.0/Pl * asin(Ratio);
  Delta[1]=90.0-180.0/Pl * (theta + acos(Ratio));
  Alpha[1]=90.0-360.0/Pl * asin(Ratio);
  if (fabs(Delta[0] >240.0))
    if (Delta[0] < 0) Delta[0] = Delta[0] + 360.0;
    else Delta[0] = Delta[0] - 360.0;
  if (fabs(Delta[1] >240.0))
  Alt_element = 2;
  if (fabs(Delta[0]) >= 120)
    if (Delta[0] < 0) Delta[Alt_element] = Delta[0] + 360.0;
    else Delta[Alt_element] = Delta[0] - 360.0;
    Alpha[Alt_element] = Alpha[0];
    Alt_element = Alt_element + 1;
  if (fabs(Delta[1]) >= 120)
    else Delta[Alt_element] = Delta[1] - 360.0;
    Alpha[Alt_element] = Alpha[1];
    Alt_element = Alt_element + 1;
  for (Array = 0; Array <= Alt_element-1; Array++)
    -720.5459770*(Alpha[Array]-90));
  Number_of_Solutions = Alt_element - 1;
  return;
}
/* discrete.C: controls all functions of the dc servo motors */

#include <stdio.h>
#include <conio.h>
#include <ctype.h>
#include <time.h>
#include <tire_e.h>
#include <graph.h>
#include <dos.h>

/* Mask_bit array is used to mask off bits that are not needed for test
these will be for switches 1-18 */
int Mask_bit[19] = {0x00,0x00,0x00,

0x40,0x80,0x20,
0x10,0x00,0x00,
0x04,0x08,0x02,
0x01,0x00,0x00,
0x00,0x80,0x20,
0x10};

int Activation_bit[19] = {0x00,0x00,0x00,

0x40,0x80,0x00,
0x00,0x00,0x00,
0x04,0x08,0x00,
0x00,0x00,0x00,
0x40,0x80,0x00,
0x00};

int Index_Mask_bit[7] = {0x0000,0x0004,0x0008,0x2000,

0x1000,0x8000,0x4000};

int Detect_Bit_5V = (0x01);
int Detect_Bit_12V = (0x02);

int Sync_Pulse_Detect[3] = {0x0600,0x0500,0x0300};
int Sync_pulse_data[16] = {0,0,0,3,0,2,1,0,0,0,0,0,0,0,0,0};

int Sync_Number;

extern unsigned Port_A_Address; /* Address of port A on 8255 */
extern unsigned Port_B_Address; /* Address of port B on 8255 */
extern unsigned Port_C_Address; /* Address of port C on 8255 */
extern unsigned Port_373; /* Address for the input post 373*/
extern Set to_actual;

unsigned Port_Address; /* Variable used to store address of
port to be read or written to */
int elaspedtime = 0;

/* Function used to test for switch activation */
int Check_for_switch_Activation(int Switch_Number)
{
int Detect_count = 0;
int Switch_data = 0;
int Switch_detect = 0;
int read_loop;

if ((Switch_Number >= 0) && (Switch_Number <= 14))
Port_Address = Port_A_Address;
else
Port_Address = Port_B_Address;

Switch_data = inp(Port_Address) & Mask_bit[Switch_Number];

Detect_count = 0;
if (Switch_data == Activation_bit[Switch_Number])
 { /* Delay reading the switch position for x amount of time */
   for (elapsedtime = 20000; elapsedtime > 0; elapsedtime = elapsedtime - 1);
   for (read_loop = 1; read_loop <= 3; read_loop ++)
   { /* Delay reading the switch position for x amount of time */
     for (elapsedtime = 5000; elapsedtime > 0; elapsedtime = elapsedtime - 1);
     Switch_data = inp(Port_Address) & Mask_bit[Switch_Number];
   }
   if (Switch_data == Activation_bit[Switch_Number])
   { 
     Detect_count = Detect_count+1;
   }
 } /* End of for (read_loop = 1; read_loop <= 3; read_loop ++) */
/* If switch was closed for 3 reads indicate that switch closure 
was detected */
if (Detect_count == 3)
  Switch_detect = 1;
else Switch_detect = 0;
} /* End of (Switch_data == Activation_bit[Switch_Number]) */
/* Run_Switch_Bit will be used to test the switches to see which ones 
are activated */
int Run_Switch_Bit(int Channel_to_test,int Display_output)
 { 
   int Number_of_Switch[12] =(3,4,5,6,9,10,11,12,15,16,17,18);
   int Act_switch = 0;
   int Switch_flag = 0;
   int Loop_count = 0;
   int Start,End;
   float Channel_number;
   if (Channel_to_test == 1)
   { 
     Start = 0; End = 3; Channel_number = 2.5;
     } 
   else if (Channel_to_test == 2)
   { 
     Start = 4; End = 7; Channel_number = 1.0;
     } 
   else
   { 
     Start = 8; End = 11; Channel_number = 4.0;
     }
   for (Loop_count = Start; Loop_count <= End; Loop_count++)
   { 
     Act_switch = Check_for_switch_Activation(Number_of_Switch[Loop_count]);
     if (Act_switch == 1)
     { 
       printf("Switch \% activated \n", Number_of_Switch[Loop_count]);
       Switch_flag = 1;
     }
   } /* End of for loop_count */
   if ((Switch_flag == 0) && (Display_output == 1))
   { 
     printf("\n No Switches found to be activated on Channel \%2.1f\n", 
       Channel_number);
     return Switch_flag;
   } /* End of function Run_Switch_Bit */
/* Check_for_Index_Mark checks for the occurs of an index mark */
```c
int Check_for_Index_Mark(int Motor_Number)
{
    int Index_data = 0;
    int Index_detect = 0;
    if ((Motor_Number == 1) || (Motor_Number == 2))
        Port_Address = Port_B_Address;
    else
        Port_Address = Port_373;
    Index_data = inpw(Port_Address) & Index_Mask_bit[Motor_Number];
    if (Index_data == 0)
    {
        /* Delay reading the switch position for x amount of time */
        for (elapsedtime = 1000; elapsedtime > 0; elapsedtime = elapsedtime - 1);
        Index_data = inpw(Port_Address) & Index_Mask_bit[Motor_Number];
        if (Index_data == 0)
    ....
    Index_detect = 1;
    }
    return Index_detect;
}

/* End of function Check_for_index_mark */

/* Function checks for to see if 5v supply is on */
int Check_for_5Volts(void)
{
    int Volt_Data = 0;
    int Active_5Volts = 0;
    Port_Address = Port_B_Address;
    Volt_Data = inpw(Port_Address) & Detect_Bit_5V;
    if (Volt_Data == Detect_Bit_5V)
    Active_5Volts = 1;
    return Active_5Volts;
}

/* Function checks for to see if +24v supply is on */
int Check_for_24Volts(void)
{
    int Volt_Data = 0;
    int Active_24Volts = 0;
    Port_Address = Port_B_Address;
    Volt_Data = inpw(Port_Address) & Detect_Bit_12V;
    if (Volt_Data == Detect_Bit_12V)
    Active_24Volts = 1;
    return Active_24Volts;
}

int Check_for_Motor_Power(void)
{
    int option_key;
    int loop;
    int exit = 0;
    char key_buffer[80];
    int Pwr_On24, Pwr_On5 = 0;
    Pwr_On24 = Check_for_24Volts();
    Pwr_On5 = Check_for_5Volts();
    if ((Pwr_On24 != 1) || (Pwr_On5 != 1))
    {
        /* Set all position registers to actual position before turning power */
```
222  Set_Motor_Position(Motor_Pair1_Address,Set_to_actual);
223  Set_Motor_Position(Motor_Pair2_Address,Set_to_actual);
224  Set_Motor_Position(Motor_Pair3_Address,Set_to_actual);
225  
226  while ((Pwr_On24 != 1) && (Pwr_On5 != 1))
227  {
228      /* Clear screen and ask for what channel we need to change */
229      _clearscreen(_GCLEARSCREEN); /* Clear screen for next output */
230      _settextposition(1,1);
231      printf("\n No Power to motor detected\n");
232      printf("\n1. Turn on digital amd motor power and press '1' to continue \n");
233      printf("\n2. Exit to main menu \n");
234      printf("\nSelect the one of the above options\n");
235  
236      option_key = -1; /* Reset value key value */
237  
238  do
239  {
240      if (option_key == 2)
241      {
242          exit = 1; break;
243      }
244      
245      get( key_buffer);
246      option_key = atoi(key_buffer);
247      while ((option_key <= 0) || (option_key > 2));
248      
249      Pwr_On24 = Check_for_24Volts();
250      Pwr_On5 = Check_for_5Volts();
251      
252  /* end of while (Power_On != 1)*/
253      
254  return exit;
255  
256  
257  /* Function Check_for_Active_Sync_Pulse checks for active sync pulse */
258  int Check_for_Active_Sync_Pulse(void)
259  {
260      
261      Sync_Number = (inpw(Port_373) & 0x00F0) >> 8;
262  
263      return Sync_pulse_data[Sync_Number];
264  
265  }
I / Display routine is used to provide the menu for manual control of the motors */

#include <stdio.h>
#include <conio.h>
#include <ctype.h>
#include <time.h>
#include <graph.h>
#include <dos.h>
#include <time.h>
#include <stdlib.h>

extern unsigned Motor_Pair1_Address; /*Address to first pair of motor controllers */
extern unsigned Motor_Pair2_Address; /*Address to first pair of motor controllers */
extern unsigned Control_word_address; /* Address of Control word on 8255 */
extern unsigned Port_A_Address; /* Address of port A on 8255 */
extern unsigned Port_B_Address; /* Address of port B on 8255 */
extern unsigned Port_C_Address; /* Address of port C on 8255 */
extern unsigned Port_373; /* Address of 74LS373 input port */
extern unsigned Motor_Pair2_Address;
extern unsigned Motor_Pair3_Address;
extern unsigned Control_word_address;
extern unsigned Port_A_Address;
extern unsigned Port_B_Address;
extern unsigned Port_C_Address;
extern unsigned Port_373;

extern int Starting_Position[2][3];
extern long int Current_Position[2][3];
extern double Azimuth[3],Elevation[3];
extern double Azimuth,Current Elevation;

int Stop_Serial_Data[3] = (0xFFBF,0xFF7F,0xFFEF);
int Initiate_Serial_Trans[3] = (0x40,0x80,0x10);

long int timeLapsed = 2000000; /* Variable for delay function */
int Channel_number; /* Number of channel to be selected */
int Error = 0; /* Variable used for receiving error number */
int Passed_check;
extern int C_Port_Data;
extern int Step,Menu_option,Initial_alignment,loop,Solution;
long int DiskAstep,DiskBstep,Power_on;

void Wait_for_key_Press(void);
void Process_Main_Option (int Key_pressed);
int Display_MainMenu(void);
char key_buffer[80];
extern FILE *Manual_test_ptr;
extern FILE *AzEl_Ptr;

int Display_Main_Menu(void)
{
    int key;
    */Print the main menu to the screen */
    clearscreen(_GCLEARSCREEN );
    settextposition(1,1); /*Position the cursor at position 1,1*/
printf( "** MANUAL CONTROL MENU **\n\n");

printf( "1. Set mode of motor controllers\n");
printf( "2. Read Status register\n");
printf( "3. Write to analog port\n");
printf( "4. Write to PWM port\n");
printf( "5. Set sampler timer\n");
printf( "6. Set digital filter parameters\n");
printf( "7. Read digital filter parameters\n");
printf( "8. Set motor acceleration\n");
printf( "9. Set motor velocity\n");
printf( "10. Read motor acceleration\n");
printf( "11. Read max motor velocity\n");
printf( "12. Read actual position of motor\n");
printf( "13. Clear position register \n\n");
printf( "14. Write final position to controller \n\n");
printf( "15. Read final position to controller \n\n");
printf( "16. Write command position to controller \n\n");
printf( "17. Read command position to controller \n\n");
printf( "18. Run motor in trapezoidal mode \n\n");
printf( "19. Read input discretes \n\n");
printf( "20. Align channel\n\n");
_settextposition(3,42);
printf( "21. Move optics arm\n\n");
_settextposition(4,42);
printf( "22. Exercise channel through limits\n\n");
_settextposition(5,42);
printf( "23. Test safety switches\n\n");
_settextposition(6,42);
printf( "24. Test serial link \n\n");
_settextposition(7,42);
printf( "25. Test PC to PC serial link \n\n");
_settextposition(8,42);
printf( "26. Backlash test \n\n");
_settextposition(9,42);
printf( "27. Step response test \n\n");
_settextposition(10,42);
printf( "28. Eccentricity test \n\n");
_settextposition(11,42);
printf( "29. Compute Azimuth & Elevation \n\n");
_settextposition(12,42);
printf( "30. Change Starting Azimuth & Elevation\n\n");
_settextposition(13,42);
printf( "31. Exercise multiple channels\n\n");
_settextposition(14,42);
printf( "32. Reset motor controllers\n\n");
_settextposition(15,42);
printf( "33. Check power supply discretes\n\n");
_settextposition(16,42);
printf( "34. Move arm to designated Az and El\n\n");
_settextposition(17,42);
printf( "35. Select channels to run\n\n");
_settextposition(23,1);
printf("Select the one of the above options\n")
key = 0; /* Reset value key value */

/* Read in data from the keyboard that is an integer format */
gets( key_buffer);
key = atoi(key_buffer);

if (key_buffer[0] == 'e') Close_down();

/*/ Test to see if it is within range of menu values */
while ((key <= 0) || (key > 35))
{
    gets( key_buffer);
    key = atoi(key_buffer);
    if (key_buffer[0] == 'e') Close_down();
}
return key;
int Display_Secondary_Menu(void)
{
    int key;
    /*Print the main menu to the screen*/
    _ctearscreen( GCLEARSCREEN );
    _settextposition(1,1);
    printf( " ** MANUAL CONTROL MENU **
    
    * Move Channel Commands *
    "I. Move optics arm delta & alpha increment
    "2. Move arm to designated Az and El
    "3. Exercise single channel through limits
    "4. Exercise all channels through limits
    "5. Compute channel Az & El
    
    * Alignment Command *
    "6. Channel alignment
    
    * System Test Commands *
    "7. Motor safety switch test
    "8. DSP to PC serial link test
    "9. Host PC to Monitoring PC serial link test
    "10. Power monitoring discrete test
    "11. Motor drive backlash test
    "12. Motor step response test
    "13. Channel eccentricity test
    "14. Reset Motor Command"
    
    _settextposition(23,1);
    printf("Select the one of the above options\n");
    key = 0; /* Reset value key value */
    /* Read in data from the keyboard that is an integer format */
    gets( key_buffer);
    key = atoi(key_buffer);
    if (key_buffer[0] == 'e') Close_down();
    /* Test to see if it is within range of menu values */
    while ((key <= 0) || (key > 15))
    {
        gets( key_buffer);
        key = atoi(key_buffer);
        if (key_buffer[0] == 'e') Close_down();
    }
    /* The following command is used to convert secondary menu options
to reflect options from the primary menu */
    key = Command_options[key];
    return key;
}
void Process_Main_Option (int Key_pressed)
{
    int read_data[3];
    long int read_long_data[3];
    int write_data[3];
    long int write_long_data[3];
    int Address_of_Motor;
    int time,Switch_activated,print_results;
    char buffer[80];
    int Channels_to_Exercise[4];
    int Channel;
/* Clear screen and ask for what channel we need to change */
_clearscreen( _CLEARSCREEN ); /* Clear screen for next output */
_Write_system_data(); /* Save position of motors to file */

if ((Key_pressed == 19) || (Key_pressed == 30) || (Key_pressed == 35)
    || (Key_pressed == 31) || (Key_pressed == 32) || (Key_pressed == 33))
{
    _settextposition(10,1); /* Position the cursor at position 1,1 */
    Channel_number = 1;
}
else
{
    _settextposition(10,1); /* Position the cursor at position 1,1 */
    printf("Enter the number of the channel (1, 2, 3) ");

    gets(buffer);
    Channel = atoi(buffer);
    Channel_number = 0;

/* Set address of motor based on the input from the keyboard */
if (Channel == 2)
{
    Address_of_Motor = Motor_Pair1_Address;
    Channel_number = 1;
}
else if (Channel == 1)
{
    Address_of_Motor = Motor_Pair2_Address;
    Channel_number = 2;
}
else if (Channel == 3)
{
    Address_of_Motor = Motor_Pair3_Address;
    Channel_number = 3;
}
/* End of Key_pressed != 19 */

/* Check to see that +24 volt power is on */
/* If exit was chosen jump to main menu */

if ((Key_pressed == 20) || (Key_pressed == 21) || (Key_pressed == 22)
    || (Key_pressed == 26) || (Key_pressed == 27) || (Key_pressed == 28)
    || (Key_pressed == 29) || (Key_pressed == 31) || (Key_pressed == 34))
{
    Passed_check = 0;
    Passed_check = Check_for_Motor_Power();
    if (Passed_check == 1) Key_pressed = 100;
}
if ((Channel_number >= 1) & (Channel_number <= 3))
{
    switch (Key_pressed)
    {
    case 1: /* Set control mode of motor controllers */
    {
        int option_key;
        unsigned statusword;
        printf("** Select mode of operation ** \n");
        printf("1. Software reset of controller \n");
        printf("2. Initialization/idle mode \n");
        printf("3. Align Mode \n");
        printf("4. Control Mode \n");
        printf("Select the one of the above options\n");
option_key = 0; /* Reset value key value */

/* Read in data from the keyboard that is an integer format */
gets(key_buffer);
option_key = atoi(key_buffer);

/* Test to see if it is within range of menu values */
while ((option_key <= 0) || (option_key > 4))
gets(key_buffer);
option_key = atoi(key_buffer);

statusword = Read_status_reg(Address_of_Motor);

if (option_key == 1)
Write_to_program_counter(Address_of_Motor,0x0000);
else if (option_key == 2)
Write_to_program_counter(Address_of_Motor,0x0101);
else if (option_key == 3) {
if ((statusword & 0x2020) == 0x2020) /* Check for Idle mode */
Write_to_program_counter(Address_of_Motor,0x0202);
else {
printf("Controller is not in IDLE mode. Set controller to IDLE mode to continue \n");
Wait_for_key_Press();
}
}
else if (option_key == 4) {
if ((statusword & 0x2020) == 0x2020) /* Check for Idle mode */
{ int zero[3] = {0,0,0};
int zero_long[3] = {0,0,0};
/* Clear flag to disable trapezoidal move */
Write_to_fflag_registor(Address_of_Motor,0x0000);
/* Set the controller motor control to bipolar operation */
Write_to_fflag_registor(Address_of_Motor,0x0202);
/* Clear flag to disable proportional velocity mode */
Write_to_fflag_registor(Address_of_Motor,0x0303);
/* Clear flag to disable integral velocity mode */
Write_to_fflag_registor(Address_of_Motor,0x0505);
/* Initialize command position to 0 */
Write_command_pos(Address_of_Motor,zero_long);
/* Initialize actual position to 0 */
Reset_Actual_Position(Address_of_Motor,zero);
/* Initialize final position to 0 */
Write_Final_pos(Address_of_Motor,zero_long);
Write_to_program_counter(Address_of_Motor,0x0303);
}
else {
printf("Controller is not in IDLE mode. Set controller to IDLE mode to continue \n");
Wait_for_key_Press();
}
break;
}

case 2: /* Read Status register */

unsigned status;
status = Read_status_reg(Address_of_Motor);
printf("\n\nValue of the status register for DiskB motor controller = %x \n",status & 0xff);
printf("Value of the status register for DiskA motor controller = %x \n",status >> 8);
Wait_for_key_Press();
break;
} /* End of case 2 */
case 3: /* Write to analog port */
{
printf("Enter the value to be outputted to the DiskB motor controller\n");
printf("\n\nanalog port Hex (0-FF)\n");
cscanf("%x", &write_data[0]);
printf("Enter the value to be outputted to the DiskA motor controller\n");
printf("\n\nanalog port in Hex (0-FF)\n");
cscanf("%x", &write_data[1]);
Write_to_8bit_port(Address_of_Motor,(write_data[0] & OxOOff)\n(write_data[1] << 8));
break;
} /* End of case 3 */

} /* End of case 4 */
case 5: /* Set sampler timer */
{
printf("\nEnter the value to be outputted to the timer of the controller in hex\n");
cscanf("%x", &write_data[0]);
Write_to_sampler_timer(Address_of_Motor,write_data[0]);
break;
} /* End of case 5 */

} /* End of case 6 */
case 6: /* Set digital filter parameters */
{
printf("Same filter parameters will be used for both\n");
printf("\nEnter the pole for the digital filter in hex\n");
cscanf("%x", &write_data[0]);
printf("\nEnter the zero for the digital filter in hex\n");
cscanf("%x", &write_data[1]);
printf("\nEnter the gain for the digital filter in hex\n");
cscanf("%x", &write_data[2]);
Write_Filter_Pole(Address_of_Motor,write_data[0]);
Write_Filter_Zero(Address_of_Motor,write_data[1]);
Write_Filter_Gain(Address_of_Motor,write_data[2]);
break; /* Exit the case statement at this point */

} /* End of case 6 */

case 7: /* Read digital filter parameters */
{
  /* Read filter data pack from controller*/
  read_data[0] = Read_Filter_Zero(Address_of_Motor);
  read_data[1] = Read_Filter_Pole(Address_of_Motor);
  read_data[2] = Read_Filter_Gain(Address_of_Motor);

  printf("** DiskB Motor Controller Filter Parameters **\n");
  printf("Zero of digital filter (Hex) = \%x\n", (read_data[0] & 0xff));
  printf("Pole of digital filter (Hex) = \%x\n", (read_data[1] & 0xff));
  printf("Gain of digital filter (Hex) = \%x\n", (read_data[2] & 0xff));

  printf("** DiskA Motor Controller Filter Parameters **\n");
  printf("Zero of digital filter (Hex) = \%x\n", ((read_data[0] & 0xff00) >> 8));
  printf("Pole of digital filter (Hex) = \%x\n", ((read_data[1] & 0xff00) >> 8));
  printf("Gain of digital filter (Hex) = \%x\n", ((read_data[2] & 0xff00) >> 8));

  Wait_for_key_Press(); /* Wait for a key to be pressed before continuing */
  break;

} /* End of case 7 */

} /* End of case 8 */

case 8: /* Set motor acceleration */
{
  printf("Enter the acceleration for DiskB motor in hex \n");
  cscanf("\%x", &write_data[0]);
  printf("Enter the acceleration for DiskA motor in hex \n");
  cscanf("\%x", &write_data[1]);

  /* Output the values of acceleration to the controllers */
  Write_Max_Accel(Address_of_Motor,write_data);
  break;

} /* End of case 8 */

} /* End of case 9 */

} /* End of case 10 */
case 11: /* Read PWM port */
{
    read_data[0] = Read_Max_Velocity(Address_of_Motor);
    printf("\n DiskB motor max. velocity is set at (Hex) %x\n", read_data[0] & 0xff);
    printf("\n DiskA motor max. velocity is set at (Hex) %x\n\n", read_data[0] >> 8);
    Wait_for_key_Press();
    break;
} /* End of case 11 */

case 12: /* Read actual position of motor */
{
    getch(); /* Clear all data out of buffer */
timetasped = 20000;
    while(!kbhit()) /* Wait for key on keyboard to be pressed */
    {
        Read_Actuat_Pos(Address_of_Motor,read_long_data);
        printf("\n DiskB motor position = %li \n", read_long_data[0]);
        printf(" DiskA motor position = %li \n\n", read_long_data[1]);
        /* delay the program be requesting new data */
        for (timetasped = 20000; timetasped > 0; timetasped = timetasped - 1)
            time = time + 1;
        time = 0;
    }
    /* Use getch to throw away. */
    getch();
    break;
} /* End of case 12 */

case 13:
{
    Reset_Actual_Position(Address_of_Motor,0x0000);
    break;
} /* End of case 13 */

case 14: /* Write final position */
{
    /* Enter position to which motor will be moved to */
    printf("\n Enter new position for DiskB motor movement - \n\n");
    gets(key_buffer);
    write_long_data[0] = atol(key_buffer);
    /* Enter position to which DiskA motor will be moved to */
    printf("\n Enter new position for DiskA motor movement - \n\n");
    gets(key_buffer);
    write_long_data[1] = atol(key_buffer);
    /* Output the final position tho the controller*/
    Write_Final_pos(Address_of_Motor,write_long_data);
    break;
} /* End of case 14 */

case 15: /* Read final position register */
{
    Read_Final_Pos(Address_of_Motor,read_long_data);
    printf("\n The final position of the DiskB motor is = %li \n\n", read_long_data[0]);
    printf("\n The final position of the DiskA motor is = %li \n\n", read_long_data[1]);
    Wait_for_key_Press();
break;

case 16: /* Write command position */
    
    break;

    printf("Enter the command position to run the DiskB motor to \
");
    gets( key_buffer);
    write_long_data[0] = atol(key_buffer);
    printf("Enter the command position to run the DiskA motor to \n");
    gets( key_buffer);
    write_long_data[0] = atol(key_buffer);

    printf("The command position of the DiskB motor is = %li \\
");
    printf("The command position of the DiskA motor is = %li \\
");
    Wait_for_key_Press();

    case 17: /* Read command position register */

    Read_Command_Pos(Address_of_Motor,read_long_data);

    printf("The command position of the DiskB motor is = %li \\
");
    printf("The command position of the DiskA motor is = %li \\
");
    Wait_for_key_Press();

    break;

    case 18: /* Start motor */

    Write_to_flag_register(Address_of_Motor,0x0808);

    printf("Port A Discretes = _x \\n");
    printf("Port B Discretes = _x \\
");
    printf("Controller Port = %x",read_data[1],read_data[2]);

    /* delay the program be requesting new data */
    for (timetasped = 25; timetasped > 0; timetasped = timetasped - 1)
        time = time +1;
    time = 0;

    /* Use getch to throw key away. */
    getch();

    case 19: /* Read out discretes and print to the screen */

    Read_Command_Pos(Address_of_Motor,read_long_data);

    printf("Port A Discretes = _x \\
");
    printf("Port B Discretes = _x \\
");
    printf("Controller Port = %x",read_data[1],read_data[2]);

    break;

    case 20: /* Align selected channel */

    printf(" ** Alignment options ** \n");
    printf("1. Encoder initialization ");
    printf("2. Initial alignment calibration ");
    printf("3. Exit without alignment ");
printf("\n Select desired option ");

do {
    gets(key_buffer);
    Menu_option = atoi(key_buffer);
} while ((Menu_option <= 0) || (Menu_option > 3));

if (Menu_option == 2) {
    Initial_alignment = 1;
    Set_Motor_Position(Address_of_Motor,Reset_to_zero);
    Align_Channel(Address_of_Motor,Channel_number,Initial_alignment);
} else if (Menu_option == 1) {
    Initial_alignment = 0;
    Align_Channel(Address_of_Motor,Channel_number,Initial_alignment);
}

break;

case 21: /* Move optics arm */
{
    long int Offset_Position[3];
    long int alpha = 0;
    long int delta = 0;
    int in_range_flag = 0; /* Flag indicating data inputted is in range */

    while (in_range_flag == 0) {
        printf("\nEnter change in delta position of arm \n");
        gets(key_buffer);
        delta = atof(key_buffer);
        printf("\nEnter change in alpha position of arm \n");
        gets(key_buffer);
        alpha = atof(key_buffer);
        if ((delta <= 480) && (delta >= -480) &&
            (alpha <= 40) && (alpha >= -40))
            in_range_flag = 1;
        else printf("\nInvalid range select again!! \n");
    }
    /* End of while in_range_flag == 0 */

    /* Read in actual position of the DiskA and DiskB motors for offset */
    Read_Actual_Pos(Address_of_Motor,Offset_Position);

    /* Compute position for DiskB motor */
    write_long_data[0] = (1967 * delta) + (alpha * 769) + Offset_Position[0];

    /* Compute position for DiskA motor command */

    /* Output the final position to the controller*/
    Write_Final_pos(Address_of_Motor,write_long_data);

    /* Start up motors for arm movement */
    Write_to_flag_register(Address_of_Motor,0x0808);
    break;
}

} /* End of case 21 */

case 22: /* Exercise channel through its limits */
{
    Exercise_Channel(Address_of_Motor,Channel_number);
    break;
}

} /* End of case 22 */

case 23: /* Run test on switches for designated channel */
printf("\n  Switch Test Results  \n");
print_results = 1;
Switch_activated = Run_Switch_Bit(Channel_number,print_results);
Wait_for_key_Press();
break;
}

case 24: /* Run test on serial link */
{
Init_Com_port(); /*Initialize serial port */
ComFlushRx(); /* Flush out data before starting loop */
printf("\nPower receiver electronics\n");
Wait_for_key_Press();
printf("\n\nTesting channel %i's serial link. Please wait!\n", Channel);
Error = Test_Serial_Link(Channel_number,C_Port_Data);
if (Error == 0) printf("\n\nSerial link communications good\n");
ComCloseAll();
Wait_for_key_Press();
break;
}

case 25: /* Run test on serial link */
{
Init_Com_port(); /*Initialize serial port */
ComFlushRx(); /* Flush out data before starting loop */
printf("\n\nHit any key to exit option\n");
while(!kbhit()) /* Wait for key on keyboard to be pressed */
{
    Error = Test_Serial_Link(Channel_number,C_Port_Data);
    getch();
ComCloseAll();
break;
}

case 26: /* Run backlash test */
{
printf("\n  1.) Test Backlash Disk B channel (U_error)\n");
printf("\n  2.) Test Backlash Disk A channel\n");
printf("\n  3.) Test Backlash Disk A & B channel together (V_error)\n");
printf("\n  4.) Test Backlash Disk A & B channel in opposite directions\n");
printf("\n  5.) Exit test\n");
printf("\n Select desired option \n");
do (
    gets( key_buffer);
    Menu_option = atoi(key_buffer);
) while ((Menu_option <= 5) || (Menu_option > 5));
if (Menu_option != 5)
{
    Init_Com_port(); /*Initialize serial port */
    ComFlushRx(); /* Flush out data before starting loop */
    Run_Back_Lash_Test(Address_of_Motor,C_Port_Data,
                        Channel_number,Menu_option);
ComCloseAll();
}
break;
}

case 27: /* Test motor for step response */
{
printf("\n Command move test options ");
printf("\n 1.) Test Disk B motor ");
printf("\n 2.) Test Disk A motor ");
printf("\n 3.) Test both Disk B & A motors running in same direction ");
printf("\n 4.) Test both Disk B & A motors running in opposite direction ");
printf("\n Trapezoidal move test options ");
printf("\n 5.) Test Disk B motor ");
printf("\n 6.) Test Disk A motor ");
printf("\n 7.) Test both Disk B & A motors running in same direction ");
printf("\n 8.) Test both Disk B & A motors running in opposite direction ");
printf("\n 9.) Exit test ");
printf("\n Select desired option ");

/* End of if Menu_option != 4 */
break;

/* Run eccentricity test on gears */

Init_Com_port(); /*Initialize serial port */
ComFlushRx(); /* Flush out data before starting loop */
Run_Eccen_Test(Address_of_Rotor, Channel_number);
ComCloseAll()
break;

/* Read data from system file on present position */
Read_AzEl_data();

Read_Current_System_data();

/* Store values read in program into temporary array */
for (loop = 0; loop <= 2; loop++)
{
  Temp_Azimuth[loop] = Azimuth[loop];
  Temp_Elevation[loop] = Elevation[loop];
}

printf("\n *** Compute Azimuth and Elevation parameters *** ");
printf("\n 1.) Use current disks position on Channel %d, Channel ");
printf("\n 2.) Enter disks positions by hand for Channel %d, Channel ");
3.) Compute AZ & EL for all three channels using current positions
4.) Exit option

Select desired options

```
do
    gets(key_buffer);
    Menu_option = atoi(key_buffer);
)
while ((Menu_option <= 0) || (Menu_option > 4));
/* Exit case statement at this time */
if (Menu_option == 4) break;
if (Menu_option == 2)
    printf("Enter IstepA position for computing Az & El \n");
    gets(key_buffer);
    DiskAstep = atoi(key_buffer);
    printf("Enter IstepB position for computing Az & El \n");
    gets(key_buffer);
    DiskBstep = atoi(key_buffer);
    printf("\n\nAzimuth & Elevation for Channel \ni",Channel);
    Compute_AZ_EL(DiskAstep,DiskBstep,1,El_Az,Alpha_Delta);
    Temp_Azimuth[Channel_number -1] = El_Az[0]*57.29577951;
    Temp_Elevation[Channel_number -1] = El_Az[1]*57.29577951;
if (Menu_option == 1)
    { 
    DiskAstep = Current_Position[1][Channel_number - 1];
    DiskBstep = Current_Position[0][Channel_number - 1];
    printf("\n\nAzimuth & Elevation for Channel \ni",Channel);
    Compute_AZ_EL(DiskAstep,DiskBstep,1,El_Az,Alpha_Delta);
    Temp_Azimuth[Channel_number -1] = El_Az[0]*57.29577951;
    Temp_Elevation[Channel_number -1] = El_Az[1]*57.29577951;
}
if (Menu_option == 3)
    { 
    for (loop = 0; loop <= 2; loop++)
    { 
    if (loop == 0) Channel = 2;
    else if (loop == 1) Channel = 1;
    else if (loop == 2) Channel = 3;
    DiskAstep = Current_Position[1][loop];
    DiskBstep = Current_Position[0][loop];
    printf("\n\nAzimuth & Elevation for Channel \ni",Channel);
    Compute_AZ_EL(DiskAstep,DiskBstep,1,El_Az,Alpha_Delta);
    Temp_Azimuth[loop] = El_Az[0]*57.29577951;
    Temp_Elevation[loop] = El_Az[1]*57.29577951;
}
Menu_option = 0;
printf("1.) Save changes to file ");
printf("2.) Continue without saving changes");
printf("3.) Select option\n");
do
    gets(key_buffer);
```
Menu_option = atoi(key_buffer);

while (((Menu_option <= 0) || (Menu_option > 2));
if (Menu_option == 1)
{
for (loop = 0; loop <= 2; loop++)
{
Azimuth[loop] = Temp_Azimuth[loop];
Elevation[loop] = Temp_Elevation[loop];
}
Write_AzEl_data();
break;
}

{ case 30:
  Select_Azel_Init(); /*Request user for new azimuth and elevation parameters*/
  break;
}
}

{ case 31:
  Channels_to_Exercise[0] = 0; /* Disable channel */
  Channels_to_Exercise[1] = 1;
  Channels_to_Exercise[2] = 1;
  Channels_to_Exercise[3] = 1; /* Enable channel */
  Exercise_Multiple_Channels(Channels_to_Exercise);
  break;
}

{ case 32: /* Reset all motor controllers to present actual position */
  Address_of_Motor = Motor_Pair1_Address;
  Set_Motor_Position(Address_of_Motor, Set_to_actual);
  Address_of_Motor = Motor_Pair2_Address;
  Set_Motor_Position(Address_of_Motor, Set_to_actual);
  Address_of_Motor = Motor_Pair3_Address;
  Set_Motor_Position(Address_of_Motor, Set_to_actual);
  break;
}

{ case 33: /* Check power supply */
  Power_on = Check_for_24Volts();
  if (Power_on == 1)
    printf("\n +24 Volt power supply on\n");
  else printf("\n +24 Volt power supply off\n");
  Power_on = Check_for_5Volts();
  if (Power_on == 1)
    printf("\n +/-5 Volt power supply on\n");
  else printf("\n +/-5 Volt power supply off\n");
  Wait_for_key_Press();
  break;
}

{ case 34: /* Move arm to new position */
  Menu_option = 0;
  Read_AzEl_data();
  /* Read data from system file on present position */
  Read_Current_System_data();
  /* Use getch to throw key away. */
  while (kbhit()) getch();
  printf("\n(1) Enter absolute AZ & EL values ");
printf("\n2.) Move arm with keyboard arrows\n");
printf("\n3.) Run scan pattern\n");
printf("\n4.) Exit without moving arm\n");
printf("\n Select option\n");
do
{
    gets(key_buffer);
    Menu_option = atoi(key_buffer);
}
while ((Menu_option <= 0) || (Menu_option > 4));
/* Use getch to throw key away. */
while (kbhit()) getch();
if (Menu_option == 1)
{
    Move_to_New_AZEl(Channel_number,Address_of_Motor);
}
if ((Menu_option == 2) || (Menu_option == 3))
{
    DiskAstep = Current_Position[1][Channel_number-1];
    DiskBstep = Current_Position[0][Channel_number-1];
    Compute_AZ_El(DiskAstep,DiskBstep,0,EL_Az,Alpha_Delta);
    Current_Azimuth = EL_Az[0];
    Current_Elevation = EL_Az[1];
    Init_Com_port();  /* Initialize serial port */
    if (Menu_option == 2)
    {
        Position(Current_Azimuth,Current_Elevation,
                Channel_number,Address_of_Motor);
    }
    else
    {
        printf("\nEnter level of trip point for scan (+0.2 to +39.0 Volts)\n");
        Stop_point = 0;
        while ((Stop_point < 0.2) || (Stop_point > 39.0))
        {
            gets(key_buffer);
            Stop_point = atof(key_buffer);
        }
        /* Use getch to throw key away. */
        while (kbhit()) getch();
        Run_Scan_Pattern(Current_Azimuth,Current_Elevation,Channel_number,
                Address_of_Motor,Stop_point);
    }
    while (kbhit()) getch();
    printf("\n1.) Save AZ & EL values for tracking\n");
    printf("\n2.) Exit without saving Az & EL\n");
    printf("\n Select option\n");
do
    {
        gets(key_buffer);
        Menu_option = atoi(key_buffer);
    }
    while ((Menu_option <= 0) || (Menu_option > 2));
    if (Menu_option == 1)
    {
        Write_system_data();
    }
Read_Current_System_data();
DiskAstep = Current_Position[1][Channel_number-1];
DiskBstep = Current_Position[0][Channel_number-1];
Compute_AZ_EL (DiskAstep,DiskBstep,0,El_Az,Alpha_Delte);
Azimuth[Channel_number-1] = El_Az[0]*57.29577951;
Elevation[Channel_number-1] = El_Az[1]*57.29577951;
Write_AzEl_data();
Write_AzEl_Track_Data();
ComCloseAll();
break;
}
case 35:
 Select_Channels_to_Run();
 default:
broadcast */
 } /* End of switch statement */
} /* End of if Channel_number >=1) & (Channel_number <= */
/* End of function Call_Routine */
void Wait_for_key_Press(void)
{
 printf("Press any key to continue");
 /* Display message until key is pressed. */
 while( lkbhit()); /* Wait for key on keyboard to be pressed */
 /* Use getch to throw key away. */
 while (kbhit()) getch();
}
FILE.C: Used to open all necessary files for diagnostic */

#include <stdio.h>
#include <conio.h>
#include <ctype.h>
#include <time.h>
#include <dos.h>
#include <time.h>
#include <cport.h>
#include <math.h>
#include <stdlib.h>
#include <graph.h>

FILE *output_ptr; /* Set up pointer for file to output error data */
FILE *Data_Ptr; /* Set up pointer for file to output test data */
FILE *System_Ptr; /* Set up pointer for file that contains system data */
FILE *AzEl_Ptr; /* Set up pointer for file that contains Azimuth & Elevation data for all three channels*/
FILE *Track_AzEl_Ptr; /* Set up pointer for tracking par. for tracking software */
FILE *Align_Ptr; /* Set up pointer with alignment info */
FILE *Manual_test_ptr; /* Set up pointer for file to output error data */
FILE *Input_Ptr; /* Set up pointer for data that contains input data */

long int Disk_Position[2];
long int Disk_Leo_Position[2];
lng int Starting_Position[2][3];
long int Current_Position[2][3];
long int LEO_Position[2][3];
long int Align_Position[2][3];
double Azimuth[3],Elevation[3];
double Track_Az[3],Track_El[3];
int Chan,Chan_Address;
int Channel_data;

/* Declare the following variables external these var. are declared in motor.c */
extern unsigned Motor_Pair1_Address; /*Address to first pair of motor controllers */
extern unsigned Motor_Pair2_Address; /*Address to first pair of motor controllers */
extern unsigned Motor_Pair3_Address; /*Address to first pair of motor controllers */
extern unsigned int Offse_Position[3][4];
extern int Selected Channels[3][2];

void Read_system_data(void);
void Open_files(int track);
void Write_system_data(void);

void Open_files(int track)
{
    /* If tracking mode is used open these additional files */
    if (track == 1)
    {
        if ((output_ptr=fopen("c:\nasa\output.txt","at")) == NULL)
        {
            printf("Cannot open file\n");
            exit(1);
        }
        if ((Data_Ptr=fopen("c:\nasa\Test.dat","at")) == NULL)
        {
            printf("Cannot open file\n");
            exit(1);
        }
        if ((Manual_test_ptr=fopen("c:\nasa\manual.dat","a+t")) == NULL)
        {
            printf("Cannot open file\n");
            exit(1);
        }
    }
    /* Open only if using manual mode */
    if (track == 0)
    {
        if ((Manual_test_ptr=fopen("c:\nasa\manual.dat","a+t")) == NULL)
        {
            printf("Cannot open file\n");
            exit(1);
        }
    }
printf("Cannot open file
");
exit(1);
}

if ((AzEl_Ptr=fopen("c:\nasa\System.dat","r+t")) == NULL)
{
printf("Cannot open file
");
exit(1);
}
if ((Track_AzEl_Ptr=fopen("c:\nasa\System.dat","r+t")) == NULL)
{
printf("Cannot open file
");
exit(1);
}
if ((Align_Ptr=fopen("c:\nasa\Align.dat","r+t")) == NULL)
{
printf("Cannot open file
");
exit(1);
}
if ((Input_Ptr=fopen("c:\nasa\Input.dat","r+t")) == NULL)
{
printf("Cannot open file
");
exit(1);
}
Read_system_data();

/* Print header file for data */
fprintf(Data_Ptr,"|TimeTag| Sig Lev | V_Error | U_Error |
|ActB| DiskBC | Istep8 | ActA| DiskAC | IStepA|

void Read_system_data(void)
{
rewind (System_Ptr);
for (Chan = 0; Chan <= 2; Chan++)
{
 fscanf(System_Ptr, "%ld %ld", &Disk_Position[1],&Disk_Position[0]);
 Starting_Position[0][Chan] = Disk_Position[0];
 Starting_Position[1][Chan] = Disk_Position[1];
}
for (Chan = 0; Chan <= 2; Chan++)
{
 fscanf(System_Ptr, "%ld %ld", &Channel_data);
 Selected_Channels[Chan][1] = Channel_data;
}

void Read_Current_System_data(void)
{
rewind (System_Ptr);
for (Chan = 0; Chan <= 2; Chan++)
{
 fscanf(System_Ptr, "%ld %ld", &Disk_Position[1],&Disk_Position[0]);
 Current_Position[0][Chan] = Disk_Position[0];
 Current_Position[1][Chan] = Disk_Position[1];
}

void Write_system_data(void)
{
/* Move file point to beginning of file */
rewind(System_Ptr);
for (Chan = 1; Chan <= 3; Chan++)
{
  switch (Chan)
  {
  case 1:
  Chan_Address = Motor_Pair1_Address; break;
  }
  case 2:
  Chan_Address = Motor_Pair2_Address; break;
  }
  case 3:
  Chan_Address = Motor_Pair3_Address; break;
  }
Read_Actual_Pos(Chan_Address,Disk_Position);
Disk_Position[0] = Starting_Position[0] [Chan-1] - Disk_Position[0];
fprintf(System_Ptr,"%ld %ld\n",Disk_Position[1],Disk_Position[0]);
/* Write out data of active channels */
for (Chan = 0; Chan <= 2; Chan++)
  fprintf(System_Ptr,"%i\n"Selected_Channels[Chan][1]);
} /* End of function Write_system_data */
/* Reads in data for user changing of azimuth and elevation errors*/
void Read_AzEt_data(void)
{
  /* Move file point to beginning of file */
  rewind(AzEl_Ptr);
  for (Chan = 0; Chan <= 2; Chan++)
  { fscanf(AzEl_Ptr,"%lf\n",Azimuth[Chan]);
    fscanf(AzEl_Ptr,"%lf\n",Elevation[Chan]);
  }
  } /* End of Read_AzEl_data function*/

/* Writes out new azimuth and elevation data */
void Write_AzEt_data(void)
{
  /* Move file point to beginning of file */
  rewind(AzEl_Ptr);
  for (Chan = 0; Chan <= 2; Chan++)
  { fprintf(AzEl_Ptr,"%lf\n",Azimuth[Chan]);
    fprintf(AzEl_Ptr,"%lf\n",Elevation[Chan]);
  }
  } /* End of Write_AzEl_data function */

/* Writes out new azimuth and elevation data */
void Write_AzEl_Track_Data(void)
{
  int loop = 0;
  double dummy = 0;
  int zero;
  int switch_channel[3] = (1,0,2);
  /* Move file point to beginning of file */
```c
    rewind (Track_AzEl_Ptr);
    for (Chan = 0; Chan <= 2; Chan++)
    {
        if (Selected_Channels[(switch_channel[Chan])][1] != 0)
        {
            fprintf(Track_AzEl_Ptr,"%lf
",Azimuth[(switch_channel[Chan])]);
            fprintf(Track_AzEl_Ptr,"%lf
",Elevation[(switch_channel[Chan])]);
            loop++;
        }
        if (loop < 3)
        {
            for (zero = 1; zero <= (3 - loop); zero++)
            {
                fprintf(Track_AzEl_Ptr,"%lf
",dummy);
                fprintf(Track_AzEl_Ptr,"%lf
",dummy);
            }
        }
    } /* End of Write_AzEl_data function */

    /* Writes out new Alignment Data to file */
    void Write_Align_data(void)
    {
        /* Move file point to beginning of file */
        rewind (Align_Ptr);
        for (Chan = 0; Chan <= 2; Chan++)
        {
            fprintf(Align_Ptr," %ld
",Align_Position[0][Chan]);
            fprintf(Align_Ptr," %ld
",Align_Position[1][Chan]);
        }
    }

    /* Reads in alignment data for all channels */
    void Read_Align_data(void)
    {
        /* Move file point to beginning of file */
        rewind (Align_Ptr);
        for (Chan = 0; Chan <= 2; Chan++)
        {
            fscanf(Align_Ptr,"%ld
",&Align_Position[0][Chan]);
            fscanf(Align_Ptr,"%ld
",&Align_Position[1][Chan]);
        }
    }

    /* Create and open temporary file. */
    FILE *outfile;
    char tmp[120];
    int loop, line_count, Channels_To_Run;
    outfile = tmpfile();
    /* Get each line from input and write to output. */
    for (line_count = 1; line_count <= 12; line_count++)
    {
        fgets( tmp, 120, Input_Ptr);
        fputs( tmp, outfile );
        Channels_To_Run = 0;
        for (loop = 0; loop <= 2; loop++)
        {
            if (Selected_Channels[loop][1] != 0)
            {
                Channels_To_Run = Channels_To_Run + 1;
            }
```
296         rewind (Input_Ptr);
297         rewind (outfile);
298         for (line_count = 1; line_count <= 12; line_count++)
299         {
300             fgets( tmp, 120, outfile);
301             /* Add 48 to the number of channels to indicate the ascii char. */
302             if (line_count == 12) tmp[0] = Channels_To_Run + 48;
303             fputs( tmp, Input_Ptr );
304         }
305     }
306     }
/* Display routine is used to provide the menu for manual control of the motors */
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <ctype.h>
#include <time.h>
#include <graph.h>
#include <dos.h>
#include <math.h>
#define BLK_SZ 8
#define Pi 3.141592654

/* Declare the following variables external these var. are declared in motor.c */
extern unsigned Motor_Pair1_Address; /*Address to first pair of motor controllers*/
extern unsigned Motor_Pair2_Address; /*Address to first pair of motor controllers*/
extern unsigned Motor_Pair3_Address; /*Address to first pair of motor controllers*/
extern unsigned Port_A_Address; /*Address of port A on 8255*/
extern unsigned Port_B_Address; /*Address of port B on 8255*/
extern unsigned Port_C_Address; /*Address of port C on 8255*/

long int timelasped;
int delay;
int Reset_to_zero = 1; /* These 2 variables will be used global for setting motor position */
int Set_to_actual = 0;
struct rccoord pos;

void Initialize_Controllers (void);

void Initialize_Controllers (void)
{
  int Ch_num,Address_of_Channel;
  int long zero[2] = (0,0);
  int Accel_data[2];

  /* Declare the digital filer parameters */
  /* Format of the Filter pole array are as follows */
  Filter_Pole[0] Bits 16 - 9 Channel 1 Arm motor pole
  Filter_Pole[0] Bits 8 - 1 Channel 1 Drum motor pole
  Filter_Pole[1] Bits 16 - 9 Channel 2 Arm motor pole
  Filter_Pole[1] Bits 8 - 1 Channel 2 Drum motor pole
  Filter_Pole[2] Bits 8 - 1 Channel 3 Drum motor pole */
  int Filter_Pole[3] = (0xe6e6,0xe6e6,0xe6e6);

  /* Format of the Filter zero array are as follows */
  Filter_Zero[0] Bits 16 - 9 Channel 1 Arm motor Zero
  Filter_Zero[0] Bits 8 - 1 Channel 1 Drum motor Zero
  Filter_Zero[1] Bits 8 - 1 Channel 2 Drum motor Zero
  int Filter_Zero[3] = (0xdbdb,0xdbdb,0xdbdb);

  /* Format of the Filter Gain array are as follows */
  Filter_Gain[0] Bits 16 - 9 Channel 1 Arm motor Gain
  Filter_Gain[0] Bits 8 - 1 Channel 1 Drum motor Gain
  Filter_Gain[1] Bits 8 - 1 Channel 2 Drum motor Gain
  int Filter_Gain[3] = (0x9090,0x9090,0x9090);

  /* Format of the Sampler Timer array are as follows */
  Sample_Time[0] Bits 16 - 9 Channel 1 Arm motor timer
  Sample_Time[0] Bits 8 - 1 Channel 1 Drum motor timer
74    Sample_Time[1] Bits 16 - 9 Channel 2 Arm motor timer
75    Sample_Time[1] Bits 8 - 1 Channel 2 Drum motor timer
77    Sample_Time[2] Bits 8 - 1 Channel 3 Drum motor timer */
78
79    int Sample_Time[3] = {0x0F0, 0x0F0, 0x0F0};
80
81    /* Format of the Max_velocity array are as follows
82    Max_Velocity[0] Bits 16 - 9 Channel 1 Arm motor max velocity
83    Max_Velocity[0] Bits 8 - 1 Channel 1 Drum motor max velocity
84    Max_Velocity[1] Bits 16 - 9 Channel 2 Arm motor max velocity
85    Max_Velocity[1] Bits 8 - 1 Channel 2 Drum motor max velocity
87    Max_Velocity[2] Bits 8 - 1 Channel 3 Drum motor max velocity */
88
89    int Max_Velocity[3] = {0x0202, 0x0202, 0x0303};
90
91    /* Format of the Max_Accel array are as follows
92    Max_Accel[0] Bits 16 - 9 Channel 1 Arm motor max Accel
93    Max_Accel[1] Bits 8 - 1 Channel 1 Drum motor max Accel
94    Max_Accel[2] Bits 16 - 9 Channel 2 Arm motor max Accel
95    Max_Accel[2] Bits 8 - 1 Channel 2 Drum motor max Accel
96    Max_Accel[3] Bits 16 - 9 Channel 3 Arm motor max Accel
97    Max_Accel[3] Bits 8 - 1 Channel 3 Drum motor max Accel */
98
99    int Max_Accel[6] = {0x1000, 0x1000, 0x1000, 0x1000, 0x1000};
100
101    for (Ch_num = 0; Ch_num < 3; Ch_num = Ch_num + 1)
102      {
103          /* Set address of motor based on loop counter */
104
105          if (Ch_num == 0)
106              Address_of_Channel = Motor_Pair1_Address;
107          else if (Ch_num == 1)
108              Address_of_Channel = Motor_Pair2_Address;
109          else if (Ch_num == 2)
110              Address_of_Channel = Motor_Pair3_Address;
111
112          /* Do a soft reset on the drum and motor controller of channel x */
113          Write_to_program_counter(Address_of_Channel, 0x0000);
114          Write_to_flag_register(Address_of_Channel, 0x0000);
115
116          /* Set the controller motor control to bipolar operation */
117          Write_to_flag_register(Address_of_Channel, 0x0202);
118          Write_to_flag_register(Address_of_Channel, 0x0303);
119          Write_to_flag_register(Address_of_Channel, 0x0505);
120
121          /* Initialize command position to 0 */
122          Write_command_pos(Address_of_Channel, zero);
123          Reset_Actual_Position(Address_of_Channel, zero);
124          Write_Final_pos(Address_of_Channel, zero);
125
126          /* Set up sample timer for the controller */
127          Write_to_sampler_timer(Address_of_Channel, Sample_Time[Ch_num]);
128          Write_Filter_Pole(Address_of_Channel, Filter_Pole[Ch_num]);
129          Write_Filter_Zero(Address_of_Channel, Filter_Zero[Ch_num]);
130          Write_Filter_Gain(Address_of_Channel, Filter_Gain[Ch_num]);
131
132          Accel_data[0] = Max_Accel[Ch_num];
Accel_data[1] = Max_Accel[Ch_num + 1];
/* Output the values of acceleration to the controllers */
Write_Max_Accel(Address_of_Channel,Accel_data);
 /* Output the velocity value to the controllers*/
Write_Max_Velocity(Address_of_Channel,Max_Velocity[Ch_num]);
 /* Put the motor controller into the control mode */
Write_to_program_counter(Address_of_Channel,0x0303);
) /* End of for loop Ch_num < 3 */
 } /* End of function Call_Routine */

int Set_Motor_Position(unsigned Address_of_Reset,int Reset)
{
long int Position[2] = (0,0);
/* If reset pin is equal to 1 reset all position counters to zero */
if (Reset == 1)
Reset_Actual_Position(Address_of_Reset,Position);
else
 /* Read actual position of motors */
Read_Actual_Pos(Address_of_Reset,Position);
 /* Put motor controller pair into idle mode */
Write_to_program_counter(Address_of_Reset,0x0101);
for (timelasped = 5000; timelasped > 0; timelasped = timelasped - 1);
 /* Initialize command position to actual position */
Write_command_pos(Address_of_Reset,Position);
 /* Initialize final position to actual position */
Write_Finat_pos(Address_of_Reset,Position);
 /* Put motor controller pair back into control mode */
Write_to_program_counter(Address_of_Reset,0x0303);
for (timetasped = 10000; timetasped > 0; timetasped = timetasped - 1);
)

/* Function Reached_Commanded_Pos checks to see if the motors have reached
 their commanded position */
int Reached_Commanded_Pos(unsigned Mot_Add,long int *Target_value,int Channel,int Difference)
{
int Target_Reached = 1;
long int Pres_Pos[2];
long int Delta_pos1,Delta_pos0;
int delay;
int Check_count = 0;
while ((Target_Reached == 1) && (Check_count < 6))
{
Read_Actual_Pos(Mot_Add,Pres_Pos);
/* Compute deltas between present and target position */
Delta_pos1 = labs(Target_value[1] - Pres_Pos[1]);
Delta_pos0 = labs(Target_value[0] - Pres_Pos[0]);
if ((Delta_pos1 > Difference) || (Delta_pos0 > Difference))
{ Target_Reached = 0; Check_count = 0; }
else
{ Target_Reached = 1; Check_count ++;
for (delay = 0; delay < 5000; delay++);
}
return Target_Reached;
}
/* Manual.C: Main executive for manual control of the dc servo motors */

#include <stdio.h>
#include <conio.h>
#include <ctype.h>
#include <time.h>
#include <graph.h>
#include <dos.h>

extern FILE *Manual_test_ptr; /* Set up pointer for file to output error data */
extern FILE *Align_Ptr;
extern FILE *System_Ptr;
extern FILE *AzEl_Ptr;
extern FILE *Track_AzEl_Ptr;
extern int C_Port_Data = 0;
extern unsigned Port_C_Address;
extern Set_to_actual;

#define Motor_Pair1_Address
#define Motor_Pair2_Address
#define Motor_Pair3_Address

#define Pwr_24Volts_On
#define Pwr_24Volts_Off

tong int Offset_Position[3][4];
int mccle = 0;
extern unsigned Control_word_address; /* Address to which 8255 command word is being set */
extern unsigned Control_word; /* Sets all ports A,B to inputs & C to outputs */

extern struct Channel_data {
    unsigned Time_tag;
    double V_error;
    double U_error;
    double Signal_strength;
    signed char Transmit_Data;
} Channel[3], Test[3];

void main ()
{
    int Key_data = 0;
    int Primary_Menu = 0;

    /* Assign channel assignments to data array */
    Test[0].Transmit_Data = 0;
    Test[1].Transmit_Data = 1;
    Test[2].Transmit_Data = 2;

    _setvideomode(_HERCMONO); /* Set mode of graphics to Hercules monitor */

    /* Open all files as required mode = 0 indicates manual mode */
    Open_files(mode);

    /* Output control word to 8255 */
    outp(Control_word_address, Control_word);

    /* Initialize motor controllers */
    InitializeControllers();

    /* Turn on 24 Volt power supply */
    C_Port_Data = C_Port_Data | Pwr_24Volts_On;
    outp(Port_C_Address, C_Port_Data);

    while(1)
    {
    
    }
if (Primary_Menu == 1)
    Key_data = Display_Main_Menu();
else
    Key_data = Display_Secundary_Menu();

Process_Main_Option(Key_data); /*Process key that was pressed*/

CLOSemdown(void)
{
    int ch;
    ConCloseAll(); /* Close all serial ports */

    /* Turn off 24 Volt power supply */
    C_Port_Data = C_Port_Data & Pwr_24Volts_Off;
    outp(Port_C_Address,C_Port_Data);

    /* Set all position registers to actual position to stop motors */
    Set_Motor_Position(Motor_Pair1_Address,Set_to_actual);
    Set_Motor_Position(Motor_Pair2_Address,Set_to_actual);
    Set_Motor_Position(Motor_Pair3_Address,Set_to_actual);

    Write_system_data(); /* Read position of motors and store data */
    fclose(Manual_test_ptr); /* Close file when done */
    fclose(Align_Ptr);
    fclose(System_Ptr);
    fclose(AzEl_Ptr);
    fclose(Track_AzEl_Ptr);

    exit (1);
}
```c
#include <stdio.h>
#include <conio.h>
#include <ctype.h>
#include <time.h>
#include <graph.h>
#include <dos.h>
#include <time.h>
#include <float.h>
#include <stdlib.h>

extern long int Starting_Position[2][3];
extern double Azimuth[3], Elevation[3];

/* Selected Channel array is defined as follows first element indicates
  electronic & software channel and second element denotes optical channel */
int Selected_Channels[3][2] = {{1,2}, {2,1}, {3,3}};
int Channel_to_Run[4] = {0, 0, 0, 0};
int Number_of_Channels_To_Run = 0;

char key_buffer[80];
char buffer[80];

int exit_loop, loop_test;
int option_key, chan;
unsigned statusword;
double temp_az[3], temp_et[3];

void Select_Azel_Changes (void)
{
  exit_loop = 0;
  Read_AzEl_date();
  /* Store values read in program into temporary array */
  for (chan = 0; chan <= 2; chan++)
  {
    temp_az[chan] = Azimuth[chan];
    temp_et[chan] = Elevation[chan];
  }
  while (exit_loop != 1)
  {
    _clearscreen(_GCLEARSCREEN);
    _settextposition(1,1); /*Position the cursor at position 1,1*/
    printf(" 
    ** Azimuth and Elevation options menu ** \n\n");
    printf("Channel 1 Azimuth = %8.6f,(Deg.) Channel 1 Elevation = %8.6f,(Deg.)\n", temp_az[1], temp_et[1]);
    printf("Channel 2 Azimuth = %8.6f,(Deg.) Channel 2 Elevation = %8.6f,(Deg.)\n", temp_az[0], temp_et[0]);
    printf("Channel 3 Azimuth = %8.6f,(Deg.) Channel 3 Elevation = %8.6f,(Deg.)\n", temp_az[2], temp_et[2]);
    printf("1. Change Azimuth and Elevation for Channel 1 \n\n");
    printf("2. Change Azimuth and Elevation for Channel 2 \n\n");
    printf("3. Change Azimuth and Elevation for Channel 3 \n\n");
    printf("4. Continue without saving changes \n\n");
    printf("5. Save changes and continue \n\n");
    printf("\nSelect the one of the above options\n");
    option_key = -1; /* Reset value key value */
    do
    {
      gets(key_buffer);
      option_key = atoi(key_buffer);
    } while ((option_key <= 0) || (option_key > 5));
```
switch (option_key)
{
    case 1:
        (option_key = 2; break;)
        option_key = 1;
    }
/* Read in data from the keyboard that is an integer format */
if ((option_key == 1) || (option_key == 2) || (option_key == 3))
{
    printf("Enter new value for azimuth in Degrees (Range +/-11)\n");
    gets(key_buffer);
    temp_az[option_key-1] = atof(key_buffer);
    printf("Enter new value for elevation in Degrees (Range +/-11)\n");
    gets(key_buffer);
    temp_el[option_key-1] = atof(key_buffer);
}
else if (option_key == 4)
{
    exit_loop = 1; /* Set flag to exit main loop */
}
else if (option_key == 5)
{
    exit_loop = 1;
    /* Store changes back into array*/
    for (chan = 0; chan <= 2; chan++)
    {
        Azimuth[chan] = temp_az[chan];
        Elevation[chan] = temp_el[chan];
    }
    Write_AzEl_data();
    Write_AzEl_Track_Data();
}/* End of while exit_loop != 1*/
} /* End of function Select_AzEl_Changes */

/* Function Move_to_New_AZEL used to move to new azel location */
void Move_to_New_AZEL(channel,Motor_Address)
{ double Delta[4], Alpha[4];
    long int _StepA[4], _StepB[4];
    long int Command_Pos[2];
    int a,b, Number_Of_Solutions,Loop,Solution;
    long int Present_Delta = 10000000;
    double New_Azimuth = 12.0;
    double New_Elevation = 12.0;
    _clearscreen(_GCLEARSCREEN);
    _settextposition(1,1); /*Position the cursor at position 1,1*/
    printf(" ** Azimuth and Elevation Channel Move ** \n\n");
    printf("Enter new azimuth for move in Degrees \n");
    while ((New_Azimuth > 11.0) || (New_Azimuth < -11.0))
    {
        gets(key_buffer);
        New_Azimuth = atof(key_buffer);
        if ((New_Azimuth > 11.0) || (New_Azimuth < -11.0))
            printf("Out of range enter new value!! \n\n");
    }
    printf("Enter new elevation for move in Degrees\n");
    while ((New_Elevation > 11.0) || (New_Elevation < -11.0))
    {
        gets(key_buffer);
        New_Elevation = atof(key_buffer);
        if ((New_Elevation > 11.0) || (New_Elevation < -11.0))
            printf("Out of range enter new value!! \n\n");
        }
New_Azimuth = New_Azimuth/57.29577951;
New_Elevation = New_Elevation/57.29577951;

AzEl2AB(New_Azimuth, New_Elevation, I_StepA, I_StepB, Delta,
Alpha,&Number_of_Solutions);

printf("\n Select one of the following alpha & delta pairs to move the channel to \\

");

for (loop=0; loop <= Number_of_Solutions; loop++)
{
  printf("%d.) Delta = %.6f (Deg.) Alpha = %.6f (Deg.)\n", loop+1,Delta[loop],Alpha[loop]);

}

printf("%d.) Exit option without moving motors\n",Number_of_Solutions+2);

do
{
  gets(key_buffer);
  Solution = atoi(key_buffer);
}
while ((Solution < 1) || (Solution > Number_of_Solutions+2));

if (Solution == (Number_of_Solutions+2))
{
  /* Set up the commanded position for the DiskB Motor */
  Command_Pos[0] = Starting_Position[0][channel-1] - I_StepB[Sotution-1];

  /* Set up the commanded position for the DiskA Motor */
  Command_Pos[1] = (-1 * Starting_Position[1][channel-1]) + I_StepA[Solution-1];

  /* Output new position to the motor controller */
  Write_Final_pos(Motor_Address,Command_Pos);

  /* Start trapezoidal move */
  Write_to_flag_registor (Motor_Address,0x0808);
}

void ScreenPrompt(void)
{
  char Key_number;
  char tmpbuf[128];
  cursorscreen ( GCLEARSCREEN );
  _settextposition(10,1); /*Position the cursor at position 1,1*/

  printf("Welcome to the Multiaccess terminal tracking program\n");

  /* Display DOS-style date and time. */
  strftime( tmpbuf );
  printf("\n\nTime: %s\n", tmpbuf);
  strftime( tmpbuf );
  printf("Data: %s\n", tmpbuf);

  _settextposition(20,1); /*Position the cursor at position 1,1*/
  printf("Press 'e' to exit program at this time or return to continue\n");

  while ( !kbhit() )
  {
    if (getch() == 'e') Close_down();
  }

  void Select_Channels_to_Run(void)
  {
    char Key_number;
    char tmpbuf[128];
    int Menu_option = 0;
    int Channel_select = 0;
    ...
Read_AzEl_data();

while (Menu_option != 4)
{
  _clearscreen(_GCLEARSCREEN);
  _settextposition(10,1); /*Position the cursor at position 1,1*/
  printf(" ** Channel selection menu **\n\n\n");
  printf("The following channels have been selected: ");
  if (Selected_Channels[1][1] != 0) printf("%i",Selected_Channels[1][1]);
  if (Selected_Channels[0][1] != 0) printf("%i",Selected_Channels[0][1]);
  if (Selected_Channels[2][1] != 0) printf("%i",Selected_Channels[2][1]);
  printf("\n\n\n");
  printf("1.) Add channel\n");
  printf("2.) Remove channel\n");
  printf("3.) Select all channels\n");
  printf("4.) Continue and save selections\n");
  do
  { gets( tmpbuf);
    Menu_option = atoi(tmpbuf);
  } while ((Menu_option <= 0) || (Menu_option > 4));

  if ((Menu_option == 1) || (Menu_option == 2))
  { printf("\nEnter channel number [1,2,3]\n");
    do
    { gets( tmpbuf);
      Channel_select = atoi(tmpbuf);
    } while ((Channel_select != 1) && (Channel_select != 2) && (Channel_select != 3));

    if (Menu_option == 1)
    {
      if (Channel_select == 1) Selected_Channels[1][1] = 1;
      else if (Channel_select == 2) Selected_Channels[0][1] = 2;
      else if (Channel_select == 3) Selected_Channels[2][1] = 3;
    }
    else if (Menu_option == 2)
    {
      if (Channel_select == 1) Selected_Channels[1][1] = 0;
      else if (Channel_select == 2) Selected_Channels[0][1] = 0;
      else if (Channel_select == 3) Selected_Channels[2][1] = 0;
    }
  }
  /* End of if (Menu_option == 1) || (Menu_option == 2) */

  if (Menu_option == 3)
  {
    Selected_Channels[0][1] = 2;
    Selected_Channels[1][1] = 1;
    Selected_Channels[2][1] = 3;
  }
  /* End of while menu option != 4 */
/* Compute the number of channels selected */

Number_of_Channels_To_Run = 0;

for (loop = 0; loop <= 2; loop++)
{
  if (Selected_Channels[loop][1] != 0)
  {
    Number_of_Channels_To_Run = Number_of_Channels_To_Run + 1;
    Channel_to_Run[Number_of_Channels_To_Run] = loop + 1;
  }
}

if((Selected_Channels[0][1] != 0) && (Selected_Channels[1][1] != 0))
{
  Channel_to_Run[1] = 2;
  Channel_to_Run[2] = 1;
}

Write_system_data();
Write_AzEl_data();
Write_AzEl_Track_data();
Change_Input_Data();

} /* End of function select channels */

void Check_System_Power(void)
{
  int Power_on24,Power_on5;
  Power_on5 = Check_for_5Volts();
  if (Power_on5 != 1)
  {
    printf("\n Turn +/-5 Volt power supply on");
    Wait_for_key_Press();
  }
  Power_on24 = Check_for_24Volts();
  if (Power_on24 != 1)
  {
    printf("\n Turn +24 Volt power supply on");
    Wait_for_key_Press();
  }
}

void Check_Switch_Status()
{
  int Channel_Number,Print_output,Switch_results,Switch_activated;
  Print_output = 0;
  Switch_results = 0;
  Switch_activated = 0;
  for (Channel_Number = 1; Channel_Number <= 3; Channel_Number++)
  {
    Switch_activated = Run_Switch_Bit(Channel_Number,Print_output);
    if (Switch_activated == 1) Switch_results = 1;
  }
  if (Switch_results == 1)
    Wait_for_key_Press();
}

void Check_Serial_Links(int Data_Port)
{
  int Error,loop,l_channel;
  for (loop = 1; loop <= Number_of_Channels_To_Run; loop ++)
  {
    l_channel = Channel_to_Run[loop];
    Error = Test_Serial_Link(l_channel,Data_Port);
  }
  if (Error == 0) printf("\n Serial link communications good\n");
  ComCloseAll();
370 }


*/
The following include statements are required to allow
program to link up with Turbo C++ libraries */

#include <cport.h>
#include <bios.h>
#include <conio.h>
#include <dos.h>
#include <ctype.h>
#include <stdio.h>
#include <math.h>

#define BLK_SZ 9
#define Number_of_samples 24

FILE *outfile[3];

int Break_handler(void);

/* Start main program here at this point */
main(void)
{
/* Declarations of these variables will cause them to be stored in CPU
Registers */
    int rv,out,status,Channel_number;
    int i, Channel=0;
    char output[80];
    unsigned int input_buffer[1024];
    unsigned int rs;
    int Elevation_error,Azimuth_error,Total_signal_strength;
    unsigned int Time_tag[3], Old_Time_tag[3];
    char *str = "Hello world";
    int Frame_count[3],Process_data_flag;
    int Relative_flag='N';
    char Mode_flag,File_flag,
    log=*N', a;
    int Good_data,Length_of_Message;
    int delay_time[10] = (0,5,4,4,3,3,2,2,1,0);
    double Elevation_average,Azimuth_average,Signal_average, Signal_offset[3];
    double Azimuth_deviation,Elevation_deviation,Signal_deviation;
    double Azimuth_deviation_square,Elevation_deviation_square,Signal_deviation_square;
    double Elevation,Azimuth,Signal_strength, last_signal[3];
    double Azimuth_total[3];
    double Azimuth_total_square[3];
    double Elevation_total[3];
    double Elevation_total_square[3];
    double Signal_strength_total[3];
    double Signal_strength_total_square[3];

    /* Set variables equal to zero */
    for (i=0;i<3;i++)
    { Azimuth_total[i] = 0.0;
    Azimuth_total_square[i] = 0.0;
    Elevation_total[i] = 0.0;
    Elevation_total_square[i] = 0.0;
    Signal_strength_total[i] = 0.0;
    Signal_strength_total_square[i] = 0.0;
    Time_tag[i]=0;
    Old_Time_tag[i]=0;
    }
    Frame_count[0] =0;
    Frame_count[1] =0;
    Frame_count[2] =0;
    Process_data_flag = 0;
    /* Set up handler for when ctrl break key is pressed */
    /* ctrl_brk(break_handler);*/
    /* Initialize RS-232 port */
    /* rv = 2 bad 'com' parameter, rv = 3 no uart chip detected */
    /* rv = 4 receive queue allocation error, rv = 4 transmit queue allocation error */
the following line will be used for the compound computer at 115kbaud */
rv = ComOpen(COM1, B19200, W8|S1|NONE, 1024, 512);
/* Set up COM1 port to be active */
ComActive(COM1);
/* Remove all data from the receive buffer */
ComFlushRx();
/* Remove all data from the transmit buffer */
ComFlushTx();
/* User interaction */
printf("Do you what to run program in continous mode? Press 'y'");
printf("for continous mode or any other key for manual mode");
Mode_flag = toupper(getch());
printf("y
Do you what to log the data to a file? Press 'y'");
File_flag = toupper(getch());
clrscr();
if (File_flag=='Y') {
  outfile[0]=fopen("Channel0.raw","w+");
  outfile[1]=fopen("Channel1.raw","w+");
  outfile[2]=fopen("Channel2.raw","w+");
  fprintf(outfile[0],"Raw El.	Raw Az.	Raw Sig.
Az	El	Signal
");
  fprintf(outfile[1],"Raw El.	Raw Az.	Raw Sig.	Az	El	Signal
");
  fprintf(outfile[2],"Raw El.	Raw Az.	Raw Sig.	Az	El	Signal
");
  log='L';
}
ComFlushRx();
while(1)
{
  if (kbhit())
    a=getch();
  if (a == 'e')
    ComCloseAll();
  if (File_flag=='Y')
    close(outfile[0]);
  close(outfile[1]);
  close(outfile[2]);
  exit(0);
}
if (a=='d')
  Process_data_flag = 1; /* If 'd' is hit process data */
log = 'L';
if ((a=='L'))
  log='L';
if ((a=='S'))
  log='S';
if ((a=='R'))
  if (Relative_flag=='Y')
    Relative_flag='N';
  else
    Relative_flag='Y';
if ((a=='Z'))
  if (Relative_flag=='Y')
    (Signal_offset[0]=last_signal[0];
    Signal_offset[1]=last_signal[1];
    Signal_offset[2]=last_signal[2];
  )
  if (Relative_flag=='Y')
    (Signal_offset[0]=0;
    Signal_offset[1]=0;
    Signal_offset[2]=0;
  )
if (Mode_flag == 'Y')
{
  Process_data_flag = 1;
}
else
{
  (kbhit())
  if (a == 'd')
    Process_data_flag = 1;
  log = 'L';
}
if ((a== 'L') || (a== 'l')) log = 'L';
if ((a== 's') || (a== 'S'))
  log = 'S';
} /* End of if mode_flag = Y */
Good_data = 0;
while (Good_data != 1)
{
  /* Wait for data */
  while (ComLenRx() == 0);
  Length_of_Message = ComLenRx();
  /* Test to see if message is longer than 9 words */
  if (Length_of_Message > 9)
    Length_of_Message = (Length_of_Message / 9) * 9;
  /* Delay program long enough to get rest of data */
  delay(delay_time[Length_of_Message]);
  if (ComLenRx() > BLK_SZ) ComFlushRx();
  else
  {
    rs = ComIn(input_buffer,BLK_SZ);
    if (input_buffer[0] >= 0) || (input_buffer[0] <= 2))
      Good_data = 1;
  else
    ComFlushRx();
  }
} /* Check to see if flag is set to process data */
if (Process_data_flag == 1)
{
  /* Increment the counter for the number of frame samples 
    that have been read in */
  Channel=(int)(input_buffer[0]) & 0x03;
  Frame_count[Channel]++;
  Time_tag[Channel] = (int)(input_buffer[2]) & OxFF;
  Time_tag[Channel] = Time_tag[Channel] << 8;
  Time_tag[Channel] = Time_tag[Channel] | ((int)(input_buffer[1]) & OxFF);
  Elevation_error = (int)(input_buffer[6]) & OxFF;
  Elevation_error = Elevation_error << 8;
  Elevation_error = Elevation_error | ((int)(input_buffer[3]) & OxFF);
  Azimuth_error = (int)(input_buffer[6]) & OxFF;
  Azimuth_error = Azimuth_error << 8;
  Azimuth_error = Azimuth_error | ((int)(input_buffer[5]) & OxFF);
  Azimuth_error = Azimuth_error;
  Total_signal_strength = (int)(input_buffer[8]) & OxFF;
  Total_signal_strength = Total_signal_strength << 8;
  Total_signal_strength = Total_signal_strength | ((int)(input_buffer[7]) & OxFF);
  /*Change integer value to a floating point value for normalizing*/
  Signal_strength = Total_signal_strength;
  if (Signal_strength <= 1e-4)
Signal_strength = 0.0001; /* Set signal_strength to small number if equal 0.0 */

/* Normalize Azimuth compute total azimuth error for n samples as well as the sum of the squares */
Azimuth = Azimuth_error / Signal_strength;
Azimuth_total[Channel] += Azimuth;
Azimuth_total_square[Channel] += Azimuth * Azimuth;

/* Normalize Elevation compute total azimuth error for n samples as well as the sum of the squares */
Elevation = Elevation_error / Signal_strength;
Elevation_total[Channel] += Elevation;
Elevation_total_square[Channel] += Elevation * Elevation;

/* Scale signal strength to a DC value from a 12 bit D/A with a +/- 10V input range */
Signal_strength = Signal_strength/204.7;
last_signal[Channel]=Signal_strength;

if (File_flag=='Y' && (log=='L' || log=='l') && Mode_flag == 'Y')
{ fprintf(outfile[Channel],"%d\t%f\tSf\n",Elevation,
Azimuth,
Signal_strength);
}

/* Compute total signal strength for x samples as well as the sum of the squares */
Signal_strength_total[Channel] += Signal_strength;
Signal_strength_total_square[Channel] += Signal_strength * Signal_strength;

if (Frame_count[Channel] == Number_of_samples)
{ Azimuth_average = Azimuth_total[Channel]/Number_of_samples;

Azimuth_deviation_square = (Azimuth_total_square[Channel] / Number_of_samples) - (Azimuth_average * Azimuth_average);
/* test to see if we have a negative number */
if (Azimuth_deviation_square < 0.0) Azimuth_deviation_square = 0.0;
Azimuth_deviation = sqrt(Azimuth_deviation_square);

Elevation_average = Elevation_total[Channel]/Number_of_samples;
Elevation_deviation_square = (Elevation_total_square[Channel] / Number_of_samples) - (Elevation_average * Elevation_average);
/* test to see if we have a negative number */
if (Elevation_deviation_square < 0.0) Elevation_deviation_square = 0.0;
Elevation_deviation = sqrt(Elevation_deviation_square);

Signal_average = Signal_strength_total[Channel]/Number_of_samples;
Signal_deviation_square = (Signal_strength_total_square[Channel] / Number_of_samples) - (Signal_average * Signal_average);
/* test to see if we have a negative number */
if (Signal_deviation_square < 0.0) Signal_deviation_square = 0.0;
Signal_deviation = sqrt(Signal_deviation_square);

else if (Channel == 1) Channel_number = 0;
else Channel_number = 2;

gotoxy(1,(5*Channel_number) + 1);
cleol();

printf("Channel %d\n",Channel_number + 1);

cleol();

printf("Time = %-1.5f  \n u = %-1.5f \n Total = %-2.3f \n", 
  Time_tag[Channel],Elevation_average,Azimuth_average, 
  Signal_average);

if (Relative_flag=='Y') 
  printf("R\n"); 
  cleol();

printf("S.Dev\n v = %-1.5f \n u = %-1.5f \n Total = %-2.4f\n", 
  Elevation_deviation,Azimuth_deviation, 
  Signal_deviation);

cleol();

gotoxy(1,22);

cleol();

if (File_flag=='Y' && (log=='L' || log=='T')) 
  printf("Press 'S' to stop logging. ");
else 
  if (File_flag=='Y') 
    printf("Press 'L' to begin logging. ");

if (Relative_flag=='Y') 
  printf("Press 'Z' to zero signal strength ");
else 
  printf("Press 'R' to toggle relative mode ");

if (Mode_flag != 'Y') 
  printf("\nPress 'd' to sample data");

fprintf(outfile[Channel],"Time = %d\n u = %-1.5f \n Total = %-2.3f\n", 
  Time_tag[Channel],Elevation_average,Azimuth_average, 
  Signal_average);

*/Reset variables used to total average amd sum of squares */
Azimuth_total[Channel] = 0.0;
Azimuth_total_square[Channel] = 0.0;
Elevation_total[Channel] = 0.0;
Elevation_total_square[Channel] = 0.0;
Signal_strength_total[Channel] = 0.0;
Signal_strength_total_square[Channel] = 0.0;

/*/Reset frame counter*/
Frame_count[Channel] = 0;

/*/ Reset process_data_flag */
Process_data_flag = 0;

*/ End of if Frame_count = Number_of_samples */

*/ End of if process_flag = 1 */

*/ End of while (1) else */
/* Display routine is used to provide the menu for manual control of the motors */
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <ctype.h>
#include <time.h>
#include <graph.h>
#include <dos.h>
#include <time.h>
#include <math.h>
#define BLK_SZ 8
#define Pi 5.141592654

/* Declare the following variables external these var. are declared in motor.c */
extern unsigned Motor_Pair1_Address; /*Address to first pair of motor controllers */
extern unsigned Motor_Pair2_Address; /*Address to first pair of motor controllers */
extern unsigned Motor_Pair3_Address; /*Address to first pair of motor controllers */
extern unsigned Port_A_Address; /*Address of port A on 8255 */
extern unsigned Port_B_Address; /*Address of port B on 8255 */
extern unsigned Port_C_Address; /*Address of port C on 8255 */

extern long int Align_Position[2][3]; /* Data containing alignment positions */
extern long int Starting_Position[2][3]; /* Array containing starting position */
extern FILE *Manuat_test_ptr;
long int timelasped;

/* The following variables are used for open loop testing */
double Step_A = 0.0;
double Step_B = 1.0;
long int DiskAstep,DiskBstep;
int Number_of_Frames = 1600;
int Range = 400;
long int Motor_Pos[2][3000];
long int Act_post2[2];
tong int Zero_Position[2] = {0,0);
int delay;
int Exit_requested;
int long Temp_actual,Temp_new;
extern int Reset_to_zero; /* These 2 variables will be used global for setting motor position */
extern int Set_to_actual;

struct rccoord pos;
int Sync_Pulse_Disable[3] = (0xBF,0x7F,0xEF);
int Sync_Pulse_Enable[3] = (0x40,0x80,0x10);

void Align_Channel(unsigned Address_of_Control,int Number_of_channel,int Stop_Flag);
void Compute_Open_Loop_Commands(unsigned Channel_Address,int Number_of_Motor,
double Alpha_arm);

/* Align Channel will be used to align the channels to a particular point */
void Align_Channel(unsigned Address_of_Control,int Number_of_channel, int Store_Flag)
(int Long Step_size = 2000000;
    int Index_step_size = 1;
    int DiskB_Motor_number = 0;
    int DiskA_Motor_number = 0;
    int DiskA_Align_switch = 0;
    int DiskB_Align_switch = 0;
    int All_Switches_Good = 0x30;
    int SwitchActivated;
    int IndexMark_Found;
    long int Delta = 250;
    long int alpha = -38;

    int Delta_limit = 3;
    long int DiskB_delta_pos = 0;
    long int DiskA_delta_pos = 0;
    long int New_Position[2] = (0,0);
    long int Actual_Position[2] = (0,0);
    int Aligned_Flag = 0;
    int Target_Reached;

    switch(Number_of_channel)
    {
        case 1:
            DiskA_Align_switch = 3;
            DiskB_Align_switch = 5;
            DiskB_Motor_number = 2;
            DiskA_Motor_number = 1;
            break;
        case 2:
            DiskA_Align_switch = 9;
            DiskB_Align_switch = 11;
            DiskB_Motor_number = 4;
            DiskA_Motor_number = 3;
            break;
        case 3:
            DiskA_Align_switch = 15;
            DiskB_Align_switch = 17;
            DiskB_Motor_number = 6;
            DiskA_Motor_number = 5;
            break;
    } /*End of switch case Channel_number */

    /* Read in actual position of motors at this time */
    Read_Actual_Pos(Address_of_Control,Actual_Position);
    Read_Actual_Pos(Address_of_Control,Actual_Position);
    printf("\n Disk B motor position = %li ",Actual_Position[0]);
    printf(" Disk A motor position = %li ",Actual_Position[1]);

    /* Set final position counters to actual position */
    New_Position[0] = Actual_Position[0];
    New_Position[1] = Actual_Position[1];

    /* Beginning of loop to align system */
    while (Aligned_Flag == 0)
/* Test for switch activation */
Switch_activated = Check_for_switch_Activation(DiskA_Align_switch);
printf("\nMoving disks to activate switch \%i",DiskA_Align_switch);
/* Move Drum and Arm gears together till switch SW3 has been hit */
while (Switch_activated != 1)
{
    /* Set up positions for the move command */
    New_Position[0] = New_Position[0] + Step_size;
    /* Add delta to arm motor position */
    /* Output new position to the motor controller */
    Write_Final_pos(Address_of_Control,NewPosition);
    /* Start trapezoidal move */
    Write_to_flag_registor(Address_of_Control,Ox0808);
    /* Delay reading of motor till it has a chance to move */
    for (timelasped = 5000; timelasped > 0; timelasped = timelasped - 1);
    /* Check to see if we have made it to new position or switch has been pressed */
    do
    { 
        Switch_activated = Check_for_switch_Activation(DiskA_Align_switch);
    }
    while (Switch_activated != 1);
    for (timelasped = 60000; timelasped > 0; timelasped = timelasped - 1);
    Read_Actual_Pos(Address_of_Control,Actual_Position);
    printf("\n Disk B motor position = \%li ",Actual_Position[0]);
    printf(" Disk A motor position = \%li ",Actual_Position[1]);
    /* End of while (Switch_data != Sw3_detect) */
    printf("\n Switch \%i limit hit ",DiskA_Align_switch);
    for (timelasped = 200000; timelasped > 0; timelasped = timelasped - 1);
    /* Set all position registers to actual position */
    Set_Motor_Position(Address_of_Control,Set_to_actual);
    printf("\n *** Start reversal of Disk A and Disk B gears till index mark on ***
 *** Disk A motor is found 
 ***
    /* Read position of arm and drum motor */
    Read_Actual_Pos(Address_of_Control,Actual_Position);
    printf("\nReverse direction of gear rotation till Disk A index mark is found\n");
    printf("\n DiskB Actual DiskB Command DiskA Actual DiskA Command Index\n");
    /* Check for index mark */
    /* Index_Mark_Found = Check_for_Index_Mark(DiskA_Motor_number); */
    Switch_activated = Check_for_switch_Activation(DiskA_Align_switch);
    /* Set delta limit for delta between actual and command position*/
    Delta_limit = 30;
    New_Position[0] = Actual_Position[0];
    New_Position[1] = Actual_Position[1];
    while (Switch_activated == 1)
    {
222 /* Add delta to arm motor position */
224
225 /* Output new position to the motor controller */
226 Write_command_pos(Address_of_Control,New_Position);
227
228 /* Decrease window between actual and command position once switch
229  is activated */
230 if ((Switch_activated == 0) && (Delta_limit > 15))
231   Delta_limit = Delta_limit - 1;
232
233 /* Check to see if the we have made it to new position or switch
234  has been pressed */
235 do
236
237 /* Read in data to look for index mark and switch activation*/
238 /* Index_Mark_Found = Check_for_Index_Mark(DiskA_Motor_number); */
239 Switch_activated = Check_for_Switch_Activation(DiskA_Align_switch);
240
241 /* Check to see if commended position has been met */
242 Target_Reached = Reached_Commanded_Pos(Address_of_Control,New_Position,
243 Number_of_channel,Delta_limit);
244
245 if ((Target_Reached == 0) && (Switch_activated == 1))
246 {
247   Read_Actual_Pos(Address_of_Control,Actual_Position);
249   Write_command_pos(Address_of_Control,New_Position);
250 }
251
252 Read_Actual_Pos(Address_of_Control,Actual_Position);
253
254 printf("%12ld %12ld %12ld %12ld %2d %2d \n", 
255 Actual_Position[0],New_Position[0], 
256 Actual_Position[1],New_Position[1], 
257 Index_Mark_Found,Switch_activated);
258
259 while ((Target_Reached == 0) && (Switch_activated == 1));
260
261 while ((Target_Reached == 0 ) && (Switch_activated == 1));
262
263 /* End of while Index_Mark_Found and
264  & Switch_activated == 1 */
265 printf("In Disk 'A' gear is aligned !!");
266
267 Read_Actual_Pos(Address_of_Control,Actual_Position);
268
269 /* Store position of disk A */
270 DiskA_Pos_at_Alignment = Actual_Position[1];
271
272 /* Hold Disk A gear steady and move Disk B gear till ****/
273 /* *** switch Sw5 is activated  ****/
274 printf("In Moving Disk B to activate switch %i",DiskB_Align_switch);
275
276 /* Read Disk A and Disk B motors current position */
277 Read_Actual_Pos(Address_of_Control,Actual_Position);
278
279 New_Position[0] = Actual_Position[0];
280 New_Position[1] = DiskA_Pos_at_Alignment;
281
282 /* Check for switch 5 activation */
283 Switch_activated = Check_for_Switch_Activation(DiskB_Align_switch);
284
285 while (Switch_activated == 1)
286 {
287 /* Set up positions for the move command */
288 /* Add delta to disk B motor position */
289 New_Position[0] = New_Position[0] + Step_size;
290 /* Add delta to disk A motor position */

New_Position[1] = DiskA_Pos_at_Alignment;

/* Output new position to the motor controller */
Write_Final_pos(Address_of_Control, New_Position);

/* Start trapezoidal move */
Write_to_flag_register(Address_of_Control, 0x0080);

/* Check to see if the we have made it to new position or switch has been pressed */
do
{
    Switch_activated = Check_for_switch_Activation(DiskB_Align_switch);
}
while (Switch_activated != 1);

Read_Actuat_Pos(Address_of_Control, Actual_Position);

printf("\n Disk B motor position = %li ", Actual_Position[0]);
printf(" Disk A motor position = %li ", Actual_Position[1]);

} /* End of while Switch_activated != 1 */

printf(" \nSwitch %i limit hit ", DiskB_Align_switch);

for (timelasted = 200000; timelasted > 0; timelasted = timelasted - 1);

Set_Motor_Position(Address_of_Control, Set_to_actual);

/*** Reverse direction of Disk B gears till index mark on Disk B motor ***/
/*** is found Disk A gear will be held steady ***/

printf(" \nReverse direction of gear rotation till Disk B index mark is found\n");

/* Read position of arm and drum motor */
Read_Actuat_Pos(Address_of_Control, Actual_Position);

New_Position[0] = Actual_Position[0];
New_Position[1] = DiskA_Pos_at_Alignment;

/* Test for index mark and switch activation */
/* Index_Mark_Found = Check_for_Index_Mark(DiskB_Motor_number); */
Switch_activated = Check_for_switch_Activation(DiskB_Align_switch);

/* Set delta limit for delta between actual and command position*/
Delta_limit = 30;

printf("\n DiskB Actual DiskB Command DiskA Actual DiskA Command Index/");
printf("\n Position Position Position Position Switch\n");

while (Switch_activated == 1)
{
    /* Add delta to drum motor position */
    New_Position[0] = New_Position[0] - Index_step_size;
    /* Add delta to arm motor position */
    New_Position[1] = DiskA_Pos_at_Alignment;

    /* Output new position to the motor controller */
    Write_command_pos(Address_of_Control, New_Position);

    /* Decrease window between actual and command position once switch is activated */
    if ((Switch_activated == 0) && (Delta_limit > 15))
    { 
        Delta_limit = Delta_limit - 1;
    }

    /* Check to see if the we have made it to new position or switch has been pressed */
    do
{
    /* Read in data to look for index mark */
    /* Index_Mark_Found = Check_for_Index_Mark(DiskB_Motor_number); */
Switch_activated = Check_for_switch_Activation(DiskB_Align_switch);

/* Check to see if commnaded position has been met */
Target_Reached = Reached_Commanded_Pos(Address_of_Control,New_Position,
Number_of_channel,Deltalimit);

/* Resend position command if delta becomes to large and
switch is not activated */
if ((Target_Reached == 0) && (Switch_activated == 1))
{
    Read_Actual_Pos(Address_of_Control,Actual_Position);
    New_Position[0] = Actual_Position[0];
    Write_command_pos(Address_of_Control,New_Position);
}

while ((Target_Reached == 0) && (Switch_activated == 1));

if ((Switch_activated == 0) && (Index_Mark_Found == 1))
{
    Read_Actual_Pos(Address_of_Control,Actual_Position);
    printf("%12ld %12ld %12ld %2d %2d \n",
Actual_Position[0],New_Position[0],
Actual_Position[1],New_Position[1],
Index_Mark_Found, Switch_activated);
}

while ((Target_Reached == 0) && (Switch_activated == 1));

/

New_Position[0] = ActuaL_Position[0];
DiskB_Pos_at_Alignmnt = Actual_Position[0];
New_Position[1] = DiskA_Pos_at_Alignment;
Write_command_pos(Address_of_Control,New_Position);
Aligned_Flag = 1;
/

/* End of while aligned = 0 */

delta_limit = 15;
/

******* Move arm back to home position *************/

/* Reset all position registers to zero */
Set_Motor_Position(Address_of_Control,Reset_to_zero);
New_Position[0] = -25000;
New_Position[1] = 0;

/* Initiate movement of motor to swing over arm */
Write_Final_pos(Address_of_Control,New_Position);
Write_to_flag_register (Address_of_Control,0x0808);
Target_Reached = 0;

while (Target_Reached != 1)
{
    /* Check to see if commnaded position has been met */
    Target_Reached = Reached_Commanded_Pos(Address_of_Control,New_Position,
Number_of_channel,Deltalimit);
}

/* Store data as required and move to home position */
if (Store_Flag == 1)
{
    New_Position[0] = -DiskB_Pos_at_Alignment;
    New_Position[1] = -DiskA_Pos_at_Alignment;
/* Read in data with present values for alignment */
```c
Read_Align_data();
Align_Position[0][Number_of_channel-1] = DiskB_Pos_at_ALIGNMENT;
Align_Position[1][Number_of_channel-1] = DiskA_Pos_at_ALIGNMENT;

/* Write out new data for alignment */
Write_Align_data(); /* Write out new alignment data */
else
(  Read_Align_data(); /* Read in data for alignment purposes */
  */ Compute position for Disk B motor for arm only move */
  New_Position[0] = Align_Position[0][Number_of_channel-1];
  New_Position[1] = Align_Position[1][Number_of_channel-1];
)

/* Initiate movement of motor */
Write_Final_pos(Address_of_Control,New_Position);
Write_to_flag_register(Address_of_Control,0x0080);
Target_Reached = 0;
while (Target_Reached != 1)
(  /* Check to see if commanded position has been met */
    Target_Reached = Reached_Commanded_Pos(Address_of_Control,New_Position,
                                         Number_of_channel,Deltat_limit);
    printf("Waiting for channel to reach home position\n");
    for (timelapsed = 10000; timelapsed > 0; timelapsed = timelapsed - 1);
  )
/* Wait for motors to stop running */
for (timelapsed = 60000; timelapsed > 0; timelapsed = timelapsed - 1);
/* Reset motor to 0,0 position */
Set_Motor_Position(Address_of_Control,Reset_to_zero);
/* Store new starting position after alignment */
Starting_Position[0][Number_of_channel-1] = 0;
Starting_Position[1][Number_of_channel-1] = 0;
)/* End of alignment function */

/* Function will be used to rotate platter through is paces */
int Exercise_Channel(unsigned Control_Address, int chan_number)
(  int long alpha_move = -40;
  int long delta_move = 100;
  int long New_Ps[2] = (0,0);
  int long Send_Ps[2] = (0,0);
  int long Actual_Ps[2] = (0,0);
  int switch number = 3;
  int count;
  int Activated_switch;
  Exit_requested = 0;
  while (Exit_requested == 0)
  (  for (count = 1; count <= 4 ; count = count + 1)
    (  switch (count)
      (  case 1:
          (  if (chan_number == 1) switch_number = 3;
          else if (chan_number == 2) switch_number = 9;
          else if (chan_number == 3) switch_number = 15;
          alpha_move = 0;
          delta_move = +500;
          break;
```
case 2:
    
    if (chan_number == 1) switch_number = 4;
    else if (chan_number == 2) switch_number = 10;
    else if (chan_number == 3) switch_number = 16;
    
    alpha_move = 0;
    delta_move = -500;
    break;

}  

case 3:
    
    if (chan_number == 1) switch_number = 5;
    else if (chan_number == 2) switch_number = 11;
    else if (chan_number == 3) switch_number = 17;
    
    alpha_move = -100;
    delta_move = 40;
    break;

}  

case 4:
    
    if (chan_number == 1) switch_number = 6;
    else if (chan_number == 2) switch_number = 12;
    else if (chan_number == 3) switch_number = 18;
    
    alpha_move = -100;
    delta_move = 0;
    break;

}  

}  

/* End of switch count case*/

Set_Motor_Position(Control_Address,Set_to_actual);

/* Delay reading of motor till it as a chance to move */
for (timelasped = 50000; timelasped > 0; timelasped = timelasped - 1);

Read_Actual_Pos(Control_Address,Actual_Pos);

/*Set up positions for the move command */
/*Add delta to Disk B and Disk A motor position */
New_Pos[0] = (alpha_move * 769) + (1967 * delta_move) + Actual_Pos[0];

/* Output new position to the motor controller */
Write_Final_pos(Control_Address,New_Pos);

/* Start trapozoidal move */
Write_to_flag_register (Control_Address,0x0808);

/* Check to see if the we have made it to new position or switch has been pressed */
do
    
    if (kbhit())
        
        Exit_requested = 1;
        break;
    
    Activated_switch = Check_for_switch_Activation(switch_number);

while (Activated_switch != 1);

/* Exit for loop if key has been pressed */
if (Exit_requested == 1) break;

/* Delay reading of motor till it as a chance to move */
for (timelasped = 50000; timelasped > 0; timelasped = timelasped - 1);

if (count == 4) /* Move arm to center position */
    
    Set_Motor_Position(Control_Address,Set_to_actual);

/* Delay reading of motor till it as a chance to move */
for (timelasped = 50000; timelasped > 0; timelasped = timelasped - 1);
Read_Actual_Pos(Control_Address,Actual_Pos);
delta_move = 0;
alpha_move = 40;

/* Set up positions for the move command */
/* Add delta to Disk B and Disk A motor position */
New_Pos[0] = (alpha_move * 769) + (1967 * delta_move) + Actual_Pos[0];

Send_Pos[0] = New_Pos[0];
Send_Pos[1] = New_Pos[1];

/* Output new position to the motor controller and start move*/
Write_Final_pos(Control_Address,Send_Pos);
Write_to_flag_register (Control_Address, Ox0080);

/* Wait till arm makes its way back to home position */
while (labs(Actual_Pos[0] - New_Pos[0]) > 25)
{
    /* Delay to give motor a chance to respond to trapezoidal move */
    for (timelapsed = 5000; timelapsed > 0; timelapsed = timelapsed - 1);
    Read_Actual_Pos(Control_Address,Actual_Pos);
}

/* End of for loop */
/* End of while 1*/
/* Stop motors and reset all positions counters to actual position */
Set_Motor_Position(Control_Address,Set_to_actual);
/* End of function */

void Run_Back_Lash_Test(unsigned Address_Motor, int Port_C_Data, int NumbChan, int Number_of_Motor)
{
    int Frame_Count = 0;
double Alpha;

    /* Print out title indicating test to be run */
    if (Number_of_Motor == 1)
        fprintf(Manual_test_ptr, "nBacklash test for Disk B (U_error)"
    else if (Number_of_Motor == 2)
        fprintf(Manual_test_ptr, "nBacklash test for Disk A "
    else if (Number_of_Motor == 3)
        fprintf(Manual_test_ptr, "nBacklash test for Disk A & B moving together (V_error)"
    else if (Number_of_Motor == 4)
        fprintf(Manual_test_ptr, "nBacklash test for Disk A & B moving a part"

    /* Print header file for data */
    fprintf(Manual_test_ptr, "nTimeTag \Sig Lev\U_Error\V_Error"
    fprintf(Manual_test_ptr, "nActB\DiskBC\ActA\DiskAC"

    Set_Motor_Position(Address_Motor,Set_to_actual);

    /* Read data from system file on present position */
    Read_Current_System_data();

    DiskAstep = Current_Position[1][Numb Chan - 1];
    DiskBstep = Current_Position[0][Numb Chan - 1];

    Compute_AZ_EL (DiskAstep,DiskBstep, 0,El_Az,Alpha_Delta);

    Alpha = Alpha_Delta[0]/57.29577951;

    Compute_Open_Loop_Commands(Address_Motor,Number_of_Motor,Alpha);  /* Compute commands for open loo
Port_C_Data = Port_C_Data | Sync_Pulse_Enable[Numb Chan - 1];
outp(Port_C_Address,Port_C_Data);

ComFlushRx();
while (Frame_Count <= Number_of_Frames)
while (ComLenRx() < BLK_SZ); /* Wait for data to come over */
Read_Actual_Pos(Address_Motor,Act_pos);
Step_Position[0] = Motor_Pos[0][Frame_Count + 1];
Step_Position[1] = Motor_Pos[1][Frame_Count + 1];
Write_Final_pos(Address_Motor,Step_Position);
Write_to_flag_register(Address_Motor,0x0808);
Read_Channel_Data(&Test[Numb Chan - 1]);
if (Act_pos[0] >= Ox7ffffff)
Act_pos[0] -= Oxffffff;
if (Act_pos[1] >= Ox7ffffff)
Act_pos[1] -= Oxffffff;
fprintf(Manual test_ptr,"\%Xu, %2.3f, %1.5f, %t%1.5f, %t%f, %t%f, %t%f, %t%f"
Test[Numb Chan - 1].Time_tag,Test[Numb Chan - 1].Signal_strength/204.7,
Test[Numb Chan - 1].V_error,Test[Numb Chan - 1].V_error,
Act_pos[0],Motor_Pos[0][Frame_Count],
Act_pos[1],Motor_Pos[1][Frame_Count];
Frame_Count++;
/* End of while loop */
Port_C_Data = Port_C_Data & Sync_Pulse_Disable[Numb Chan - 1];
outp(Port_C_Address,Port_C_Data);
/* End of function Run_Back_Lash_Test */

void Run_Step_Test(unsigned Add_of_Motor,int C_number,int Step_value,int Option)
{
int long Act_step_pos_array_B[1000];
int long Act_step_pos_array_A[1000];
int Sample_Total = 1000;
int Number_of_samples = 0;
int Array_count = 0;
unsigned AD_data[1000];
int AD_dataA,AD_dataB;
unsigned UAD_dataA,UAD_dataB;
Read_Actual_Pos(Add_of_Motor,Step_Position);
if ((Option == 1) || (Option == 5))
Step_Position[0] = Step_value + Step_Position[0]; /* Assign step value for disk b motor */
Step_Position[1] = 0 + Step_Position[1];
else if ((Option == 2) || (Option == 6))
Step_Position[1] = Step_value + Step_Position[1]; /* Assign step value for disk b motor */
Step_Position[0] = 0 + Step_Position[0];
else if ((Option == 3) || (Option == 7))
Step_Position[1] = - Step_value + Step_Position[1]; /* Assign step value for disk b motor */
Step_Position[0] = Step_value + Step_Position[0]; /* to run in same direction */
else if ((Option == 4) || (Option == 8))
Step_Position[1] = Step_value + Step_Position[1]; /* Assign step value for disk b motor */
Step_Position[0] = Step_value + Step_Position[0];

if (Option >= 1 && Option <= 4)
    Write_command_pos(Add_of_Motor,Step_Position);
else
    Write_Final_pos(Add_of_Motor,Step_Position);
    Write_to_flag_registor(Add_of_Motor,0x0808);

while (Number_of_samples < Sample_total)

    AD_data[Number_of_samples] = Read_8bit_port(Add_of_Motor);
    Read_Actual_Pos(Add_of_Motor,Act_pos);
    Act_step_pos_array_B[Number_of_samples] = Act_pos[0];
    Act_step_pos_array_A[Number_of_samples] = Act_pos[1];
    Number_of_samples++;

    fprintf(Manual_test_ptr, "Number of samples taken = %i", Number_of_samples);
*/ Store sampled data */
for (Array_count = 0; Array_count < Number_of_samples; Array_count++)

    UAD_dataB = AD_data[Array_count] & 0xFF;
    AD_dataB = (UAD_dataB - 128);
    UAD_dataA = AD_data[Array_count] & 0xFF00;
    AD_dataA = UAD_dataA >> 8;
    fprintf(Manual_test_ptr, "Y, d", Act_step_pos_array_B[Array_count]);
    fprintf(Manual_test_ptr, "Xd", AD_dataB);
    fprintf(Manual_test_ptr, "X", AD_dataA);

void Run_Eccen_Test(unsigned Motor_Address, int CPort, int Number_of_Channel)

    DA_data, AD_dataB, AD_dataA;
    unsigned UAD_dataB, UAD_dataA;
    long int Present_Position[2];
    long int Command_Pos[2];
    int Left_Limit[3] = (4,10,16);
    int Right_Limit[3] = (3,9,15);
    int Limit_Hit = 0;

    fprintf(Manual_test_ptr, "DiskBApos D/A DiskAApos D/A");
    /* Set up position to move to designated limit */
    Command_Pos[0] = 1000000;
    Command_Pos[1] = -1000000;

    Read_Actual_Pos(Motor_Address,Present_Position);
    Command_Pos[0] = Present_Position[0] + Command_Pos[0];

    if (Number_of_samples < Sample_total)
        Limit_Hit = Check_for_switch_Activation(Right_Limit[Number_of_Channel - 1]);
        Limit_Hit = 0;
        /* Output new position to the motor controller and start move*/
        Write_Final_pos(Motor_Address,Command_Pos);
        Write_to_flag_registor(Motor_Address,0x0808);
        /* Wait till first limit has been reached */
        while (Limit_Hit == 0)
            Limit_Hit = Check_for_switch_Activation(Right_Limit[Number_of_Channel - 1]);

    Set_Motor_Position(Motor_Address,Set_to_actual);
    /* Set up position to move to designated limit */
    Command_Pos[0] = -1000000;
814  Command_Pos[1] = +1000000;
815  Command_Pos[0] = Present_Position[0] + Command_Pos[0];
817
818  /* Output new position to the motor controller and start move*/
819  Write_Final_pos(Motor_Address,Command_Pos);
820  Write_to_flag_register(Motor_Address,0x0808);
821
822  /* Enable receiver electronics for timing purposes */
823  CPort = CPort | Sync_Pulse_Enable(Number_of_Channel - 1);
824  outp(PortC_Address,CPort);
825
826  ComFlushRx();
827  Limit_Hit = 0;
828
829  while (Limit_Hit == 0)
830  
831  
832  (while (ComLenRx() < BLK_SZ); /* Wait for data to come over */
833  ComFlushRx(); /* Flush out receive buffer to avoid overflow */
834  DA_data = Read_Bbit_port(Motor_Address);
835  Read_Actual_Pos(Motor_Address,Present_Position);
836  if (Present_Position[1] > Ox7fffff)
837  Present_Position[1] -= Oxffffff;
838  if (Present_Position[0] > Ox7fffff)
839  Present_Position[0] -= Oxffffff;
840
841  UAD_dataB = DA_data & Ox00FF;
842  AD_dataB = UAD_dataB - 128;
843  UAD_dataA = DA_data & OxFF00;
844  AD_dataA = UAD_dataA >> 8;
845
846  AD_dataA = UAD_dataA - 128;
847
848  /* Output data to file for receive */
849  fprintf(Manual_test_ptr,"_10td",Present_Position[0]);
850  fprintf(Manual_test_ptr," _5d ",AD_dataB);
851  fprintf(Manual_test_ptr," _10td,,,Present_Position[1]);
852  fprintf(Manual_test_ptr," X5d ",AD_dataA);
853  Limit_Hit = Check_for_switch_Activation(Left_Limit[Number_of_Channel - 1]);
854
855  /* Disable sync pulse to receiver electronics */
856  CPort = CPort & Sync_Pulse_Disable(Number_of_Channel - 1);
857  outp(PortC_Address,CPort);
858
859  } /* Function Compute_Open_Loop_Command */
860
861  void Compute_Open_Loop_Commands(unsigned Channel_Address, int Motor_Number,
862  double Alpha.arm)
863  {
864  int Frame_count = 0;
865  double Int_Motor_PosA = 0;
866  double Int_Motor_PosB = 0;
867
868  if (Motor_Number == 1) /* Move only disk b motor */
869  
870  Step_A = 0;
871  Step_B = 1.0;
872
873  } else if (Motor_Number == 2) /* Move only disk a motor: */
874  
875  Step_A = -1.0;
876  Step_B = 0;
877
878  } else if (Motor_Number == 3) /* Move only disk A & B motor:
879  disk will run in same direction*/
888  
889  Step_A = 2.729 - sin((double)(Alpha_arm/2.0) - (Pi/4.0));
890  Step_B = 2.729;
891  
892  else if (Motor_Number == 4) /* Move only disk A & B motor:
893        disk will run in same direction*/
894    
895    Step_A = -1.0;
896    Step_B = 1.0;
897  
898 
899  /* Read in actual position of the arm and drum motors for offset */
900  Read_Actual_Pos(Channel_Address, Act_pos);
901 
902  Motor_Pos[0][Frame_count] = Act_pos[0];
903  Motor_Pos[1][Frame_count] = Act_pos[1];
904  Int_Motor_PosB = (double)Act_pos[0];
905  Int_Motor_PosA = (double)Act_pos[1];
906  
907  Frame_count = Frame_count + 1;
908  
909  /* Check to see if range of frame counts is such that both motors will
910     be moved together */
911  
912  while (Frame_count <= Number_of_Frames)
913    
914    if (Frame_count <= Range)
915      
916      Int_Motor_PosB = Int_Motor_PosB + Step_B;
917      Motor_Pos[0][Frame_count] = (long)Int_Motor_PosB;
918      Int_Motor_PosA = Int_Motor_PosA + Step_A;
919      Motor_Pos[1][Frame_count] = (long)Int_Motor_PosA;
920  
921    else if ((Frame_count <= (3*Range)) && (Frame_count > Range))
922      
923      Int_Motor_PosB = Int_Motor_PosB - Step_B;
924      Motor_Pos[0][Frame_count] = (long)Int_Motor_PosB;
925      Int_Motor_PosA = Int_Motor_PosA + Step_A;
926      Motor_Pos[1][Frame_count] = (long)Int_Motor_PosA;
927  
928    else if ((Frame_count <= (4*Range)) && (Frame_count > (3*Range)))
929      
930      Int_Motor_PosB = Int_Motor_PosB + Step_B;
931      Motor_Pos[0][Frame_count] = (long)Int_Motor_PosB;
932      Int_Motor_PosA = Int_Motor_PosA - Step_A;
933      Motor_Pos[1][Frame_count] = (long)Int_Motor_PosA;
934  
935    else if ((Frame_count <= (5*Range)) && (Frame_count > (4*Range)))
936      
937      Int_Motor_PosB = Int_Motor_PosB - Step_B;
938      Motor_Pos[0][Frame_count] = (long)Int_Motor_PosB;
939      Int_Motor_PosA = Int_Motor_PosA + Step_A;
940      Motor_Pos[1][Frame_count] = (long)Int_Motor_PosA;
941  
942    else if ((Frame_count <= (7*Range)) && (Frame_count > (5*Range)))
943      
944      Int_Motor_PosB = Int_Motor_PosB - Step_B;
945      Motor_Pos[0][Frame_count] = (long)Int_Motor_PosB;
946      Int_Motor_PosA = Int_Motor_PosA + Step_A;
947      Motor_Pos[1][Frame_count] = (long)Int_Motor_PosA;
948    
949    else if ((Frame_count <= (8*Range)) && (Frame_count > (7*Range)))
950      
951    else if ((Frame_count <= (8*Range)) && (Frame_count > (7*Range)))
952      
953      Int_Motor_PosB = Int_Motor_PosB + Step_B;
954      Motor_Pos[0][Frame_count] = (long)Int_Motor_PosB;
955      Int_Motor_PosA = Int_Motor_PosA + Step_A;
956      Motor_Pos[1][Frame_count] = (long)Int_Motor_PosA;
957    
958    else if ((Frame_count <= (8*Range)) && (Frame_count > (7*Range)))
959      
960
Frame_count = Frame_count + 1;

Motor_Pos[0][Frame_count] = Motor_Pos[0][0];
Motor_Pos[1][Frame_count] = Motor_Pos[1][0];

} /* End of Compute_Open_Loop_Commands function */
/* Motor.C: controls all functions of the dc servo motors */

#include <stdio.h>
#include <conio.h>
#include <ctype.h>
#include <time.h>
#include <graph.h>
#include <dos.h>
#include <time.h>

unsigned Motor_Pair1_Address = Ox300; /*Address to first pair of motor controllers */
unsigned Motor_Pair2_Address = Ox306; /*Address to first pair of motor controllers */
unsigned Motor_Pair3_Address = Ox310; /*Address to first pair of motor controllers */

unsigned Control_word_address = Ox31e; /*Address to which 8255 command word is being set */
unsigned Port_A_Address = Ox318; /*Address of port A on 8255 */
unsigned Port_B_Address = Ox31a; /*Address of port g on 8255 */
unsigned Port_C_Address = Ox31c; /*Address of port C on 8255 */

Control_word = 0x92; /*Sets ports A,B to inputs and C to outputs */
Port_373 = Ox30C; /*Address for 74LS373 latch */

/*Set up all variables that will define the HCTL-1X00 motor controller registers */

int Flag_Reg_Address = Ox0000; /*Address for writing to flag register */
int Prog.Counter_Address = Ox0505; /*Address of program counter */
int Status_Reg_Address = Ox0707; /*Address of Status Register */
int AD_Motor_Com_Address = Ox0808; /*Address at which we can write data out to the 8 bit port that feeds an A/D */
int Pwm_Motor_Com_Address = Ox0909; /*Address at which we can command the PWM port of the controller */
int Command_Pos3_Address = Ox0c0c; /*Address to the MSB of the command position */
int Command_Pos2_Address = Ox0d0d; /*Address to the middle byte of the command position */
int Command_Pos1_Address = Ox0e0e; /*Address to the LSB of the command position */
int Sampler_Timer_Address = Ox1010; /*Address for setting up timer in controller */
int Actual_Pos3_Address = Ox1212; /*Address to the MSB of the command to read the actual position of the motor */
int Actual_Pos2_Address = Ox1313; /*Address to the middle byte of the command to read the actual position of the motor */
int Actual_Pos1_Address = Ox1414; /*Address to the LSB of the command to read the actual position of the motor */
int Comm_Ring_Address = Ox1818; /*Address to set up the length of commutation cycle */
int Comm_Vel_Timer_Address = Ox1919; /*Address to command the amount of phase advance at a given advance */
int X_Address = Ox1a1a; /*Address to command the interval that only one phase is active */
int Y_Phase_Overlay_Address = Ox1b1b; /*Address to command the interval that two seq. phase are active */
int Offset_Address = Ox1c1c; /*Address to command the relative start of the commutation cycle with respect to the index pulse */
int Max_Phase_Adv_Address = 0x1ff1; /* Address to command upper limit of
    phase advance */
int Filter_Zero_Address = 0x2020; /* Address to set up value for zero in
    digital filter */
int Filter_Pole_Address = 0x2121; /* Address to set up value for pole in
    digital filter */
int Filter_Gain_Address = 0x2222; /* Address to set gain of the digital filter */
int Comm_Velocity1_Address = 0x2323; /* Address to input LSB of command velocity */
int Comm_Velocity2_Address = 0x2424; /* Address to input MSB of command velocity */
int Accel_LSB_Address = 0x2626; /* Address to input LSB of command acceleration */
int Accel_MSB_Address = 0x2727; /* Address to input MSB of command acceleration */
int Max_Velocity_Address = 0x2828; /* Address to max velocity in the trapezoidal
    profile */
int Final_Pos1_Address = 0x2929; /* Address (LSB) to write or read the position
    at which the motor is commanded to go to */
int Final_Pos2_Address = 0x2a2a; /* Address to write or read the position
    at which the motor is commanded to go to */
int Final_Pos3_Address = 0x2b2b; /* Address (MSB) to write or read the position
    at which the motor is commanded to go to */
int Actual_Vel1_Address = 0x3434; /* Address (LSB) to read actual velocity of motor */
int Actual_Vel2_Address = 0x3535; /* Address (MSB) to read actual velocity of motor */
/* Listed below is all the functions in the program */
int Read_Filter_Zero(unsigned Motor_Address);
int Display_Main_Menu(void);
void Wait_for_key_Press(void);
void Process_Main_Option(int Key_pressed);
int Display_Main_Menu(void);
void Write_Filter_Zero(unsigned Motor_Address, int Filter_Zero);
void Write_Filter_Pole(unsigned Motor_Address, int Filter_Pole);
void Write_Filter_Gain(unsigned Motor_Address, int Filter_Gain);
int Read_Filter_Zero(unsigned Motor_Address);
int Read_Filter_Pole(unsigned Motor_Address);
int Read_Filter_Gain(unsigned Motor_Address);
void Write_to_BBit_port(unsigned Motor_Address, int Digital_word);
void Write_to_PWM_port(unsigned Motor_Address, int PWM_data);
void Write_to_sampler_timer(unsigned Motor_Address, int Sampler_data);
unsigned Read_Status_reg(unsigned Motor_Address);
void Write_Max_Velocity(unsigned Motor_Address, int MaxVelocity);
void Write_Max_Accel(unsigned Motor_Address, int Max_Acc_data);
void Write_to_program_counter(unsigned Motor_Address, int Program_counter);
void Write_to_max_Accel(unsigned Motor_Address, int Motor_Acceleration);
void Read_Max_Velocity(unsigned Motor_Address);
void Read_Command_Pos(unsigned Motor_Address, int *Command_pos);
void Write_command_pos(unsigned Motor_Address, long *Position_data);
/* The following function will be used to write to both the drum and arm
motor controller at the same time */
void Write_to_motor_controller(unsigned Address, int Dataword, int Register) {
    outpw (Address, Register); /*Generate an Ale signal and send over the
    register address that we want to write
The following function will be used to read data from the both the drum and motor controllers:

```c
int Read_from_motor_controller(unsigned Address, int Registor)
{
    int Dumb_data,Controller_data;
    outpw(Address, Registor); /*Generate an ALE signal and send over the register address that we want to write to */
    Dumb_data = inpw(Address + 2); /*Generate chip select to motor controller */
    Controller_data = inpw(Address + 4); /* Generate OE signal and read data */
    return Controller_data;
}
```

The following function will write to the flag register:

```c
void Write_to_flag_registor (unsigned Motor_Address,int Flagword)
{
    Write_to_motor_controtLer(Motor_Address,Flagword,Flag_Reg_Address);
}
```

The following function will write to the program counter:

```c
void Write_to_program_counter (unsigned Motor_Address,int Program_counter)
{
    Write_to_motor_controtLer(Motor_Address,Programcounter,Prog_CounterAddress);
}
```

The following function will write to the status registor:

```c
void Write_to_statusreg(unsigned Motor_Address, int Statusword)
{
    Write_to_motor_controtLer(Motor_Address,Statusword,Status_Reg_Address);
}
```

The following function will read the status registor:

```c
unsigned Read_status_reg(unsigned Motor_Address)
{
    unsigned Status;
    Status = Read_from_motor_controller(Motor_Address,Status_Reg_Address);
    return Status;
}
```

The following function will write to the 8 bit port:

```c
void Write_toBbit_port(unsigned Motor_Address,int Digital_word)
{
    Write_to_motor_controtLer(Motor_Address,Digitalword,AD_Motor_ComAddress);
}
```

The following function will read the data in the 8 bit port:

```c
int Read_Bbit_port(unsigned Motor_Address)
{
    int Port_data;
    Port_data = Read_from_motor_controller(Motor_Address,AD_Motor_Com_Address);
}
/* The following function will write data to the PWM port*/

void Write_to_PWM_port(unsigned Motor_Address, int PWM_data)
{
    Write_to_motor_controller(Motor_Address, PWM_data, Pwm_Motor_Com_Address);
}

/* The following function will read the data from the pwm command registor*/

int Read_PWM_port(unsigned Motor_Address)
{
    int PWM_Port_data;
    PWM_Port_data = Read_from_motor_controller(Motor_Address, Pwm_Motor_Com_Address);
    return PWM_Port_data;
}

/* The following function will write data to the Command position registor*/

void Write_command_pos(unsigned Motor_Address, long int *Position_data)
{
    long int New_Pos_Data[2];
    int Pos_data = 0;
    /* Invert the sign of the command if channel 1 is written to */
    if (Motor_Address == Motor_Pair1_Address)
    {
        New_Pos_Data[0] = Position_data[0] * -1;
    }
    else
    {
        New_Pos_Data[0] = Position_data[0];
        New_Pos_Data[1] = Position_data[1];
    }
    /* Output the most significant byte to the motor controller */
    Pos_data = (((New_Pos_Data[0] >> 16) & 0xff) | ((New_Pos_Data[1] >> 8) & 0xff00));
    Write_to_motor_controller(Motor_Address, Pos_data, Command_Pos3_Address);
    /* Output the middle byte to the motor controller */
    Pos_data = (((New_Pos_Data[0] >> 16) & 0xff) | (New_Pos_Data[1] >> 8) & 0xff00);
    Write_to_motor_controller(Motor_Address, Pos_data, Command_Pos2_Address);
    /* Output the least significant byte to the motor controller */
    Pos_data = ((New_Pos_Data[0] & 0xff) | (New_Pos_Data[1] << 8) & 0xff00);
    Write_to_motor_controller(Motor_Address, Pos_data, Command_Pos1_Address);
}

/* The following function will read the command motor position registor*/

Read_Command_Pos(unsigned Motor_Address, long int *Command_pos)
{
    int Com_data1, Com_data2, Com_data3;
    /* Read the most significant byte of the command position from the motor controller */
    Com_data3 = Read_from_motor_controller(Motor_Address, Command_Pos3_Address);
    /* Read the middle byte of the command position from the motor controller */
    Com_data2 = Read_from_motor_controller(Motor_Address, Command_Pos2_Address);
    /* Read the least significant byte of the command position from the motor controller */
    Com_data1 = Read_from_motor_controller(Motor_Address, Command_Pos1_Address);
/* Determine actual pos for drum motor */
Command_pos[0] = Command_pos[0] << 8; (Com_data2 & 0xff);
Command_pos[1] = Command_pos[1] << 8; (Com_data1 & 0xff);

/* Determine actual pos for arm motor */
Command_pos[1] = Command_pos[1] << 8; (Com_data2 & 0xff00);
Command_pos[1] = Command_pos[1] << 8; (Com_data1 & 0xff00);
Command_pos[1] = Command_pos[1] << 8; (Com_data1 & 0xff00) >> 8));

if (Motor_Address == Motor_Pair1_Address)
{
    Command_pos[0] = Command_pos[0] * -1;
}

/* The following function will write data to the Sampler timer register*/
void Write_to_sampler_timer(unsigned Motor_Address, int Sampler_data)
{
    /* Output the timer value to the motor controller */
    Write_to_motor_controller(Motor_Address, Sampler_data, Sampler_Timer_Address);
}

/* The following function will read the actual motor position register*/
Read_Actual_Pos(unsigned Motor_Address, long int * Actual_pos)
{
    int Actual_data1, Actual_data2, Actual_data3;

    /* Read the least significant byte of the command position from the motor controller */
    Actual_data1 = Read_from_motor_controller(Motor_Address, Actual_Pos1_Address);
    Actual_data2 = Read_from_motor_controller(Motor_Address, Actual_Pos2_Address);
    Actual_data3 = Read_from_motor_controller(Motor_Address, Actual_Pos3_Address);

    /* Shift and or the data read from motor controller and store in a 21 byte word */
    if (Actual_data1 > 0x7fffff)
    Actual_data1 = 0x7fffff;
    if (Actual_data2 > 0x7fffff)
    Actual_data2 = 0x7fffff;
    if (Actual_data3 > 0x7fffff)
    Actual_data3 = 0x7fffff;

    /* Determine actual pos for drum motor */
    Actual_pos[0] = Actual_pos[0] = Actual_data3 & 0xff;
    Actual_pos[0] = (Actual_pos[0] << 8) | (Actual_data2 & 0xff);  
    Actual_pos[0] = (Actual_pos[0] << 8) | (Actual_data1 & 0xff);

    if (Actual_pos[0] > 0x7fffff)
    Actual_pos[0] = 0x7fffff;
    /* Determine actual pos for arm motor */
    Actual_pos[1] = Actual_pos[1] = Actual_data3 & 0xff00;
    Actual_pos[1] = (Actual_pos[1] << 8) | (Actual_data2 & 0xff00);  
    Actual_pos[1] = (Actual_pos[1] | (Actual_data1 & 0xff00) >> 8));

    if (Actual_pos[1] > 0x7fffff)
    Actual_pos[1] = 0x7fffff;
    if (Motor_Address == Motor_Pair1_Address)
    {
        Actual_pos[0] = Actual_pos[0] * -1;
    }
```c
/* The following function reset the actual position register*/
void Reset_Actual_Position(unsigned Motor_Address, int Reset_Pos)
{
    /* Write to register 15H to reset actual position counter */
    Write_to_motor_controller(Motor_Address, Reset_Pos, Actual_Pos2_Address);
}

/* The following function writes the zero of the digital filter to the
   to the motor controller */
void Write_Filter_Zero(unsigned Motor_Address, int Filter_Zero)
{
    Write_to_motor_controller(Motor_Address, Filter_Zero, Filter_Zero_Address);
}

/* The following function will read value of the register that contains the
   zero of the digital filter*/
int Read_Filter_Zero(unsigned Motor_Address)
{
    int Zero;
    Zero = Read_from_motor_controller(Motor_Address, Filter_Zero_Address);
    return Zero;
}

/* The following function writes the pole of the digital filter to the
   to the motor controller */
void Write_Filter_Pole(unsigned Motor_Address, int Filter_Pole)
{
    Write_to_motor_controller(Motor_Address, Filter_Pole, Filter_Pole_Address);
}

/* The following function will read value of the register that contains the
   pole of the digital filter*/
int Read_Filter_Pole(unsigned Motor_Address)
{
    int Pole;
    Pole = Read_from_motor_controller(Motor_Address, Filter_Pole_Address);
    return Pole;
}

/* The following function writes the Gain of the digital filter to the
   to the motor controller */
void Write_Filter_Gain(unsigned Motor_Address, int Filter_Gain)
{
    Write_to_motor_controller(Motor_Address, Filter_Gain, Filter_Gain_Address);
}

/* The following function will read value of the register that contains the
   gain of the digital filter*/
```
int Read_Filter_Gain(unsigned Motor_Address)
{
    int Gain;
    Gain = Read_from_motor_controller(Motor_Address,Filter_Gain_Address);
    return Gain;
}

/* The following function will write data to the Max Acceleration register*/
void Write_Max_Accel(unsigned Motor_Address, int *Max_Acc_data)
{
    int Acc_data = 0;

    /* Output the most significant byte to the motor controller */
    Acc_data = ((Max_Acc_data[0] >> 8) & 0xff) | (Max_Acc_data[1] & 0xff00);
    Write_to_motor_controller(Motor_Address,Acc_data,Accel_MSB_Address);

    /* Output the least significant byte to the motor controller */
    Acc_data = (Max_Acc_data[0] & 0x0ff) | (Max_Acc_data[1] << 8);
    Write_to_motor_controller(Motor_Address,Acc_data,Accel_LSB_Address);
}

/* The following function will read the Maximum acceleration register*/
int Read_Max_Accel(unsigned Motor_Address, int *Motor_Acceleration)
{
    int Accel_data1,Accel_data2;

    /* Read the most significant byte max. acceleration from the motor controller */
    Accel_data2 = Read_from_motor_controller(Motor_Address,Accel_MSB_Address);

    /* Read the least significant byte of the command position from the motor controller */
    Accel_data1 = Read_from_motor_controller(Motor_Address,Accel_LSB_Address);

    /* Shift and or the data read from motor controller and store in a 21 byte word */
    Motor_Acceleration[0] = Accel_data2 & 0xff;
    Motor_Acceleration[0] = (Motor_Acceleration[0] << 8) | (Accel_data1 & 0xff);
    Motor_Acceleration[1] = Accel_data2 & 0xff00;
    Motor_Acceleration[1] = Motor_Acceleration[1] | (Accel_data1 & 0xff00) >> 8);
}

/* The following function writes the Max Velocity for trapezoidal mode to the motor controller */
void Write_Max_Velocity(unsigned Motor_Address, int MaxVelocity)
{
    Write_to_motor_controller(Motor_Address,MaxVelocity,Max_Velocity_Address);
}

/* The following function reads the value of the register that contains the Max velocity for trapezoidal mode*/
int Read_Max_Velocity(unsigned Motor_Address)
{
    int Velocity;
}
Velocity = Read_from_motor_controller(Motor_Address, Max_Velocity_Address);
return Velocity;

/* The following function will write data to the Final position register*/
void Write_Final_pos(unsigned Motor_Address, long int *Final_data)
{
    long int New_final_data[2];
    int Final_data = 0;
    if (Motor_Address == Motor_Pair1_Address)
    {
        New_final_data[0] = Final_data[0] * -1;
    }
    else
    {
        New_final_data[0] = Final_data[0];
        New_final_data[1] = Final_data[1];
    }
    /* Output the most significant byte to the motor controller */
    Final_data = (((New_final_data[0] >> 16) & 0xff) | ((New_final_data[1] >> 8) & 0xff00));
    Write_to_motor_controller(Motor_Address, Final_data, Final_Pos3_Address);
    /* Output the middle byte to the motor controller */
    Final_data = (((New_final_data[0] >> 8) & 0xff) | (New_final_data[1] & 0xff00));
    Write_to_motor_controller(Motor_Address, Final_data, Final_Pos2_Address);
    /* Output the least significant byte to the motor controller */
    Final_data = (New_final_data[0] & 0xff) | (New_final_data[1] << 8) & 0xff00;
    Write_to_motor_controller(Motor_Address, Final_data, Final_Pos1_Address);
}

/* The following function will read the Final motor position register*/
Read_Final_pos(unsigned Motor_Address, long int *Final_pos)
{
    int Final_data1, Final_data2, Final_data3;
    /* Read the most significant byte of the command position from the motor controller */
    Final_data3 = Read_from_motor_controller(Motor_Address, Final_Pos3_Address);
    /* Read the middle byte of the command position from the motor controller */
    Final_data2 = Read_from_motor_controller(Motor_Address, Final_Pos2_Address);
    /* Read the least significant byte of the command position from the motor controller */
    Final_data1 = Read_from_motor_controller(Motor_Address, Final_Pos1_Address);
    /* Determine the final position of the drum motor */
    Final_pos[0] = (Final_pos[0] << 8) | (Final_data2 & 0xff);
    Final_pos[0] = (Final_pos[0] << 8) | (Final_data1 & 0xff);
    /* Determine final position of the arm motor */
    Final_pos[1] = (Final_pos[1] << 8) | (Final_data2 & 0xff00);
    Final_pos[1] = (Final_pos[1] << 8) | (Final_data1 & 0xff00);
    if (Motor_Address == Motor_Pair1_Address)
    {
        Final_pos[0] = Final_pos[0] * -1;
    }
if (Final_pos[0] > 0xffffff)
Final_pos[0] -= 0xffffff;

if (Final_pos[1] > 0xffffff)
Final_pos[1] -= 0xffffff;

}
/* Display routine is used to provide the menu for manual control of the motors */

#include <stdio.h>
#include <conio.h>
#include <ctype.h>
#include <time.h>
#include <graph.h>
#include <dos.h>
#include <time.h>
#include <stdlib.h>
#include <math.h>
#include <cport.h>

#define LEFTARROW 75
#define RIGHTARROW 77
#define UPARROW 72
#define DOWNARROW 80

signed char output_buffer[512];
signed char input_buffer[1024];
unsigned int rs;

BLK_SZ = 8;

/* Declare the following variables external these var. are declared in motor.c */
extern unsigned Motor_Pair1_Address;
/* Address to first pair of motor controllers */
extern unsigned Motor_Pair2_Address;
/* Address to first pair of motor controllers */
extern unsigned Motor_Pair3_Address;
/* Address to first pair of motor controllers */
extern unsigned Port_A_Address;
/* Address of port A on 8255 */
extern unsigned Port_B_Address;
/* Address of port B on 8255 */
extern unsigned Port_C_Address;
/* Address of port C on 8255 */
extern int Set_to_actual;
extern int C_Port_Data;
extern long int Starting_Position[2][3];
extern long int Current_Position[2][3];
extern int Stop_Serial_Data[3];
extern int Initiate_Serial_Trans[3];

extern struct Channel_data
   {
      unsigned Time_tag;
      double Elevation_error;
      double Azimuth_error;
      double Signal_strength;
      signed char Transmit_Data;
   } Channel[3],Test[3];

long int timelased;

int Exercise_Multiple_Channels(int *Channels_to_Execute)
{
   int Switch_number_array[3][5] = {{3,4,5,6,3},{9,10,11,12,9},{15,16,17,18,15}};
   long int Delta_move_array[3][4] = {{-500,40,0,500},{-500,40,0,500},{-500,40,0,500}};
   long int Delta_move_only = (40);
   long int Delta_move = (0);
   int numb_switch,chan_number,Activated_switch,Address,Move_finished
   long int Target_Position[2];
   long int Position_of_Motor[2];
   int exit_loop = (0);
   while (kbhit()) getch(); /* clear out keyboard buffer */

   /* Start all requested channels running */
   numb_switch = 4;
   for (chan_number = 1; chan_number <=3; chan_number++)
   {
      if (Channels_to_Execute[chan_number] == 1)
```c
switch (chan_number)
{
    case 1:
        Address = Motor_Pair1_Address; break;
    case 2:
        Address = Motor_Pair2_Address; break;
    case 3:
        Address = Motor_Pair3_Address; break;
}

Move_Motor(Address,Alpha_move_array[chan_number-1][numb_switch-1],
            Delta_move_array[chan_number-1][numb_switch-1]);

printf("\r\nMoving channel %i to limit switch %i", chan_number,
       Switch_number_array[chan_number-1][numb_switch]);

while (exit_loop != 1)
{
    for (chan_number = 1; chan_number <= 3; chan_number++)
    {
        if (kbhit()) exit_loop = 1;
        while (kbhit()) getch(); /* clear out keyboard buffer */
        if (Channels_to_Exercise[chan_number] == 1)
        {
            for (numb_switch = 1; numb_switch <= 4; numb_switch++)
            {
                Activated_switch = Check_for_switch_Activation(Switch_number_array[chan_number-1][numb_switch-1]);
                if (Activated_switch == 1)
                {
                    /* Delay program till motor stops */
                    for (timelasped = 60000; timelasped > 0; timelasped = timelasped - 1);
                    switch (chan_number)
                    {
                        case 1:
                            Address = Motor_Pair1_Address;
                            Alpha_move_only = 25;
                            break;
                        case 2:
                            Address = Motor_Pair2_Address;
                            Alpha_move_only = 40;
                            break;
                        case 3:
                            Address = Motor_Pair3_Address;
                            Alpha_move_only = 55;
                            break;
                    }
                    if (numb_switch == 4)
                    {
                        /* Stop motors and set to actual position */
                        Set_Motor_Position(Address, Set_to_actual);
                        Read_Actual_Pos(Address, Position_of_Motor);
                        Target_Position[0] = (Alpha_move_only * 769) + Position_of_Motor[0];
                        Target_Position[1] = Position_of_Motor[1];
                        Move_Motor(Address, Alpha_move_only, Delta_move);
                        Move_finished = 0;
                        /* Wait for arm to swing over before continuing */
                    }
                }
            }
        }
    }

    if (numb_switch != 4)
    {
        /* Delay program till motor stops */
        for (timelasped = 60000; timelasped > 0; timelasped = timelasped - 1);
        switch (chan_number)
        {
            case 1:
                Address = Motor_Pair1_Address;
                Alpha_move_only = 25;
                break;
            case 2:
                Address = Motor_Pair2_Address;
                Alpha_move_only = 40;
                break;
            case 3:
                Address = Motor_Pair3_Address;
                Alpha_move_only = 55;
                break;
        }
    }
}
```

while (Move_finished != 1) {
    Move_finished = Reached_Commanded_Pos(Address, Target_Position, chan_number, 30);
    if (kbhit()) exit_loop = 1; /* clear out keyboard buffer */
}

void Position(double Azimuth, double Elevation, int channel, unsigned Motor_Address, int Incremental) {
    double Delta[4], Alpha[4], AZ_Deg, EL_Deg, Command_Length, U_Scale, V_Scale;
    double Level_of_Signal, Level_of_UError, Level_of_VError;
    long int I_StepA[4], I_StepB[4], Step_Dif;
    long int Command_Pos[2], Actual_Pos[2];
    int a, b, Number_of_Solutions, loop, Solution;
    long int Present_Delta = 10000000;
}
int t = 1;
long int time_out;
int Numb_of Chan, Loop_count;
Loop_count = 0;
Level_of Signal = 0;
Level_of Error = 0;
Level_of VError = 0;
AZ_Deg = (Azimuth) * 57.29577951;
EL_Deg = (Elevation) * 57.29577951;
AzEl2AB(Azimuth, Elevation, I_StepA, I_StepB, Delta, Alpha, &Number_of_Solutions);
for (loop = 0; loop <= Number_of_Solutions; loop++)
{
    Step_Dif = Current_Position[][channel-1] - I_StepA[loop];
    if (fabs(Step_Dif) < Present Delta)
    {
        Present Delta = fabs(Step_Dif);
        Solution = loop;
    }
}
_clearscreen(_GCLEARSCREEN);
printf("Use cursor keys to move the head in the X and Y coordinates.
");
printf("Press 'E' to exit.
");
Read_Actual_Pos(Motor_Address,Actual_Pos);
Actual_Pos[0] = Starting_Position[0][channel-1] - Actual_Pos[0];
Actual_Pos[1] = Starting_Position[1][channel-1] + Actual_Pos[1];
if (channel == 1) Numb_of Chan = 2;
else if (channel == 2) Numb_of Chan = 1;
else Numb_of Chan = 3;
if (channel == 3)
{
    U_Scale = -0.0002718;
    V_Scale = 0.0002453;
}
else
{
    U_Scale = 0.0002718;
    V_Scale = -0.0002453;
}
do {
    /* Turn DSP for data retrieval */
    ComFlushRx(); /* Flush out data before starting loop */
    timo_out = 0;
/* Wait for data before reading out data */
while ((ComLenRx() < BLK_SZ) && (time_out <= 50000))
    time_out = time_out + 1;

if (time_out < 50000)
    { Read_Channel_Data(&Test[channel-1]);
    /* Clear out keyboard buffer before continuing */
    else
        { while (kbhit()) b = getch();
            ComFlushRx();
        }
/* Turn off sync pulse to x channel */
C_Port_Data = C_Port_Data & Stop_Serial_Data[channel - 1];
outp(Port_C_Address, C_Port_Data);
Level_of_Signal = Test[channel - 1].Signal_strength/204.7 + Level_of_Signal;
Level_of_UError = Test[channel - 1].Azimuth_error*U_Scale + Level_of_UError;
Level_of_VError = Test[channel - 1].Elevation_error*V_Scale + Level_of_VError;

        if (Loop_count == 10)
            _settextposition(16,1); /*Position the cursor at position 1,5*/

        printf("\n\tTime_tag = %u\t Signet Level: %10.6f\n",
                Test[channel-1].Time_tag,
                Level_of_Signal/Loop_count);
        printf("\n\tU_Error: %10.6f \t V_Error: %10.6f \n",
                Level_of_UError/Loop_count,
                Level_of_VError/Loop_count);
        Level_of_Signal = 0;
        Level_of_UError = 0;
        Loop_count = 0;
})
        }
)
while (!kbhit());
a=getch();
if (a==0) a=getch();
switch(a)
    { case LEFTARROW: Azimuth+=10e-6;
        break;
    case RIGHTARROW: Azimuth-=10e-6;
        break;
    case UPARROW: Elevation-=10e-6;
        break;
    case DOWNARROW: Elevation+=10e-6;
        break;
    case '4': Azimuth+=100e-6;
        break;
    case '6': Azimuth-=100e-6;
        break;
    case '2': Elevation+=100e-6;
        break;
    case '8': Elevation-=100e-6;
break;

default: break;
)

Read_Actual_Pos(Motor_Address, Actual_Pos);

Actual_Pos[0] = Starting_Position[0][channel-1] - Actual_Pos[0];
Actual_Pos[1] = Starting_Position[1][channel-1] + Actual_Pos[1];

AzEl2AB(Azimuth, Elevation, I_StepA, I_StepB, Delta, Alpha,&Number_of_Solutions);

/*/ Search for the nearest solution */
Present_Delta = 10000000;

for (loop=0; loop <= Number_of_Solutions; loop++)
{
    Step_Dif = Actual_Pos[1] - I_StepA[loop];
    if (labs(Step_Dif) < Present_Delta)
        Present_Delta = labs(Step_Dif);
        Solution = loop;
}

/*/ Set up the commanded position for the DiskB Motor */
Command_Pos[0] = Starting_Position[0][channel-1] - I_StepB[Solution];

/*/ Set up the commanded position for the DiskA Motor */
Command_Pos[1] = (-1 * Starting_Position[1][channel-1]) + I_StepA[Solution];

/*/ Output new position to the motor controller */
Write_Final_pos(Motor_Address, Command_Pos);

/*/ Start trapezoidal move */
Write_to_flag_register(Motor_Address, 0x0808);

while (kbhit()) b = getch();

while ((a!='e') && (a!='E'))
{

}

void Run_Scan_Pattern(double Azimuth, double Elevation, int channel,
unsigned Motor_Address, double trip_point)
{
    double Delta[4], Alpha[4], AZ_Deg, EL_Deg, Command_Length;
    long int I_StepA[4], I_StepB[4], Step_Dif;
    long int Command_Pos[2], Actual_Pos[2];
    int key, b, Number_of_Solutions, loop, Solution, change_par;
    long int Present_Delta = 10000000;
    int t = 1;
    long int time_out, loop_count, setpoint;
    int exit = 0;

    AZ_Deg = (Azimuth) * 57.29577951;
    EL_Deg = (Elevation) * 57.29577951;

    _clearscreen( _GCLEARSCREEN );

    AzEl2AB(Azimuth, Elevation, I_StepA, I_StepB, Delta, Alpha,&Number_of_Solutions);

    for (loop=0; loop <= Number_of_Solutions; loop++)
    {
        Step_Dif = Current_Position[1][channel-1] - I_StepA[loop];
        if (labs(Step_Dif) < Present_Delta)
            Present_Delta = labs(Step_Dif);
            Solution = loop;
    }
444  
445  
446  Read_Actual_Pos(Motor_Address, Actual_Pos);
447  
448  Actual_Pos[0] = Starting_Position[0][channel-1] - Actual_Pos[0];
449  Actual_Pos[1] = Starting_Position[1][channel-1] + Actual_Pos[1];
450  
451  setpoint = 0;
452  Test[channel-1].Signal_strength = 0.0;
453  
454  while ((Test[channel-1].Signal_strength/204.7 < trip_point) && (exit == 0))
455  
456  for (change_par = 1; change_par <= 4; change_par++)
457  
458  /* Exit if signal level is above trip point */
459  if (((Test[channel-1].Signal_strength/204.7) > trip_point) || (exit == 1)) break;
460  
461  if ((change_par == 1) || (change_par == 3))
462  setpoint = setpoint + 1;
463  
464  for (loop_count = 1; loop_count <= setpoint; loop_count++)
465  
466  if (kbhit())
467  
468  
469  
470  _settextposition(2,5); /* Position the cursor at position 1,5 */
471  
472  AZ_Deg = (Azimuth) * 57.29577951;
473  EL_Deg = (Elevation) * 57.29577951;
474  
475  printf("Position:
	Azimuth: %8.6f Elevation: %8.6f\n", AZ_Deg, EL_Deg);
476  
477  printf("Delta: %10.6f Alpha: %10.6f\n", Delta[Solution], Alpha[Solution]);
478  
479  printf("\n\n\nI Step: %11.6f 1 Step: %11.6f\n", I_StepA[Solution], I_StepB[Solution]);
480  
481  printf("\n\n\nActualA: %11.6f ActualB: %11.6f\n", Actual_Pos[1], Actual_Pos[0]);
482  
483  printf("\n\n\nTest[channel-1].Signal_strength = %3.6f", Test[channel-1].Signal_strength/204.7);
484  
485  /* Turn DSP for data retrieval */
486  C_Port_Data = C_Port_Data | Initiate_Serial_Tran[channel - 1];
487  outp(Port_C_Address, C_Port_Data);
488  
489  ComFlushRx(); /* Flush out data before starting loop */
490  
491  time_out = 0;
492  
493  /* Wait for data before reading out data */
494  while (((ComLenRx()) < BLK_SZ) && (time_out <= 80000))
495  
496  time_out = time_out + 1;
497  
498  if (time_out < 80000)
499  Read_Channel_Data(&Test[channel - 1]);
500  else if (time_out >= 80000) ComFlushRx();
501  
502  /* Turn off sync pulse to x channel */
503  C_Port_Data = C_Port_Data & Stop_Serial_Data[channel - 1];
504  outp(Port_C_Address, C_Port_Data);
505  
506  
507  
508  /* Break out of for loop if signal is large enough */
509  if (((Test[channel-1].Signal_strength/204.7) > trip_point) || (exit == 1)) break;
510  
511  switch(change_par)
512  
513  { case 1: Azimuth+=50e-6; break;
514  case 2: Elevation+=50e-6; break;
515  case 3: Azimuth-=50e-6; break;
case 4:  
    Elevation-=50e-6;
    break;
  
default: break;
}

Read_Actual_Pos(Motor_Address,Actual_Pos);

Actual_Pos[0] = Starting_Position[0][channel-1] - Actual_Pos[0];
Actual_Pos[1] = Starting_Position[1][channel-1] + Actual_Pos[1];

AzEl2AB(Azimuth, Elevation, I_StepA, I_StepB, Delta, 
    Alpha,&Number_of_Solutions);

/* Search for the nearest solution */
Present_Delta = 10000000;

for (loop=0; loop <= Number_of_Solutions; loop++)
{
    Step_Dif = Actual_Pos[1] - I_StepA[loop];
    if (labs(Step_Dif) < Present_Delta)
    {
        Present_Delta = labs(Step_Dif);
        Solution = loop;
    }
}

/* Set up the commanded position for the DiskB Motor */
Command_Pos[0] = Starting_Position[0][channel-1] - I_StepS[Solution];

/* Set up the commanded position for the DiskA Motor */
Command.Pos[1] = (-1 * Starting_Position[1][channel-1]) + I_StepA[Solution];

/* Output new position to the motor controller */
Write_Final_pos(Motor_Address,Command_Pos);

/* Start trapezoidal move */
Write_to_flag_register (Motor_Address,0x0808);

) /* End of for (loop_count = 1; loop_count <= setpoint; loop_count++) */

) /* End o for (change_par = 1; change_par < 4; change_par++) */

/* Turn off sync pulse to x channel */
C_Port_Data = C_Port_Data & Stop_Serial_Data[channel - 1];
outp(Port_C_Address,C_Port_Data);

)
The following include statements are required to allow
program to link up with Turbo C++ libraries */

#include <port.h>
#include <bios.h>
#include <conio.h>
#include <dos.h>
#include <ctype.h>
#include <stdio.h>
#include <math.h>
#define BLK_SZ 8

signed char output_buffer[512];
signed char input_buffer[1024];
unsigned int rs;
int V_error,U_error,Total_signal_strength;
int Frame_count,Number_of_samples,Toggle_count,Process_data_flag,Time_tag;
char Mode_flag;
char tmpbuf[128]; /* Char array for storing time data */
tmpt:xJfl[128]; /* Char array for storing time data */

struct Channel_data
{
    unsigned Time_tag;
    double V_error;
    double U_error;
    double Signal_strength;
    signed char Transmit_Data;
} Channel[3],Test[3];

double Offset = .03;

extern unsigned Port_A_Address; /* Adress of port A on 8255 */
extern unsigned Port_B_Address; /* Adress of port B on 8255 */
extern unsigned Port_C_Address; /* Adress of port C on 8255 */
extern FILE *output_ptr; /* Pointer for file to output error data */

void Init_Com_port(void);
void Read_Channel_Data(struct Channel_data *Ch.ptr);
int Test_Serial_Link(int Chan_Number, int Port_C_Data);

/* Read_Channel_Data reads in the data and formats as required */

void Read_Channel_Data(struct Channel_data *Ch_ptr)
{
    /* Read data from dsp via com port*/
    rs = ComIn(input_buffer,BLK_SZ);

    /* Send over data to the monitor PC */
    ComActive(COM2);
    ComOut(&(Ch.ptr->Transmit_Data),1);
    ComOut(input_buffer,rs);
    ComActive(CON); /*Convert input data to 16 bit format for time tag
    input_buffer[0] and input_buffer[1] time tag data*/
    Ch.ptr->Time_tag = (int)input_buffer[1];
    Ch.ptr->Time_tag = (Ch.ptr->Time_tag & Oxff) << 8;
    Ch.ptr->Time_tag += (int)input_buffer[0] & Oxff;

    /* Convert buffer inputs 2,3 to 16 bit elevation error */
    V_error = (int)input_buffer[3];
    V_error = (V_error & Oxff) << 8;
    V_error += (int)input_buffer[2] & Oxff;

    /* Convert buffer inputs 4,5 to 16 bit azimuth error */
    U_error = (int)input_buffer[5];
    U_error = (U_error & Oxff) << 8;
    U_error += (int)input_buffer[4] & Oxff;

    /* Convert buffer inputs 6,7 to 16 bit total_strength error */
Total_signal_strength = (int) input_buffer[7];
Total_signal_strength = (Total_signal_strength & 0xff) << 8;
Total_signal_strength += (int) input_buffer[6] & 0xff;

/* Change integer value to a floating point value for normalizing*/
Ch_ptr->Signal_strength = (double) Total_signal_strength - Offset;

if (Ch_ptr-> Signal_strength < 0.00006)
  {
    /* Set signal_strength to small number if equal 0.0 */
    Ch_ptr-> Signal_strength = 0.000006;
  }

/* Normalize azimuth error */
Ch_ptr->U_error = (double) U_error / Ch_ptr->Signal_strength;

/* Normalize Elevation error */
Ch_ptr->V_error = (double) V_error / Ch_ptr->Signal_strength;

} /* End of read channel data */

void Init_Com_port(void)
{
  int rv = 0;

  /* Initialize RS-232 port COM2 will be for sending data and monitoring
     PC and COM3 will be the receiver for DSP Data*/
  rv = 2 bad com parameter, rv = 3 no uart chip detected */
  if (rv = 4 receive queue allocation error, rv = 4 transmit queue allocation error

  /* the following line will be used for the compadd computer at 115k baud */
  rv = ComOpen(COM1, 8815200, U81S1NONE, 16, 16);

  /* Used for 19.2k option */
  rv = ComOpen(COM2, 8919200, U81S1NONE, 16, 16);

  /* Set up COM2 port to be active */
  ComActive(COM2);

  /* Remove all data from the receive buffer */
  ComFlushRx();

  /* Remove all data from the transmit buffer */
  ComFlushTx();

  /* Set up COM3 port to be active */
  ComActive(COM3);

  /* Remove all data from the receive buffer */
  ComFlushRx();

  /* Remove all data from the transmit buffer */
  ComFlushTx();

} /* End of Init_Com_port */

/* Function Test_Serial_Link is used to test serial link from DSP to PC*/

int Test_Serial_Link(int Chan_Number, int Port_C_Data)
{
  int Stop_Serial_Data[3] = {0xBF, 0x7F, 0xEF};
  int Initiate_Serial_Trans[3] = {0x40, 0x80, 0x10};
  int Data_Link_Length = 0;
  int Serial_Link_Error = 0;
  int Time_out = 0;
  int long test_loop;
  int delay_loop;
  int Error_number; /* Initialize Error number to be sent to
     calling program */
Time_out = 0;
Error_number = 0; /* Initialize error number to zero */

Port_C_Data = Port_C_Data | Initiate_Serial_Tran(Chan_Number - 1);
outp(Port_C_Address, Port_C_Data);

for (test_loop = 1; test_loop <= 50; test_loop++)
{
    Time_out = 0;
    /* Check to see if data is available from the receiver electronics */
    while (ComLenRx() < BLK_SZ)
    {
        Data_Link_Length = ComLenRx();
        Serial_Link_Error = ComError();
        if (Serial_Link_Error != 0)
        {
            /* Display date and time of error*/
            printf("RS232 link error detected
", Serial_Link_Error);
            printf("RS232 message length
", Data_Link_Length);
            Error_number = 1;
            break;
        }
        Time_out++;
        /* Increment timeout count for serial port */
    }
    if (Time_out == 32000)
    {
        printf("Serial link timeout error
");
        Error_number = 2;
        break;
    }
    /* End of while (ComLenRx() < BLK_SZ) */
    Data_Link_Length = ComLenRx();
    /* If data length is greater than report error */
    if (Data_Link_Length == 8) Read_Channel_Data(&Test[Chan_Number - 1]);
    else
    {
        printf("Message length error. Length of message = %", Data_Link_Length);
        ComFlushRx();
    }
    if (Error_number != 0) break; /* Exit for loop if failure has occurred */
    for (delay_loop = 1; delay_loop <= 5000; delay_loop++)
        ComFlushRx();
    /* End of for test loop */
    /* Turn off sync pulse to x channel */
    Port_C_Data = Port_C_Data & Stop_Serial_Data[Chan_Number - 1];
    outp(Port_C_Address, Port_C_Data);
    return Error_number;
}

void Report_Com_Error(int Error_on_link, int Channel)
{
    int Data_length;
    /* Display date and time of error*/
    Data_length = ComLenRx();
    strftime(tmpbuf);
    _strdate(tmpbuf1);
    fprintf(output_ptr, "\n Time & Date of error = %s %s ",
    tmpbuf, tmpbuf1);
    fprintf(output_ptr, "RS232 link error detected %x on Channel %x \n",
    Error_on_link, Channel);
    fprintf(output_ptr, "RS232 message length %x\n", Data_length);
/* Function Serial_Port_Timeout reports timeout failure of serial port */

void Serial_Port_Timeout(int Channel)
{
    printf("Serial Port timeout has occurred on Channel %i", Channel);
    _strtime( tmpbuf ); /* Retrieve system time and date */
    _strdate( tmpbuf1);
    printf("\n Time of timeout = %s", tmpbuf);
    fprintf(output_ptr, "\n Time & Date of error = %s %s ",
    tmpbuf,tmpbuf1);
    printf(output_ptr, "Serial Port timeout has occurred on channel %i",Channel);
}

void Initialize_Dsp_Link(void)
{
    int Stop_Data[3] = (OxBF,Ox7F,OxEF);
    int Initiate_Serial[3] = (0x40,0x80,0x10);
    long int long_delay;
    int Init_Chan;
    int Port_data = (0x00);
    for (Init_Chan = 1; Init_Chan <= 3; Init_Chan++)
    {
        /* Turn on sync pulse to DSP */
        Port_data = Port_data | Initiate_Serial[Init_Chan];
        outp(Port_C_Address,Port_data);
        for (long_delay = 1; long_delay <= 500000; long_delay++);
        /* Turn off sync pulse to channel */
        Port_data = Port_data & Stop_Data[Init_Chan];
        outp(Port_C_Address,Port_data);
    }
}

/* End of function Initialize_Dsp_Link */
/* Track.C: Main executive for controlling the tracking software */

#include <stdio.h>
#include <conio.h>
#include <ctype.h>
#include <time.h>
#include <dos.h>
#include <time.h>
#include <cport.h>
#include <math.h>
#include <stdlib.h>
#include <graph.h>

void Move_Chan_To_Starting_Pos(void);
void Init_Channel_Address(void);
void Compute_Offs_t(int i_chan);
void Init_Array(void);

extern void _fortran POSlTI_(int *ICHANNEL_, int *INTRACK,
   double *U, double *V,
   long *ISTEPA, long *ISTEPB);
extern void _fortran INIT(int *REALTIME);
extern unsigned Control_word_address;
extern unsigned Control_word;
extern unsigned Motor_Pair1_Address;
extern unsigned Motor_Pair2_Address;
extern unsigned Motor_Pair3_Address;
extern unsigned Port_A_Address;
extern unsigned Port_B_Address;
extern unsigned Port_C_Address;
extern int Reset_to_zero;
extern int Set_to_actual;
extern struct Channel data {
   unsigned Time_tag;
   double V_error;
   double U_error;
   double Signal_strength;
   signed char Transmit_Data;
} Channel[3],Test[3];
unsigned Channel_Time_Tag[4];
int Aquire_New_Time_Tag[4];
int Sample_Counts = 100;
int Sample_Count[4] = (0,0,0,0);
int Inter_Count[4] = (0,0,0,0);
double Signal_Pwr[4][6] = (((0.0,0.0,0.0,0.0,0.0,0.0),(0.0,0.0,0.0,0.0,0.0,0.0),(0.0,0.0,0.0,0.0,0.0,0.0),(0.0,0.0,0.0,0.0,0.0,0.0)),
   (0.0,0.0,0.0,0.0,0.0,0.0),(0.0,0.0,0.0,0.0,0.0,0.0));

/* Declare all variables used for 8255 control */
extern unsigned Control_word_address; /* Address of Control word on 8255 */
extern unsigned Port_A_Address; /* Address of port A on 8255 */
extern unsigned Port_B_Address; /* Address of port B on 8255 */
extern unsigned Port_C_Address; /* Address of port C on 8255 */
extern int Reset_to_zero; /* These 2 variables will be used global for setting motor position */
extern int Set_to_actual;

/* Declare all variables used for 8255 control */
extern unsigned Control_word_address; /* Address of Control word on 8255 */
extern unsigned Motor_Pair1_Address; /* Address to first pair of motor controllers */
extern unsigned Motor_Pair2_Address; /* Address to first pair of motor controllers */
extern unsigned Motor_Pair3_Address; /* Address to first pair of motor controllers */
extern unsigned Control_word_address; /* Address to which 8255 command word is being set */
extern unsigned Control_word; /* Sets all ports A,B to inputs & C to outputs */

/* Array of data used to read in RS232 data back from the DSP */
extern struct Channel_data {
   unsigned Time_tag;
   double V_error;
   double U_error;
   double Signal_strength;
   signed char Transmit_Data;
} Channel[3],Test[3];
unsigned Channel_Time_Tag[4];
int Aquire_New_Time_Tag[4];

/* Declarations for channel bias offsets */
double Uoffset[6] = {0.0,0.0,0.5,0.446};
double Voffset[4] = {0.0,0.0,-0.446,0.390};
double UDelta[6] = (0.00, 1.0, -2.0, 1.0, 0.00, 0.00);
double V_Delta[6] = (0.00, 0.00, 0.00, -2.0, 0.00, 0.00);
double MaxU_Delta[6] = (1.00, 1.00, 1.00, 2.00, 0.00, 0.00);
double MaxV_Delta[6] = (1.00, 1.00, 1.00, 2.00, 0.00, 0.00);
double Stepsize = .05;

/* Threshold for determining intrack condition */
int Sample_Counts = 100;
int Sample_Count[4] = (0,0,0,0);
int Inter_Count[4] = (0,0,0,0);
double Signal_Pwr[4][6] = (((0.0,0.0,0.0,0.0,0.0,0.0),(0.0,0.0,0.0,0.0,0.0,0.0),(0.0,0.0,0.0,0.0,0.0,0.0),(0.0,0.0,0.0,0.0,0.0,0.0)),
   (0.0,0.0,0.0,0.0,0.0,0.0),(0.0,0.0,0.0,0.0,0.0,0.0));
double track_threshold[4] = (0.5, 0.5, 0.5, 0.5);
double High_threshold = 0.5;
double Low_threshold = 0.1;

unsigned int Offset_Position[3][4];
unsigned int Channel_Address[4];
extern FILE *output_ptr; /* Pointer for file to output error data */
extern FILE *Data_Ptr; /* Pointer for file to output test data */
extern FILE *System_Ptr; /* Pointer for file to read and store system parameters */
extern FILE *AzEL_Ptr; /* Pointer for file to read system parameters */
extern FILE *Atign_Ptr; /* Pointer for file to read system parameters */
extern FILE *Track_AzEl_Ptr; /* Pointer for file to read system parameters */
extern FILE *Input_Ptr; /* Pointer for file to read system parameters */
extern int Starting_Position[2][3];
extern int Selected_Channels[3][2];
int Port_Data = 0x00; /* Initialize variable for port c data reg. */
int Pwr_24Volts_On = 0x20; /* Variable used to turn on 24 volt supply */
int Pwr_24Volts_Off = 0xDF; /* Variable used to turn off 24 volt supply */
int Track_Option; /* Flag used to determine what tracking option to do */
long int In_track = 0; /* Flag passed to control software indicating intrack position */
long int Track_Flag[4] = (0, 0, 0, 0); /* Array used to used to store intrack flag */
double U_signal = 0.0; /* Passes U error to control software */
double V_signal = 0.0; /* Passes V error to control software */
long int I_stepA = 0; /* Control software commanded position for arm */
long int I_stepB = 0; /* Control software commanded position for drum */
long int Act_pose3];
long int Present_Position[3];
int real_time = 1; /* Flag used to indicate real_time mode */
int Motor_Command[3]; /* Size of message from DSP in bytes */
int First_time_flag = 1; /* Set to zero for open loop testing */
int first_time_in_track[4] = (0, 0, 0, 0);
int Out_of_track_count[4] = (0, 0, 0, 0);
int Loop_count = 1;
int count, loop;
int Number_of_Samples = 0;

extern int Channel_to_Run[4];
extern int Number_of_Channels_To_Run;
int Track_Channel = 1;
int l_channel = 2;
int Active_Channel;

/* Array used to disable sync pulse generation */
int Disable_Sync_Pulse[4] = (0x00, 0xBF, 0x7F, 0xEF);
/* Array used enable sync pulse generation */
int Enable_Sync_Pulse[4] = (0x00, 0x40, 0x80, 0xa0);
char tmpbuf[128]; /* Char array for storing time data */
char tmpbuf1[128]; /* Char array for storing time data */
long int delay;

/* Set up function prototype */
void Compute_OpenLoop_Commands(void);
void Wait_for_key_Press(void);
void Check_for_Key_Pressed(void);
void Sync_Serial_Link(void);
unsigned Link_error = 0;
unsigned Data_length = 0;
unsigned Time_out_count = 0;
int Record_Flag[4] = (0, 0, 0, 0); /* Set recording data option to zero */
char key_buffer[80];
int Align = 0;
int mode = 1;
void main ()
{
    /* Assign channel assignments to data array */
    Channel[0].Transmit_Data=0;
    Channel[1].Transmit_Data=1;
    Channel[2].Transmit_Data=2;
    Screen_Prompt();
    _clearscreen(_GCLEARSCREEN);
    Open_files(mode); /* Open all required files mode = 1 Indicates track mode */
    /* Output control word to 8255 */
    Port_Data = 0x00;
    outp(Control_word_address, Control_word);
    outp(Port_C_Address,Port_Data);
    Init_Array();
    Select_Channels_to_Run(); /* Request the user what channels to run */
    Select_Azel_Changes(); /*Request user for new azimuth and elevation parameters*/
    /* Close files so that Avtec software can read data */
    fclose(Input_Ptr);
    fclose(Track_AzEt_Ptr);
    INIT(&Real_time); /* Initialize Avtec control software */
    Init_Channel_Address(); /* Address of n_tor controller assignment*/
    /* Turn off 24 Volt power supply */
    Port_Data = Port_Data & Pwr_24Volts_Off;
    outp(Port_C_Address,Port_Data);
    Initialize_Controllers(); /* Initialize controllers to a predefined parameters */
    /* Turn on 24 Volt power supply */
    Port_Data = Port_Data | Pwr_24Volts_On;
    outp(Port_C_Address,Port_Data);
    for (delay = 1; delay <= 70000; delay++)
    {
        Check_Switch_Status();
        Check_System_Power();
        for (loop = 1; loop <= Number_of_Channels_To_Run; loop++)
        {
            track_threshold[loop] = High_threshold;
            I_channel = Channel_to_Run[loop];
            Offset_Position[0][I_channel] = Starting_Position[0][I_channel-1];
            Offset_Position[1][I_channel] = -Starting_Position[1][I_channel-1];
        }
        Move_CHAN_To_Starting_Pos();
        /* Initialize serial port for receiving data */
        Init_Com_port(); /* Initialize com port for receiving data*/
        _clearscreen(_GCLEARSCREEN);
        printf("\n\n Tracking program is now running !! ");
        printf("\n\n Press 'e' to exit program ");
        printf("\n\n ** Channel Status ** ");
        for (loop = 1; loop <= Number_of_Channels_To_Run; loop++)
        {
            I_channel = Channel_to_Run[loop];
            Port_Data = Port_Data | Enable_Sync_Pulse[I_channel];
        }
    }
outp(Port_C_Address,Port_Data); /* Send out command to initiate sync pulses */

/* Sync host pc to receiver electronics */
Sync_Serial_Link();

while(1)
{
  for (Track_Channel = 1; Track_Channel <= Number_of_Channels_To_Run;
       Track_Channel++)
  {
    I_channel = Channel_to_Run[Track_Channel];
    /* If only one channel is running make sure that sync pulse goes
     high before continuing */
    if (Number_of_Channels_To_Run == 1)
      while (I_channel == Check_for_Active_Sync_Pulse());
    /* Wait till sync pulse goes low */
    while (I_channel != Check_for_Active_Sync_Pulse());
    Active_Channel = Check_for_Active_Sync_Pulse();
    /* Check to see if data is available from the receiver electronics
     * Sync pulse is still active*/
    port_data = Port_Data | Ox01;
    outp(Port_C_Address,Port_Data); /* Send out command to initiate sync pulses */
    if (ComLenRx() < BLK_SZ) &&
      (I_channel == Active_Channel))
      { Link_error = ComError();
        if (Link_error != 0)
          Report_Com_Error(Link_error, I_channel);
        Active_Channel = Check_for_Active_Sync_Pulse();
      } /* End of while (ComLenRx() < BLK_SZ) */
    Port_Data = Port_Data | Ox01;
    outp(Port_C_Address,Port_Data); /* Send out command to initiate sync pulses */
    if (ComLenRx() >= BLK_SZ)
      { Read_Channel_Data(&Channel[I_channel - 1]);
        if (Aquire_New_Time_Tag[I_channel] == 1)
          { Channel_Time_Tag[I_channel] = Channel[I_channel - 1].Time_tag;
            Aquire_New_Time_Tag[I_channel] = O;
          }
        else
          { Channel[I_channel - 1].Signal_strength = 1.0;
            ComFlushRx();
          }
        if (Channel[I_channel - 1].Time_tag == Channel_Time_Tag[I_channel])
          { /* Increment next time tag for next frame comparison */
            Channel_Time_Tag[I_channel]++;
          }
        /* Test to see if errors are out of range */
        if (((Channel[I_channel - 1].Signal_strength/204.7) >= track_threshold[I_channel])
          && (fabs(Channel[I_channel - 1].V_error) < 1.0) &&
          (fabs(Channel[I_channel - 1].U_error) < 1.0))
          { Track_Flag[I_channel] = 1;
            Sample_Count[I_channel]++;
            if (Sample_Count[I_channel] > 5)
              Signal_Pwr[I_channel][Inter_Count[I_channel]] =
              Channel[I_channel - 1].Signal_strength +
              Signal_Pwr[I_channel][Inter_Count[I_channel]];
if (Sample_Count[I_channel] == Sample_Counts)
{
    Compute_Offset(I_channel);
    Sample_Count[I_channel] = 0;
}

/* Convert normalized values into microradians */
/* Channel 1 & 2 have a sign reversal */
if (I_channel == 3)
{
    U_signal = (Channel[I_channel - 1].U_error + Uffset[I_channel])
    * -0.0002718;
    V_signal = (Channel[I_channel - 1].V_error + Voffset[I_channel])
    * -0.0002453;
}
else
{
    U_signal = (Channel[I_channel - 1].U_error + Uffset[I_channel])
    * 0.0002718;
    V_signal = (Channel[I_channel - 1].V_error + Voffset[I_channel])
    * 0.0002453;
}

if (first_time_in_track[I_channel] == 1)
{
    _settextposition(Track_Channel + 7,2);
    /* printf("\nChannel %i in track mode",I_channel);*/
    track_threshold[I_channel] = Low_threshold;
    Out_of_track_count[I_channel] = 0;
    first_time_in_track[I_channel] = 0;
}
else
{
    if (Out_of_track_count[I_channel] > 5)
    {
        Track_Flag[I_channel] = 0;
        first_time_in_track[I_channel] = 1;
        track_threshold[I_channel] = High_threshold;
        /* Print out og track condition */
        if (Out_of_track_count[I_channel] == 6)
        {
            _settextposition(Track_Channel + 7,2);
            /* printf("\nChannel %i out of track",I_channel);*/
        }
    }
    Out_of_track_count[I_channel] ++; /* Increment out of track counter */
}
V_signal = 0; /* Set U and V errors to zero so motors will */
U_signal = 0; /* not be command to move on bad data */

In_track = Track_Flag[I_channel];
Read_Actual_Pos(Channel_Address[I_channel],Act_pos);
I_step8 = Offset_Position[O][I_channel] - Act_pos[O];
I_stepA = Act_pos[1] - Offset_Position[1][I_channel];
/* Call Avtec control software for new position commands */
POSITION(&Track_Channel, &In_track, &U_signal, &V_signal, &I_stepA, &I_stepB);
/* Set up the commanded position for the Disk8 Motor */
Motor_Command[0] = Offset_Position[0][I_channel] - I_step8;
/* Set up the commanded position for the DiskA Motor */
Write_Final_pos(Channel_Address[I_channel], Motor_Command);
370 Write_to_flag_register (Channel_Address[I_channel],&x0808);
371 /* Output data to file if requested */
372 if (Record_Flag[I_channel] == 1)
373 {
374   
375   fprintf(Data_Ptr," %xu, %x+3.3f, %x+3.3f, %x+9li, %x+9li, %x+9li, %x+9li, %x+9li",
376       Channel[I_channel].Time_tag,
377       Channel[I_channel].Signal_strength/204.7,
378       V_signal * 1000000,u_signal * 1000000,
379       Present_Position[0],Motor_Command[0],I_step8,
380       Present_Position[1],Motor_Command[1],l_stepA);
381     Number_of_Samples ++; /* Record number of samples taken */
382     
383     if (Number_of_Samples >= 1201)
384     {
385       Record_Flag[I_channel] = 0;
386       Number_of_Samples = 0;
387       printf("Max samples taken !!!");
388     }
389     
390     } /* End of if record flag = 1 */
391  
392  } /* End of if Channel.Time_tag == Time_Tag */
393  
394  else
395  {
396    Sync_Serial_Link();
397    break;
398  }
399  
400  /* Send out command to toggle line that frame is done */
401  Port_Data = Port_Data & 0xFE;
402  outp(Port_C_Address,Port_Data); /* Send out command to initiate sync pulses */
403  
404  Check_for_Key_Pressed();
405  
406  } /* End of for I_channel = 1 to 3 */
407  
408  } /* End of while = 1 */
409  
410  } /* End of main function */
411  
412  /* Function Init_Channel_Address of motor controller assignment */
413  void Init_Channel_Address(void)
414  {
415    Channel_Address[I] = Motor_Pair1_Address;
416    Channel_Address[2] = Motor_Pair2_Address;
417    Channel_Address[3] = Motor_Pair3_Address;
418  }
419  
420  /* Close_Down function will shut down the system in a orderly fashion */
421  void Close_down(void)
422  {
423    
424    int Ch;
425    
426    ComCloseAll(); /* Close all serial ports */
427    
428    for (Ch = 1; Ch <= 3; Ch ++)
429    {
430    /* Disable sync pulse to all channels */
431    Port_Data = Port_Data & Disable_Sync_Pulse[Ch];
432    outp(Port_C_Address,Port_Data);
433    
434    } /* Turn off 24 Volt power supply */
435    Port_Data = Port_Data & Pwr_24Volts_Off;
436    outp(Port_C_Address,Port_Data);
437    
438    /* Set all position registers to actual position to stop motors */
439    for (Ch = 1; Ch <= 3; Ch ++)
440    Set_Motor_Position(Channel_Address[Ch],Set_to_actual);
void Move_Chan_To_Starting_Pos(void)
{
    long int Initial_Pos[3];
    long int Target_Position[2][4];
    long int Target[3];
    int At_Target = 0;
    int Delta_Limit = 25; /* Delta between target and actual move to indicate the arm is at its designated position */

    for (Track_Channel = I; Track_Channel <= Number_of_Channels_To_Run; Track_Channel++)
    {
        i_channel = Channel_to_Run[Track_Channel];
        V_signal = O;
        U_signal = O;

        /* Call control software to get initial position to were arm should go */
        POSITION(&Track_Channel, &In_track, &U_signal, &V_signal, &I_stepA, &I_stepB);

        /* Set up the commanded position for the Disk B Motor */
        Motor_Command[0] = Offset_Position[0][i_channel] - I_stepB;
        Target_Position[0][I_channel] = Motor_Command[0];

        /* Set up the commanded position for the Disk A Motor */
        Motor_Command[1] = Offset_Position[1][i_channel] + I_stepA;
        Target_Position[1][I_channel] = Motor_Command[1];

        /* Write out the new motor positions that we will need to move to */
        Write_Final_pos(Channel_Address[i_channel], Motor_Command);

        /* Start Trapizodial move */
        Write_to_flag_register (Channel_Address[i_channel], Ox0808);

        printf("Channets being sent to their starting positions");

        /* Wait for channels to reach their destinations */
        for (Track_Channel = 1; Track_Channel <= Number_of_Channels_To_Run; Track_Channel++)
        {
            i_channel = Channel_to_Run[Track_Channel];

            Target[0] = Target_Position[0][I_channel];
            Target[1] = Target_Position[1][I_channel];

            while (At_Target != 0)
            {
                Check_for_Key_Pressed();

                At_Target = Reached_Commanded_Pos(Channel_Address[i_channel], Target, i_channel, Delta_Limit);
            }

            printf("Channels have reached their starting positions");
        }
    }
}

/* End of function Move_Channels_To_Start */
518 */ Function Check_for_Key_Pressed (void) will be used check for an exit key */
519 void Check_for_Key_Pressed(void)
520 {
521     char Key_number;
522     if (kbhit())
523     {
524         Key_number = getch();
525         switch(Key_number)
526         {
527             case 'e': Close_Down();
528             default: break;
529         }
530         /* End of switch statement */
531         /* End of if kbhit() */
532     }
533 }
534 /* End of function Check_for_Key_Pressed */
535
536 void Sync_Serial_Link(void)
537 {
538     /* Run through one cycle of sync pulses to synchronize communications */
539     for (Track_Channel = 1; Track_Channel <= Number_of_Channels_To_Run; Track_Channel++)
540         ....
541         I_channel = Channel_to_Run[Track_Channel];
542         while (I_channel != Check for Active_Sync_Pulse());
543         while (I_channel == Check for Active_Sync_Pulse());
544         Aquire_New_Time_Tag[I_channel] = 1;
545         ....
546         ComFlushRx(); /* Flush out data before starting loop after synchronizing
547         serial link */
548 }
549 }
550
551 void Compute_Offset(int I_chan)
552 {
553     double Max_Sig;
554     int Number_of_Max;
555     int loop;
556     
557     Inter_Count[I_chan]++;
558     if ((Inter_Count[I_chan] >= 0) && (Inter_Count[I_chan] <= 4))
559     {
560         Offset[I_chan] = Offset[I_chan] + U_DELTA[Inter_Count[I_chan]];
561         Offset[I_chan] = Offset[I_chan] + V_DELTA[Inter_Count[IChan]];
562     }
563     else
564     {
565         Inter_Count[I_chan] = 0;
566         Max_Sig = 0;
567         /* Determine max power signal for x samples */
568         for (loop = 0; loop <= 4; loop++)
569         {
570             if (abs(Signal_Pwr[I_chan][loop] - Max_Sig) > Max_Sig)
571             {
572                 Max_Sig = Signal_Pwr[I_chan][loop];
573                 Number_of_Max = loop;
574             }
575         }
576         /* End of for loop 0 to 4 */
577     }
578     if (IChan == 2)
579         fprintf(Data_Ptr, 
580         /* Channel[1][chan - 1].Time_tag, Signal_Pwr[IChan][0], */
581         /* Channel[IChan - 1].Time_tag, Signal_Pwr[IChan][0], */
582         /* Channel[IChan - 1].Time_tag, Signal_Pwr[IChan][0], */
583     }
592      Signal_Pwr[l_chan][1], Signal_Pwr[l_chan][2],
593      Signal_Pwr[l_chan][3], Signal_Pwr[l_chan][4],
594      Max_Sig, Number_of_Max);
595
598
599      /* Reinitialize signal strength array back to zero */
600      for (loop = 0; loop <= 5; loop++)
601          Signal_Pwr[l_chan][loop] = 0;
602      ) /* End of Inter_Count[l_chan] == 5 */
603
604      ) /* End of function Compute_offset */
605
606    void Init_Array(void)
607    {
608      int i;
609      for (i = 0; i<=5; i++)
610      {
611          U_Delta[i] = U_Delta[i] * Stepsize;
612          V_Delta[i] = V_Delta[i] * Stepsize;
613          MaxU_Delta[i] = MaxU_Delta[i] * Stepsize;
614          MaxV_Delta[i] = MaxV_Delta[i] * Stepsize;
615      }
616    }
617
618    void Wait_for_key_Press(void)
619    {
620      printf("Press any key to continue");
621      /* Display message until key is pressed. */
622      while (lkbhit()); /* Wait for key on keyboard to be pressed */
623      /* Use getch to throw key away. */
624      while (kbhit()) getch();
625    }
<p>| 1  | 5.5  | Predicted azimuth of LEO 1 (deg) |
| 2  | 3.0  | Predicted elevation of LEO 1 (deg) |
| 3  | -7.5 | Predicted azimuth of LEO 2 (deg) |
| 4  | 4.0  | Predicted elevation of LEO 2 (deg) |
| 5  | 8.0  | Predicted azimuth of LEO 3 (deg) |
| 6  | -3.5 | Predicted elevation of LEO 3 (deg) |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.02</td>
<td>! Duration of simulation in hours (REAL*8)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>! Number of disc channels (INTEGER)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>! Tracking rate in Hz (INTEGER)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.5</td>
<td>! Tracking gain (REAL*8)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>.FALSE.</td>
<td>! Write flag for detailed output (LOGICAL)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>42164.</td>
<td>! HEO orbital elements at time 0 (REAL*8)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>0.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>! Number of LEO satellites (INTEGER)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>6528.</td>
<td>! LEO #1 orbital elements at time 0 (REAL*8)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>0.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>30.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>25.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>20.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>0.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>6878.</td>
<td>! LEO #2 orbital elements at time 0 (REAL*8)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>0.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>28.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>345.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>310.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>0.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>6528.</td>
<td>! LEO #3 orbital elements at time 0 (REAL*8)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>0.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>30.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>330.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>0.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>0.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
.module/ram/boot=0 adsys;

(Module ad_conv will be used to for all A/D convertor related calls )

(Number of samples per frame)

.Counter used to track the number of data points taken)

(Counts number of convert pulses to A/D)

(Send_convert will be used to generate the convert pulse to the A/D)

 ENTRY send_convert;

send_convert:

ENA Sec_Reg;  (Select secondary registers)

ax1 = dm(Points_per_Frame);

ay1 = dm(Convert_pulse_count);

ar = ax1 - ay1;

if eq jump Stop_convert;  (Test to see if enough points)

( have been processed if so exit interrupt)

PM(14,M7) = ax0;  (Write out to the program memory to)

(A low pulse of 80+ usec. will be )

(A init converter of analog data)

ar = ay1 + 1;  (Increment the number of pulse counts

in present frame)

dm(Convert_pulse_count) = ar;

Stop_convert:

DIS Sec_Reg;  (Switch registers back to primary mode)

rti;

(Initialize_AD initializes A/D with doing one read from the A/D)

 ENTRY Initialize_AD;

Initialize_AD:

ay0 = PM(17,M7);

rts;

(Declare array that for storage of raw data from A/D All four channels

will be stored in array. Assignment is as follows:

Channel 1 will store in elements 1,5,9...

Channel 2 will store in elements 2,6,10...

Channel 3 will store in elements 3,7,11...

Channel 4 will store in elements 4,8,12...)

(*************************************************************************)
(Read data interrupt from A/D
& reads data from all four A/D channels)

ENTRY Read_data_int;

Read_data_int:

EA Sec_Reg; (Select secondary registers)

se = 4; (Set up se register for multifunction shift inst)

ar = PM(17,M7); (Read 1st channel of data from A/D)

sr = ASHIFT ar BY 4 (HI); (Shift data over to right)

sr = ASHIFT sr1 BY 4 (HI); (to extend sign bit and rescale back)

sr = ASHIFT sr1 BY 4 (HI); (to proper scale and store data into array)

ar = PM(17,M7); (Read 2nd channel of data from A/D)

DM(16,M5) = sr1, sr = ASHIFT ar (HI); (Shift data over to right to extend sign)

sr = ASHIFT sr1 BY -4 (HI); (bit and shift back left to rescale & store data for channel 1 in array)

sr = ASHIFT sr1 BY -4 (HI); (for channel 1 in array)

ar = PM(17,M7); (Read 3rd channel of data from A/D)

DM(16,M5) = sr1, sr = ASHIFT ar (HI); (Shift data over to right to extend sign)

sr = ASHIFT sr1 BY -4 (HI); (bit and shift back left to rescale & store data for channel 2 in array)

ar = PM(17,M7); (Read 4th channel of data from A/D)

DM(16,M5) = sr1, sr = ASHIFT ar (HI); (Shift data over to right to extend sign)

sr = ASHIFT sr1 BY -4 (HI); (bit and shiftback left to rescale & store data)

sr = ASHIFT sr1 BY -4 (HI); (for channel 3 in array)

DM(16,M5) = sr1; (Store data for channel 4 in array)

ayl = dm(Data_Ready_to_be_Processed); (Increment number of points read in)

ar = ayl + 1;

DM(Data_Ready_to_be_Processed) = ar;

DIS Sec_Reg; (Switch registers back to primary mode)

rd;

.endmod;
Module/ram/boot=0 Algorithm;
(Module algorithm will be used to compute the error signal for all four quadrants on the detector)

Total number of samples that will be stored at any one time

(number_of_samples = 40 used for eprom version 40 samples equals 10 samples of data per quadrant)

(const number_of_filter_samples = 40);

(number_of_samples = 20000 used for emulator version 200 samples equals 500 samples of data per quadrant)

(Declare array that for storage of the outputs from the bandpass filter will be stored in array. Assignment is as follows:
Channel 1 will store in elements 1, 5, 9...
Channel 2 will store in elements 2, 6, 10...
Channel 3 will store in elements 3, 7, 11...
Channel 4 will store in elements 4, 8, 12...
)

(Array to store the output of the bandpass filter)
var/dm/ram/circ Filtered_data[number_of_filter_samples];
var/dm/ram/circ Filtered_data;

(Array to store the output of the bandpass filter)
var/dm/ram/circ Lowpass_Filt_data[number_of_filter_samples];
var/dm/ram/circ Lowpass_Filt_data;

(Pointer, length, and count increment of array for bandpass filter)
var/dm/ram Filtered_data_add_ptr;
var/dm/ram Filtered_data_len_ptr;
var/dm/ram Filtered_data_count_ptr;
var/dm/ram Filtered_data_add_ptr;
var/dm/ram Filtered_data_len_ptr;

(Pointer, length, and count increment of array for Lowpass filter)
var/dm/ram Low_Filt_data_add_ptr;
var/dm/ram Low_Filt_data_len_ptr;
var/dm/ram Low_Filt_data_count_ptr;
var/dm/ram Low_Filt_data_add_ptr;
var/dm/ram Low_Filt_data_len_ptr;

(The following variables will be the final results from a frames worth of data)
var/dm/ram Total_signal_strength;
var/dm/ram Elevation_error;
var/dm/ram Azimuth_error;
var/dm/ram Total_signal_strength;
var/dm/ram Elevation_error;
var/dm/ram Azimuth_error;

external Process_data_add_ptr;
external Process_data_len_ptr;
external Bandpass_Filt;
external Lowpass_Filt;

external Delay_1;  (Bandpass filter delays for channel 1)
external Delay_2;  (Bandpass filter delays for channel 2)
.external Delay_3; (Bandpass filter delays for channel 3)
.external Delay_4; (Bandpass filter delays for channel 4)

.external Pointer_1; (Bandpass filter delays for channel 1)
.external Pointer_2; (Bandpass filter delays for channel 2)
.external Pointer_3; (Bandpass filter delays for channel 3)
.external Pointer_4; (Bandpass filter delays for channel 4)

.external Delay_1L; (Lowpass filter delays for channel 1)
.external Delay_2L; (Lowpass filter delays for channel 2)
.external Delay_3L; (Lowpass filter delays for channel 3)
.external Delay_4L; (Lowpass filter delays for channel 4)

.external Pointer_1L; (Lowpass filter delays for channel 1)
.external Pointer_2L; (Lowpass filter delays for channel 2)
.external Pointer_3L; (Lowpass filter delays for channel 3)
.external Pointer_4L; (Lowpass filter delays for channel 4)

.ENTRY Filter_error_signal;

Filter_error_signal:

(Filter channel 1 thru bandpass)
L5 = %Delay_1; (Set up the length of the filter delay array for the FIR bandpass filter)
I5 = DM(Pointer_1); (15 points to delay elements)
MY1 = DM(10,MO); (Send channel 1 data to input of Bandpass filter)
Call Bandpass_filt; (Run data point from channel 1 thru bandpass filter)
DM(Pointer_1) = I5; (Store pointer of last sample to be saved)
DM(12,MO) = MR1; (Store output from filter)

(Filter channel 2 thru bandpass)
I5 = DM(Pointer_2); (15 points to delay elements)
MY1 = DM(10,MO); (Send channel 2 data to input of Bandpass filter)
Call Bandpass_filt; (Run data point from channel 2 thru bandpass filter)
DM(Pointer_2) = I5; (Store pointer of last sample to be saved)
DM(12,MO) = MR1; (Store output from filter)

(Filter channel 3 thru bandpass)
I5 = DM(Pointer_3); (15 points to delay elements)
MY1 = DM(10,MO); (Send channel 3 data to input of Bandpass filter)
Call Bandpass_filt; (Run data point from channel 3 thru bandpass filter)
DM(Pointer_3) = I5; (Store pointer of last sample to be saved)
DM(12,MO) = MR1; (Store output from filter)

(Filter channel 4 thru bandpass)
I5 = DM(Pointer_4); (15 points to delay elements)
MY1 = DM(IO, MO);  
Call Bandpass_filt;  
DM(Pointer_4) = I5;  
DM(I2, MO) = MR1;  

(Change I2 and L5 for lowpass filter compututation)  
Modify(I2, M3);  
I5 = %delay_1L;  

(Filter channel 1 thru lowpass filter)  
I5 = DM(Pointer_1L);  
ax0 = DM(I2, MO);  
ar = abs ax0;  
my1 = ar;  
Call Lowpass_filt;  
DM(Pointer_1L) = I5;  
DM(I3, MO) = MR1;  

(Filter channel 2 thru lowpass)  
I5 = DM(Pointer_2L);  
ax0 = DM(I2, MO);  
ar = abs ax0;  
my1 = ar;  
Call Lowpass_filt;  
DM(Pointer_2L) = I5;  
DM(I3, MO) = MR1;  

(Filter channel 3 thru lowpass)  
I5 = DM(Pointer_3L);  
ax0 = DM(I2, MO);  
ar = abs ax0;  
my1 = ar;  
Call Lowpass_filt;  
DM(Pointer_3L) = I5;  
DM(I3, MO) = MR1;  

(Send channel 4 data to input of Bandpass filter)  
(Run data point from channel 4 thru bandpass filter)  
(Store pointer of last sample to be saved)  
(Store output from filter)  
(Decrement the address pointer I2 by the value of M3)  
(Set up length of FIR lowpass filter delay line)  
(15 points to delay elements )  
(Send channel 1 data to input of Bandpass filter)  
(Rectify signal by taking the absolute value of ax0)  
(Input parameter to lowpass filter)  
(Run data point from channel 1 thru bandpass filter)  
(Store pointer of last sample to be saved)  
(Store output from filter)  
(15 points to delay elements )  
(Send channel 2 data to input of Bandpass filter)  
(Rectify signal by taking the absolute value of ax0)  
(Input parameter to lowpass filter)  
(Run data point from channel 2 thru bandpass filter)  
(Store pointer of last sample to be saved)  
(Store output from filter)  
(15 points to delay elements )  
(Send channel 3 data to input of Bandpass filter)  
(Rectify signal by taking the absolute value of ax0)  
(Input parameter to lowpass filter)  
(Run data point from channel 3 thru bandpass filter)  
(Store pointer of last sample to be saved)  
(Store output from filter)  

(Fiter channel 1 thru lowpass)  
(Fiter channel 2 thru lowpass)  
(Fiter channel 3 thru lowpass)
\begin{verbatim}
222      (filter channel 4 thru lowpass)
223      15 = DM(Pointer_4L);  \hfill (15 points to delay elements )
224      ax0 = DM(12,MO); \hfill (Send channel 4 data to input of
225      ar = abs ax0;  \hfill Bandpass filter)
226      my1 = ar; \hfill (Rectify signal by taking the
227      Call Lowpass_filt; \hfill absolute value of ax0)
228      (Run data point from channel 4
229      DM(Pointer_4L) = 15; \hfill thru Bandpass filter)
230      DM(13,MO) = MR1; \hfill ($store pointer of last sample to be
231      (Increment the number of data points processed through algorithm)
232      ay1 = dm(Number_of_Data_Points_Processed);
233      ar = ay1 + 1;
234      dm(Number_of_Data_Points_Processed) = ar;
235                   \hfill (store output from filter)
236                   \hfill (Rove address pointer back four places
237                   \hfill to pull out last computed values from frame)
238                   \hfill (Store quadrant A data (channel 1) in ax0)
239                   \hfill (Store quadrant B data (channel 2) in ay0)
240                   \hfill (Store quadrant C data (channel 3) in ax1)
241                   \hfill (Store quadrant D data (channel 4) in ay1)
242                          \hfill (Calculate total signal strength)
243      ar = ax0 + ay0; \hfill (Compute Quadrant A + B magnitude)
244      af = ax1 + ay1; \hfill (Compute Quadrant C + D magnitude)
245      DM(Var Total_signal_strength) = ar;
246                          \hfill (Sum all Quardrants A + B + C + D)
247      ar = ar + af; \hfill (Compute Azimuth error)
248      DM(Var Azimuth_error) = ar;
249                          \hfill (Compute elevation)
250      ar = ax0 + ay0; \hfill ( Compute Quadrant A + B magnitude)
251      ar = ar - af; \hfill (Subtract Quardrants A + B - C - D where
252      DM(Var Elevation_error) = ar; \hfill af = C + D)
253                          \hfill (Compute elevation)
254      ar = ax0 + ay1; \hfill (Add Quadrant A + D)
255      af = ax1 + ay0; \hfill (Add Quadrant C + B)
256      ar = ar - af; \hfill (Subtract Quadrant (A + D) - (C + B) )
257      DM(Var Elevation_error) = ar; \hfill (Store Azimuth error)
258                   \hfill RTS;
259                   \hfill ENDMOD;
\end{verbatim}
(* Analog Dev. ADSP-210x Filter Realization Copyright (C) by HYPERCEPTION, INC. *)

Filter Generated from File: kwband.FIR

Filter Order=42

Direct form Realization of the following convolution sum:

\[ y(n) = \sum_{k=0}^{N-1} h(k) x(n-k) \]

(*)

The following main module example may be copied to a separate filename (e.g. Main.dsp) and uncommented to provide a test of the filter. A typical example of assembly and link commands needed to produce executable filter test code would then be:

dspma Main <cr>
dspma kwband <cr>
dspml Main kwband -a archname.ACH -g <CR>

NOTE: The Linker expects an Architecture Description File (.ACH) which is created by the System Builder.

*)

.MODULE /RAM/boot=0 FIR_Filt; ( Q15-Notation (scaling) is in effect )

.CONST Length=42; ( Length of impulse response )

.VAR/P murdered CIRC Delay_1[Length];

.VAR/P murdered CIRC Delay_2[Length];

.VAR/P murdered CIRC Delay_3[Length];

.VAR/P murdered CIRC Delay_4[Length];

.Global Delay_1;

.Global Delay_2;

.Global Delay_3;

.Global Delay_4;

.VAR/DM Pointer_1[1];

.VAR/DM Pointer_2[1];

.VAR/DM Pointer_3[1];

.VAR/DM Pointer_4[1];

.global Pointer_1;

.global Pointer_2;

.global Pointer_3;

.global Pointer_4;

.VAR/DM Coefficients[Length];

global Coefficients;
VAR/PM Band_Coefficients[Length];
.global Band_Coefficients;

( Current circular buffer pointer )

( NOTE: The user may wish to store the following coefficients to an external .DAT file in hexadecimal format. Keep in mind that these source code coefficients have been quantized. )

.INIT Band_Coefficients: 13,-78,-121,40,316,271,-293,
-773,-305,889,1305,-17,-1780,-1598,
2773,-532,-3072,-1876,1350,2653,806,
-1598,-1780,-17,1350,889,-305,-773,
316,271,316,40,-121,-78,13;

.INIT Delay_1: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.INIT Delay_2: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.INIT Delay_3: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.INIT Delay_4: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.ENTER Bandpass_Filt;

( Input is in MYI register, )
( output is in MR1 register. )

Bandpass_Filt:

II = 'Coefficients;

I6 points to filter coefficients

CNTR = Length - 2;

MR = 0, MXO = DM(I1,M1), MYO = PM(I5,M5); ( init mr mxo = coeff myo = delay )

DO convolve UNTIL CE;

convolve:

MR = MR + MXO*MYO (ss), MXO = DM(I1,M1), MYO = PM(I5,M5);

( mr = mr + coeff*delay ) get next coeff get next delay )

MR = MR + MXO*MY1 (ss), MY1 = PM(15,M6);

( store sample in delay last multiply )

MR = MR (rnd);

( If MV SAT MR; ) ( If an overflow condition has occurred sat

RTS;

(Init_Bandpass_Coeff subroutine is used to move coefficients from

program memory to data memory )

.ENTER Init_Bandpass_Coeff;

Init_Bandpass_Coeff:

LT = 0;

L5 = 0;

M1 = 1;

MS = 1;

II = 'Coefficients; (Set up pointer for data memory)

I5 = 'Band_Coefficients; (Set up pointer for program memory)

CNTR = Length;
DO init_band_coef UNTIL CE;
    sr1 = PM(15, M5);
    sr0 = px;  \(\text{Store 8 data bits from px into sr0}\)
    sr = sr or \(\text{shift sr1 by -8 (M1)}\);
    init_band_coef: \(DM(11, M1) = sr0;\)
    RTS;
    .ENDMOD;
Filter Generated from File: kwlow.FIR
Filter Order=45

Direct Form Realization of the following convolution sum:

\[ y(n) = \sum_{k=0}^{N-1} h(k)x(n-k) \]

\[ \begin{array}{cccccc}
  & -1 & -1 & -1 & \ldots & -1 \\
  x(n) & \downarrow & \downarrow & \downarrow & \ldots & \downarrow \\
  v h(0) & v h(1) & v h(2) & \ldots & v h(N-2) & v h(N-1) \\
  \downarrow & \downarrow & \downarrow & \ldots & \downarrow & \downarrow \\
  y(n) & 0 & 0 & 0 & \ldots & 0
\end{array} \]

---

MODULE /ram/boot = LOW_FIR_Filt; (Q15-Notation (scaling) is in effect)

CONST Length_low=45; (Length of impulse response)

(Declare delay line arrays for lowpass filter)

VAR/PM/CIRC Delay_1L[Length_low];
VAR/PM/CIRC Delay_2L[Length_low];
VAR/PM/CIRC Delay_3L[Length_low];
VAR/PM/CIRC Delay_4L[Length_low];

.Global Delay_1L;
.Global Delay_2L;
.Global Delay_3L;
.Global Delay_4L;

(Declare pointers that will point to the delay line array elements)

VAR/DM Pointer_1L[1];
VAR/DM Pointer_2L[1];
VAR/DM Pointer_3L[1];
VAR/DM Pointer_4L[1];

.global Pointer_1L;
global Pointer_2L;
global Pointer_3L;
global Pointer_4L;

VAR/DM Coefficients_low[Length_low];
.Global Coefficients_low;

VAR/PM Low_Coeff[Length_low];
.Global Low_Coeff;

.INIT Low_Coeff:
26,35,58,88,127,176,236,
306,387,477,577,683,795,909,
1023,1134,1239,1333,1416,1483,1532,
1563,1573,1563,1532,1483,1416,1333,
1239,1134,1023,909,795,683,577,
477,387,306,236,176,127,88,
58,35,26;

.INIT Delay_1L: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.INIT Delay_2L: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.INIT Delay_3L: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.INIT Delay_4L: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.ENDMOD;

58,35,26;

.INIT Delay_1L: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.INIT Delay_2L: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.INIT Delay_3L: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.INIT Delay_4L: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.ENDMOD;

58,35,26;

.INIT Delay_1L: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.INIT Delay_2L: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.INIT Delay_3L: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.INIT Delay_4L: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.ENDMOD;

58,35,26;

.INIT Delay_1L: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.INIT Delay_2L: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.INIT Delay_3L: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.INIT Delay_4L: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.ENDMOD;

58,35,26;

.INIT Delay_1L: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.INIT Delay_2L: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.INIT Delay_3L: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.INIT Delay_4L: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.ENDMOD;

58,35,26;

.INIT Delay_1L: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.INIT Delay_2L: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.INIT Delay_3L: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.INIT Delay_4L: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.ENDMOD;

58,35,26;

.INIT Delay_1L: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.INIT Delay_2L: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.INIT Delay_3L: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;

.INIT Delay_4L: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
.module/ram/boot=0/abs=0 MAIN;

(Module main will be used to initialize all system parameters )
(and service all interrupts)

.include <c:\nasa\dspcode\ezlab\def2101.h>;  ( file has sys. cntl. registers )

(Total number of samples that will be stored at any one time)

( Used for emulator version )
 (.const number_of_samples = 2000;)

( Used for Eprom version )
 .const number_of_samples = 400;

(Declare array that for storage of raw data from A/D All four channels will be stored in array. Assignment is as follows: Channel 1 will store in elements 1,5,9...
Channel 2 will store in elements 2,6,10...
Channel 3 will store in elements 3,7,11...
Channel 4 will store in elements 4,8,12...
)
.var/dm/ram/circ AD_Raw_Date[number_of_samples];

(Pointer, length, and count increment of array that receives data from A/D)
.var/dm/ram AD_rawdata_add_ptr;
.var/dm/ram AD_rawdata_len_ptr;
.var/dm/ram AD_rawdata_count_ptr;

(Points to the element in the array of raw data that is being processed)
.var/dm/ram Process_data_add_ptr;
.global Process_data_add_ptr;
.var/dm/ram Process_data_len_ptr;  (Length of array)
.global Process_data_len_ptr;

(Counter used to track the number of data points taken)
.var/dm/ram Data_Ready_to_be_Processed;
.global Data_Ready_to_be_Processed;

(Maximum number of data points to be taken per frame)
.var/dm/ram Points_per_Frame;
.global Points_per_Frame;
(.init Points_per_Frame: 100;)  (number of samples of data per frame)

(Counts number of convert pulses to A/D)
.var/dm/ram Convert_pulse_count;
.global Convert_pulse_count;

.count irqO;
.var/dm/ram Read_adc_add_ptr;
.var/dm/ram Convert_pulse_add_ptr;
.var/dm/ram Sync_flag_enable;

(Time tag for data to be sent over to host computer)
.var/dm/ram Frame_tag;
.global Frame_tag;
(.init Frame_tag: 000; )

(Memory mapped I/O declarations)
.port read_adc;  (Mapped I/O address that A/D is read)
.global read_adc;
.port convert_pulse;  (Mapped I/O address that initiates)
.global convert_pulse;  (conversion on A/D)
.port watch_dog;  (Mapped I/O address to reset watchdog timer)
.global watch_dog;

.external Number_of_Data_Points_Processed;  (Counter for tracking number of points processed)
.external Delay_1;  (Bandpass filter delays for channel 1)
.external Delay_2;  (Bandpass filter delays for channel 2)
.external Delay_3;  (Bandpass filter delays for channel 3)
.external Delay_4;  (Bandpass filter delays for channel 4)
.external Pointer_1;  (Bandpass filter delays for channel 1)
.external Pointer_2;  (Bandpass filter delays for channel 2)
.external Pointer_3;  (Bandpass filter delays for channel 3)
.external Pointer_4;  (Bandpass filter delays for channel 4)
.external Delay_1L;  (Lowpass filter delays for channel 1)
.external Delay_2L;  (Lowpass filter delays for channel 2)
.external Delay_3L;  (Lowpass filter delays for channel 3)
.external Delay_4L;  (Lowpass filter delays for channel 4)
.external Pointer_1L;  (Lowpass filter delays for channel 1)
.external Pointer_2L;  (Lowpass filter delays for channel 2)
.external Pointer_3L;  (Lowpass filter delays for channel 3)
.external Pointer_4L;  (Lowpass filter delays for channel 4)
.external Filtered_data_add_ptr;
.external Filtered_data_len_ptr;
.external Filtered_data;
.external Coefficients;
.external Low_Filt_data_add_ptr;
.external Low_Filt_data_len_ptr;
.external Lowpass_Filt_data;
.external Coefficients_low;

(Declarations of external subroutines in other modules)
.external Filter_error_signal;  (Subroutine for filtering)
.external Initialize_AD;  (Subroutine to initialize A/D)
.external Send_convert;  (Subroutine used to generate convert pulse to A/D convertor)
.external Read_Data_Int;  (Subroutine reads in all 4 channels of data from the A/D and sign extends the data)
.external Compute_Tracking_Error;  (Subroutine to compute error signals)
.external Transmit_data;  (Subroutine used to transmitt data to host computer)
.external Send_data;  (Subroutine used to service sportO transmit interrupt)
.external Init_Lowpass_Coeff;  (Subroutine used to initialize Lowpass coefficients)
.external Init_Bandpass_Coeff;  (Subroutine used to initialize Bandpass coefficients)
.external Set_up_uartparameters;  (Subroutine used to initialize uarts parameters)

(Listed below is the interrupt jump table for the 2101)
(Resetting flag_out disables the rs232 driver chip)
jump start;  nop;  nop;  nop;  ( reset vector)
jump read_data_int;  nop;  nop;  nop;  ( irq2 interrupt vector)
jump Send_data;  nop;  nop;  nop;  ( sport0 transmit vector)
rtl;  nop;  nop;  nop;  ( sport0 receive vector)
jump send_convert;  nop;  nop;  nop;  ( irq1 interrupt vector)
set flag_out;  ( irq0 interrupt vector)
call Reset_Control_Flags;
call Sync_pulse_disable;
rti;

DM(Watch_dog) = ax0;  (timer interrupt vector)
DM(reset_flag_out);  ( used for sync mode)
rti;
nop;
nop;

(call Reset_Control_Flags; rti; nop; nop;) (timer interrupt
(End of the interrupt jump table)

start:

call Init_sys_par;  (Initialize system parameters)
call Init_Lowpass_Coeff;  (Initialize lowpass coefficients)
call Init_Bandpass_Coeff;  (Initialize bandpass coefficients)
call Set_Primary_Data_Reg;  (Initialize all data registers)
call Set_up_uart_parameters;  (Initialize uarts parameters)
call Initialize_AD;  (Initialize A/D for operation)
call Reset_Control_Flags;  (Reset counters for number of samples
taken and processed)
reset_flag_out;  (Disable serial port driver)
ax1 = dm(Points_Per_Frame);  (Set points_per_frame and)
dm(Convert_Pulse_Count)= ax1;  (Convert_pulse_count equal)
ax0 = 1;
dm(Sync_flag_enable) = ax0;
ifc = 0x3f;  (Clear all interrupts before enabling )
imask = 0x37;  (Enable interrupts Irq2,
Sport0 Transmit, Irq1, and Irq0 for
core mode,timer)
imask = 0x35;  (Enable interrupts Irq2,
Sport0 Transmit Irq1, and Timer for
non sync mode)

(Enable timer used for sync mode of operation it will be used to
reset the watchdog timer)
ENA Timer;  (Start timer for watchdog timer updates)

main_loop:

(Wait for sync pulse to occur before continuing )
ax0 = dm(Sync_flag_enable);  
ar = Pass ax0;  
if ne jump main_loop;  
imask = 0x35;  (Disable interrupt IRQ0)

Inner_loop:
ax1 = dm(Data_Ready_to_be_Processed);
ay1 = dm(Number_of_Data_Points_Processed);
ar = ax1 - ayl;
if eq jump Inner_loop;  (Test and see if data needs to be processed
if not wait for next data to be input read from
A/D)
call Filter_Error_Signal;  (Call algorithm to filter error signal)
ax1 = dm(Points_per_Frame);
ayl = dm(Number_of_Data_Points_Processed);

ar = axl - ayl;

if ne jump inner_loop;

call Compute_Tracking_Error;

ca[C°ntinue_TrackingError;

call Transmit_data;

DM(Watch_dog) = axO;

DIS AR_SAT;

ayl = dm(frame_tag);

ar = ayl + 1;

dm(frame_tag) = ar;

Wait_for_enabte:

axO = clm(Sync_flag_disable);

ar = Pass axO;

if ne jump Wait_for_enable;

ifc = 0x02;

imask = 0x37;

ENA AR_SAT;

jump main_loop;

(Disable_irq1 will disable irq1 for conversion)

disable_irq1:

ayO = Ox3b;

axO = imask;

ar = axO AND ayO;

imask = ar;

rts;

(Disable_irq2 will disable irq1 for conversion)

disable_irq2:

ayO = Ox1f;

axO = imask;

ar = axO AND ayO;

imask = ar;

rts;

(Disable_timer will disable timer for conversion)

disable_timer:

dis timer;

rts;

(Reset control flags used to reset counters)
.entry Reset_Control_Flags;

Reset_Control_Flags:

```assembly
ar = 0;
dm(Data_Ready_to_be_Processed) = ar;
dm(Number_of_Data_Points_Processed) = ar;
dm(Convert_pulse_count) = ar;

rts;

Reset_Timer:
ax0 = 7499;
dm(Tcount_Reg) = ax0; (Set up timer to timeout after 6 milli sec.)
nop; (Enable RS232 driver for data transmission)
rts;

(Reset timer function at beginning of 40 Hz. sample frame)

Init_sys_par:
ax0 = 0x1819;
dm(Sys_Ctrl_Reg) = ax0; (Sport 0 enabled, Sport 1 enabled &)
               (config as F1,F0,Irq0,1,2, Boot page 0,)
               (One wait states for boot memory)
               (One wait states for program memory)

ax0 = 0x3200;
dm(Dm_wait_Reg) = ax0; (Set Data memory wait states to )
               (the following settings DWAIT0,3,4=O)
               (DWAIT2 = 1 wait state for reading)
               (data from A/D DWAIT1 = 3 for )
               (generating output pulse to watchdog)
               ( timer )

Set up timer parameters)
ax0 = 9;
(axo = 39;)
dm(Tscale_Reg) = ax0;

(Tscale_reg will be used for 40 Hz. update)
(ax0 = 39 to allow for PC to keep up
used for debug only sets update to .1 sec)
 decrement tcount every 10 th instruction cycle)
(T timer)

ax0 = 30719;
dm(Tperiod_Reg) = ax0;

( This value in timer counter corresponds )
( to interrupts generated every 25 milliseconds)
( for the frame timer)

ax0 = 10000;

(Delay the first interrupt by approx. 8.3 milli seconds)

(tcount_Reg) = ax0;

(ax0 = 0x4400;)
dm(Sport1_Ctrl_Reg) = ax0; (Internal clock of Sport1 enabled)

ax0 = 225;
dm(Sport1_Sclkdiv) = ax0;

(Set up timing for Sport1 serial clock)

(ax0 = 0;)

(dm(SPORT0_RX_WORDS1) = ax0; (disable Sport 0 Multichannel capability)
dm(SPORTO_TX_WORDD0) = axO;

axO = 0x4e09;

dm(SPORTO_CTRL_REG) = axO;  
(Internal serial clock generation
Transmit frame sync and sync width
Internal transmit frame sync enabled
Data format right justified zero filled
Unused MSB's Serial length word = 10
SLEN + 1)

(axO = 319;)
(Internal transmit frame sync enabled)

axO = 52;  
(Set up clock for 115.9k baud data transmission)

dm(SPORTO_SCLDIV) = axO;

axO = 0;  
(No receive frame sync divide)

dm(SPORTO_RFSDIV) = axO;

dm(SPORTO_AUTOBUF_CTRL) = axO;  
(Autobuffer enabled)

Set up misc. control registers

imask = 0x00;
(i All interrupts Disabled)
icnt = 0x7;  
(irqO, irq1, irq2 are edge sensitive)
ifc = 0x3f;  
(Initialize by clearing all interrupts)

axO = 0;  
(Initialize frame tag counter)

dm(frame_tag) = axO;

axO = 100;  
(Initialize frame markers)

dm-points_per_frame) = axO;

axO = 1;  
(Enable sync_flag_enable flag)

dm(Sync_flag_enable) = axO;

ENA_AR_SAT;  
(Initialize frame tag counter)

Set up miscellaneous control registers

rts;

Set_input_array_ptr sets up the pointers for all array required for this program

Set_primary_data_reg:

I0 = "AD Raw Data;  
(Set up address pointer for processing AD Raw data array)

dM(Process_data_add_ptr) = I0;  
(Store address pointer for pointing to data in the array that will be processed)

L0 = %AD_Raw_Data;  
(Set length of buffer for a circular array)

dM(Process_data_len_ptr) = L0;  
(Store length of buffer data to be processed)

I1 = "Coefficients;  
(Initialize I1 for address of bandpass filter I1 will for both lowpass and bandpass filters)

L1 = 0;  
(Set L6 equal 0 to indicate that the array is not circular)

I2 = "Filtered Data;  
(Set up address pointer for array of outputs from bandpass filter)
DM(Filtered_data_add_ptr) = 12;  # Store address pointer
L2 = &Filtered_data;  # Set length of buffer for a circular array
DM(Filtered_data_len_ptr) = L2;  # Store length of output filter (buffer for bandpass filter)

L3 = &Lowpass_Filt_data;  # Set up address pointer for array of outputs from bandpass filter
DM(Low_Filt_data_add_ptr) = L3;  # Store address pointer for pointing to data in the array that will be processed
L3 = &Lowpass_Filt_data;  # Set length of buffer for lowpass filter array
DM(Low_Filt_data_len_ptr) = L3;  # Store length of output filter (buffer for lowpass filter)

DM(Convert_Pulse_Add_ptr) = &Convert_Pulse;  # Set up address pointer for generating convert pulse

DM(Pointer_1) = &Delay_1;  # Set up address of delay line of bandpass filter and store address there is 1 delay line for each quadrant
DM(Pointer_2) = &Delay_2;
DM(Pointer_3) = &Delay_3;
DM(Pointer_4) = &Delay_4;

DM(Pointer_1L) = &Delay_1L;  # Set up address of delay line of lowpass filter and store address there is 1 delay line for each quadrant
DM(Pointer_2L) = &Delay_2L;
DM(Pointer_3L) = &Delay_3L;
DM(Pointer_4L) = &Delay_4L;

L5 = &Delay_1;  # Reset address of 15 to initial delay element that processed
L5 = &Delay_1;  # L5 will be used as the delay element length for both bandpass and lowpass filter

DM(AD_Raw_Data) = &AD_Raw_Data;  # Set up address pointer for array AD_Raw_data
DM(AD_rawdata_add_ptr) = l6;  
(Store address pointer for input array)

L6 = %AD_Raw_Data;  
(Set length of buffer for a circular array)

DM(AD_rawdata_len_ptr) = L6;  
(Store length on input array)

I7 = 'Read_Adc;  
(Set up address pointer for reading A/D)

DM(Read_Adc_Add_Ptr) = I7;  
(Store address)

M0 = 1;  
(Set up address counter for incrementing through the Raw_data_array, and intermediate arrays after the bandpass and lowpass filter)

M1 = 1;  
(Set address counter to be used for bandpass & lowpass filters)

M3 = -4;  
(Set up address counter to -4 to decrement address between bandpass and lowpass filtering)

M5 = 1;  
(Set address counter to be used for bandpass & lowpass filters)

M6 = 2;  
(Set up address counter to increment delay storage in bandpass and lowpass filter this provides the required offset in storing the data)

M7 = 0;  
(Set up address counter to zero such that when reading the A/D and Sending the convert pulse that we don't change the address)

(The following counters and array length indicators are spare and will be sent to zero)

M2 = 0;
M4 = 0;
L4 = 0;
L7 = 0;

.rts;

.Entry Sync_pulse_enable;

Sync_pulse_enable:
ax0 = 1;
dm(Sync_flag_enable) = ax0;  
(Set up flag for enabling IRQ0)
.rts;

Sync_pulse_disable:
ax0 = 0;
dm(Sync_flag_enable) = ax0;  
(Set up flag for disabling IRQ0)
.rts;

Check_sync_flag:
ax0 = dm(Sync_flag_enable);
er = Pass ax0;
if eq jump Disable_IRQ0;
ifc = 0x20;
imask = 0x37;  
(Clear irq0 before enabling interrupt)
(Enable IRQ0)
.rts;

Disable_IRQ0;
imask = 0x35;  
(Disable IRQ0)
.rts;

.endmod;
.module/ram/boot=0 uart;

(Output_Buffer will be used to store the formatted Data words)

.var/dm/ram/circ       Output_Buffer(buffer_length);

.var/dm/ram           Number_of_words_sent;

.var/dm/ram           Words_to_be_transmitted;

(Set up temporary buffers for storing IO,MO,LO)

.var/dm/ram           Temp_IO_buffer;

.var/dm/ram           Temp_LO_buffer;

.var/dm/ram           Temp_MO_buffer;

(Declare external subroutines used)
.extern Total_signal_strength;
.extern Elevation_error;
.extern Azimuth_error;
.extern Frame_tag;
.extern Reset_Control_Flags;
.extern Sync_Pulse_Enable;
.extern Ten__IObuffer;
.extern Temp_LO_buffer;
.extern Temp_MO_buffer;

(Macro delay loop used to generate a delay .8 usec per count)

.MACRO DELAY(XO);
.LOCAL Delay_loop;
    cntr = XO;
    DO delay_loop until cntr;
    nop; nop; nop; nop; nop; nop; nop;
    delay_loop: nop;
.ENDMACRO;

(Transmit_data will be used to transmit data to the host computer)

.ENTRY Transmit_data;

Transmit_data:
    ar = 0; (Reset counter to zero)
    dm(Number_of_words_sent) = ar;
    Call Set_address_pointers; (Set up address pointers)
    (Call Set_up_test_data;) (Subroutine used for testing purposes
    sets up predefined data for transmission)
    Call Format_data; (Format data into 8 bit words)
    IO = 'Output_buffer;
    ar = DM(IO,MO); (Transmit first word to host PC)
    TXO = ar;
    ay1 = dm(Number_of_words_sent);
    ar = ay1 + 1;
    dm(Number_of_words_sent) = ar;
    RTS;

(Set address pointer store primary array address and length and
temporary assigns new values to IO,LO,MO)

Set_address_pointers:
74
75     dm(Temp_IO_buffer) = IO; \hspace{1cm} \text{(Store IO, L0, and MO)}
76     dm(Temp_LO_buffer) = LO;
77     dm(Temp_MO_buffer) = MO;
78
79     IO = 'Output_buffer; \hspace{1cm} \text{(Set IO to address of output buffer)}
80     LO = '0Output_buffer; \hspace{1cm} \text{(Set L0 to length of output buffer)}
81     MO = 1;
82
83     RTS;
84
85
86     \text{~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~}
87     \text{(Reset address pointer store primary array address and length and}
88     \text{ temporary assigns new values to IO, L0, MO This subroutine is called}
89     \text{when Sport O has transmitted all 8 data words)}
90
91     Reset_address_pointer:
92
93     IO = dm(Temp_IO_buffer); \hspace{1cm} \text{(Primary data back to IO, L0, and MO)}
94     LO = dm(Temp_LO_buffer);
95     MO = dm(Temp_MO_buffer);
96
97     Delay(1000); \hspace{1cm} \text{(Delay disabling driver by .8 mSec.)}
98     reset flag_out; \hspace{1cm} \text{(Disable serial driver till next sync pulse)}
100
101     RTS;
102
103
104     \text{~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~}
105     \text{(Format_data is used to format the data to 8 bit words for data transmission)}
106
107     Format_data:
108
109     axO = 0xff00;
110     ax1 = 0x00ff;
111
112     ayO = DM(Frame_tag);
113     call Bit_reversal;
114
115     ar = axO and ayO; \hspace{1cm} \text{(mask off 8 thru 1 and set bits to zero)}
116
117     sr1 = 0x001;
118
119     sr = sr or lshift ar by -7 (HI); \hspace{1cm} \text{(Shift data left 7 places}
120     \hspace{4cm} \text{Bits 16 thru 9 now become bits}
121     \hspace{4cm} \text{9 thru 2 and set bit 10 to one)}
122
123     DM(IO,M0) = sr1; \hspace{1cm} \text{(Transmit data)}
124
125
126     ar = ax1 and ayO; \hspace{1cm} \text{(mask off 16 thru 9 and set bits to zero)}
127
128     sr1 = 0x001;
129
130     sr = sr or lshift ar by 1 (HI); \hspace{1cm} \text{(Shift data right 1 place}
131     \hspace{4cm} \text{Bits 8 thru 1 now become bits 9}
132     \hspace{4cm} \text{thru 2 set bit 10 to one)}
133
134     DM(IO,M0) = sr1; \hspace{1cm} \text{(Transmit data)}
135
136
137
138
139     ayO = DM(Elevation_error);
140     call Bit_reversal;
141
142     ar = axO and ayO; \hspace{1cm} \text{(mask off 8 thru 1 and set bits to zero)}
143
144     sr1 = 0x001;
145
146     sr = sr or lshift ar by -7 (HI); \hspace{1cm} \text{(Shift data left 7 places}
DM(IO,MO) = sr1;

Bits 16 thru 9 now become bits 9 thru 2 and set bit 10 to one)

ar = ax1 and ay0;  (mask off 16 thru 9 and set bits to zero)
sr1 = 0x001;
sr = sr or lshift ar by 1 (HI);  (Shift data right 1 place Bits 8 thru 1
now become bits 9 thru 2)

DM(IO,MO) = sr1;

(Transmit data)

ar = ax1 and ay0;  (mask off 16 thru 9 and set bits to zero)
sr = sr or tshift ar by 1 (HI);  (Shift data right 1 place Bits 8 thru 1
now become bits 9 thru 2)

DM(IO,MO) = srl;

(Term data)

ar = ax1 and ay0;  (mask off 16 thru 9 and set bits to zero)
sr = sr or tshift ar by 1 (HI);  (Shift data right 1 place Bits 8 thru 1
now become bits 9 thru 2)

DM(IO,MO) = srl;

(Term data)

ar = ax1 and ay0;  (mask off 16 thru 9 and set bits to zero)
sr = sr or tshift ar by 1 (HI);  (Shift data right 1 place Bits 8 thru 1
now become bits 9 thru 2)

DM(IO,MO) = srl;

(Term data)

ar = ax1 and ay0;  (mask off 16 thru 9 and set bits to zero)
sr = sr or tshift ar by 1 (HI);  (Shift data right 1 place Bits 8 thru 1
now become bits 9 thru 2)

DM(IO,MO) = srl;

(Term data)

ar = ax1 and ay0;  (mask off 16 thru 9 and set bits to zero)
sr = sr or tshift ar by 1 (HI);  (Shift data right 1 place Bits 8 thru 1
now become bits 9 thru 2)

DM(IO,MO) = srl;

(Term data)

ar = ax1 and ay0;  (mask off 16 thru 9 and set bits to zero)
sr = sr or tshift ar by 1 (HI);  (Shift data right 1 place Bits 8 thru 1
now become bits 9 thru 2)

DM(IO,MO) = srl;

(Term data)

ar = ax1 and ay0;  (mask off 16 thru 9 and set bits to zero)
sr = sr or tshift ar by 1 (HI);  (Shift data right 1 place Bits 8 thru 1
now become bits 9 thru 2)

DM(IO,MO) = srl;

(Term data)

ar = ax1 and ay0;  (mask off 16 thru 9 and set bits to zero)
sr = sr or tshift ar by 1 (HI);  (Shift data right 1 place Bits 8 thru 1
now become bits 9 thru 2)

DM(IO,MO) = srl;

(Term data)

ar = ax1 and ay0;  (mask off 16 thru 9 and set bits to zero)
sr = sr or tshift ar by 1 (HI);  (Shift data right 1 place Bits 8 thru 1
now become bits 9 thru 2)

DM(IO,MO) = srl;

(Term data)

ar = ax1 and ay0;  (mask off 16 thru 9 and set bits to zero)
sr = sr or tshift ar by 1 (HI);  (Shift data right 1 place Bits 8 thru 1
now become bits 9 thru 2)

DM(IO,MO) = srl;

(Term data)

ar = ax1 and ay0;  (mask off 16 thru 9 and set bits to zero)
sr = sr or tshift ar by 1 (HI);  (Shift data right 1 place Bits 8 thru 1
now become bits 9 thru 2)

DM(IO,MO) = srl;

(Term data)

ar = ax1 and ay0;  (mask off 16 thru 9 and set bits to zero)
sr = sr or tshift ar by 1 (HI);  (Shift data right 1 place Bits 8 thru 1
now become bits 9 thru 2)

DM(IO,MO) = srl;

(Term data)

ar = ax1 and ay0;  (mask off 16 thru 9 and set bits to zero)
sr = sr or tshift ar by 1 (HI);  (Shift data right 1 place Bits 8 thru 1
now become bits 9 thru 2)

DM(IO,MO) = srl;

(Term data)

ar = ax1 and ay0;  (mask off 16 thru 9 and set bits to zero)
sr = sr or tshift ar by 1 (HI);  (Shift data right 1 place Bits 8 thru 1
now become bits 9 thru 2)

DM(IO,MO) = srl;

(Term data)

ar = ax1 and ay0;  (mask off 16 thru 9 and set bits to zero)
sr = sr or tshift ar by 1 (HI);  (Shift data right 1 place Bits 8 thru 1
now become bits 9 thru 2)

DM(IO,MO) = srl;

(Term data)

ar = ax1 and ay0;  (mask off 16 thru 9 and set bits to zero)
sr = sr or tshift ar by 1 (HI);  (Shift data right 1 place Bits 8 thru 1
now become bits 9 thru 2)

DM(IO,MO) = srl;

(Term data)

ar = ax1 and ay0;  (mask off 16 thru 9 and set bits to zero)
sr = sr or tshift ar by 1 (HI);  (Shift data right 1 place Bits 8 thru 1
now become bits 9 thru 2)

DM(IO,MO) = srl;

(Term data)

ar = ax1 and ay0;  (mask off 16 thru 9 and set bits to zero)
sr = sr or tshift ar by 1 (HI);  (Shift data right 1 place Bits 8 thru 1
now become bits 9 thru 2)

DM(IO,MO) = srl;

(Term data)

ar = ax1 and ay0;  (mask off 16 thru 9 and set bits to zero)
sr = sr or tshift ar by 1 (HI);  (Shift data right 1 place Bits 8 thru 1
now become bits 9 thru 2)

DM(IO,MO) = srl;

(Term data)

ar = ax1 and ay0;  (mask off 16 thru 9 and set bits to zero)
sr = sr or tshift ar by 1 (HI);  (Shift data right 1 place Bits 8 thru 1
now become bits 9 thru 2)

DM(IO,MO) = srl;

(Term data)

ar = ax1 and ay0;  (mask off 16 thru 9 and set bits to zero)
sr = sr or tshift ar by 1 (HI);  (Shift data right 1 place Bits 8 thru 1
now become bits 9 thru 2)

DM(IO,MO) = srl;

(Term data)

ar = ax1 and ay0;  (mask off 16 thru 9 and set bits to zero)
sr = sr or tshift ar by 1 (HI);  (Shift data right 1 place Bits 8 thru 1
now become bits 9 thru 2)

DM(IO,MO) = srl;

(Term data)

ar = ax1 and ay0;  (mask off 16 thru 9 and set bits to zero)
sr = sr or tshift ar by 1 (HI);  (Shift data right 1 place Bits 8 thru 1
now become bits 9 thru 2)

DM(IO,MO) = srl;

(Term data)

ar = ax1 and ay0;  (mask off 16 thru 9 and set bits to zero)

(rts;

(Send data sends the next data word to the host PC after receiving
a Sport0 transmit interrupt)
.ENTRY Send_data;
Send_data:
ENA Sec_reg;  (Enable secondary registers)
ayl = dm(Number_of_words_sent);  (Test to see if all data has been)
ax1 = dm(Words_to_be_transmitted);  (sent)
ar = ax1 - ayl;
if gt jump Transmit;  (If not done with data transmission
jump to transmit else quit transmission )
call Reset_address_pointer;  (If all data words have
been sent reset control flags
and exit interrupt routine)
call Sync_Pulse_Enable;
DIS Sec_reg;  (Disable secondary registor)
nop;
RTI;
Transmit:
ar = dm(10,M0);  (Transmit next data word to host PC)

txO = ar;
ar = ayl + 1;  (Increment counter of words that has been sent)

dm(Number_of_words_sent) = ar;
DIS Sec_reg;  (Disable secondary registor)
nop;
RTI;

Remark: {Set up uart_parameters is used to initialize}

ENTRY Set_up_uart_parameters;
Set_up_uart_parameters:
ar = 0;
dm(Number_of_words_sent) = ar;
ar = 8;
dm(Words_to_be_transmitted) = ar;
RTS;

Remark: {Set up test_data is a subroutine used to set up predefined data}

ENTRY Set_up_test_data;
Set_up_test_data:
ar = 0x01e0;  (Set up "T" for transmission)
(dm(Frame_tag) = ar;)

(ar = 0x5556;  (Set up "S" for transmission)
DM(Elevation_error) = ar;

ar = 0x5678;  (Set up "T" for transmission)
Dm(Azimuth_error) = ar;

ar = Oxabc;  (Set up "U" for transmission)
Dm(Totat_signal_strength) = ar;

RTS;

Remark: (Bit reversal routine will rotate bit order 1 to 16 to the order of 16 to 1)
cntr = 16;
mr0 = 0x0001;
mr1 = 0x8000;
ar = 0;
ay1 = 0;
se = 1;

Do Reverse_bit Until CE;

af = mr0 and ay0;  (Test to see if bit is a one)
if ne ar = mr1 or ay1;  (Add bit to af register if bit x was a one)
sr = lshift mr0 (hi), ay1 = ar;  (Shift masking bit right by one)
          (Store new reversed data in ay1)
mr0 = sr1;  (Store new value of mask word)
sr = lshift mr1 by -1 (hi);  (Shift bit used for setting one to left) .
Reverse_bit: mr1 = sr1;
ay0 = ar;  (Store data back into ay0 for next step of formatting)

rts;

.endmod;
COMMON /ATTBLK/ MACOS,TSMALLSCAN
1 COMMON /CIRCBLK/ PI,TWOPI,HALFPi,RAD
COMMON /DISCBLK/ DISCDEGMAX,DISCVEL,DISCACC,DISCACCIM,DRATIO,
ALSETDEG,NSTEPS,ISTEPMAX,MAXSLEWSTEPS,MAXTRACKSTEPS,DLDEGPARK,
ALDEGPARK,ISTEPAK,ISTEPBPARK
INTEGER*4 NSTEPS,ISTEPMAX,MAXSLEWSTEPS,MAXTRACKSTEPS
INTEGER*4 ISTEPAPARK,ISTEPPARK
1 COMMON /FOVBLK/ FOVDEG,BEAMW,ARML
1 COMMON /MISCBLK/ REALTIME,TEST,WFLAG,IRATE,TRKGAIN
2 LOGICAL REALTIME,TEST,WFLAG
1 COMMON /ORBITBLK/ EPHH,EPHL,PRDH,PRDL(6)
1 COMMON /SAVEBLK/ INTRACKSAVE,USAVE,VSAVE,ISTEPASAVE,ISTEPBSAVE
2 INTEGER*4 ISTEPASAVE,ISTEPBSAVE
3 LOGICAL INTRACKSAVE
SUBROUTINE ALDLDLAB(AL,DL,ICHL,ISOL,INITDIR,TCUR,DLA',DLB',
   1 ISTEPA,ISTEPB,ALSTEP)

   This routine transforms from disc & arm angles to disc driver angles.

   Prototype Version 1.0  Avtec Systems, Inc.  August 1992

   Inputs:
   12   AL = Arm angles (alpha)
   13   DL = Disc angles (delta)
   14   ICHL = Channel index
   15   ISOL = Selected alpha & delta solution (1 or 2)
   16   INITDIR = Initialize disc direction flags
   17   TCUR = Current time
   18   ISTEPMAX = Disc step at maximum rotation (DISCBLK)
   19   WFLAG = Write flag (MISCBLK)

   Outputs:
   22   DLA(ICHL) = Disc driver A angle (delta A)
   23   DLB(ICHL) = Disc driver B angle (delta B)
   24   ISTEPA(ICHL) = Disc A resolution step
   25   ISTEPB(ICHL) = Disc B resolution step
   26   ALSTEP(ICHL) = Arm angle corresponding to ISTEPA(ICHL) and ISTEPB(ICHL)
   27   INITDIR(ICHL) = Initialize disc direction flag

   IMPLICIT REAL*8 (A-H,O-Z)
   REAL*8 AL(6,2),DL(6,2),DLA(6),DLB(6),ALSTEP(6)
   INTEGER*4 ISTEPA(6),ISTEPB(6),ISTEPAOLD(6),ISTEPBOLD(6)
   INTEGER*2 INITDIR(6),ISIGNOLD(6)
   LOGICAL REVERSE
   INCLUDE 'CIRCBLK.FCB'
   INCLUDE 'DISCBLK.FCB'
   INCLUDE 'MISCBLK.FCB'

   DATA INIT /1/

   IF(INIT.EQ.1) THEN
     IF(DISCDEGMAX.LT.180 .OR. DISCDEGMAX.GT.270) STOP 115
     ALSET = ALSETDEG / RAD
     DISCSTEP = TROOP / NSTEPS
     ISTEPNORP = NSTEPS - ISTEPMAX
     INIT=0
   END IF

   REVERSE = .FALSE.
   JSTEPA = NINT(DL(ICHL,ISOL)/DISCSTEP)
   DLBCONT = DL(ICHL,ISOL) - (AL(ICHL,ISOL) - ALSET) / DRATIO
   JSTEPB = NINT(DLBCONT/DISCSTEP)

   C   If disc direction is already initialized and both old disc steps could be
   C   negative or positive, maintain the same sign.

   IF(INITDIR(ICHL).EQ.0 .AND. IABS(ISTEPAOLD(ICHL)).GE.ISTEPMORP
   1 .AND. IABS(ISTEPBOLD(ICHL)).GE.ISTEPMORP) THEN
     IF(ISIGN(1,JSTEPA).NE.ISIGNOLD(ICHL)) THEN
       JSTEPA = JSTEPA + ISIGNOLD(ICHL)*NSTEPS
       JSTEPB = JSTEPB + ISIGNOLD(ICHL)*NSTEPS
     END IF
   END IF

   END
74 C If either disc step exceeds maximum value, reverse both disc directions.
75 C
76 IF(IABS(JSTEPA).GT.ISTEMAX OR IABS(JSTEPB).GT.ISTEMAX) THEN
77 REVERSE = .TRUE.
78 JSTEPA = JSTEPA + ISIGN(NSTEPS,JSTEPA)
79 JSTEPB = JSTEPB + ISIGN(NSTEPS,JSTEPB)
80 END IF
81 IF(IABS(JSTEPA).GT.ISTEMAX OR IABS(JSTEPB).GT.ISTEMAX) STOP 116
82 IF(INITDIR(ICHL).EQ.0 .AND. REVERSE) THEN
83 IF(WFLAG) WRITE(15,20)
84 WRITE(20,21) TCUR,ICHL
85 FORMAT(/17X,'ROTATION LIMIT / DIRECTION REVERSED')
86 WRITE(12,12) ' TIME =',F12.3,' SEC',sx,'CHANNEL',I2,
87 1 ' ROTATION LIMIT / DIRECTION REVERSED')
88 END IF
89 ISTEPA(ICHL) = JSTEPA
90 ISTEPB(ICHL) = JSTEPB
91 C Compute disc angles and arm angle.
92 C
93 DLA(ICHL) = JSTEPA * DISCSTEP
94 DLB(ICHL) = JSTEPB * DISCSTEP
95 ALSTEP(ICHL) = ALSET + DRATIO * (DLA(ICHL) - DLB(ICHL))
96 ISTEPAOLD(ICHL) = JSTEPA
97 ISTEPBOLD(ICHL) = JSTEPB
98 ISIGNOLD(ICHL) = ISIGN(1,JSTEPA)
99 INITDIR(ICHL) = 0
100 RETURN
101 END
SUBROUTINE ANGLEPR (AZS, ELS, AZAHEAD, ELAHEAD, DLATRUE,

 In simulation mode, this routine computes and returns the Angle Processor data.

 In real-time mode, this routine returns the Angle Processor data previously saved.

 Prototype Version 1.0 Avtec Systems, Inc. August 1992

 IMPLICIT REAL*8 (A-H,O-Z)

 REAL*8 AZS(6), ELS(6), AZAHEAD(6), ELAHEAD(6)
 REAL*8 DLATRUE(6), ALTRUE(6), U(6), V(6)
 LOGICAL INTRACK(6)

 INCLUDE 'CIRCBLK.FCB'
 INCLUDE 'FOVBLK.FCB'
 INCLUDE 'MISCBLK.FCB'
 INCLUDE 'SAVEBLK.FCB'

 DATA RMSERR /1.60D-5/
 DATA RES /1.50D-6/
 DATA TRMIS /10D-5/
 DATA INIT /1/

 IF(REALTIME) THEN
 INTRACK(ILEO) = INTRACKSAVE
 U(ILEO) = USAVE
 V(ILEO) = VSAVE
 RETURN
 END IF

 IF(INIT.EQ.1) THEN
 TANHALF = DTAN(5D-I*FOVDEG/RAD)
 HALFBEAMSQ = (5D-I*BEAMM)**2
 SIGMA = RMSERR / DSQRT(2D0)
 CALL RNDC(Unif)
 ORIENTMIS = TWOPI*Unif
 TRMISHALF = 5D-I*TRMIS
 INIT = 0
 END IF
C Determine coordinates of arm pivot point.

\[
\begin{align*}
XA &= ARML \cdot DCOS(HALFPI-DLATRUE(ICHL)) \\
YA &= ARML \cdot DSIN(HALFPI-DLATRUE(ICHL))
\end{align*}
\]

C Determine coordinates of center of optics.

\[
\begin{align*}
COSAL &= DCOS(ALTRUE(ICHL)) \\
SINAL &= DSIN(ALTRUE(ICHL)) \\
AZOPT &= DATAN(TANHALF \cdot (XA + COSAL \cdot YA - SINAL \cdot XA)) \\
ELOPT &= DATAN(TANHALF \cdot (YA - SINAL \cdot YA - COSAL \cdot XA))
\end{align*}
\]

C Include transmit/receive misalignment.

\[
\begin{align*}
ANGLMIS &= ORIENTMIS - DLATRUE(ICHL) - ALTRUE(ICHL) \\
COSMIS &= DCOS(ANGLMIS) \\
SINMIS &= DSIN(ANGLMIS) \\
AZOPTT &= AZOPT + COSMIS \cdot TRMISHALF \\
ELOPTT &= ELOPT + SINMIS \cdot TRMISHALF \\
AZOPTR &= AZOPT - COSMIS \cdot TRMISHALF \\
ELOPTR &= ELOPT - SINMIS \cdot TRMISHALF
\end{align*}
\]

C Determine coordinates of LEO when HEO beam arrives. Note: errors in point-ahead angles caused by small attitude and ephemeris errors are negligible.

\[
\begin{align*}
AZLEO &= AZS(ILEO) + AZAHEAD(ILEO) \\
ELLEO &= ELS(ILEO) + ELAHEAD(ILEO)
\end{align*}
\]

C Assume in track if LEO is within HEO beam.

\[
\begin{align*}
IF ((AZLEO-AZOPTT)**2 + (ELLEO-ELOPTT)**2 < HALFBEAMSQ) THEN \\
INTRACK(ILEO) = .TRUE.
\end{align*}
\]

C Determine difference coordinates of center of received spot.

\[
\begin{align*}
AZDIF &= AZS(ILEO) - AZOPTR \\
ELDIF &= ELS(ILEO) - ELOPTR
\end{align*}
\]

C Determine rotation from azimuth, elevation to u,v.

\[
\begin{align*}
ROT &= -(HALFPI + DLATRUE(ICHL) + ALTRUE(ICHL)) \\
COSROT &= DCOS(ROT) \\
SINROT &= DSIN(ROT)
\end{align*}
\]

C Determine random u,v errors.

\[
\begin{align*}
UMEAN &= AZDIF \cdot COSROT + ELDIF \cdot SINROT \\
VMEAN &= -AZDIF \cdot SINROT + ELDIF \cdot COSROT \\
U(ILEO) &= \text{Gausscl}(UMEAN, \text{SIGMA}, 3DO) \\
V(ILEO) &= \text{Gausscl}(VMEAN, \text{SIGMA}, 3DO)
\end{align*}
\]
V(ILEO) = MIN(VCONT/RES) * RES

C-------------------------------------------------------------
C Assume not in track if LEO is outside HEO beam.
C-------------------------------------------------------------

ELSE

INTRACK(ILEO) = .FALSE.

END IF

RETURN

END
DOUBLE PRECISION FUNCTION ARKTNS(N,X,Y)
FOUR QUAD ARC TANGENT X=COS(ANGLE) - Y=SIN(ANGLE)
N=180 GIVES PV BETWEEN -PI & +PI
N=360 GIVES PV BETWEEN 0 & +2PI
IMPLICIT REAL*8 (A-H,O-Z)
PI=3.14159265358979323800
TPI=2*PI
IF (X.NE.0.000) GO TO 10
IF (Y.GT.0.000) T=0.500*PI
IF (Y.LT.0.000) T=1.500*PI
IF (Y.EQ.0.000) T=0.000
GO TO 20
T=DATAN(Y/X)
IF (X.LT.0.000) T=T+PI
IF (T.LT.0.000) T=T+TPI
IF (N.EQ.360) GO TO 30
IF (T.GT.PI) T=T-TPI
ARKTNS=T
RETURN
END
SUBROUTINE ASGNMFN (HEONAV, LEONAV, ECLIPSE, AZS, ELS, ILEO, ICHL, 
1 LEOCHL, ICHLLLEO, TCUR)

The Assignment Function performs the following:
Assigns disc channels to LEO satellites.
Determines azimuth, elevation and point-ahead angles.
Selects one of two possible solutions for the disc angle.
Sets the scan reset flag for acquisition.

Prototype Version 1.0    Avtec Systems, Inc.    August 1992

Entry points:
- ASGNMFN: Assignment Function
- ASGNMFNR: Read the predicted Az & El of LEO satellites and select alpha & delta solutions (in real-time mode)

Inputs (ENTRY ASGNMFN):
- HEONAV = Geocentric-equatorial coordinates of HEO satellite from Navigation Processor
- LEONAV = Geocentric-equatorial coordinates of LEO satellites from Navigation Processor
- ECLIPSE = Eclipse flags from Scene Processor
- AZS = Azimuths of LEO satellites from Scene Processor
- ELS = Elevations of LEO satellites from Scene Processor
- ILEO = LEO satellite index
- ICHL = Channel index
- LEOCHL = LEO channel assignments
- ICHLLLEO = Channel LEO assignments
- TCUR = Current time
- IRATE = Tracking rate (MISCBLK)
- REALTIME = Real-time flag (MISCBLK)
- WFLAG = Write flag (MISCBLK)

Inputs (ENTRY ASGNMFNR):
- NLEOS = Number of LEO satellites
- NCHANNELS = Number of disc channels
- ICHLPRSEQ = Channel priority sequence

IMPLICIT REAL*8 (A-H,O-Z)
REAL*8 HEOSOON(6),LEOSOON(6,6)
REAL*8 HEONAV(6),LEOMAV(6,6),HEONAVSOON(6),LEONAVSOON(6,6)
REAL*8 AZAHEAD(6),ELAHEAD(6)
REAL*8 ALNAV(6,2),DLNAV(6,2)
INTEGER*2 LEOCHL(6),ICHLLEO(6),ICHLPRSEQ(6),IALDLSOL(6),INITDIR(6)
LOGICAL ECLIPSE(6),ECLPNAV(6),ECLPNAVSOON(6),INTRACK(6),SCANRES(6)
LOGICAL DONE
CHARACTER MODE(6)*13,DASH(72)*1
INCLUDE 'CIRCBLK.FCB'
INCLUDE 'DISCBLK.FCB'
INCLUDE 'MISCBLK.FCB'
DATA MODE /6*'PARK'/ ! Channel modes
DATA INTRACK /6*.FALSE./ ! In-track flags
DATA INITDIR /6*1! ! Initialize disc direction flags
DATA DASH /72**'-'/ !
DATA INIT /1/

IF(INIT.EQ.1) THEN
TSLEWMAX = 2*DISCDEGMAX/DISCVEL
IF(WFLAG) WRITE(14,20) DASH
20 FORMAT(//'ASSIGNMENT FUNCTION'//',72A1)
INIT = 0
END IF
IF(ILEO.NE.0) THEN
  C ...............................................................................
  IF(a) LEO is not currently assigned a channel, and
  IF(b) Eclipse soon flag from Navigation Processor data is false
  THEN (1) Assign the highest priority free channel, and
  (2) Set mode to 'slew to acq', and
  (3) Set scan reset flag to true.
  C ...............................................................................
IF(LEOCHL(ILEO).EQ.0) THEN
  IF(.NOT.REALTIME) THEN
    TSOON = TCUR + TSLEWMAX
    CALL QVAL(TSOON,ILEO,HEOSOON,LEOSOON)
    CALL NAV.GPR2(HEOSOON,LEOSOON,ILEO,TSOON,HEONAVSOON,
    1 ' LEONAVSOON)
    CALL XYZAZEL(HEONAVSOON,LEONAVSOON,ILEO,'N','T',
    1 ECLPNAVSOON,AZNAVT,ELNAVT,AZAHEAD,ELAHEAD)
  END IF
  IF(.NOT.ECLPNAVSOON(ILEO)) THEN
    LEOCHL(ILEO) = ICHL
    ICHLLEO(ICHL) = ILEO
    INITDIR(ICHL) = 1
    MODE(ICHL) = 'SLEW TO ACQ'
    SCANRESC(ILEO) = .TRUE.
    IF(WFLAG) WRITE(14,40) TCUR,ICHL,ILEO
    FORMAT(/' TIME =',F12.3,' SEC'/8X,'CHANNEL',I2,' RELEASED
      FROM LEO',I2)
    WRITE(20,45) TCUR,ICHL,ILEO
    FORMAT(/' TIME =',F12.3,' SEC',8X,'CHANNEL',I2,' RELEASED
      FROM LEO',I2)
    LEOCHL(ILEO) = 0
    ICHLLEO(ICHL) = 0
    INITDIR(ICHL) = 1
    MODE(ICHL) = 'PARK'
   ELSE IF(.NOT.REALTIME) CALL
    XYZAZEL(HEONAV,LEONAV,ILEO,'N','T',ECLPNAV,AZNAVT,ELNAVToAZAHEAD,ELAHEAD)
   IF(MODE(ICHL).NE.'SLEW TO ACQ' .AND. ECLPNAV(ILEO) .AND. .NOT.INTRACK(ILEO)) THEN
     IF(WFLAG) WRITE(14,50) TCUR,ICHL,ILEO
     FORMAT(/' TIME =',F12.3,' SEC'/8X,'CHANNEL',I2,' RELEASED
       FROM LEO',I2)
     WRITE(20,55) TCUR,ICHL,ILEO
     FORMAT(/' TIME =',F12.3,' SEC',8X,'CHANNEL',I2,' RELEASED
       FROM LEO',I2)
     LEOCHL(ILEO) = 0
     ICHLLEO(ICHL) = 0
     INITDIR(ICHL) = 1
     MODE(ICHL) = 'PARK'
   END IF
C ...............................................................................
C If (a) LEO is currently assigned a channel, and
C (b) Mode is not 'slew to acq', and
C (c) Eclipse flag from Navigation Processor data is true, and
C (d) In-track flag is false
ELSE
  IF(.NOT.REALTIME) CALL XYZAZEL(HEONAV,LEONAV,ILEO,'N','T',ECLPNAV,AZNAVT,ELNAVToAZAHEAD,ELAHEAD)
  IF(MODE(ICHL).NE.'SLEW TO ACQ' .AND. ECLPNAV(ILEO) .AND. .NOT.INTRACK(ILEO)) THEN
    IF(WFLAG) WRITE(14,50) TCUR,ICHL,ILEO
    FORMAT(/' TIME =',F12.3,' SEC'/8X,'CHANNEL',I2,' RELEASED
      FROM LEO',I2)
    WRITE(20,55) TCUR,ICHL,ILEO
    FORMAT(/' TIME =',F12.3,' SEC',8X,'CHANNEL',I2,' RELEASED
      FROM LEO',I2)
    LEOCHL(ILEO) = 0
    ICHLLEO(ICHL) = 0
    INITDIR(ICHL) = 1
    MODE(ICHL) = 'PARK'
END IF

END IF

C If LEO is currently assigned a channel, convert from azimuth and elevation
to alpha and delta and select solution 1 (in simulation mode).

IF(LEOCHL(ILEO).NE.0) THEN

IF(.NOT.REALTIME) THEN

CALL AZELALDL(AZNAVT,ELNAVT,ILEO,ICHL,12,ALNAV,DLNAV)

IALDLSOL(ICHL) = 1

END IF

IF(WFLAG) THEN

ISOL = IALDLSOL(ICHL)

WRITE(14,60) TCUR, ICHL, ILEO, AZNAVT(ILEO)*RAD,

ELNAVT(ILEO)*RAD, DLNAV(ICHL,ISOL)*RAD,

ALNAV(ICHL,ISOL)*RAD, ISOL, AZAHEAD(ILEO)*ID6,

ELAHEAD(ILEO)*ID6

60 FORMAT(I/'TIME'=I,F12.3,' SEC',SX,'CHL',

I8X,'LEO',12//

8X,'AZ NAV =',FII.5,TX,'EL NAV =',FI0.5,' (DEG)'/8X,

2 'DELTA =',FII.5,?X,'ALPHA =',FI0.5o'

3 'SOL',12/BX,'AZ AHD =',F7.1,11X,'EL AHD =',F6.1,6X,

4 'MICRORAD')

END IF

END IF

C Proceed to Control Function.

CALL CNTRLFN(ECLPNAV,AZNAVT,ELNAV,IALDLSOL,AZAHEAD,ELAHEAD,

ECLIPSE,MODE,INITDIR,INTRACK,SCANRES,ICHL,ILEO,

AZS,ELS,TCUR)

RETURN

ENTRY ASGNMFNR(NLEOS,NCHANNELS,ICHLPRSEQ)

C Read the predicted Az & EL of LEO satellites and select alpha & delta
solutions (in real-time mode).

OPEN(2,FILE='AZEL.DAT',STATUS='OLD')

DO I=1,MIN(NLEOS,NCHANNELS)

READ(2,*) AZPRED,ELPRED

AZNAV(I) = AZPRED/RAD

ELNAV(I) = ELPRED/RAD

ECLPNV(I) = .FALSE.

ECLPNVSOON(I) = .FALSE.

AZAHEAD(I) = 0

ELAHEAD(I) = 0

J = ICHLPRSEQ(I)

CALL AZELALDL(AZNAV,ELNAV,J,12,ALNAV,DLNAV)

WRITE(*,90) I, AZPRED, ELPRED, DLNAV(J,1)*RAD, ALNAV(J,1)*RAD,

1 DLNAV(J,2)*RAD, ALNAV(J,2)*RAD
DONE=.FALSE.
DO WHILE (.NOT.DONE)
WRITE(*,91)
FORMAT(' Select solution (1 or 2)')
READ(*,*) IALDLSOL(J)
IF(IALDLSOL(J).EQ.1 .OR. IALDLSOL(J).EQ.2) DONE = .TRUE.
END DO
END DO
RETURN
END
SUBROUTINE AZELALDL(AZ,EL,ILEO,ICHL,ISOL,AL,DL)

This routine transforms FOV azimuth & elevation to
arm angle alpha and disc angle delta.

Prototype Version 1.0  Avtec Systems, Inc.  August 1992

Inputs:
AZ = Azimuths of LEO satellites
EL = Elevations of LEO satellites
ILEO = LEO satellite index
ICHL = Channel index
ISOL = 1 for solution #1
       2 for solution #2
       12 for both solutions

Outputs:
AL(ICHL,ISOL) = Arm angles alpha
DL(ICHL,ISOL) = Disc angles delta

IMPLICIT REAL*8 (A-H,O-Z)
REAL*8 AZ(6),EL(6),AL(602),DL(6,2)
INCLUDE 'CIRCBLK.FCB'
INCLUDE 'FOVBLK.FCB'
DATA INIT /I/
IF(INIT.EQ.1) THEN
    THREEHALFPi = 3*HALFPi
    TANHALF = DTAN(FOVDEG*HALFPi/180)
    IF(ARML.LE.1) STOP 111
    INIT=0
END IF
XF = DTAN(AZ(ILEO)) / TANHALF
YF = DTAN(EL(ILEO)) / TANHALF
RF = DSQRT(XF**2 + YF**2)
IF(RF.EQ.0) STOP 112
IF(RF.GT.1) STOP 112
THETA = DATAN2(YF,XF)
PHI = DACOS(RF/(2*ARML))
IF(ISOL.EQ.1 .OR. ISOL.EQ.12) THEN
    DELTA = HALFPI - THETA + PHI
    DL(ICHL,1) = DELTA
    AL(ICHL,1) = THREEHALFPi - 2*PHI
END IF
IF(ISOL.EQ.2 .OR. ISOL.EQ.12) THEN
    DELTA = HALFPI - THETA - PHI
    DL(ICHL,2) = DELTA
    AL(ICHL,2) = 2*PHI - HALFPI
END IF
RETURN
END
BLOCK DATA

IMPLICIT REAL*8 (A-H,O-Z)

INCLUDE 'ATTBLK.FCB'

INCLUDE 'CIRCBLK.FCB'

INCLUDE 'DISCBLK.FCB'

INCLUDE 'FOVBLK.FCB'

INCLUDE 'MISCBLK.FCB'

INCLUDE 'ORBITBLK.FCB'

INCLUDE 'SAVEBLK.FCB'

DATA NACQS /0/

DATA TSMALLSCAN /1D30/

DATA PI /3.141592653589793D0/

DATA TWOPI /6.283185307179586D0/

DATA HALFPi /1.5707963267948971D0/

DATA PI /3.141592653589793D0/

DATA DISCDEGMAX /2.4D2/

DATA DISCVEL /7.7D0/

DATA DISCACCM /2.674D3/

DATA DISCACCTM /2.880-3/

DATA DRATIO /2.7294D0/

DATA ALSETDEG /901/

DATA NSTEPS /708000/

DATA DLDEGPARK /0D0/

DATA ALDEGPARK /35D0/

DATA ISTEPAPARK /0/

DATA ISTEPBPARK /39630/

DATA FOVDEG /2.2D1/

DATA BEAMW /.5D-4/

DATA ARML /1.39600/

DATA TEST / .FALSE./

DATA EPHH /2D-1/

DATA EPHL /5D-1/

DATA EPHH /2D-1/

DATA EPHL /5D-1/

END
SUBROUTINE CNTRLFN(ECLPNAV,AZNAV,ELNAV,IALDLSOL,AZAHEAD,ELAHEAD,
  1    ECLIPSE,MODE,INITDIR,INTRACK,SCANRES,ICH1,ILEO,
  2    AZS,ELS,TCUR)

The Control Function consists of:

  Offset Processor: Error Processor, Offset Generator, and Scan Generator
  Disc & Arm Controller: Determines angles for Disc Drivers

Prototype Version 1.0 Avtec Systems, Inc. August 1992

Inputs:

  ECLPNAV = Eclipse flags from Navigation Processor data
  AZNAV = Azimuths of LEO satellites from Navigation Processor data
  ELNAV = Elevations of LEO satellites from Navigation Processor data
  IALDLSOL = Selected alpha & delta solutions (1 or 2)
  AZAHEAD = Azimuth point-ahead angles
  ELAHEAD = Elevation point-ahead angles
  ECLIPSE = Eclipse flags from Scene Processor
  MODE = Channel modes
  INITDIR = Initialize disc direction flags
  INTRACK = In-track flags
  SCANRES = Scan reset flags
  ICH1 = Channel index
  ILEO = LEO satellite index
  AZS = Azimuths of LEO satellites from Scene Processor
  ELS = Elevations of LEO satellites from Scene Processor
  TCUR = Current time
  IRATE = Tracking rate (MISCBLK)
  REALTIME = Real-time flag (MISCBLK)
  TRKGAIN = Tracking gain (MISCBLK)
  WFLAG = Write flag (MISCBLK)
  NACQS = Number of acquisitions (ATTBLK)

Outputs:

  MODE(ICH1) = Channel mode
  INTRACK(ILEO) = In-track flag
  NACQS = Number of acquisitions (ATTBLK)
  ISTEPASAVE = Disc A resolution step in real-time mode (SAVEBLK)
  ISTEPBSAVE = Disc B resolution step in real-time mode (SAVEBLK)

IMPLICIT REAL*8 (A-H,O-Z)

REAL*8 AZNAV(6),ELNAV(6),AZAHEAD(6),ELAHEAD(6),AZS(6),ELS(6)
REAL*8 AZSCAN(6),ELSCAN(6),ACQDWDUR(6)
REAL*8 AZSUM(6),ELSUM(6),AZERR(6),ELERR(6),U(6),V(6)
REAL*8 AZOFFSET(6),ELOFFSET(6),AZOFFSETP(6),ELOFFSETP(6)
REAL*8 AL(6,2),DL(6,2),ALSTEP(6)
REAL*8 DLA(6),DLB(6),DLATRUE(6),DLBTRUE(6),ALTRUE(6)
INTEGER*4 STEPA(6),ISTEPB(6),ISTEPAOLD(6),ISTEPBOLD(6)
INTEGER*4 INHSTEPS,INHCOUNT(6),INHCHECK(6)
INTEGER*2 IALDLSOL(6),INITCHL(6),INITDIR(6)
LOGICAL ECLPNAV(6),ECLIPSE(6),INTRACK(6),SCANRES(6)
CHARACTER DASH(7'9) ,MODE(6)*13,INNMODE(6)*13,MODECHG*13

DATA ACQDWDUR /1D-1/
DATA ISTEPOLAL /10-1/ ! Duration of an acquisition dwell (sec)
DATA ISTEPOLALT /50/ ! Disc step tolerance for starting acquisition
DATA ISTEPOLALT /50/ ! Disc step tolerance for tracking
DATA FACTORINH /1.5D0/
DATA DASH /79*'-'/ ! Scan inhibit step factor
DATA DASH /79*'-'/
DATA INITCHL /6*1/
DATA INIT /1/
DATA DLRUNE /6*OOD/, DLBTRUE /6*OOD/, ALTRUE /6*OOD/
DATA DLA /6*OOD/, ALSTEP /6*OOD/ 
DATA ISTEPAOLD /6*0/, ISTEPBOLD /6*0/ 

IF(INIT.EQ.1) THEN
  QRTRBEAM = BEAMW/4
  DLPARK = DLDEGPARK/RAD
  ALPARK = ALDEGPARK/RAD
  ACQODWUR = ACWOODUR - 10-5
  ISTEPMAX = NINT(NSTEPS*DISCDEGMAX/3.602)
  MAXSLEWSTEPS = DISCVEL/IRATE * NSTEPS/3.602
  IF(DISACCTM .GE. 5D-1/IRATE) THEN
    MAXTRACKSTEPS = DISCACC/(2*IRATE)**2 * NSTEPS/3.602
  ELSE
    MAXTRACKSTEPS = (DISCACC*DISACCTM**2 + DISCACC*DISACCTM*(IDO/IRATE-2)*DISACCTM) * NSTEPS/3.602
  END IF
  INHSTEPS = MAXTRACKSTEPS / FACTORINH
  IF(WFLAG) WRITE(15,20) DASH
  FORMAT(//, CONTROL FUNCTION'//, 79A1)
  INIT = 0
  END IF
ENDIF

IF(TEST) THEN
  MODE(INCHL) = 'TRACKING'
  AZOFFSET(ILEO) = 0
  ELOFFSET(ILEO) = 0
ENDIF

IF(WFLAG) WRITE(15,40) TCUR,ICHL,ILEO
40 FORMAT(//' TIME = ',F12.3,' SEC',SX,'CHL',I2,4X,,LEO,,I2)

C ...............................................................................
C Get the true disc positions.
C .............................................................
CALL DISCPOS(ISTEPAOLD,ISTEPBOLD,MODE,INITCHL,ICNL,ISTEPATRUE, 
  ISTEPBTRUE,DLRUNE,DLBTRUE,ALTRUE)

IF(WFLAG) WRITE(15,42) ISTEPATRUE(INCHL), ISTEPBTRUE(INCHL), 
  1 DLATRUE(INCHL)*RAD, DLBTRUE(INCHL)*RAD, ALTRUE(INCHL)*RAD
42 FORMAT(//3X,'TRUE POSITION'/17X,'STEP A = ',Ill,5X,'STEP B = ',Ill 
  /17X,'DELTA A = ',F11.5,5X,'DELTA B = ',F11.5, (DEG)')

C ...............................................................................
C If mode is 'slew to acq' and eclipse flag from Navigation Processor 
C data is false and true disc steps are close to previous commands, then 
C change mode to 'acquisition'.
C ...............................................................
IF(MODE(INCHL).EQ. 'SLEW TO ACQ' .AND. .NOT.ECLPNAV(ILEO)) THEN
  IF(INITDIR(INCHL).EQ.0 .AND. 
    1 IABS(ISTEPATRUE(INCHL)-ISTEPAOLD(INCHL)) .LE. ISTEPTOLA .AND.
    2 IABS(ISTEPBTRUE(INCHL)-ISTEBPOLD(INCHL)) .LE. ISTEPTOLA) THEN
    MOOE(INCHL) = 'ACQUISITION'
    WRITE(20,47) TCUR,ICHL 
    47 FORMAT(//' TIME = ',F12.3,' SEC',8X,'CHANNEL',I2, 
    1 ' ACQUISITION STARTED')
ACQODWEND(ILEO) = TCUR + ACQODWUR
END IF
ENDIF

C ...............................................................
C
C If mode is 'slew to reaq' and true disc steps are close to previous
C commands, then change mode to 'reacquisition'.

C IF(MODE(ICHL).EQ.'SLEW TO REAQ') THEN

C IABS(ISTEPATRUE(ICHL)-ISTEPAOLD(ICHL)) .LE. ISTEPTOLA .AND.
C IABS(ISTEPBTRUE(ICHL)-ISTEPBOLD(ICHL)) .LE. ISTEPTOLA THEN

C MODE(ICHL) = 'REACQUISITION'

C WRITE(20,48) TCUR,ICHL
C FORMAT('/' TIME =',F12.3,' SEC',BX,'CHANNEL',12,
C 8X,' REACQUISITION STARTED')

C ACQDWEND(ILEO) = TCUR + ACQDWUR

END IF
END IF

C IF(WFLAG) THEN
WRITE(15,55) MOOE(ICHL)
C FORMAT(/A16)
C MODECHG = '

C END IF

C C If mode is 'acquisition', 'reacquisition' or 'tracking':
C
C IF(MODE(ICHL).EQ.'ACQUISITION' .OR.
C MODE(ICHL).EQ.'REACQUISITION' .OR.
C MODE(ICHL).EQ.'TRACKING') THEN

C If eclipsed, the LEO is not in track.

C IF(ECLIPSE(ILEO)) THEN
INTRACK(ILEO) = .FALSE.
C IF(WFLAG) WRITE(15,60)
FORMAT(17X,'NOT IN TRACK / ECLIPSE')
C IF(MODE(ICHL).EQ.'TRACKING') WRITE(20,65) TCUR,ICHL
C FORMAT('/' TIME =',F12.3,' SEC',BX,'CHANNEL',12,
C 8X,' NOT IN TRACK / ECLIPSE')
C ELSE

C IF true disc steps are close to previous commands, then get information
C from the Angle Processors.

C IF((IABS(ISTEPATRUE(ICHL)-ISTEPAOLD(ICHL)) .LE. ISTEPTOLT .AND.
C IABS(ISTEPBTRUE(ICHL)-ISTEPBOLD(ICHL)) .LE. ISTEPTOLT) .OR.
C TEST) THEN

C CALL ANGLEPR(AZS,ELS,AZAHEAD,ELAHEAD,DLATRUE,ALTRUE,ILEO,ICHL,INTRACK,U,V)

C C If in-track flag is true and mode is 'acquisition' or 'reacquisition'
C then change mode to 'tracking'.

C IF(INTRACK(ILEO)) THEN

C IF(MODE(ICHL).EQ.'ACQUISITION') THEN
NACQS = NACQS + 1
WRITE(20,70) TCUR,ICHL
FORMAT('/' TIME =',F12.3,' SEC',BX,'CHANNEL',12,
8X,'ACQUISITION COMPLETED')
END IF
IF(MODE(ICHL).EQ.'REACQUISITION')WRITE(20,72)TCUR,ICHL
    FORMAT(/' TIME =',F12.3,' SEC',8X,'CHANNEL',I2,
       ' REACQUISITION COMPLETED')
227
IF(MODE(ICHL).NE.'TRACKING') THEN
228  MODE(ICHL) = 'TRACKING'
229  MOCSCG = 'TRACKING'
230  END IF
231
232 C-----------------------------------------------------------------------
233 C Error Processor transforms u,v errors from Angle Processors to
234 C azimuth and elevation.
235 C-----------------------------------------------------------------------
236 ROT = HALFP + DLA(ICHL) + ALSTEP(ICHL)
237 COSROT = DCOS(ROT)
238 SINKROT = DSIN(ROT)
239
240 AZERR(ILEO) = U(ILEO)*COSROT + V(ILEO)*SINKROT +
241              AZAHEAD(ILEO)
242 1
243 ELERR(ILEO) = -U(ILEO)*SINKROT + V(ILEO)*COSROT +
244              ELAHEAD(ILEO)
245 1
246
247 IF(WFLAG) WRITE(15,80) U(ILEO)*ID6, V(ILEO)*ID6,
248          AZERR(ILEO)*ID6, ELERR(ILEO)*ID6,
249 80 FORMAT(17X,'U ERR =',F11.1,SX,'V ERR =',F11.1,
250          '(MICRORAD)'/17X,'AZ ERR =',F11.1,5X,'EL ERR =',
251          F11.1,' (MICRORAD)')
252
253 C-----------------------------------------------------------------------
254 C Offset Generator updates azimuth and elevation offsets with
255 C respective errors.
256 C-----------------------------------------------------------------------
257 AZOFFSET(ILEO) = AZOFFSET(ILEO) + TRKGAIN*AZERR(ILEO)
258 ELOFFSET(ILEO) = ELOFFSET(ILEO) + TRKGAIN*ELERR(ILEO)
259
260 ELSE
261
262 IF(WFLAG) WRITE(15,90)
263 90 FORMAT(17X,'NOT IN TRACK / OUTSIDE HE0 BEAM')
264
265 IF(MODE(ICHL).EQ.'TRACKING') THEN
266  IF(REALTIME) WRITE(20,95) TCUR,ICHL
267  IF(.NOT.REALTIME) WRITE(20,96) TCUR,ICHL
268 95 FORMAT(/' TIME =',F12.3,' SEC',8X,'CHANNEL',I2,
269          ' NOT IN TRACK')
270 96 FORMAT(/' TIME =',F12.3,' SEC',8X,'CHANNEL',I2,
271          ' NOT IN TRACK / OUTSIDE HE0 BEAM')
272 1
273  END IF
274  END IF
275
276 C-----------------------------------------------------------------------
277 C If true disc steps are not close to previous commands, then change mode
278 C to 'scan inhibit'.
279 C-----------------------------------------------------------------------
280 ELSE
281
282 INHMOOE(ICHL) = MODE(ICHL)
283 INHCOUNT(ICHL) = 1
284 INHCHECK(ICHL) = 0
285 MODE(ICHL) = 'SCAN INHIBIT'
286 IF(WFLAG) WRITE(15,55) MODE(ICHL)
287 55 WRITE(20,52) TCUR,ICHL
288 52 FORMAT(/' TIME =',F12.3,' SEC',8X,'CHANNEL',I2,
289          ' SCAN INHIBITED')
290 1
291  END IF
292
293 END IF
C--- If mode is 'tracking' and in-track flag is false:
C  In real-time, set mode to 'slew to reacq' and set previous tracking
C  offsets to zero.
C  In simulation, set mode to 'reacquisition' and save previous tracking
C  offsets.
C  Set scan reset flag to true.
C---------------------------------------------------------------------

IF(MODE(ICHL).EQ.'TRACKING' .AND. .NOT.INTRACK(ILEO)) THEN
C Removed the following lines of code on 8-26-92 RJP

C---------------------------------------------------------------------
C IF(REALTIME) THEN
C MODE(ICHL) = 'SLEW TO REACQ'
C MODECHG = 'SLEW TO REACQ'
C AZOFFSETP(ILEO) = 0
C ELOFFSETP(ILEO) = 0
C ELSE
C MODE(ICHL) = 'REACQUISITION'
C IF(WFLAG) WRITE(15,55) MODE(ICHL)
C WRITE(20,48) TCUR,ICHL
C AZOFFSETP(ILEO) = AZOFFSET(ILEO)
C ELOFFSETP(ILEO) = ELOFFSET(ILEO)
C END IF
C---------------------------------------------------------------------

SCANRES(ILEO) = .TRUE.

END IF

END IF

C---------------------------------------------------------------------
C If mode is 'park', set arm angle alpha and disc angle delta.
C---------------------------------------------------------------------

IF(MODE(ICHL).EQ.'PARK') THEN

ISOL = 1
AL(ICHL,ISOL) = ALPARK
DL(ICHL,ISOL) = DLPARK

ELSE

C---------------------------------------------------------------------
C If mode is 'slew to acq' or 'acquisition' and a new scan position is
C required, the Offset Generator uses the Scan Generator output.
C---------------------------------------------------------------------

IF((MODE(ICHL).EQ.'SLEW TO ACQ'.AND.SCANRES(ILEO)) .OR. 
(MODE(ICHL).EQ.'ACQUISITION' .AND. 
(SCANRES(ILEO).OR.TCUR.GT.ACQDWDEND(ILEO)))) THEN
CALL SCANGN(ILEO, ICHL,MODE,SCANRES,TCUR,AZOFFSET,ELOFFSET)
ACQDWDEND(ILEO) = TCUR + ACQDWDUR

C---------------------------------------------------------------------

ELSE IF((MODE(ICHL).EQ.'SLEW TO REACQ'.AND.SCANRES(ILEO)) .OR. 
(MODE(ICHL).EQ.'REACQUISITION' .AND. 
(SCANRES(ILEO).OR.TCUR.GT.ACQDWDEND(ILEO)))) THEN
CALL SCANGN(ILEO, ICHL,MODE,SCANRES,TCUR,AZOFFSET,ELOFFSET)
AZOFFSET(ILEO) = AZOFFSETP(ILEO) + AZSCAN(ILEO)
ELOFFSET(ILEO) = ELOFFSETP(ILEO) + ESLCAN(ILEO)

ENDIF

C---------------------------------------------------------------------

ELSE IF((MODE(ICHL).EQ.'SLEW TO REACQ'.AND.SCANRES(ILEO)) .OR. 
(MODE(ICHL).EQ.'REACQUISITION' .AND. 
(SCANRES(ILEO).OR.TCUR.GT.ACQDWDEND(ILEO)))) THEN
CALL SCANGN(ILEO, ICHL,MODE,SCANRES,TCUR,AZOFFSET,ELOFFSET)
AZOFFSET(ILEO) = AZOFFSETP(ILEO) + AZSCAN(ILEO)
ELOFFSET(ILEO) = ELOFFSETP(ILEO) + ESLCAN(ILEO)

ENDIF

ACQDWDEND(ILEO) = TCUR + ACQDWDUR
C If mode is 'scan inhibit', the Offset Generator performs a square test
C pattern.
C
ELSE IF(MODE(ICHL).EQ.'SCAN INHIBIT') THEN
AZOFFSET(ILEO) = ORTRBEAM * (-1)**(1+INHCOUNT(ICHL)/2)
ELOFFSET(ILEO) = QRTRBEAM * (-1)**((1+INHCOUNT(ICHL))/2)
END IF

IF(WFLAG) WRITE(15,45) AZOFFSET(ILEO)*ID3, ELOFFSET(ILEO)*ID3
45 FORMAT(17X,'AZ OFFS =',F11.4,5X,'EL OFFS =',F11.4,
1 ' (MILLIRAD)')

C Disc & Arm Controller adds azimuth and elevation from Navigation
C Processor data and Offset Generator.

AZSUM(ILEO) = AZNAVT(ILEO) + AZOFFSET(ILEO)
ELSUM(ILEO) = ELNAVT(ILEO) + ELOFFSET(ILEO)

C Azimuth and elevation are transformed to arm angle alpha and
disc angle delta.

ISOL = IALDLSOL(ICHL)

CALL AZELALDL(AZSUM,ELSUM,ILEO,ICHL,ISOL,AL,DL)

END IF

C Arm angle alpha and disc angle delta are transformed to disc
C driver A & B angles.

CALL ALDLDLAB(AL,DL,ICHL,ISOL,INITDIR,TCUR,DLA,DLB,
1 ISTEPA,ISTEPB,ALSTEP)

C If mode is 'scan inhibit' and changes in disc step commands are small
C enough for acquisition or reacquisition, change mode to 'slew to acq' or
C 'slew to reacq' depending on previous mode.

IF(MODE(ICHL).EQ.'SCAN INHIBIT') THEN
IF(INHCOUNT(ICHL).GE.2) THEN
IF(IABS(ISTEPA(ICHL)-ISTEPAOLD(ICHL)).LE.INHSTEPS
AND. IABS(ISTEPB(ICHL)-ISTEPBOLD(ICHL)).LE.INHSTEPS) THEN
IF(INHCHECK(ICHL).EQ.3) THEN
IF(INHMOOE(ICHL).EQ.'ACQUISITION,) THEN
MOOE(ICHL) = 'SLEW TO ACQ'
MODECHG = 'SLEW TO ACQ'
ELSE
MOOE(ICHL) = 'SLEW TO REACQ'
MODECHG = 'SLEW TO REACQ'
AZOFFSETP(ILEO) = 0
ELOFFSETP(ILEO) = 0
END IF
SCANRES(ILEO) = .TRUE.
END IF
END IF

INHCHECK(ICHL) = INHCHECK(ICHL) + 1
ELSE
    INHCHECK(ICHL) = 0
END IF

END IF

INHCOUNT(ICHL) = INHCOUNT(ICHL) + 1

END IF

IF(WFLAG) WRITE(15,50) MODECHG, DL(ICHL,ISOL)*RAD,
1 AL(ICHL,ISOL)*RAD, ISOL, ISTEPA(ICHL), ISTEPB(ICHL),
2 DLAC(ICHL)*RAD, DLB(ICHL)*RAD, ALSTEPA(ICHL)*RAD
END IF

50 FORMAT(Al/16X,'DELTA == ',F11.5,' (DEG)',
1 5X,'ALPHA == ',F11.5,' (DEG)',
2 17X,'STEP A == ',F11.5,' (DEG)',
3 42X,'STEP AL == ',F11.5,' (DEG)')
C...............................................................................
C In real-time mode, save disc step commands.
C.............................................................................
C...............................................................................
C IF(REALTIME) THEN
1 IF(TEST) THEN
2 ISTEPASAVE = ISTEPAPARK
3 ISTEPBSAVE = ISTEPBPARK
4 ELSE
5 ISTEPASAVE = ISTEPA(ICHL)
6 ISTEPBSAVE = ISTEPB(ICHL)
7 END IF
END IF

ELSE
INITCHL(ICHL) = 0
ISTEPAOLD(ICHL) = ISTEPA(ICHL)
ISTEPBOLD(ICHL) = ISTEPB(ICHL)
RETURN
END
SUBROUTINE COPYVEC(U,N,V)
REAL*8 U(N),V(N)
DO I=1,N
   V(I)=U(I)
END DO
RETURN
END
SUBROUTINE CROSS(U,V,W)

REAL*8 U(3),V(3),W(3)

W(1)=U(2)*V(3)-U(3)*V(2)
W(2)=U(3)*V(1)-U(1)*V(3)
W(3)=U(1)*V(2)-U(2)*V(1)

RETURN

END
SUBROUTINE DFQ(X,DX,DT)

Differential equations for satellite state X.

Courtesy of Bob Dasenbrock of NRL. Modified for OMA.

IMPLICIT REAL*8 (A-H,O-Z)
REAL*8 MU,MUR
DIMENSION X(6),DX(6)
DATA MU /'3.9860095/.
R=DSQRT(X(1)**2+X(2)**2+X(3)**2)
MUR=MU/(R**3)
DX(1)=X(4)*DT
DX(2)=X(5)*DT
DX(3)=X(6)*DT
DX(4)=MUR*X(1)*DT
DX(5)=MUR*X(2)*DT
DX(6)=MUR*X(3)*DT
RETURN
END
SUBROUTINE DISCPOS (ISTEPAOLD,ISTEPOLD,MODE,INITCHL,ICHL, ISTEPATRUE,ISTEBTRUE,DLATRUE,DLBTRUE,ALTRUE)

In simulation mode, this routine computes and returns the true disc positions.

In real-time mode, this routine returns the true disc positions previously saved.

Prototype Version 1.0 Avtec Systems, Inc. August 1992

Inputs:
ISTEPAOLD = Previous disc A step commands
ISTEPOLD = Previous disc B step commands
MODE = Channel modes
INITCHL = Channel initializations
ICHL = Channel index
ISTEPMAX = Disc step at maximum rotation (DISCBLK)
MAXSLESTEP = Max #steps in an update interval if slewing (DISCBLK)
MAXTRACKSTEPS = Max #steps in an update interval if tracking (DISCBLK)
IRATE = Tracking rate (MISCBLK)
REALTIME = Real-time flag (MISCBLK)
ISTEPASAVE = True disc A step in real-time mode (SAVEBLK)
ISTEBSAVE = True disc B step in real-time mode (SAVEBLK)

Outputs:
ISTEPATRUE(ICHL) = True disc A step
ISTEBTRUE(ICHL) = True disc B step
DLATRUE(ICHL) = True disc A angle (delta A) in simulation mode
DLBTRUE(ICHL) = True disc B angle (delta B) in simulation mode
ALTRUE(ICHL) = True arm angle (alpha) in simulation mode

IMPLICIT REAL*8 (A-H, O-Z)
REAL*8 DLATRUE(6),DLBTRUE(6),ALTRUE(6),APHS(6),BPHS(6)
INTEGER*4 ISTEPAOLD(6),ISTEPOLD(6),ISTEPATRUE(6),ISTEBTRUE(6)
INTEGER*4 ISTEPMAX(6),ISTEPASAVE(6),ISTEBSAVE(6)
INTEGER*2 MAXSTEPS
INTEGER*13 MODE(6)
CHARACTER*13 MODE(6)
INCLUDE 'CIRCBLK.FCB'
INCLUDE 'DISCBLK.FCB'
INCLUDE 'MISCBLK.FCB'
INCLUDE 'SAVEBLK.FCB'
DATA ACCUR /200/ ! Disc step accuracy
DATA F /ADD/ ! Period factor for accuracy model
DATA MAXSTEPERR /2/ ! Maximum random step error
DATA INIT /1/

In real-time mode, return the true disc positions previously saved.

IF(REALTIME) THEN
ISTEPATRUE(ICHL) = ISTEPASAVE
ISTEBTRUE(ICHL) = ISTEPBSAVE
RETURN
ENDIF

In simulation mode, compute and return the true disc positions.

IF(INIT.EQ.1) THEN
DISCSTEP = T90PI/NSTEPS
DLMX = DISCSTEP * (ISTEPMAX + 4.9990-1)
ERRAMPL = ACCUR * DISCSTEP
ALSET = ALSETDEG / RAD

DO I=1,6
    CALL RNDM(UNIF)
    APHS(I) = TWOPI*UNIF
    CALL RNDM(UNIF)
    BPHS(I) = TWOPI*UNIF
END DO

C Compute the true disc steps. Simulation begins in park positions.

IF(INITCHL(ICHL).EQ.1) THEN
    JSTEPATRUE = ISTEPAPARK
    JSTEPBTRUE = ISTEPBPARK
ELSE
    IF(MODE(ICHL).EQ.'ACQUISITION'.OR.MOOE(ICHL).EQ.'REACQUISITION' .OR.MODE(ICHL).EQ.'TRACKING') THEN
        MAXSTEPS = MAXTRACKSTEPS
    ELSE
        MAXSTEPS = MAXSLEWSTEPS
    END IF
    ISTEPDIFF = ISTEPAOLD(ICHL) - ISTEPATRUEOLD(ICHL)
    JSTEPATRUE = NINT((JSTEPATRUE + (2*UNIF-1) * MAXSTEPERR)

C Include random disc step errors.

IF(INITCHL(ICHL).EQ.1 .OR. (MODE(ICHL).EQ.'PARK' .AND. JSTEPATRUE.EQ.ISTEPAPARK)) THEN
    DLRNDMA = DISCSTEP * ISTEPAPARK
ELSE
    CALL RNDM(UNIF)
    DLRNDMA = DISCSTEP * (JSTEPATRUE + (2*UNIF-1) * MAXSTEPERR)
END IF

C Determine true disc and arm angles.
DLATRUE(ICHL) = DLRNDMA + ERRAMPL * DSIN(F*DLRNDMA + APHS(ICHL))
DLBTRUE(ICHL) = DLRNDMB + ERRAMPL * DSIN(F*DLRNDMB + BPHS(ICHL))
ALTRUE(ICHL) = ALSET + DRATIO * (DLATRUE(ICHL) - DLBTRUE(ICHL))

C Save true disc steps.
ISTEPATRUEOLD(ICHL) = JSTEPATRUE
ISTEPBTRUEOLD(ICHL) = JSTEPBTRUE
RETURN
END
REAL*8 FUNCTION DOT(U,V)

REAL*8 U(3),V(3)

DOT=U(1)*V(1)+U(2)*V(2)+U(3)*V(3)

RETURN

END
SUBROUTINE ELSTAT (ELD, X)

C ELD IN THE INPUT KEPLERIAN STATE IN KM AND DEGREES
C X IS THE OUTPUT CARTESIAN STATE IN KILOMETERS
C
WHERE:
C ELD(1)=SEMI-MAJOR AXIS (KM)
C ELD(2)=ECCENTRICITY
C ELD(3)=INCLINATION (DEG)
C ELD(4)=LONG OF ASC NODE (DEG)
C ELD(5)=ARG OF PERIGEE (DEG)
C ELD(6)=MEAN ANOMALY (DEG)
C
X(1)=X (KM)
X(2)=Y (KM)
X(3)=Z (KM)
X(4)=XDOT (KM/S)
X(5)=YDOT (KM/S)
X(6)=ZDOT (KM/S)

IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION X(6), ELD(6), A(3,2)
XMU=398600.000
PI=3.14159265358979323800
DTR=PI/180.000
SNI=DSIN(ELD(3)*DTR)
CNI=DCOS(ELD(3)*DTR)
$OM=DSIN(ELD(4)*DTR)
COM=DCOS(ELD(4)*DTR)
XM=DMOD(ELD(6),360.000)*DTR
E=XKEP(ECC,XM,1.000-14)
SINE=DSIN(E)
COSE=DCOS(E)
STA=SQRT(1.000-ECC**2)*SINE/(1.000-ECC*COSE)
CTA=(COSE-ECC)/(1.000-ECC*COSE)
TAA=ARKTNS(180,CTA,STA)
TBB=TAA+DTR*ELD(5)
CBA=DCOS(TBB)
SBA=DSIN(TBB)
A(1,1)=+COM*CBA-SOM*CNI*SBA
A(2,1)=+SOM*CBA+COM*CNI*SBA
A(3,1)=SNI*SBA
A(1,2)=+COM*SBA-SOM*CNI*CBA
A(2,2)=+SOM*SBA+COM*CNI*CBA
A(3,2)=SNI*CBA
P=ELD(1)*(1.000-ECC**2)
R=P/(1.000+ECC*CTA)
VR=ECC*STA*SORT(XMU/P)
VT=SORT(XMU*(2.000/R-1.000/ELD(1))-VR*VR)
DO 10 K=1,3
10 X(K+3)=VR*A(K,1)+VT*A(K,2)
RETURN
END
FUNCTION GAUSCL(XMEAN,SIGMA,CUTOFF)

C GENERATES A RANDOM NUMBER FROM A CLIPPED GAUSSIAN DISTRIBUTION, WHERE THE TAILS ARE CUT OFF.

C INPUTS:
C XMEAN = MEAN OF GAUSSIAN DISTRIBUTION
C SIGMA = STANDARD DEVIATION OF GAUSSIAN DISTRIBUTION
C CUTOFF = CUTOFF VALUE IN UNITS OF SIGMA

C OUTPUT:
C GAUSCL = RANDOM NUMBER FROM CLIPPED GAUSSIAN DISTRIBUTION

C---------------------------------------------------------------------

IMPLICIT REAL*8 (A-H,O-Z)
50 GAUSCL = GAUSS(0D0,1D0)
IF(DABS(GAUSCL) .GT. CUTOFF) GO TO 50
GAUSCL = XMEAN + GAUSCL*SIGMA
RETURN
END
FUNCTION GAUSS (XMEAN, VARIANCE)

G generatiance a random number from a Gaussian distribution

Inputs:
XMEAN = mean of Gaussian distribution
VARIANCE = variance of Gaussian distribution

Output:
GAUSS = random number

Reference:

IMPLICIT REAL*8 (A-H, O-Z)

DATA TWOPI/6.2831853071795865D0/

CALL RNDM(X1)
CALL RNDM(X2)
Y = DSQRT(-2*VARIANCE*DLOG(X1))
GAUSS = XMEAN + Y*DCOS(TWOPI*X2)
RETURN
END
SUBROUTINE INTRFACE

Interface between control software and executive program.

Prototype Version 1.0  Avtec Systems, Inc.  August 1992

Entry points:

INIT  Initialization of control software
POSITION  Computation of new position commands for a disc channel

Inputs:

REALTIMEFLAG = Real-time flag (LOGICAL), argument in ENTRY INIT:

True indicates real-time mode
False indicates simulation mode

In real-time mode:

Input file name is INPUT.DAT
Az-EI file name is AZEL.DAT (see SUBROUTINE ASGNMFN)

In simulation mode, input file name (CHARACTER*12) is entered from
the keyboard

The following inputs are contained in the input file:

SIMHRS  = Duration of simulation in hrs (REAL*8)
          (used only in simulation mode)
WCHANNELS = Number of disc channels (INTEGER: 1-6)
IRATE  = Tracking rate in Hz (INTEGER)
TRKGAIN  = Tracking gain (REAL*8)
WFLAG = Write flag for output from the Truth Generator, Scene
Processor, Navigation Processor, Assignment Function
         and Control Function (LOGICAL, false in real-time mode)

Output file names (CHARACTER*12) are entered from
the keyboard

HEL(J) = Classical orbital elements for HEO at time 0 (REAL*8):

(1) = Semi-major axis (km)
(2) = Eccentricity
(3) = Inclination (deg)
(4) = Longitude of ascending node (deg)
(5) = Argument of perigee (deg)
(6) = Mean anomaly (deg)

NLEOS = Number of LEO satellites (INTEGER: 1-6)

LEL(J,I) = Classical orbital elements for LEO I at time 0 (REAL*8)

Outputs:

Event Log

Truth Generator  (if WFLAG is true)
Scene Processor  (if WFLAG is true)
Navigation Processor (if WFLAG is true)
Assignment Function (if WFLAG is true)
Control Function  (if WFLAG is true)

In real-time mode, Event Log file name is EVENT.DAT

In simulation mode, output file names (CHARACTER*12) are entered from
the keyboard
IMPLICIT REAL*8 (A-H,O-Z)
REAL*8 U [FAR]
REAL*8 V [FAR]
REAL*8 HEL(6),LEL(6,6),HEO(6),LEO(6,6),MU
INTEGER*4 ISTEPA [FAR]
INTEGER*4 ISTEPA [FAR]
INTEGER*4 ICHORDER6,ICHLPRSEQ6,ICHLLEO6,LEOCHL6
CHARACTER*12 Filename
LOGICAL REALTIMEFLAG [FAR]
LOGICAL INTRACK [FAR]
LOGICAL DONE
INCLUDE 'CIRCBLK.FCB'
INCLUDE 'MISCBLK.FCB'
INCLUDE 'ORBITBLK.FCB'
INCLUDE 'SAVEBLK.FCB'
DATA ICHLORDER /1,2,3,4,5,6/ ! Channel command order
DATA IORDER /0/ ! Channel order index
DATA ICHLPRSEQ/1,2,3,4,5,6/ ! Channel priority sequence
DATA LEOSEQ /0/ ! LEO sequence
DATA LEOCHL /6*0/ ! LEO channel assignments
DATA ICHLLEO /6*0/ ! Channel LEO assignments
DATA INITP /1/ ! Initialization of ENTRY POSITION
DATA RE /6.37814503/ ! Earth radius (km)
DATA MU /3.986000405/ ! Gravitational parameter

ENTRY INIT_REALTIMEFLAG

REALTIME = REALTIMEFLAG

IF(REALTIME) THEN
OPEN(1,FILE='INPUT.DAT',STATUS='OLD')
ELSE
WRITE(*,1)
1 FORMAT(' Enter input file name (e.g., Up to 8 characters.DAT)')
READ(*,2) Filename
2 FORMAT(A)
OPEN(1,FILE=FILENAME,STATUS='OLD')
END IF

READ(I,*) SIHHRS ! Used only in simulation mode
READ(I,*) NCHANNELS
READ(I,*) IRATE
READ(I,*) TRKGAIN
READ(I,*) WFLAG
IF(REALTIME) WFLAG = .FALSE.

IF(NCHANNELS.LT.1 OR. NCHANNELS.GT.6) STOP 151
IF(IRATE.LT.1) STOP 152
IF(TRKGAIN.LE.0) STOP 153

DO J=1,6
READ(I,*) HEL(J)
END DO
C In real-time mode, read the predicted Az & El of LEO satellites and
C select alpha & delta solutions.
C
IF(REALTIME) CALL ASGNMFNR(NLEOS,NCHANNELS,ICHLPRSEQ)
C
C Read the names of output files.
C
IF(WFLAG) THEN
  WRITE(*,41) FORMAT(' Enter file name for Truth Generator output',
                   1 '(e.g., Up to 8 characters.DAT))
  READ(*,2) FILENAME
  OPEN(11,FILE=FILENAME,STATUS='NEW')
  WRITE(*,42) FORMAT(' Enter file name for Scene Processor output',
                   1 '(e.g., Up to 8 characters.DAT))
  READ(*,2) FILENAME
  OPEN(12,FILE=FILENAME,STATUS='NEW')
  WRITE(*,43) FORMAT(' Enter file name for Navigation Processor output',
                   1 '(e.g., Up to 8 characters.DAT))
  READ(*,2) FILENAME
  OPEN(13,FILE=FILENAME,STATUS='NEW')
  WRITE(*,44) FORMAT(' Enter file name for Assignment Function output',
                   1 '(e.g., Up to 8 characters.DAT))
  READ(*,2) FILENAME
  OPEN(14,FILE=FILENAME,STATUS='NEW')
  WRITE(*,45) FORMAT(' Enter file name for Control Function output',
                   1 '(e.g., Up to 8 characters.DAT))
READ(*,2) FILENAME
OPEN(15,FILE=FILENAME,STATUS='NEW')
END IF
IF(REALTIME) THEN
OPEN(20,FILE='EVENT.DAT',STATUS='UNKNOWN')
ELSE
WRITE(*,46)
FORMAT('Enter file name for Event Log output',
1 ' (e.g., Up to 8 characters.DAT)')
READ(*,2) FILENAME
OPEN(20,FILE=FILENAME,STATUS='NEW')
END IF
WRITE(20,60)
FORMAT(1/' EVENT LOG'/',
'......... ')
TCUR = 0
TSTOP = SIMHRS*3.6D3
DT = 1D0/(IRATE*NCHANNELS)
RETURN
C-------------------------------------------------------------
ENTRY POSITION (ICHANNEL,INTRACK,U,V,ISTEPA,ISTEPB)
C Computation of new position commands for a disc channel.
C Inputs (real-time mode only):
C ICHANNEL = Channel index
C INTRACK = In-track flag
C U,V = Tracking detector errors (rad)
C ISTEPA = Current disc A resolution step
C ISTEPB = Current disc B resolution step
C Outputs (real-time mode only):
C ISTEPA = New command for disc A resolution step
C ISTEPB = New command for disc B resolution step
C-------------------------------------------------------------
IF(REALTIME) THEN
IF(U.GT.101 .OR. V.GT.1D1) THEN
TEST = .TRUE.
INTRACK = .TRUE.
U = 0
V = 0
ELSE
TEST = .FALSE.
END IF
INTRACKSAVE = INTRACK
USAVE = U
VSAVE = V
ISTEPASAVE = ISTEPA
ISTEPBSAVE = ISTEPB
ICHL = ICHANNEL
IORDER = 0
DONE = .FALSE.
DO WHILE (.NOT.DONE)
IORDER = IORDER + 1
IF (IORDER.GT.NCHANNELS) STOP 114
ICHLOD = ICHLORDER(IORDER)
IF (ICHLOD.EQ.IORDER) DONE = .TRUE.
END DO

IF(INITP.EQ.1) THEN
INTERVALS = IORDER - 1
ELSE
INTERVALS = IORDER - IORDEROLD
IF (INTERVALS.LE.0) INTERVALS = NCHANNELS + INTERVALS
END IF

TCUR = TCUR + INTERVALS * DT
IORDEROLD = IORDER
ILEO = ICHLLEO(ICHL)
IF (ILEO.EQ.0) THEN
J = 0
DONE = .FALSE.
DO WHILE (.NOT.DONE)
J = J + 1
ICHLPR = ICHLPRSEQ(J)
IF (ICHLLEO(ICHLPR).EQ.0) THEN
IF (J.LE.NLEOS) ILEO = J
DONE = .TRUE.
END IF
IF (J.EQ.NCHANNELS) DONE = .TRUE.
END DO
END IF

ELSE
IORDER = MOD(IORDER,NCHANNELS) + 1
ICHLOD = ICHLORDER(IORDER)
IF(INITP.EQ.0) TCUR = TCUR + DT
TCURM = TCUR/6D1
ITCURM = MINT(TCURM)
IF (DABS(TCURM-ITCURM).LT.1D-6) WRITE(*,80) ITCURM
80 FORMAT(IT, ' MIN')
IF (TCUR.GT.TSTOP) THEN
WRITE(*,90) TSTOP
90 FORMAT(/' END OF SIMULATION')
WRITE(20,95) TSTOP
95 FORMAT(/' TIME =',F12.3,' SEC','END OF SIMULATION')
STOP
END IF
ILEO = ICHLLEO(ICHL)
IF (ILEO.EQ.0) THEN
J = 0
DONE = .FALSE.
DO WHILE (.NOT.DONE)
J = J + 1
ICHLPR = ICHLPRSEQ(J)
IF (ICHLLEO(ICHLPR).EQ.0) THEN
J = 0
ENDIF
ENDIF
DO WHILE (.NOT.DONE)
  ILEOSEQ = MOD(ILEOSEQ,NLEOS) + 1
  IF(ILEOCHL(ILEOSEQ).EQ.0) THEN
    ILEO = ILEOSEQ
    DONE = .TRUE.
  END IF
  I = I + 1
  IF(I.EQ.NLEOS) DONE = .TRUE.
END DO
END IF

END IF

END IF

IF(J.EQ.NCHANNELS) DONE = .TRUE.
END DO

C .............................................................................
C
C Proceed to Navigation Function.
C .............................................................................
C
CALL NAVIGFN(HEO,LEO,NLEOS,ILEO,NCHANNELS,ICHL,LEOCHL,
1 ICHLLEO,TCUR)

C .............................................................................
C In real-time mode, get disc step commands.
C .............................................................................
C
IF(REALTIME) THEN
  ISTEPA = ISTEPASAVE
  ISTEPB = ISTEPBSAVE
END IF

INITP = 0
RETURN
END
SUBROUTINE INV3X3(MAT)

REAL*8 MAT(3,3), TEMP(3,3), DET

DET = MAT(1,1)*MAT(2,2)*MAT(3,3) + MAT(1,2)*MAT(2,3)*MAT(3,1) +
& MAT(1,3)*MAT(2,1)*MAT(3,2) - MAT(1,3)*MAT(2,2)*MAT(3,1) -
& MAT(1,1)*MAT(2,3)*MAT(3,2) - MAT(1,2)*MAT(2,1)*MAT(3,3)

TEMP(1,1)=MAT(2,2)*MAT(3,3)-MAT(2,3)*MAT(3,2)
TEMP(1,2)=MAT(2,3)*MAT(3,1)-MAT(2,1)*MAT(3,3)
TEMP(1,3)=MAT(2,1)*MAT(3,2)-MAT(2,3)*MAT(3,1)

TEMP(2,1)=MAT(1,3)*MAT(3,2)-MAT(1,2)*MAT(3,3)
TEMP(2,2)=MAT(1,1)*MAT(3,3)-MAT(1,3)*MAT(3,1)
TEMP(2,3)=MAT(1,3)*MAT(3,1)-MAT(1,1)*MAT(3,2)

TEMP(3,1)=MAT(1,2)*MAT(2,3)-MAT(1,3)*MAT(2,2)
TEMP(3,2)=MAT(1,1)*MAT(2,1)-MAT(1,2)*MAT(2,3)
TEMP(3,3)=MAT(1,1)*MAT(2,2)-MAT(1,2)*MAT(2,1)

DO I=1,3
  DO J=1,3
    MAT(I,J)=TEMP(J,1)/DET
  END DO
END DO
RETURN
END
SUBROUTINE NAVIGFN (HEO, LEO, NLEOS, ILEO, NCHANNELS, ICHL,
  1 LEOCHL, ICHLLEO, TCUR)

This routine simulates the Navigation Function, which consists of the
Truth Generator, Scene Processor, and Navigation Processor.
Prototype Version 1.0  Avtec Systems, Inc. August 1992

Inputs:
- HEO = Geocentric-equatorial coordinates of HEO satellite
- LEO = Geocentric-equatorial coordinates of LEO satellites
- NLEOS = Number of LEO satellites
- ILEO = LEO satellite index
- NCHANNELS = Number of disc channels
- ICHL = Channel index
- LEOCHL = LEO channel assignments
- ICHLLEO = Channel LEO assignments
- TCUR = Current time
- IRATE = Tracking rate (NISCBLK)
- REALTIME = Real-time flag (HISCBLK)
- WFLAG = Write flag (HISCBLK)
- PRDH = Period of HEO satellite (ORBITBLK)
- PRDL = Periods of LEO satellites (ORBITBLK)

INPLICIT REAL*8 (A-H,O-Z)
REAL*8 HEO(6), LEO(6,6), HEONAV(6), LEONAV(6,6)
REAL*8 AZS(6), ELS(6)
INTEGER*2 LEOCHL(6), ICHLLEO(6), INITLEO(6)
LOGICAL ECLIPSE(6), OLDECLIPSE(6)
CHARACTER*I DASH(73), ECLIND

INCLUDED 'CIRCBLK.FCB'
INCLUDED 'HISCBLK.FCB'
INCLUDED 'ORBITBLK.FCB'

DATA ECLIPSE /6*.FALSE./
DATA DASH /73''/'/
DATA INIT /1/

C Truth Generator simulates orbital dynamics of HEO and LEO satellites.

IF(ILEO .NE. 0) THEN
  IF(.NOT.REALTIME) THEN
    CALL TRUTHGN(HEO, LEO, NLEOS, ILEO, TCUR)
    IF(WFLAG) THEN
      IF(INIT.EQ.1) WRITE(11,20) DASH
      WRITE(11,25) TCUR,(HEO(J),J=1,6)
      WRITE(11,26) ILEO,(LEO(J,ILEO),J=1,6)
    END IF
  END IF
  END IF

C Scene Processor determines FOV coordinates of LEO satellites.

IF(.NOT.REALTIME) THEN
CALL SCENEPR(HEO,LEO,NLEOS,Ileo,NCHANNELS,TCUR,
ECLIPSE,AZS,ELS)

IF(INITLEO(Ileo).EQ.1) THEN
  IF(ECLIPSE(Ileo)) WRITE(20,50) TCUR,ileo
  ELSE
    IF(ECLIPSE(Ileo).AND..NOT.OLEDCLIPSE(Ileo))
      WRITE(20,50) TCUR,ileo
  IF(.NOT.ECLIPSE(Ileo).AND.OLEDCLIPSE(Ileo))
  WRITE(20,55) TCUR,ileo
END IF

OLEDCLIPSE(Ileo) = ECLIPSE(Ileo)

IF(WFLAG) THEN
  IF(INIT.EQ.1) WRITE(12,60)
  ECLIND = ' '
  IF(ECLIPSE(Ileo)) ECLIND = 'E'
  WRITE(12,65) TCUR,ileo,ECLIND,AZS(Ileo)*RAD,ELS(Ileo)*RAD
END IF

END IF

50 FORMAT(/' TIME =',F12.3,' SEC',8X,'LEO',I2,' ECLIPSED')
55 FORMAT(/' TIME =',F12.3,' SEC',8X,'LEO',I2,' NOT
ECLIPSED')
60 FORMAT(/' SCENE PROCESSOR',22X,'ECLIPSE',5X,'AZ(DEG),
I 6X,'EL(DEG)')
65 FORMAT(/' TIME =',F12.3,' SEC',5X,'LEO',I2,A9,F15.5,F13.5)

C ...............................................................................
C Navigation Processor superimposes ephemeris errors.
C ...............................................................................

IF(.NOT.REALTIME) THEN
  CALL NAVGPR(HEO,LEO,NLEOS,Ileo,TCUR,HEONAV,LEONAV)
  IF(WFLAG) THEN
    IF(INIT.EQ.1) WRITE(13,80)
    WRITE(13,25) TCUR,(HEONAV(J),J=1,6)
    WRITE(13,26) Ileo,(LEONAV(J,ileo),J=1,6)
  END IF
END IF

80 FORMAT(/' NAVIGATION PROCESSOR (KM)',16X,'X',12X,'Y',12X,'Z'
/24X,'(KM/SEC)',12X,2HX,'11X,2HY',11X,2HZ/'//',73AI)
124 INITLO(Ileo) = 0
125 INIT = 0

127 END IF

C .............................................................................
C Proceed to Assignment Function.
C .............................................................................

CALL ASGNMFN(HEONAV,LEONAV,ECLIPSE,AZS,ELS,Ileo,ICHL,LEOCHL,
ICHILLEO,TCUR)
SUBROUTINE NAVIGPR (HEO,LEO,NLEOS,ILEO,TCUR,HEONAV,LEONAV)
C  This routine simulates the Navigation Processor, which superimposes
ephemeris errors on the true satellite positions and velocities.
C  Prototype Version 1.0  Avtec Systems, Inc.  August 1992
C  Entry points:
C  NAVIGPR  Initialization is performed in first call
C  NAVIGPR2 Initialization is not performed
C  Inputs:
C  HEO = Geocentric-equatorial coordinates of HEO satellite
C  LEO = Geocentric-equatorial coordinates of LEO satellites
C  NLEOS = Number of LEO satellites
C  ILEO = LEO satellite index
C  TCUR = Current time
C  PRDH = Period of HEO orbit (ORBITBLK)
C  PRDL = Periods of LEO orbits (ORBITBLK)
C  Outputs:
C  HEONAV = Coordinates of HEO satellite including ephemeris errors
C  LEONAV(ILEO) = Coordinates of LEO satellite including ephemeris errors

IMPLICIT REAL*8 (A-H,O-Z)
REAL*8 HEO(6),LEO(6,6),HEONAV(6),LEONAV(6,6)
REAL*8 HPHS(3),LPHS(3,6),LFREQ(6)
INCLUDE 'CIRCBLK.FCB'
INCLUDE 'ORBITBLK.FCB'
DATA INIT /1/
IF(INIT.EQ.1) THEN
  DO J=1,3
      CALL RNDM(UNIF)
      HPHS(J) = TWOP*XUNIF
  DO I=1,NLEOS
      CALL RNDM(UNIF)
      LPHS(J,I) = TWOP*XUNIF
  END DO
  END DO
  HFREQ = TWOP/PRDH
  DO I=1,NLEOS
      LFREQ(I) = TWOP/PRDL(I)
  END DO
  INIT = 0
END IF
ENTRY NAVIGPR2 (HEO,LEO,ILEO,TCUR,HEONAV,LEONAV)
C
TEMPH = HFREQ*TCUR
TEMPL = LFREQ(ILEO)*TCUR
DO J=1,3
  ARG = TEMPH*HPHS(J)
  HEONAV(J) = HEO(J) + EPHH*DSIN(ARG)
  HEONAV(J+3) = HEO(J+3) + EPHHWHFREQWDCOS(ARG)
ARG = TEMPL*LPHS(J,ILED)
LEONAV(J,ILED) = LEO(J,ILED) + EPHL*DSIN(ARG)
LEONAV(J+3,ILED) = LEO(J+3,ILED) + EPHL*LFREQ(ILED)*DCOS(ARG)
END DO
RETURN
END
C-------------------------------------------------------------
1
2
3 PROGRAM OMA
4
5 C Control software simulation for the Optical Multiple Access (OMA) system.
6 C (compatible with Microsoft FORTRAN Version 5.0 and VAX FORTRAN V5.5)
7 C
8 C This main program simulates the real-time executive program.
9 C
10 C Prototype Version 1.0 Avtec Systems, Inc. August 1992
11 C
12 C
13 C Inputs and outputs: see SUBROUTINE INTERFACE
14 C
15 C
16 C-------------------------------------------------------------
17
18 REAL*8 U,V
19 INTEGER*2 ICHANNEL
20 INTEGER*4 ISTEPA,ISTEPB
21 LOGICAL REALTIRE,INTRACK
22
23 DATA REALTIRE / .FALSE. /
24
25 C-------------------------------------------------------------
26 C In simulation mode:
27 C
28 C REALTIME is false.
29 C ICHANNEL,INTRACK,U,V,ISTEPA,ISTEPB are not used.
30 C
31 C-------------------------------------------------------------
32 CALL INIT(REALTIRE)
33
34 10 CALL POSITION(ICHANNEL,INTRACK,U,V,ISTEPA,ISTEPB)
35 GO TO 10
36
37 END
SUBROUTINE OFIT(HINTG,LINTG,NLEOS,T1,DTINTG)

This routine fits a quadratic function of time to each position coordinate and velocity component using 3 points.

Prototype Version 1.0  Avtec Systems, Inc. August 1992

IMPLICIT REAL*8 (A-H,O-Z)

REAL*8 HEO(6),LEO(6,6),HINTG(6,3),LINTG(6,6,3)
REAL*8 TMAT(3,3),HCOEFF(6,3),LCOEFF(6,6,3)

TMAT(1,2) = T1
TMAT(2,2) = T1 + DTINTG
TMAT(3,2) = T1 + 2*DTINTG

DO J = 1,3
   TMAT(J,1) = 1
   TMAT(J,3) = TMAT(J,2)**2
END DO

CALL INV3X3(TMAT)

DO ICOORD = I,6
   DO ITERM = I,3
      TEMP = 0
      DO J = 1,3
         TEMP = TEMP + HINTG(ICOORD,J)*TMAT(ITERM,J)
      END DO
      HCOEFF(ICOORD,ITERM) = TEMP
   END DO
   DO I = 1,NLEOS
      TEMP = 0
      DO J = 1,3
         TEMP = TEMP + LINTG(ICOORD,I,J)*TMAT(ITERM,J)
      END DO
      LCOEFF(ICOORD,I,ITERM) = TEMP
   END DO
END DO

ENTRY QVAL(TCUR,ILEO,HEO,LEO)

Evaluate quadratic functions of position coordinates and velocity components.

Inputs:
TCUR = Current time
ILEO = LEO satellite index

Outputs:
HEO = Geocentric-equatorial coordinates of HEO satellite
LEO = Geocentric-equatorial coordinates of LEO satellites
TCURSQ = TCUR**2
DO ICOORD=1,6
   HEO(ICOORD) = HCOEFF(ICOORD,1) + HCOEFF(ICOORD,2)*TCUR +
            HCOEFF(ICOORD,3)*TCURSQ
   LEO(ICOORD,ILEO) = LCOEFF(ICOORD,ILEO,1) +
            LCOEFF(ICOORD,ILEO,2)*TCUR + LCOEFF(ICOORD,ILEO,3)*TCURSQ
END DO
RETURN
END
SUBROUTINE RNDM (Z)

C GENERATES RANDOM NUMBERS HAVING A UNIFORM DISTRIBUTION, USING THE
C MULTIPlicative CONGRUENTIAL METHOD
C
C REFERENCE:
9 C B. CARNAHAN, H.A. LUTHER AND J.D. WILKES, "APPLIED NUMERICAL METHODS"
10 C JOHN WILEY & SONS, NEW YORK, NY, 1969, pp.545-549.
C
C---------------------------------------------------------------------

IMPLICIT REAL*8 (A-H,O-Z)
INTEGER*4 A,M,X
DATA I /1/
IF(I .EQ. 1) THEN
I = 0
M = 1048576
FM = M
X = 566387
A = 1027
END IF
X = MOD(A*X,M)
Z = X/FM
RETURN
END
SUBROUTINE RUK(X,DT,XNEW)

Fourth order Runge-Kutta integrator.

Courtesy of Bob Dasenbrock of NRL. Modified for OMA.

Inputs:

X(1,2,3) = current satellite position in geocentric coordinates (km)
X(4,5,6) = current satellite velocity in geocentric coordinates (km/sec)
DT = propagation time interval (sec)

Outputs:

XNEW(1,2,3) = satellite position propagated ahead DT sec
XNEW(4,5,6) = satellite velocity propagated ahead DT sec

IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION D(6),F(6),U(6),X(6),XNEW(6)

CALL DFQ(X,D,DT)

DO K=1,6
    U(K)=X(K)+0.5D0*D(K)
END DO

CALL DFQ(U,F,DT)

DO K=1,6
    D(K)=D(K)+2.0D0*F(K)
    U(K)=X(K)+0.5D0*F(K)
END DO

CALL DFQ(U,F,DT)

DO K=1,6
    D(K)=D(K)+2.0D0*F(K)
    U(K)=X(K)+F(K)
END DO

CALL DFQ(U,F,DT)

DO K=1,6
    XNEW(K)=X(K)+(F(K)+6.0D0*(D(K)+F(K))
END DO

RETURN
END
SUBROUTINE SCANGN (ILEO, ICHL, MODE, SCANRES, TCUR, AZSCAN, ELSCAN)

Scan Generator implements acquisition and reacquisition patterns.

Prototype Version 1.0  Avtec Systems, Inc.  August 1992

Inputs:
- ILEO = LEO satellite index
- ICHL = Channel index
- MODE = Channel modes
- SCANRES = Scan reset flags
- BEAMW = HEO beamwidth
- TCUR = Current time
- TSMALLSCAN = Earliest time for small scan pattern (ATTBLK)
- WFLAG = Write flag (MISCBLK)

Outputs:
- AZSCAN(ILEO) = Azimuth coordinate of scan position
- ELSCAN(ILEO) = Elevation coordinate of scan position
- MODE(ICHL) = Channel mode

 IMPLICIT REAL*8 (A-H,O-Z)
 REAL*8 AZSCAN(6), ELSCAN(6)
 INTEGER*2 IPOSIT(6), ICOUNT(6), ISIZE(6), ISIZE2(b), ISIGN(6)
 LOGICAL SCANRES(6), SMALLSCAN
 CHARACTER*13 MODE(6)

 DATA ISQUARE /1089/  ! Number of beams in square pattern
 DATA ISMALLSQUARE /121/  ! Number of beams in small square pattern
 DATA SMALLSCAN/.FALSE./
 DATA IPOSIT /6*0/
 DATA INIT /1/

 IF(INIT.EQ.1) THEN
  HALFBEAM = BEAMW/2
  QRTRBEAM = BEAMW/4
  INIT = 0
 END IF

 IF(.NOT.SMALLSCAN .AND. TCUR.GE.TSMALLSCAN) THEN
  ISQUARE = ISMALLSQUARE
  SMALLSCAN = .TRUE.
 END IF

 IF(IPPOSIT(ILEO).GE.ISQUARE .AND. .NOT.SCANRES(ILEO)) THEN
  SCANRES(ILEO) = .TRUE.
  IF(WFLAG) WRITE(15,20)
   20 FORMAT(17X,'REPEAT SCAN')
  WRITE(20,21) TCUR, ICHL
   21 FORMAT(/' TIME = ',F12.3,' SEC',8X,'CHANNEL',I2,' REPEAT SCAN')
  IF(MODE(ICHL).EQ.'ACQUISITION') THEN
   MODE(ICHL) = 'SLEW TO ACQ'
  ELSE
   MODE(ICHL) = 'SLEW TO REACQ'
  END IF
  IF(WFLAG) WRITE(15,23) MODE(ICHL)
   23 FORMAT(A16)
END IF

C If scan reset flag is true, initialize the scan pattern.

IF(SCANRES(ILEO)) THEN
  IPOSIT(ILEO) = 1
  ICOUNT(ILEO) = 0
  ISIZE(ILEO) = 1
  ISIZE2(ILEO) = 2
  ISIGN(ILEO) = 1
  SCANRES(ILEO) = .FALSE.
  IF(ISQUARE .EQ. ISQUARE/2 * 2) THEN
    AZSCAN(ILEO) = -QRTRBEAM
    ELSSCAN(ILEO) = -QRTRBEAM
  ELSE
    AZSCAN(ILEO) = 0
    ELSSCAN(ILEO) = 0
  END IF
ELSE
  C If scan reset flag is false, continue the scan pattern.
  IPOSIT(ILEO) = IPOSIT(ILEO) + 1
  ICOUNT(ILEO) = ICOUNT(ILEO) + 1
  IF(ICOUNT(ILEO) .LE. ISIZE(ILEO)) THEN
    ELSSCAN(ILEO) = ELSSCAN(ILEO) + HALFBEAM*ISIGN(ILEO)
  ELSE
    AZSCAN(ILEO) = AZSCAN(ILEO) + HALFBEAM*ISIGN(ILEO)
  END IF
  IF(ICOUNT(ILEO) .EQ. ISIZE2(ILEO)) THEN
    ICOUNT(ILEO) = 0
    ISIZE(ILEO) = ISIZE(ILEO) + 1
    ISIZE2(ILEO) = ISIZE(ILEO) * 2
    ISIGN(ILEO) = -ISIGN(ILEO)
  END IF
ENDIF

IF(WFLAG) WRITE(15,50) IPOSIT(ILEO)
50   FORMAT(17X,'SCANPOS =',110)
RETURN
END
SUBROUTINE SCENEPR (HEO, LEO, NLEOS, ILEO, NCHANNELS, TCUR, 
ECLIPSE, AZS, ELS)

The Scene Processor computes the azimuth & elevation of a received
LEO beam center, including the effects of attitude errors.

Prototype Version 1.0 Avtec Systems, Inc. August 1992

Inputs:
HEO = Geocentric-equatorial coordinates of HEO satellite
LEO = Geocentric-equatorial coordinates of LEO satellites
NLEOS = Number of LEO satellites
ILEO = LEO satellite index
NCHANNELS = Number of disc channels
TCUR = Current time
IRATE = Tracking rate (MISCBLK)
PRDH = Period of HEO orbit (ORBITBLK)

Outputs:
ECLIPSE(ILEO) = True if LEO is eclipsed by the earth
AZS(ILEO) = Azimuth in HEO FOV
ELS(ILEO) = Elevation in HEO FOV

IMPLICIT REAL*8 (A-H,O-Z)
REAL*8 HEO(6), LEO(6,6), AZS(6), ELS(6), AZAHEAD(6), ELAHEAD(6)
LOGICAL ECLIPSE(6), FINDADJT, DONE

INCLUDE 'ATTBLK.FCB'
INCLUDE 'CIRCBLK.FCB'
INCLUDE 'MISCBLK.FCB'
INCLUDE 'ORBITBLK.FCB'

DATA ATTRP /1D-3/
DATA ATTRPADJ /2,5D-4/
DATA ATTY /3D-3/
DATA ATTYADJ /7.5D-4/
DATA VIBSIGMA /1.585D-5/
DATA VIBBANDW /1DO/
DATA FINDADJT / .FALSE. /
DATA ROLADJT /1D30/, PCHADJT /1D30/, YAWADJT /1D30/
DATA OLDVIBROL /ODO/, OLDVIBPCH /ODO/, OLDVIBYAW /ODO/
DATA INIT /1/

IF (INIT.EQ.1) THEN
C = TWOP1 / (PRDH/2)
C2 = 1 - DEXP(-TWOP1*VIBBANDW/(IRATE*NCHANNELS)) ! All channels called
GVAR = (2-C2)/C2 * VIBSIGMA**2
CALL RNDM(UNIF)
ROLPHS = TWOP1*UNIF
CALL RNDM(UNIF)
PCHPHS = TWOP1*UNIF
CALL RNDM(UNIF)
YAWPHS = TWOP1*UNIF
IF(NLEOS.GE.4 .AND. NCHANNELS.GE.4) FINDADJT = .TRUE.
INIT = 0
END IF

C Generate random vibrations in roll, pitch and yaw.
VIBROL = OLDVIBROL + (GAUSS(ODD,GVAR)-OLDVIBROL) * C2
VIBPCH = OLDVIBPCH + (GAUSS(ODD,GVAR)-OLDVIBPCH) * C2
VIBYAW = OLDVIBYAW + (GAUSS(ODD,GVAR)-OLDVIBYAW) * C2

C If #acquisitions is at least 2 (and there are at least 4 LEOs and channels),
74 C determine when to adjust variations in roll, pitch and yaw.
75 C---------------------------------------------------------------
76    TEMP = C1*TCUR
77    IF(FINDADJT .AND. NACQS.GE.2) THEN
78       ROLADJT = - ROLPHS
79       DONE = .FALSE.
80       DO WHILE (.NOT.DONE)
81          ROLADJT = ROLADJT + PI
82          IF(ROLADJT.GE.TEMP) DONE = .TRUE.
83       END DO
84       PCHADJT = - PCHPHS
85       DONE = .FALSE.
86       DO WHILE (.NOT.DONE)
87          PCHADJT = PCHADJT + PI
88          IF(PCHADJT.GE.TEMP) DONE = .TRUE.
89       END DO
90       YAWADJT = - YAWPHS
91       DONE = .FALSE.
92       DO WHILE (.NOT.DONE)
93          YAWADJT = YAWADJT + PI
94          IF(YAWADJT.GE.TEMP) DONE = .TRUE.
95       END DO
96       TSMALLSCAN = DMAX1(ROLADJT,PCHADJT,YAWADJT) / C1
97       FINDADJT = .FALSE.
98    END IF
99   END IF
100 C---------------------------------------------------------------
101 C Determine total errors in roll, pitch and yaw. If time is large enough,
102 C adjust attitude variations.
103 C---------------------------------------------------------------
104    IF(TEMP.LT.ROLADJT) THEN
105       ROLERR = ATTRP * DSIN(TEMP+ROLPHS) + VIBROL
106     ELSE
107       ROLERR = ATTRPADJ * DSIN(TEMP+ROLPHS) + VIBROL
108     END IF
109    IF(TEMP.LT.PCHADJT) THEN
110       PCHERR = ATTRP * DSIN(TEMP+PCHPHS) + VIBPCH
111     ELSE
112       PCHERR = ATTRPADJ * DSIN(TEMP+PCHPHS) + VIBPCH
113     END IF
114    IF(TEMP.LT.YAWADJT) THEN
115       YAWERR = ATTY * DSIN(TEMP+YAWPHS) + VIBYAW
116     ELSE
117       YAWERR = ATTYADJ * DSIN(TEMP+YAWPHS) + VIBYAW
118     END IF
119 C---------------------------------------------------------------
120 C Calculate azimuth and elevation of received LEO beam center.
121 C---------------------------------------------------------------
122    CALL XYZAZEL(HEO, LEO, ILEO,'T','R',ECLIPSE,AZS,ELS,AZAHED,ELAHED)
123 C---------------------------------------------------------------
124 C Include effects of attitude errors on azimuth and elevation.
125 C---------------------------------------------------------------
126    AZTEMP = AZS(ILEO) + PCHERR
127    ELTEMP = ELS(ILEO) + ROLERR
128    SINY = DSIN(YAWERR)
129    COSY = DCOS(YAWERR)
130    AZS(ILEO) = AZTEMP*COSY-ELTEMP*SINY
131    ELS(ILEO) = AZTEMP*SINY+ELTEMP*COSY
132
148 OLDVIBROL = VIBROL
149 OLDVIBPCH = VIBPCH
150 OLDVIBYAW = VIBYAW
151
152
153 RETURN
154 END
SUBROUTINE TRUTHGN (HEO,LEO,NLEOS,ILEO,TCUR)

Truth Generator simulates orbital dynamics of the HEO and LEO satellites.

Prototype Version 1.0    Avtec Systems, Inc.    August 1992

Inputs:

HEO = Geocentric-equatorial coordinates of HEO satellite
LEO = Geocentric-equatorial coordinates of LEO satellites
NLEOS = Number of LEO satellites
ILEO = LEO satellite index
TCUR = Current time

Outputs:

HEO = Updated geocentric-equatorial coordinates of HEO satellite
LEO = Updated geocentric-equatorial coordinates of LEO satellites

---------------------------------------------
IMPLICIT REAL*8 (A-H,O-Z)
REAL*8 HEO(6),LEO(6,6),HINTG(6,3),LINTG(6,6,3)
DATA DTINTG /1D1/
DATA INIT /1/

---------------------------------------------

IF(INIT.EQ.1) THEN
  CALL COPYVEC(HEO,6,HINTG(1,1))
  CALL RUK(HEO,DTINTG,HINTG(1,2))
  CALL RUK(HINTG(1,2),DTINTG,HINTG(1,3))
  DO I=I,NLEOS
    CALL COPYVEC(LEO(1,I),6,LINTG(1,1,1))
    CALL RUK(LEO(I,I),DTINTG,LINTG(I,I,2))
    CALL RUK(LINTG(I,I,2),DTINTG,LINTG(I,I,3))
  END DO
  CALL OFIT(HINTG,LINTG,NLEOS,TPROP,DTINTG)
  TPROP = DTINTG
  INIT = 0

---------------------------------------------

ELSE IF(TCUR.GE.TPROP) THEN
  CALL COPYVEC(HINTG(1,2),6,HINTG(1,1))
  CALL COPYVEC(HINTG(1,3),6,HINTG(1,2))
  CALL RUK(HINTG(1,2),DTINTG,HINTG(1,3))
  DO I=I,NLEOS
    CALL COPYVEC(LINTG(1,1,2),6,LINTG(1,1,1))
    CALL COPYVEC(LINTG(1,1,3),6,LINTG(1,1,2))
    CALL RUK(LINTG(1,I,2),DTINTG,LINTG(1,I,3))
  END DO
  CALL QFIT(HINTG,LINTG,NLEOS,TPROP,DTINTG)
  TPROP = TPROP + DTINTG
END IF

---------------------------------------------

Evaluate quadratic functions to obtain satellite coordinates between
74 C integration points.
75 C-----------------------------------------------
76  CALL QVAL(TCUR,ILEO,HEO,LEO)
77  RETURN
78  END
SUBROUTINE UNIT(V)
REAL*8 V(3),VLEN
VLEN=DSQRT(V(1)**2+V(2)**2+V(3)**2)
IF(VLEN.EQ.0) STOP 113
V(1)=V(1)/VLEN
V(2)=V(2)/VLEN
V(3)=V(3)/VLEN
RETURN
END
REAL*8 FUNCTION VLEN(V)

REAL*8 V(3)

VLEN=DSQRT(V(1)**2+V(2)**2+V(3)**2)

RETURN

END
DOUBLE PRECISION FUNCTION XKEP (ECC, XM, TOL)
SOLVES Kepler's Equation 
"XM = XKEP - ECC*SIN(XKEP)"
SOLVES FOR "XKEP" IN RADIANS GIVEN "XM" AND "ECC"
IMPLICIT REAL*8 (A-H,O-Z)
EOLD = XM
DO 10 K = 1,100
SEC = DSIN(EOLD)*ECC
CEC = DCOS(EOLD)*ECC
ENEW = (XM + SEC - EOLD*CEC)/(1.0D0 - CEC)
DE = DABS(ENEW - EOLD)
IF (DE .LE. TOL) GO TO 20
EOLD = ENEW
10 EOLD = ENEW
WRITE(*,100)
FORMAT ('** KEPLER'S EQUATION DID NOT CONVERGE **')
STOP
20 XKEP = ENEW
RETURN
END
SUBROUTINE XYZAZEL (HEO, LEO, ILEO, COORD, BEAM, ECLIPSE, AZ, EL, 
    1 AZAHEAD, ELAHEAD)

This routine transforms LEO X,Y,Z to HEO FOV coordinates AZ, EL.

Prototype Version 1.0 Avtec Systems, Inc. August 1992

Inputs:
1. HEO = Geocentric-equatorial coordinates of HEO satellite
2. LEO = Geocentric-equatorial coordinates of LEO satellites
3. ILEO = LEO satellite index
4. COORD = Coord source: 'T' (Truth Generator) or 'N' (Navigation Processor)
5. BEAM = Beam direction: 'R' (receive) or 'T' (transmit)

Outputs:
1. ECLIPSE(ILEO) = Eclipse flag
2. AZ(ILEO) = Azimuth
3. EL(ILEO) = Elevation
4. AZAHEAD(ILEO) = Azimuth point-ahead angle
5. ELAHEAD(ILEO) = Elevation point-ahead angle

IMPLICIT REAL*8 (A-H,O-Z)
REAL*8 HEO(6), LEO(6,6), AZ(6), EL(6), AZAHEAD(6), ELAHEAD(6)
REAL*8 AZVEC(3), ELVEC(3), DOWN(3), TOLEO(3)
LOGICAL ECLIPSE(6)
CHARACTER*1 COORD, BEAM

#include 'ORBITBLK.FCB'

DATA C /2.9979245805/ ! Speed of light (km/sec)
DATA RE /6.378145D3/ ! Earth radius (km)
DATA INIT /1/

IF(INIT.EQ.1) THEN
    RESQ = RE**2
    RSQ = (RE+DSQRT(3DO)*EPHL)**2
    INIT = 0
END IF

IF(BEAM.EQ. 'R') SIGN = -1
IF(BEAM.EQ. 'T') SIGN = 1

DO J=1,3
    DI N(J) = -HEO(J)
    AZVEC(J) = HEO(J+3)
END DO

CALL UNIT(DOWN)
CALL UNIT(AZVEC)
CALL CROSS(AZVEC,DOWN,ELVEC)
RHEO = VLEN(HEO)
IF(COORD.EQ. 'T') RHORIZ = DSQRT(RHEO**2-RESQ)
IF(COORD.EQ. 'N') RHORIZ = DSQRT(RHEO**2-RSQ)
COSTHRES = RHORIZ/RHEO

DO J=1,3
    TOLEO(J) = LEO(J,ILEO) - HEO(J)
    AZVEC(J) = HEO(J+3) ! Circular orbit
END DO
CALL UNIT(DOWN)
CALL UNIT(AZVEC)
CALL CROSS(AZVEC,DOWN,ELVEC)

RHEO = VLEN(HEO)
IF(COORD.EQ. 'T') RHORIZ = DSQRT(RHEO**2-RESQ)
IF(COORD.EQ. 'N') RHORIZ = DSQRT(RHEO**2-RSQ)
COSTHRES = RHORIZ/RHEO

DO J=1,3
    TOLEO(J) = LEO(J,ILEO) - HEO(J)
END DO
RTOLEO = VLEN(TOLEO)
CALL UNIT(TOLEO)

AZCOMP = DOT(TOLEO,AZVEC)
ELCOMP = DOT(TOLEO,ELVEC)
DNCOMP = DOT(TOLEO,DOWN)
IF(DNCOMP.LE.0) STOP 117
IF(DNCOMP.GE.COSTHRES .AND. RTOLEO.GE.RHORIZ) THEN
ECLIPSE(ILEO) = .TRUE.
ELSE
ECLIPSE(ILEO) = .FALSE.
END IF

AZTENP = DATAN(AZCOHP/DNCONP)
ELTENP = DATAN(ELCOHP/DNCONP)

TPROP = DSQRT((HEO(1)-LEO(1,ILEO))**2 + (HEO(2)-LEO(2,ILEO))**2
+ (HEO(3)-LEO(3,ILEO))**2) / C * SIGN

DO J=1,3
TOLEO(J) = LEO(J,ILEO)+TPROP*(LEO(J+3,ILEO)-HEO(J+3))-HEO(J)
END DO

CALL UNIT(TOLEO)

AZCOMP = DOT(TOLEO,AZVEC)
ELCOMP = DOT(TOLEO,ELVEC)
DNCOMP = DOT(TOLEO,DOWN)
IF(DNCOMP.LE.0) STOP 117

AZ(ILEO) = DATAN(AZCOMP/DNCOMP)
EL(ILEO) = DATAN(ELCOMP/DNCOMP)

AZAHEAD(ILEO) = 2 * (AZ(ILEO) - AZTEMP) * SIGN
ELAHEAD(ILEO) = 2 * (EL(ILEO) - ELTEMP) * SIGN

RETURN
END