This thesis develops a method of supervisory control of the five-degree-of-freedom dexterous right arm of the Beam Assembly Teleoperator (BAT).

The thesis imposes a constraint that the end effector of the right arm must move in a straight line. The joint angles are used to transform the position of the end effector into
cylindrical coordinates. With the initial and final position known, an equation for a straight line is obtained and the cylindrical coordinates of the end effector for each point on the straight line are found. The cylindrical coordinates are then transformed back into joint angles for each point and the joint angles are used to control the BAT'S right arm.

The method of moving the end effector in a straight line is implemented in two ways. The first implementation uses an unknown starting position and moves to a known finish point. Here the previously developed BAT software is used to input the starting position of the arm. Then the straight line path method uses this starting point and drives the arm to the known finish point. In the second implementation, a method is developed in which a set of points is remembered by the BAT and the straight line path method is used to drive the arm from any start point, through all the stored points and return to the original point. Further extensions using this method are discussed.

An appendix contains a block diagram description of the Beam Assembly Teleoperator and the source code for the software developed and implemented.
SUPERVISORY CONTROL OF THE RIGHT ARM OF THE BEAM ASSEMBLY
TELEOPERATOR

CPT. ANTHONY J. MANGANIELLO
HQDA, MILPERCEN (DAPC-OPA-E)
200 Stovall Street
Alexandria, VA  22332

Final Report  10 May 85

Approved for public release; distribution is unlimited.

A thesis submitted to MIT, Cambridge, MA in partial fulfillment of the requirements for the degree of Master of Science in Mechanical Engineering.
SUPERVISORY CONTROL OF THE RIGHT ARM
OF THE BEAM ASSEMBLY TELEOPERATOR

by

ANTHONY J. MANGANIELLO

B.S., United States Military Academy
(1977)

Submitted to the Department of
Mechanical Engineering
in Partial Fulfillment of the
Requirements for the
Degree of

MASTER OF SCIENCE in MECHANICAL ENGINEERING

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1985

© Anthony J. Manganiello 1985

The author hereby grants to M.I.T. permission to reproduce
and to distribute copies of this thesis document in whole
or in part.

Signature of Author

Anthony J. Manganiello
Department of Mechanical Engineering
May 10, 1985

Certified by

David L. Akin
Thesis Supervisor

Accepted by

Ain A. Sonin
Chairman, Mechanical Engineering Department Committee
SUPERVISORY CONTROL OF THE RIGHT ARM
OF THE BEAM ASSEMBLY TELEOPERATOR

by

Anthony J. Manganiello

Submitted to the Department of Mechanical Engineering
on May 10, 1985 in partial fulfillment of the
requirements for the Degree of Master of Science in
Mechanical Engineering

ABSTRACT

This thesis develops a method of supervisory control of the
five-degree-of-freedom dexterous right arm of the Beam Assembly
Teleoperator (BAT).

The thesis imposes a constraint that the end effector of the right arm
must move in a straight line. The joint angles are used to transform the
position of the end effector into cylindrical coordinates. With the
initial and final position known, an equation for a straight line is
obtained and the cylindrical coordinates of the end effector for each point
on the straight line are found. The cylindrical coordinates are then
transformed back into joint angles for each point and the joint angles are
used to control the BAT'S right arm.

The method of moving the end effector in a straight line is
implemented in two ways. The first implementation uses an unknown starting
position and moves to a known finish point. Here the previously developed
BAT software is used to input the starting position of the arm. Then the
straight line path method uses this starting point and drives the arm to
the known finish point. In the second implementation, a method is
developed in which a set of points is remembered by the BAT and the
straight line path method is used to drive the arm from any start point,
through all the stored points and return to the original point. Further
extensions using this method are discussed.

An appendix contains a block diagram description of the Beam Assembly
Teleoperator and the source code for the software developed and
implemented.

Thesis Supervisor: David L. Akin
Title: Rockwell International Assistant Professor of
Aeronautics
ACKNOWLEDGEMENTS

I would like to sincerely thank Professor David Akin for all the assistance he has given me. He was always available to help guide and educate me throughout the entire thesis. It was obvious that educating and helping students is his utmost concern.

Very special thanks to John Spofford who helped teach me everything I know about the BAT and never once made me wait when I asked for help. Good luck in the future with all your work, John.

Many thanks also go to everyone in the Space System Laboratory including those members now gone who have worked on the BAT project from its conception (in particular, Tom Waschura for his work on the software and Eric Shain for his work on the wrist). If it were not for their hard work and education this thesis would have never been conceived.

Thanks to Homayoon Kazerooni for his guidance concerning M.I.T. and the Mechanical Engineering Department. Thanks for taking me under your wing and helping me.

I would also like to thank Sandy Tepper for not only typing my thesis and helping me out from the start of my stay at M.I.T., but also for knowing more about M.I.T. than anyone else.

I would like to thank the U.S. Army for making it possible for me to obtain this degree.
Most of all, I would like to thank my wife, Susan, and my children, Christopher and Lisa, for their patience, understanding and support. I only hope I can repay them for the sacrifices they made for me while I was at M.I.T. Thanks for always being there.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>2</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>3</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>5</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>6</td>
</tr>
<tr>
<td>CHAPTER 1 - INTRODUCTION</td>
<td>11</td>
</tr>
<tr>
<td>CHAPTER 2 - A METHOD FOR SUPERVISORY CONTROL OF THE RIGHT ARM</td>
<td>15</td>
</tr>
<tr>
<td>2.1 Objective</td>
<td>15</td>
</tr>
<tr>
<td>2.2 Overview of the Procedure</td>
<td>17</td>
</tr>
<tr>
<td>2.3 Problem Set-Up</td>
<td>19</td>
</tr>
<tr>
<td>2.4 Joint Angles to Cylindrical Coordinates</td>
<td>23</td>
</tr>
<tr>
<td>2.5 Straight Line Path Generation</td>
<td>29</td>
</tr>
<tr>
<td>2.6 Cylindrical Coordinates to Joint Angles</td>
<td>36</td>
</tr>
<tr>
<td>2.7 Determination of the Total Time and the Values of the Shoulder Yaw (θ₁) at Each Time Step</td>
<td>41</td>
</tr>
<tr>
<td>2.8 Determination of the Orientation Angles</td>
<td>51</td>
</tr>
<tr>
<td>2.9 Simulation</td>
<td>52</td>
</tr>
<tr>
<td>CHAPTER 3 - IMPLEMENTATION I: UNKNOWN STARTING POSITION, KNOWN FINISH POINT</td>
<td>71</td>
</tr>
<tr>
<td>CHAPTER 4 - IMPLEMENTATION II: LEARNING TRAJECTORIES BY CONCATENATION</td>
<td>89</td>
</tr>
<tr>
<td>CHAPTER 5 - CONCLUSIONS</td>
<td>103</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>107</td>
</tr>
<tr>
<td>APPENDIX A - DESCRIPTION OF THE BAT</td>
<td>108</td>
</tr>
<tr>
<td>APPENDIX B - PROGRAM FOR SUPERVISORY CONTROL</td>
<td>147</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>NUMBER</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Sketch of the ICS and BAT</td>
<td>12</td>
</tr>
<tr>
<td>2.1</td>
<td>Diagram of the Right Arm Control System</td>
<td>16</td>
</tr>
<tr>
<td>2.2</td>
<td>Joint Angles</td>
<td>19</td>
</tr>
<tr>
<td>2.3</td>
<td>Point A on the Right Arm</td>
<td>20</td>
</tr>
<tr>
<td>2.4</td>
<td>Coordinate Frames for the Right Arm</td>
<td>21</td>
</tr>
<tr>
<td>2.5</td>
<td>Coordinate Frames for the Wrist</td>
<td>21</td>
</tr>
<tr>
<td>2.6</td>
<td>Cylindrical Coordinate Reference Frame</td>
<td>23</td>
</tr>
<tr>
<td>2.7</td>
<td>Joint Angles to Cylindrical Coordinates - Position i</td>
<td>24</td>
</tr>
<tr>
<td>2.8</td>
<td>Joint Angles to Cylindrical Coordinates - Position ii</td>
<td>26</td>
</tr>
<tr>
<td>2.9</td>
<td>Joint Angles to Cylindrical Coordinates - Position iii</td>
<td>28</td>
</tr>
<tr>
<td>2.10</td>
<td>Straight Line Path in Reference Frame ((X_0,Y_0,Z_0))</td>
<td>30</td>
</tr>
<tr>
<td>2.11</td>
<td>Projected Line in (X_0Y_0) Plane</td>
<td>30</td>
</tr>
<tr>
<td>2.12</td>
<td>Travelling Across the Projected Line</td>
<td>31</td>
</tr>
<tr>
<td>2.13</td>
<td>Slope of the Projected Line Equal to 1</td>
<td>35</td>
</tr>
<tr>
<td>2.14</td>
<td>Cylindrical Coordinates to Joint Angles - Case i</td>
<td>37</td>
</tr>
<tr>
<td>2.15</td>
<td>Cylindrical Coordinates to Joint Angles - Case ii</td>
<td>39</td>
</tr>
<tr>
<td>2.16</td>
<td>Cylindrical Coordinates to Joint Angles - Case iii</td>
<td>41</td>
</tr>
<tr>
<td>2.17a</td>
<td>Driving Function (Initial Less than Final)</td>
<td>42</td>
</tr>
<tr>
<td>2.17b</td>
<td>Driving Function (Initial Greater than Final)</td>
<td>43</td>
</tr>
<tr>
<td>NUMBER</td>
<td>TITLE</td>
<td>PAGE</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>2.18</td>
<td>Simplified Model of Arm and Beam</td>
<td>46</td>
</tr>
<tr>
<td>2.19</td>
<td>Sampled Driving Function</td>
<td>49</td>
</tr>
<tr>
<td>2.20</td>
<td>Arm Position - Simulation I</td>
<td>55</td>
</tr>
<tr>
<td>2.21</td>
<td>Shoulder Yaw Position - Simulation I</td>
<td>56</td>
</tr>
<tr>
<td>2.22</td>
<td>Shoulder Yaw Velocity - Simulation I</td>
<td>56</td>
</tr>
<tr>
<td>2.23</td>
<td>Shoulder Yaw Acceleration - Simulation I</td>
<td>57</td>
</tr>
<tr>
<td>2.24</td>
<td>Wrist Yaw Position - Simulation I</td>
<td>57</td>
</tr>
<tr>
<td>2.25</td>
<td>Wrist Roll Position - Simulation I</td>
<td>58</td>
</tr>
<tr>
<td>2.26</td>
<td>Y vs. X Position of Point A - Simulation I</td>
<td>58</td>
</tr>
<tr>
<td>2.27</td>
<td>Z vs. X Position of Point A - Simulation I</td>
<td>59</td>
</tr>
<tr>
<td>2.28</td>
<td>Z vs. Y Position of Point A - Simulation I</td>
<td>59</td>
</tr>
<tr>
<td>2.29</td>
<td>Arm Position - Simulation II</td>
<td>60</td>
</tr>
<tr>
<td>2.30</td>
<td>Shoulder Yaw Position - Simulation II</td>
<td>61</td>
</tr>
<tr>
<td>2.31</td>
<td>Shoulder Yaw Velocity - Simulation II</td>
<td>61</td>
</tr>
<tr>
<td>2.32</td>
<td>Shoulder Yaw Acceleration - Simulation II</td>
<td>62</td>
</tr>
<tr>
<td>2.33</td>
<td>Wrist Yaw Position - Simulation II</td>
<td>62</td>
</tr>
<tr>
<td>2.34</td>
<td>Wrist Roll Position - Simulation II</td>
<td>63</td>
</tr>
<tr>
<td>2.35</td>
<td>Y vs. X Position of Point A - Simulation II</td>
<td>63</td>
</tr>
<tr>
<td>2.36</td>
<td>Z vs. X Position of Point A - Simulation II</td>
<td>64</td>
</tr>
<tr>
<td>2.37</td>
<td>Z vs. Y Position of Point A - Simulation II</td>
<td>64</td>
</tr>
<tr>
<td>2.38</td>
<td>Arm Position - Simulation III</td>
<td>65</td>
</tr>
<tr>
<td>2.39</td>
<td>Shoulder Yaw Position - Simulation III</td>
<td>66</td>
</tr>
<tr>
<td>NUMBER</td>
<td>TITLE</td>
<td>PAGE</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>2.40</td>
<td>Arm Radius Velocity - Simulation III</td>
<td>66</td>
</tr>
<tr>
<td>2.41</td>
<td>Arm Radius Acceleration - Simulation III</td>
<td>67</td>
</tr>
<tr>
<td>2.42</td>
<td>Wrist Yaw Position - Simulation III</td>
<td>67</td>
</tr>
<tr>
<td>2.43</td>
<td>Wrist Roll Position - Simulation III</td>
<td>68</td>
</tr>
<tr>
<td>2.44</td>
<td>Y vs. X Position of Point A - Simulation III</td>
<td>68</td>
</tr>
<tr>
<td>2.45</td>
<td>Z vs. X Position of Point A - Simulation III</td>
<td>69</td>
</tr>
<tr>
<td>2.46</td>
<td>Z vs. Y Position of Point A - Simulation III</td>
<td>69</td>
</tr>
<tr>
<td>3.1</td>
<td>Finish Point, Implementation I</td>
<td>72</td>
</tr>
<tr>
<td>3.2</td>
<td>Arm Position, Step 1, Implementation I</td>
<td>81</td>
</tr>
<tr>
<td>3.3</td>
<td>Shoulder Yaw Position, Step 1, Implementation I</td>
<td>82</td>
</tr>
<tr>
<td>3.4</td>
<td>Wrist Yaw Position, Step 1, Implementation I</td>
<td>82</td>
</tr>
<tr>
<td>3.5</td>
<td>Wrist Roll Position, Step 1, Implementation I</td>
<td>83</td>
</tr>
<tr>
<td>3.6</td>
<td>Y vs. X Position, Step 1, Implementation I</td>
<td>83</td>
</tr>
<tr>
<td>3.7</td>
<td>Z vs. X Position, Step 1, Implementation I</td>
<td>84</td>
</tr>
<tr>
<td>3.8</td>
<td>Z vs. Y Position, Step 1, Implementation I</td>
<td>84</td>
</tr>
<tr>
<td>3.9</td>
<td>Shoulder Yaw Position, Step 2, Implementation I</td>
<td>85</td>
</tr>
<tr>
<td>3.10</td>
<td>Wrist Yaw Position, Step 2, Implementation I</td>
<td>86</td>
</tr>
<tr>
<td>3.11</td>
<td>Wrist Roll Position, Step 2, Implementation I</td>
<td>86</td>
</tr>
<tr>
<td>3.12</td>
<td>Y vs. X Position, Step 2, Implementation I</td>
<td>87</td>
</tr>
<tr>
<td>3.13</td>
<td>Z vs. X Position, Step 2, Implementation I</td>
<td>87</td>
</tr>
<tr>
<td>3.14</td>
<td>Z vs. Y Position, Step 2, Implementation I</td>
<td>88</td>
</tr>
<tr>
<td>4.1</td>
<td>Implementation II</td>
<td>91</td>
</tr>
<tr>
<td>4.2</td>
<td>Arm Position, Step 1, Implementation II</td>
<td>95</td>
</tr>
<tr>
<td>NUMBER</td>
<td>TITLE</td>
<td>PAGE</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>4.3</td>
<td>Shoulder Yaw Position, Step 1, Implementation II</td>
<td>96</td>
</tr>
<tr>
<td>4.4</td>
<td>Wrist Yaw Position, Step 1, Implementation II</td>
<td>96</td>
</tr>
<tr>
<td>4.5</td>
<td>Wrist Roll Position, Step 1, Implementation II</td>
<td>97</td>
</tr>
<tr>
<td>4.6</td>
<td>Y vs. X Position, Step 1, Implementation II</td>
<td>97</td>
</tr>
<tr>
<td>4.7</td>
<td>Z vs. X Position, Step 1, Implementation II</td>
<td>98</td>
</tr>
<tr>
<td>4.8</td>
<td>Z vs. Y Position, Step 1, Implementation II</td>
<td>98</td>
</tr>
<tr>
<td>4.9</td>
<td>Shoulder Yaw Position, Step 2, Implementation II</td>
<td>99</td>
</tr>
<tr>
<td>4.10</td>
<td>Wrist Yaw Position, Step 2, Implementation II</td>
<td>100</td>
</tr>
<tr>
<td>4.11</td>
<td>Wrist Roll Position, Step 2, Implementation II</td>
<td>100</td>
</tr>
<tr>
<td>4.12</td>
<td>Y vs. X Position, Step 2, Implementation II</td>
<td>101</td>
</tr>
<tr>
<td>4.13</td>
<td>Z vs. X Position, Step 2, Implementation II</td>
<td>101</td>
</tr>
<tr>
<td>4.14</td>
<td>Z vs. Y Position, Step 2, Implementation II</td>
<td>102</td>
</tr>
<tr>
<td>A.1</td>
<td>Sketch of BAT</td>
<td>108</td>
</tr>
<tr>
<td>A.2</td>
<td>Main Power Subsystem</td>
<td>110</td>
</tr>
<tr>
<td>A.3</td>
<td>Control Subsystem</td>
<td>112</td>
</tr>
<tr>
<td>A.4</td>
<td>Pneumatic Subsystem</td>
<td>113</td>
</tr>
<tr>
<td>A.5</td>
<td>Video Subsystem</td>
<td>115</td>
</tr>
<tr>
<td>A.6</td>
<td>Left Arm Subsystem</td>
<td>117</td>
</tr>
<tr>
<td>A.7</td>
<td>Right Arm Subsystem</td>
<td>118</td>
</tr>
<tr>
<td>A.8</td>
<td>Propulsion Subsystem</td>
<td>120</td>
</tr>
<tr>
<td>A.9</td>
<td>Sketch of ICS</td>
<td>121</td>
</tr>
<tr>
<td>A.10</td>
<td>ICS Control Panels</td>
<td>123</td>
</tr>
<tr>
<td>A.11</td>
<td>Central Controller</td>
<td>126</td>
</tr>
</tbody>
</table>
As stated previously, the concern in this chapter is to develop a method of generating a straight line path for point A. In this chapter the values of $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$ (shoulder yaw, shoulder pitch, elbow, wrist motor A, wrist motor B) are given as part of the problem. $\psi_4$ and $\psi_5$ will be tucked away until Section 2.8 (since the motor angles do not affect the position of point A).

At this time, the starting position of point A is represented in terms of $\varphi_1, \varphi_2, \varphi_3$ (shoulder yaw, shoulder pitch, elbow) and the length of the two links ($L_1$ and $L_2$). It would be easier to compute a straight line path if the joint coordinates $(\varphi_1, \varphi_2, \varphi_3, L_1, L_2)$ were changed to another coordinate system.

2.4 Joint Angles to Cylindrical Coordinates

Cylindrical coordinates $\varphi, R, Z$ (Figure 2.6) were chosen because of both the design of the manipulator and the ease of transformation. Shoulder

![Diagram](image)
formation is wrist roll, $\theta_5$. $\theta_5$ rotates about the $Y''$ axis from $\theta_5\text{MIN} = -1.745$ radians to $\theta_5\text{MAX} = 1.745$ radians. There is one other consideration concerning the wrist angles. When the wrist was developed by the Space Systems Laboratory, a differential mechanism and two motors (motor A and motor B) were used inside the wrist to enable it to have yaw and roll. A detailed description of the mechanism is described in reference [3]. The angles which are used in the control system are the angles from the wrist motors; these motors are labelled A and B. In this thesis $\psi_4$ will be used as the angle for motor A and $\psi_5$ will be used as the angle for motor B. Reference [3] develops the following transformation used to convert between $\psi_4$, $\psi_5$ and $\theta_4$, $\theta_5$ (the two motor angles and the two wrist angles).

$$\theta_4 = -\frac{1}{2} \psi_4 + \frac{1}{2} \psi_5$$

$$\theta_5 = -\frac{1}{2a} \psi_4 - \frac{1}{2a} \psi_5$$

where $a = \frac{27}{29}$, and is based on the gear ratios of the differential drive.

Also,

$$\psi_4 = -\psi_4 - a \theta_5$$

$$\psi_5 = \psi_4 - a \theta_5$$
FIGURE 2.4 Coordinate Frames for the Right Arm

FIGURE 2.5 Coordinate Frames for the Wrist
arrow in each frame in Figure 2.4 signifies the 0 radian point.) The second rotation transformation is about the $X_1$ axis. The shoulder pitch or $\theta_2$ can rotate from $\theta_2 = 1.31$ radians to $2.62$ radians. The coordinate frame is then translated along the $Y_2$ axis by the length of the first link ($L_1 = 0.41$ meters). The elbow rotation, $\theta_3$, is the next transformation. It rotates about $X_3$ from $\theta_3 = 0.52$ radians to $\theta_3^{\text{MAX}} = 3.75$ radians. The last transformation is a translation along the $Y_4$ axis the length of the second link ($L_2 = 0.36$ meters). This is point A.

The generation of a straight line path for point A is the main concern. However, the orientation angles (wrist yaw and wrist roll, $\theta_4$ and $\theta_5$) can also be changed while the arm is moving. The reference frame of the wrist is fixed to point A and shown in Figure 2.5. The first transformation is wrist yaw, $\theta_4$, which rotates about the $Z'$ axis and varies from $\theta_4^{\text{MIN}} = 0$ radians to $\theta_4^{\text{MAX}} = 3.14$ radians. The next trans-
2.3 Problem Set-Up

The position of the end effector is the main concern. This is the point that must follow a straight line path (Point, A, Figure 2.3). This gives the arm three degrees of freedom (shoulder yaw, shoulder pitch and elbow). The restriction of point A to follow a straight line path from the start point to the finish point is a two-link problem. The reference frame \((X_0, Y_0, Z_0)\) is positioned at the base of the first link (Point 0 in Figure 2.3). The first rotation transformation is about the \(Z_0\) axis (Figure 2.4). This is shoulder yaw or \(\theta_1\). It can rotate from \(\theta_{1_{\text{MIN}}} = 0.0\) radians to \(\theta_{1_{\text{MAX}}} = 2.44\) radians. (Note: the base of the
From this the central controller computes the initial position of the end effector. The procedure for converting the voltage inputs to joint angles will be covered in the next chapter. The final position is predetermined in advance and stored in the central controller. The problem is to find a trajectory for the end effector of the arm, compute the joint angles, and send them to the control system. The procedure for sending the joint angles to the control system will also be covered in the next chapter. Additionally, the position of the end effector is restricted to following a straight line path. This restriction is added to make it easier to concatenate paths (path concatenation is developed in Chapter 4).

The problem is now in the following format.

**ASSUMPTIONS:**

The reference frame is fixed to the BAT and stationary.

**GIVEN:**

1) Joint angles for the original arm configuration.
   (Shoulder Yaw ($\phi_1$), Shoulder Pitch ($\phi_2$), Elbow ($\phi_3$),
   Wrist Yaw ($\phi_4$) and Wrist Roll ($\phi_5$), Figure 2.2).
2) The final position of the arm.
3) The physical dimensions of the arm.

**FIND:**

1) A straight line path for the position of the end effector from the start point to the finish point.
2) The joint angles for each position on the path.
determine what configuration the arm is in, and start generating inputs to the control system for a specific task. These inputs would mimic the inputs that would have come from the master arm if the master were controlling the task. An important task in an assembly operation is for the arm to move from a point \((x,y,z)\) (in a three-dimensional space with a reference frame stationary and fixed to the BAT) to another point \((x,y,z)\). This thesis develops a method for the central controller to control the right arm moving from any point to a specified point. Using this method, it develops a way to concatenate specified points and move the arm through these points. This will enable the BAT to learn many trajectories which can be combined to perform assembly tasks. This will be the basis for the extensions to other types of tasks.

2.2 Overview of the Procedure

In this chapter, the task is to develop a method in which the arm moves to a specified location; for example, the operator has just grasped a beam with the right arm, and would like to move to a location next to the left arm. The central controller must:

1) know the starting configuration of the arm;
2) know the final configuration of the arm;
3) compute a trajectory;
4) transmit the appropriate command to the joint/actuator control system.

The operator starts this procedure by entering a keyboard option on the control panel at the ICS. This would let the central controller know that the supervisory control option has been selected. The original configuration is then read from the inputs of each joint of the master arm.
CHAPTER 2
SUPERVISORY CONTROL OF THE RIGHT ARM

2.1 Objective

Using the two joysticks to control the BAT's position and the master arm to control the BAT's right arm, it is easy to imagine times where controlling the operation of the BAT can become a very hard task with only two hands. The operator might also need to enter an option in the BAT's software. The control system of the right arm can be improved, so that the master arm is not needed to perform most tasks. There are some cases in an assembly operation where the master-slave relationship would be easier to use. (For example, if two beams do not fit just right and have to be jiggled to make the connection.) However, in most cases it would be much easier to simply enter a command and let the central controller move the right arm to a preprogrammed position.

Figure 2.1 depicts the entire control system of the BAT's right arm. The position of the right arm is fed back only to the joint/actuator cards and not to the central controller. Therefore, the central controller must receive the positions of the right arm from the master arm at the ICS. The assumption is made that the master arm's joint angles are equal to the right arm's joint angles, or at least the relationship is a known linear one (recall the master arm is kinematically similar to the right arm). In the supervisory mode, the commands from the master arm are halted at the central controller. The central controller can then
with a specified start and finish point, the BAT system will then be taught to not only remember the trajectory, but also to play it back at the operator's command. This will give the BAT system the capability of performing any task over and over again without the use of the master arm. There are many tasks that are repetitive during an assembly operation. It would not be an efficient use of the BAT system if the operator were to perform these tasks again and again using the master arm at the ICS. By upgrading the control system, the BAT's capabilities will be greatly enhanced. Freeing the operator's hands will further increase the man-machine productivity.
degrees of freedom. It is also equipped with two cameras providing the operator with two different perspectives of the work area.

The BAT is controlled by the ICS. The ICS has two three degree-of-freedom joysticks to control the BAT's six maneuvering degrees of freedom. There is a master arm which is kinematically similar to the BAT's dexterous right arm. The operator uses this master arm to control the BAT's right arm in a master-slave control relationship. On the ICS there is also a helmet gimbal system used to control the tilt and pan unit of the BAT's main camera. The BAT and the ICS are described in further detail in Appendix A.

To perform an assembly task with the system, the operator uses the two joysticks to position the BAT at the work area. The operator then straps his/her right arm to the ICS's master arm and positions it to where he/she would like the BAT's right arm to be. The left arm is used to hold the workpiece in place. This can be quite challenging especially while the operator is trying to control the BAT's position with the joystick controls. A method of freeing the operator from the master-slave control relationship would help the man-machine to be more productive.

In this thesis, the operator is released from certain tasks by implementing supervisory control of the right arm. Two different trajectories will be used to demonstrate the control. The first trajectory will have an unknown starting point and a specified finishing point. By selecting a keyboard option, the operator will be able to move the BAT's right arm to this specified finish point. By defining a second trajectory
a more efficient and effective system. This thesis will continue along this trend to upgrade the interface between man and machine.

The teleoperator system is made up of two major structures:

1) the Beam Assembly Teleoperator (BAT)
2) the Integrated Control Station (ICS)

The BAT has a fixed left arm and a dexterous right arm. The left arm has a gripper end effector used for grasping a beam, and a motor-driven roller used for moving the beam laterally. The right arm is a five degree-of-freedom manipulator with a two-jaw grip end effector. The BAT frame has eight trolling motors which are used to give the unit six
CHAPTER 1
INTRODUCTION

One of the most important parameters affecting the industrialization of space is the capability to assemble structures in orbit. Much larger structures can be built in space than on earth. Since there is a limit to payload capacity, the need to assemble structures in space is imperative. Experimentation with assembly tasks on earth will help in the development of the most efficient assembly method, which will increase productivity on an actual mission. Tests have already been conducted, in a simulated environment, evaluating the ability of humans to assemble objects in space. The next step is to see how a machine can help man in accomplishing these tasks. The Space Systems Laboratory at M.I.T. has already conducted tests involving assembly by humans in a simulated environment. Neutral buoyancy was used as the simulation medium. A teleoperated system was also developed which will simulate the man-machine interface in space. This device is called the Beam Assembly Teleoperator (BAT). The BAT operates in the neutral buoyancy environment.

The Space Systems Laboratory began development of the BAT in mid 1981, and has been using this device in experiments with man-machine assembly tasks. The software for master-slave control of the BAT manipulator was previously developed in the Space Systems Lab. Both the hardware and the software are being continuously improved to provide
<table>
<thead>
<tr>
<th>NUMBER</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.12</td>
<td>Joint/Actuator Control System</td>
<td>127</td>
</tr>
<tr>
<td>A.13</td>
<td>Propulsion Control System</td>
<td>129</td>
</tr>
<tr>
<td>A.14</td>
<td>Communication Link</td>
<td>130</td>
</tr>
<tr>
<td>A.15</td>
<td>Main() Function</td>
<td>132</td>
</tr>
<tr>
<td>A.16</td>
<td>Initializebat() Function</td>
<td>134</td>
</tr>
<tr>
<td>A.17</td>
<td>Calibratebat() Function</td>
<td>135</td>
</tr>
<tr>
<td>A.18</td>
<td>Screen() Function</td>
<td>137a</td>
</tr>
<tr>
<td>A.19a</td>
<td>Runbat() Function</td>
<td>137b</td>
</tr>
<tr>
<td>A.19b</td>
<td>Runbat() Function</td>
<td>138</td>
</tr>
<tr>
<td>A.20</td>
<td>Shutdownbat Function</td>
<td>146</td>
</tr>
</tbody>
</table>
yaw \( (\theta_1) \) transforms identically. The transformation of shoulder pitch and elbow \( (\theta_2 \text{ and } \theta_3) \) to \( R,Z \) is just a planar problem. The plane is defined by the location of \( \theta \). For a particular value of \( \theta \) the arm can only be in these positions.

i) Elbow angle \( (\theta_3) < \pi \) (Elbow down position)

ii) Elbow angle \( (\theta_3) > \pi \) (Elbow up position)

iii) Elbow angle \( (\theta_3) = \pi \) (Fully extended)

Position i) Elbow angle \( (\theta_3) < \pi \) (Figure 2.7)

Note: There can never be a negative \( R \) so the arm is only in Quadrant I and IV of the \( (R,Z) \) plane.

FIGURE 2.7 Joint Angle to Cylindrical Coordinates - Position i)
To find $Z, R$ for either of the two diagrams in Figure 2.7 the following procedure is used:

1) find $\gamma$ (the distance from the origin to point A) using the law of cosines,

$$\gamma = \sqrt{L_1^2 + L_2^2 - 2L_1L_2\cos\theta_3}$$

2) find $\phi$ (the angle of the triangle formed by $L_1$ and $\gamma$ at the origin) using the law of sines,

$$\phi = \sin^{-1}\left(\frac{L_2\sin\theta_3}{\gamma}\right)$$

(NOTE: $\phi$ can only be between 0 and $\pi/2$ radians, therefore, there is no ambiguity in the arcsin.)

3) find $\psi$ (the angle between $\gamma$ and the $R$ axis)

$$\psi = \frac{\pi}{2} - (\theta_2 - \phi)$$

4) find $Z, R$ using polar coordinate transformations

$$Z = \gamma \sin \psi \quad \quad R = \sqrt{\gamma^2 - Z^2}$$
Position ii) Elbow angle $(\theta_3) > \pi$ (Figure 2.8).

![Diagram showing joint angles and cylindrical coordinates](image)

**FIGURE 2.8** Joint Angle to Cylindrical Coordinates - Position ii

This configuration of the master arm is not possible given the current design of both the master arm of ICS and the dexterous arm of BAT. It is included in this thesis to ensure that this method can handle all types of inputs to the control system. (The algorithm for finding $R$ and $Z$ is similar to Position i).
1) Find \( \gamma \) (the distance from the origin to point A) using the law of cosines

\[
\gamma = \sqrt{L_1^2 + L_2^2 - 2 \times L_1 \times L_2 \times \cos(2\pi - \theta_3)}
\]

(NOTE: The angle of the triangle is now \( 2\pi - \theta_3 \) instead of \( \theta_3 \).)

2) Find \( \phi \) (the angle of the triangle formed by \( \gamma \), \( L_2 \) at the origin) using the law of sines

\[
\phi = \sin^{-1} \left( \frac{L_2 \sin(2\pi - \theta_3)}{\gamma} \right)
\]

(NOTE: \( \phi \) can only be between 0 and \( \pi/2 \).)

3) Find \( \psi \) (the angle between \( \gamma \) and the R axis)

\[
\psi = \frac{\pi}{2} - (\theta_2 - \phi)
\]

4) Find \( Z, R \)

\[
Z = \gamma \sin \psi \quad R = \sqrt{\gamma^2 - Z^2}
\]
Position iii) Elbow angle \( (\theta_3) = \gamma \) (Figure 2.9)

1) find \( \gamma \) (the distance from the origin to point A)
\[
\gamma = L_1 + L_2
\]

2) find \( \varphi \) (the angle between \( \gamma \) and the R axis)
\[
\varphi = \frac{\pi}{2} - \theta_2
\]

3) find \( Z, R \)
\[
Z = \gamma \sin \varphi \quad \quad R = \gamma \cos \varphi
\]
Upon receiving the joint angles (shoulder yaw, $\theta_1$, shoulder pitch, $\theta_2$, and elbow, $\theta_3$) the controller checks $\theta_3$ and finds $\theta, R, Z$ of the initial position of point A from one of these three algorithms.

In this chapter the final position of point A is known and has been stored in the controller. The final position is stored in cylindrical coordinates. The final orientation angles $\theta_4, \theta_5$ (wrist yaw and wrist roll) are also stored and are in radians. The controller knows it must move point A from $\theta, R, Z$ (initial) to $\theta, R, Z$ (final) in a straight line.

2.5 Straight Line Path Generation

If a path is a straight line in $X_0, Y_0, Z_0$ space, it can be projected down to a straight line in the $X, Y$ plane (Fig. 2.10). Due to the maximum and minimum values of shoulder yaw ($\theta_1$) only the first and second quadrants of the $(X_0, Y_0)$ plane are involved. Having $\theta, R$ initial and $\theta, R$ final, measured as shown in Figure 2.11, the values for $X, Y$ initial and $X, Y$ final are easily computed.

<table>
<thead>
<tr>
<th>Initial</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y_i = R_i \times \sin \theta_i$</td>
<td>$Y_f = R_f \times \sin \theta_f$</td>
</tr>
<tr>
<td>$X_i = R_i \times \cos \theta_i$</td>
<td>$X_f = R_f \times \sin \theta_f$</td>
</tr>
</tbody>
</table>

From here the equation for a straight line is then computed in terms of $X$ and $Y$. 
FIGURE 2.10 Straight Line Path in Reference Frame \((X_0, Y_0, Z_0)\)

FIGURE 2.11 Projected Line in \(X_0Y_0\) Plane
\[ Y = m(X - x_i) + y_i \]  \hspace{1cm} (2.1)

where

\[ m = \frac{y_f - y_i}{x_f - x_i} \]

[The case when \( \Delta X = 0 \) sets the flag.] The value for \( \theta \) changes every time step. The method for changing \( \theta \) (also shoulder yaw, \( \theta_i \)) will be discussed in Section 2.7. Assume for now that the value of \( \theta \) at each time step is given. Then \( R \) and \( Z \) are computed using the \( X \) and \( Y \) computed from the straight line restriction (2.1). (See Figure 2.12.)

![Figure 2.12 Travelling Across the Projected Line](image-url)
To review what has been done until now, first the joint angles \( (\alpha_1, \alpha_2, \alpha_3) \) were obtained from the arm. The joint angles were then converted to cylindrical coordinates of point A. With the initial and final cylindrical coordinates of point A, a straight line path in the \( X_0'Y_0'Z_0' \) plane was found for point A. As \( \alpha \) (which is also the shoulder yaw angle, \( \alpha_1 \)) sweeps across the straight line, the value of \( R \) and \( Z \) must be found for each \( \theta \). This will give the cylindrical coordinates of point A on the straight line. Knowing the value of \( \theta \), \( R \) and \( Z \) of point A, this will be used to convert back to joint angles (this part will be covered in Section 2.6). To find the values of \( R \) and \( Z \) for each \( \theta \), four special cases may be examined.

Case 1) \( \Delta \theta = 0, \Delta Z = 0 \)

Here, \( \theta \) and \( Z \) retain the value of \( \theta_i \) and \( Z_i \). Depending on whether \( R_i \) is less than or greater than \( R_f \), the appropriate driving function is used to determine \( R \) at each time step (this driving function is the same one used to determine \( \theta \) when \( \Delta \theta \neq 0 \) and will be covered in Section 2.7).

Case 2) \( \Delta \theta, \Delta R = 0 \)

This is done exactly as above except that \( R \) is set equal to \( R_i \) and \( Z \) is used in the driving function.

Case 3) \( \Delta \theta = 0 \)

For this condition, \( R \) is used in the driving function. \( X \) and \( Y \) are found using simple conversions.
\[ X = R \times \cos \theta \]
\[ Y = R \times \sin \theta \]

Dependent upon the value of \( \Delta X \) and \( \Delta Y \) the appropriate side of the equation for a straight line (shown below) is used to compute \( Z \).

\[
\frac{X - X_i}{X_f - X_i} = \frac{Z - Z_i}{Z_f - Z_i} = \frac{Y - Y_i}{Y_f - Y_i}
\]

Case 4) If \( \Delta R, \Delta Z = 0 \)

or \( \Delta R = 0 \)

or \( \Delta Z = 0 \)

or \( \Delta \theta, \Delta R, \Delta Z \neq 0 \)

the following procedure is used.

The value of each \( \theta \), given by the driving function, is used to compute the next \( X \) and \( Y \) along the straight line. From these two values, \( R \) is computed as shown below.

General case:

using

\[ X = \frac{Y}{\tan \theta} \]
substitute this into the equation for a straight line (2.1)

\[ Y = m\left(\frac{Y}{\tan \phi} - X_i\right) + Y_i. \]

Then, solving for \( Y \)

\[ Y = \frac{(Y_i - mX_i)}{(1 - \frac{m}{\tan \phi})}. \]

Now with \( Y \) and \( \phi \), find \( X \)

\[ X = \frac{Y}{\tan \phi}. \]

\( R \) is then found by

\[ R = \sqrt{X^2 + Y^2}. \]

Flagged cases:

- is checked as it changes, so that when the next \( \phi \) reaches a value of \( \phi_{\text{low}} \), (Figure 2.12, in this case \( \phi_{\text{low}} = 1.53 \) radians)
- retains this value until it passes \( \phi = -/2 \). At \( \phi = -/2 \), \( \phi \) is then set to \( \phi_{\text{high}} \) which is 1.60 radians. This is to account for the singularity at \( \phi = -/2 \). There are four more cases which are flagged.
Case 1) If \( x = 0 \)  
(Figure 2.13)

Then,
\[
Y = X_i \tan \theta
\]
\[
X = X_i
\]

Case 2) If \( m = 0 \)

Then
\[
X = \frac{Y_i}{\tan \theta}
\]
\[
Y = Y_i
\]

Case 3) If \( \theta = 0 \)

Then
\[
Y = 0
\]
\[
X = X_i \frac{Y_i}{m} \quad \text{(except when } m = 0)\]
Case 4) If $\phi = \frac{\pi}{2}$ (possibly as a starting point)
Then
\[
X = 0
\]
\[
Y = Y_i - mX_i
\]

Once $X$ and $Y$ are known, using the general or flagged cases, $R$ is then found using
\[
R = \sqrt{X^2 + Y^2}
\]

To find $Z$, the equation for a straight line in three-dimensional space is
\[
\frac{X - X_i}{X_f - X_i} = \frac{Z - Z_i}{Z_f - Z_i} = \frac{Y - Y_i}{Y_f - Y_i}
\]

The values of $Z_i$ and $Z_f$ are known and depending on whether $\phi X$ or $\phi Y$ is zero the right or left side of the equation is used.

\[
Z = (\frac{X - X_i}{X_f - X_i})(Z_f - Z_i) + Z_i
\]
\[
Z = (\frac{Y - Y_i}{Y_f - Y_i})(Z_f - Z_i) + Z_i
\]

2.6 Cylindrical Coordinates to Joint Angles

At this point, the values of $\phi, R, Z$ which keep point A on a straight line path are known. Therefore for a particular value of $\phi$,
finding $\varphi_2$ (shoulder pitch) and $\varphi_3$ (elbow) is a planar problem. 
Recall that $\varphi$ in cylindrical coordinates is equal to $\theta_1$ (shoulder yaw). The algorithm to find the joint angles is simply the reverse of going from $(\theta_1, \theta_2, \theta_3)$ joint angles to $(\varphi, R, Z)$ cylindrical coordinates. Only three positions can occur here also.

Position i) Elbow angle $(\varphi_3) < \pi$ (Figure 2.14)

![Diagram](attachment:image.png)

**Figure 2.14** Cylindrical Coordinates to Joint Angles - Case i

1) find 

$$\varphi_2 = \sqrt{R^2 + Z^2}$$
If the case arises where $\Delta \phi, \Delta R = 0$ (case 2, Section 2.5), $T$ is again computed using the maximum $\Delta \phi = \pi$. The driving function will now be $Z(t)$.

$$Z_f = Z_i$$

$$Z(t) = Z_i + (Z(1 - \frac{t}{T} + \frac{1}{2\pi} \sin(2\pi \frac{t}{T})))$$

$$Z_f = Z_i$$

$$Z(t) = Z_f - (Z(1 - \frac{t}{T} + \frac{1}{2\pi} \sin(2\pi \frac{t}{T})))$$

2.8 Determining the Orientation Angles

The values of the orientation angles $\phi_4, \phi_5$ (wrist yaw and wrist roll) do not affect the straight line path of point A. Therefore, wrist yaw, $\phi_4$, and wrist roll, $\phi_5$, can be changed slowly as the arm moves point A along its straight line path. Recall from the end of Section 2.3, $\phi_{4i}$ and $\phi_{5i}$ (initial) are obtained by transforming the values of $\phi_4$ (wrist motor A) and $\phi_5$ (wrist motor B). The values of the final orientation angles, $\phi_{4f}$ and $\phi_{5f}$, are stored along with the final position of point A. In order to have all of the joints of the arm finish at the same time, the time it takes to go from $\phi_{4i}, \phi_{5i}$ (initial) to $\phi_{4f}, \phi_{5f}$ (final) will be $T$ (the total time to traverse the straight line path, Section 2.7). To account for gradual acceleration and braking of the two
increment, will be the sampling time of the arm's control system 
(\Delta t \approx 0.05 \text{ seconds}). It is obvious, by looking at Figure 2.19, that 
gradiual acceleration and gradual braking will be applied to the shoulder 
yaw. At each time increment the values for \( t, T \) and \( \Delta \phi \), are used 
in the driving function to compute the value of \( \dot{\phi}_1(t) \). This value of 
\( \dot{\phi}_1 \) is then used to determine \( R \) and \( Z \) of point A on a straight line 
path as shown in Section 2.5.

For case 3, when \( \dot{\phi}_1 = \dot{\phi}_f \), the driving function, \( \dot{\phi}_1(t) \) cannot 
be used. In this case \( T \) is computed using the maximum change, \( \Delta \theta_1 = \pi \). 
The driving function will then be determined by the change in radius, \( \Delta R \). 
The same type of driving function is used for \( R(t) \) as was used for 
\( \dot{\phi}_1(t) \) in case 1 or 2. This will also provide gradual acceleration and 
braking of the arm.

\[
R_f - R_i \\
R(t) = R_i + \left( \Delta R \left( \frac{t}{T} - \frac{1}{2} \sin \left( \frac{2\pi}{T} \frac{t}{T} \right) \right) \right)
\]

\[
R_f - R_i \\
R(t) = R_f - \left( \Delta R \left( 1 - \frac{t}{T} + \frac{1}{2} \sin \left( \frac{2\pi}{T} \frac{t}{T} \right) \right) \right)
\]

The values of \( X, Y \) and \( Z \) are then computed as in case 3, Section 2.5 
(also for case 1).
small. Too small a time might cause the controller to generate a command that could saturate the actuator. Therefore, during experimentation with this method on the BAT, the value of FS is approximately 3.5. It starts out this high for equipment safety and can be decreased if the arm moves too slowly. Once the value of $\Delta \theta$ (the change in shoulder yaw) is computed from the initial and final position of point A, it is used to compute $T$, the total time to traverse the straight line path.

The value of $T$ along with the value of $\Delta \theta$ (the shoulder yaw change) is used in the driving function, $\theta(t)$ (Figure 2.17). By letting $t$ start from 0 and slowly increasing $t$ until $t = T$, the driving function, $\theta(t)$, will be sampled [1]. This is shown in Figure 2.19 for the case when $\dot{\theta}_f$ is greater than $\dot{\theta}_i$. The value of $\Delta t$, the time

---

**FIGURE 2.19** Sampled Driving Function
Integrate over $X$ to find the total drag force. To find the Drag Moment,

$$\text{Drag Moment} = \text{Drag Force} \times \text{Moment Arm}$$

$$\text{Drag Moment Arm} = D_{\text{ARM}} \times Y \quad Y : 0, L_{\text{ARM}}$$

$$\text{Drag Moment Beam} = D_{\text{BEAM}} \times Y \quad Y : L_{\text{ARM}}, L_{\text{ARM}} + \text{Diameter of Beam}$$

Integrating over $Y$ gives the total drag moment on the arm and beam for a given velocity, $V$. The two integrations give a constant (called Drag) multiplied by the $\frac{1}{2}$ term

$$\text{Drag Moment} = \text{Drag} \times \frac{1}{2} \cdot$$

The value for maximum torque ($\tau$) was computed using the maximum torque for the shoulder yaw motor and the gear train dimensions between the shoulder and the arm. To find the total time, the maximum acceleration is used along with the velocity at that time. Recall that the maximum acceleration occurs at $t = \frac{T}{4}$. The value for $t$ is substituted into the equations for acceleration and velocity (Figure 2.17). These values of acceleration and velocity are used with the constants ($J_T$, $\Omega_0$, Drag) to find the equation for $T$ (Eq. (2.2)). $FS$ is a factor of safety which is used to ensure that the time taken to traverse the straight line path is not too
Moment of Inertia:

\[ J_T = J_{\text{ARM}} + J_{\text{BEAM}} \]

\[ J_{\text{ARM}} = J_{\text{GA}} + m_{\text{ARM}} \left( \frac{L_{\text{ARM}}}{2} \right)^2 \]

\[ J_{\text{GA}} = \frac{1}{12} m_{\text{ARM}} (3 R_{\text{ARM}}^2 + L_{\text{ARM}}^2) \]

\[ J_{\text{BEAM}} = J_{\text{GB}} + m_{\text{BEAM}} \left( L_{\text{ARM}} + \frac{R_{\text{BEAM}}}{2} \right)^2 \]

\[ J_{\text{GB}} = \frac{1}{2} m_{\text{BEAM}} R_{\text{BEAM}}^2 \]

Drag Moment:

Total Drag Force = \( D_{\text{ARM}} + D_{\text{BEAM}} \)

where

\[ D = \frac{2 C_{\text{D}} A V^2}{2} \]

\[ \text{Velocity}_{\text{ARM}} = \left( \frac{\pi}{2} \right) (x) \quad X : 0, L_{\text{ARM}} \]

\[ A = (\text{Diameter})(x) \quad X : 0, L_{\text{ARM}} \]

\[ \text{Velocity}_{\text{BEAM}} = \left( L_{\text{ARM}} + x \right) \quad X : 0, \text{Diameter of Beam} \]

\[ A = (L_{\text{BEAM}})(x) \quad X : 0, \text{Diameter of Beam} \]
The values of $J_T$ (total moment of inertia of the arm and beam) and drag moment exerted on the arm and beam by the medium are constants and computed using the simplified model shown in Figure 2.18. (The drag moment is a constant when using the value for velocity at $t = \frac{T}{4}$.

![Figure 2.18 Simplified Model of Arm and Beam](image)

By overestimating the dimensions of the arm and the beam, and assuming the arm and beam are in the position shown in Figure 2.18, the values of $J_T$ and the drag moment are also overestimated. These two constants were easily computed using very straightforward equations.
the path is not of the utmost importance. The main concern is to make sure a command is not generated which will saturate the actuator. Therefore, as long as the values used in the calculation of $T$ are overestimated, the arm will move at a reasonable speed and the command will not saturate the actuator. To find $T$, the equation for the torque and drag force is used and solved for $T$.

\[
\tau = J_T \ddot{\theta} + \text{Drag Moment}
\]

where

- $\tau$ = the maximum torque applied to the arm
- $J_T$ = the total moment of inertia of the arm grasping a beam
- Drag Moment = the total drag moment on the beam and the arm.

at $t = \frac{T}{4}$

\[
\ddot{\theta}(t) = \frac{2\Delta e \dot{\theta}}{T^2}, \quad \ddot{\theta}(t)^2 = \frac{\Delta e^2}{T^2}
\]

\[
\ddot{\theta} = J_T \dddot{\theta} + \dot{\theta}^2 \text{ Drag}
\]

Substituting and solving for $T$

\[
T = \text{F.S.} \sqrt{\frac{1}{\frac{1}{T^2} [J_T \dddot{\theta}^2 + \dot{\theta}^2 \text{ Drag}]}}, \quad (2.2)
\]

where F.S. = a Factor of Safety.
Case 1) Shoulder yaw initial ($\theta_{1i}$) < shoulder yaw final ($\theta_{1f}$).

Case 2) Shoulder yaw initial ($\theta_{1i}$) > shoulder yaw final ($\theta_{1f}$).

The velocity and acceleration are simply the first and second derivatives. To account for the starting position not being at zero in Figures 2.17a and 2.17b, the two equations are adjusted as follows.

Shoulder yaw initial < shoulder yaw final

$$\dot{\theta}_1(t) = \dot{\theta}_{1i} + (|\Delta \dot{\theta}| \left( \frac{t}{T} - \frac{1}{2\pi} \sin(2\pi \frac{t}{T}) \right))$$

Shoulder yaw initial > shoulder yaw final

$$\ddot{\theta}_1(t) = \ddot{\theta}_{1f} + (|\Delta \ddot{\theta}| \left( 1 - \frac{t}{T} + \frac{1}{2\pi} \sin(2\pi \frac{t}{T}) \right))$$

Case 3, when $\theta_{1i} = \theta_{1f}$, will be discussed at the end of this section.

By knowing the maximum acceleration the motor is capable of producing under the worst conditions, $T$ (the total time it takes the arm to traverse the path) can be determined for a value of $\Delta \dot{\theta}_1$. It is apparent that the maximum acceleration occurs at $t = \frac{T}{4}$ and $t = \frac{3T}{4}$. $t = \frac{T}{4}$ will be used to compute $T$, the total time. Putting $t = \frac{T}{4}$ into the equations for $\dot{\theta}_1(t)$ and $\ddot{\theta}_1(t)$, the maximum acceleration and velocity at that time are now functions of $\Delta \dot{\theta}$ and $T$. In this thesis, the speed to complete
\( \dot{\theta}_1(t) = \Delta \theta \left( 1 - \frac{t}{T} + \frac{1}{2\pi} \sin\left(2\pi \frac{t}{T}\right) \right) \)

\( \ddot{\theta}_1(t) = -\frac{\Delta \theta}{T} \left( 1 - \cos\left(2\pi \frac{t}{T}\right) \right) \)

\( \dddot{\theta}_1(t) = -\frac{2\pi \Delta \theta}{T^2} \sin\left(2\pi \frac{t}{T}\right) \)

**FIGURE 2.17b** Driving Function (Initial Greater than Final)
\[ \dot{\theta}_1(t) = \Delta \theta \left( \frac{t}{T} - \frac{1}{2\pi} \sin \left(2\pi \frac{t}{T} \right) \right) \]

\[ \ddot{\theta}_1(t) = \frac{\Delta \dot{\theta}}{T} \left( 1 - \cos(2\pi \frac{t}{T}) \right) \]

\[ \dddot{\theta}(t) = \frac{2\pi \Delta \ddot{\theta}}{T^2} \sin(2\pi \frac{t}{T}) \]

**FIGURE 2.17a** Driving Function (Initial Less than Final)
2.7 Determine the Total Time and the Value of Shoulder Yaw ($\theta_1$) at Each Time Step

Knowing $\theta$ (initial) and $\theta$ (final) from the end of Section 2.4, gives the initial and final shoulder yaw angles. The motor at the reference base of the arm drives the shoulder yaw, $\theta_1$. The specifications of this motor and a simplified model of the arm will determine the total time it takes the arm to traverse the straight line path. With the initial and final value of $\theta_1$, a driving function, $\theta_1(t)$, is used to provide ample braking of the system (Figure 2.17a and 2.17b) [2]. The braking will be shown later when this function is sampled. Three cases can occur.
3) find $\theta_3$ (elbow)

$$\theta_3 = 2\pi - \cos^{-1}\left(\frac{L_1^2 + L_2^2 - \gamma^2}{2L_1L_2}\right) .$$

Note: The angle in the triangle is only from 0 to $\pi$.

4) find $\psi$

$$\psi = \sin^{-1}\left(\frac{L_2 \sin(2\pi - \theta_2)}{\gamma}\right)$$

$\psi$ is only from 0 to $\frac{\pi}{2}$.

5) find $\theta_2$ (shoulder pitch)

$$\theta_2 = \frac{\pi}{2} - (\psi + \phi) .$$

Position iii) Elbow angle $(\theta_3) = \pi$ (Figure 2.16)

This case cannot happen in a straight line path unless the controller computes a $\gamma$ greater than $\gamma_{\text{max}}$ due to an input error or if the manipulator starts out with $\theta_3 = \pi$. In either case, the algorithm switches to Position i). Through simple trigonometric routines, the values of $\theta_1, \theta_2, \theta_3$ (shoulder yaw, shoulder pitch and elbow) are found which will move the end effector along a straight line path.
Position ii) Elbow angle ($\theta_3$) > $\pi$ (Figure 2.15)

FIGURE 2.15  Cylindrical Coordinates to Joint Angles - Case ii

1) find $\gamma$

$$\gamma = \sqrt{Z^2 + R^2}$$

2) find $\psi$

$$\psi = \tan^{-1}\left(\frac{Z}{R}\right)$$

is only from $-\pi/2$ to $-\pi/2$.
γ is checked here to insure it does not exceed \( \gamma_{\text{max}} \). This is just in case the operator mistakenly enters in a point beyond the manipulator's reach. If \( \gamma > \gamma_{\text{max}} \), \( \gamma_{\text{max}} \) is used and \( \theta_3 \) (elbow) will be at \( \pi \).

2) find \( \psi \)

\[
\psi = \tan^{-1} \left( \frac{Z}{R} \right).
\]

Note: \( \psi \) can only go from \(-\frac{\pi}{2}\) to \(\frac{\pi}{2}\), therefore, the arctan function will have no ambiguity.

3) find \( \theta_3 \), using the law of cosines (elbow)

\[
\theta_3 = \cos^{-1} \left( \frac{1}{2L_1L_2} \right) \left( L_1^2 + L_2^2 + \gamma^2 \right).
\]

Note: The value of \( \theta_3 \) can only go from 0 to \( \pi \) so the value of the arc-cosine function will have no ambiguity.

4) find \( \phi \), using the law of sines

\[
\phi = \sin^{-1} \left( \frac{L_2 \sin \theta_3}{\gamma} \right).
\]

Note: The value of \( \phi \) can only be between 0 and \( \frac{\pi}{2} \).

4) find \( \theta_2 \) (shoulder pitch)

\[
\theta_2 = \frac{\pi}{2} - (\gamma - \phi).
\]
wrist motors, the values of $\theta_4$ and $\theta_5$ at each time step are obtained using the same type of driving functions as $\theta_1(t)$. For example, if $\theta_{4f} > \theta_{4i}$ and $\theta_{5f} < \theta_{5i}$ then the values at each time step for $\theta_4$ and $\theta_5$ will be found using

$$
\theta_4(t) = \theta_{4i} + (\Delta \theta_4 (\frac{t}{T} - \frac{1}{2\pi} \sin(2\pi \frac{t}{T})))
$$

$$
\theta_5(t) = \theta_{5f} - (\Delta \theta_5 (1 - \frac{t}{T} + \frac{1}{2\pi} \sin(2\pi \frac{t}{T})))
$$

where $T =$ the total time found in Section 2.7

$t =$ the value for time using the time increment $\Delta t$.

The two values of orientation angles $\theta_4, \theta_5$, are then converted to wrist motor (A and B) angles $\psi_4, \psi_5$, using the transformations shown at the end of Section 2.3.

2.9 Simulation

An algorithm implementing this straight line path program was written in C and simulated on a VAX 11/782. The initial values of the joint angles and the final values of point A and orientation angles were entered from the keyboard. The program follows this outline:

1) Read the initial joint angles (shoulder yaw, $\psi_1$, shoulder pitch, $\psi_2$, elbow, $\psi_3$, wrist motor, $\psi_4$, and wrist motor, $\psi_5$) from the keyboard (all values were entered in radians).
2) Convert joint angles \( (\theta_1, \theta_2, \theta_3)_i \) to cylindrical coordinates \( (\Theta, R, Z)_i \) for the initial position.

3) Convert the wrist motor angles \( (\psi_4, \psi_5) \) to wrist orientation angles (wrist yaw, \( \psi_4 \), and wrist roll, \( \psi_5 \)) for the initial position.

4) Enter the final position of point A in cylindrical coordinates \( (\Theta, R, Z)_f \) and the final orientation angles \( (\psi_4_f, \psi_5_f) \). All values are entered in meters and radians.

5) Compute constants \( (\Delta \Theta, \Delta R, \Delta Z) \), the initial and final position of point A in the X, Y plane, the equation for the straight line between the two points).

6) Using \( |\Delta \Theta_1| \) (the change in shoulder yaw, which is also \( \Delta \Theta \) in cylindrical coordinates), compute \( T \) (the total time to traverse the straight line path).

7) Start a loop (from \( t=0 \) until \( t=T \), incrementing \( t \) by \( \Delta t \) (where \( \Delta t \) used in simulation was \( .25 \) seconds instead of the sampling time (\( \approx .05 \) seconds), this was done for the sake of brevity during simulation).

   a) Compute \( \phi_1(t) \), the driving function, for the appropriate case as in Section 2.7.

   b) Compute \( R \) and \( Z \) for point A at that particular \( t \) (found in step 7a) where \( \Theta \), \( R \) and \( Z \) are constrained by the straight line equation obtained in step 5.
c) Compute the three joint angles (shoulder yaw, \( \theta_1 \), shoulder pitch, \( \theta_2 \), and elbow, \( \theta_3 \)) from the cylindrical coordinates \((\theta, R, Z)\) found in step 7b.

d) Calculate the wrist yaw, \( \theta_4 \), and the wrist roll, \( \theta_5 \), using the appropriate driving functions, \( \theta_4(t) \) and \( \theta_5(t) \).

e) Check the limits for all joint and orientation angles \((\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)\).

f) Convert wrist orientation angles \((\theta_4, \theta_5)\) to wrist motor angles \((\psi_4, \psi_5)\).

g) Compute the velocity and acceleration of the shoulder yaw using the equations in Section 2.7.

h) Plot the following graphs:
   - shoulder yaw versus time
   - shoulder yaw velocity versus time
   - shoulder yaw acceleration versus time
   - wrist yaw versus time
   - wrist roll versus time
   - the position of point A in \( X, Y, Z \) space
     - \( X \) versus \( Y \)
     - \( X \) versus \( Z \)
     - \( Y \) versus \( Z \).
Many simulations were performed using this algorithm with excellent results. The plots show how gradual acceleration and braking were applied to shoulder yaw, $\theta_1$, wrist yaw, $\theta_4$, and wrist roll, $\theta_5$. The plots of point A in the X,Y,Z plane show how the path of point A is a straight line. Three of these simulation results follow.

Simulation 1) Shoulder yaw initial, $\theta_{1i}$, < shoulder yaw final, $\theta_{1f}$
- the elbow is in the elbow down position ($\theta_3 < \pi$). (See Figure 2.20.)
FIGURE 2.21 Shoulder Yaw Position - Simulation I

FIGURE 2.22 Shoulder Yaw Velocity - Simulation I
FIGURE 2.23 Shoulder Yaw Acceleration - Simulation I

FIGURE 2.24 Wrist Yaw Position - Simulation I
FIGURE 2.25  Wrist Roll Position - Simulation I

FIGURE 2.26  Y vs. X Position of Pt. A - Simulation I
FIGURE 2.27  Z vs. X Position of Point A - Simulation I

FIGURE 2.28  Z vs. Y Position of Point A - Simulation I
Simulation 2) - Shoulder yaw initial, $-\theta_1$, greater than shoulder yaw final, $\theta_1$.
- The elbow is in the elbow up position ($-3 > \tau$).

FIGURE 2.29  Arm Position-Simulation II
FIGURE 2.30 Shoulder Yaw Position - Simulation II

FIGURE 2.31 Shoulder Yaw Velocity - Simulation II
FIGURE 2.32 Shoulder Yaw Acceleration - Simulation II

FIGURE 2.33 Wrist Yaw Position - Simulation II
FIGURE 2.34 Wrist Roll Position - Simulation II

FIGURE 2.35 Y vs. X Position of Pt. A - Simulation II
FIGURE 2.36  $I$ vs. $x$ position at Pt. 3 - Simulation II

FIGURE 2.37  $I$ vs. $x$ position of Pt. 4 - Simulation II
Simulation 3) - Shoulder yaw initial, $\gamma_{1i}$, = shoulder yaw final, $\gamma_{4f}$
- the elbow is in the elbow down position ($\gamma_{3i} = -\gamma_{3f}$)
- the initial and final orientation angles are equal, $\gamma_{4i} = -\gamma_{4f}$, $\gamma_{5i} = -\gamma_{5f}$.

**FIGURE 2.28** Arm Position - Simulation III
- increment the time step
- synchronize the control system
- start again at the beginning of the loop
  until the total time is reached.

When the final point is reached, the function returns control to the calling function. The calling function in this case is the option block. The option used to select this method of control is discussed next.

Using screen functions supplied by the compiler, the operator can select many different options by using a character on the keyboard. These options are outlined in Appendix A (Sect. 3, Part D.5). A new option was added to the option block: if the operator selects 'm', the right arm will move the point A to the specified final point, point B, and the final orientation angles. The new option is outlined below.

- select the supervisory mode
- select option 'm'
- the master arm is read to find the starting configuration of the arm
- the final coordinates of point A and the final orientation angles are recalled from memory
- the move() function is called. The function is passed the initial angular configuration of the arm in integers and the values for the final cylindrical coordinates of point A and the final orientation angles.
- convert the integers of each angle $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$ of the initial position to the appropriate radian value.
- convert the initial position of the arm to cylindrical coordinates $(\theta, R, Z)$ of point A and to wrist orientation angle $(\theta_4, \theta_5)$ [it already has the final cylindrical coordinates of point A and the final orientation angles $(\theta_4, \theta_5)$.]
- compute the constants necessary for the straight line path constraint
- start the loop
  - using the driving function compute the cylindrical coordinates of point A on the straight line path for that time step
  - using the driving functions for wrist yaw and roll, compute the wrist yaw and roll angles $(\theta_4, \theta_5)$ for that time step
  - convert the cylindrical coordinates of the point A to joint angles $\theta_1, \theta_2, \theta_3$ (shoulder yaw, shoulder pitch and elbow)
  - check each angle for their limits
  - convert the orientation angles $(\theta_4, \theta_5)$ to wrist motor angles $(\theta_4, \theta_5)$
  - convert the angles from radians to integers
  - send the integers for each angle $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$ to the control system
Since the relationship between the angular position and the integer output of the sensor (a linear resistance potentiometer) is a linear one, the conversion from integer to radian is a simple equation. The data from the diagrams shown was used to develop the equations for each angle:

\[ \theta_1, \theta_2, \theta_3, \theta_4, \theta_5 \]

Therefore, in this step the central controller uses the existing functions in the BAT software (Appendix A, Sect. 3) to read the master arm. It takes these integer values and converts them to radian values using the equations developed from the data shown in the diagrams above. The central controller now knows the configuration of the master arm in radian measurement of \( \theta_1, \theta_2, \theta_3, \theta_4, \theta_5 \). Here again the assumption is made that the configuration of the master arm is the same as the configuration of the BAT's right arm. These equations are easily inverted, so that they are used to convert from radian measurement to integer value. This is used later in the procedure.

Now, the central controller knows the initial position of the arm and the final point. For this option it calls the straight line path generation function, now called move(). The move() function is a streamlined version of the simulation program used in the last chapter. The move() function is outlined below.
The procedure for obtaining the wrist motor integers was a bit different. Knowing the limits of wrist yaw and wrist roll, the master arm's wrist was moved to these limits and the integers for the wrist motors A and B were recorded.

\[
\begin{align*}
\theta_4^\text{MIN} &= 0.0 \text{ radians} \\
\theta_4^\text{MAX} &= 3.14 \text{ radians} \\
\theta_5^\text{MIN} &= -1.74 \text{ radians} \\
\theta_5^\text{MAX} &= 1.74 \text{ radians}
\end{align*}
\]

Using the conversion from reference [3] (also in Section 2.3), the corresponding limits for the wrist motors A and B were found. The data for these motors is shown below.

\[\begin{array}{c}
\text{Wrist Motor A, } \theta_4 \\
\hline
880 \text{ H} & 6C4 \text{ H} \\
600 \text{ FFF} & 600 \text{ FFF} \\
-4.7657 \text{ MIN} & 1.6242 \text{ MAX}
\end{array}\]
Shoulder Yaw, $\theta_1$

Shoulder Pitch, $\theta_2$

Elbow, $\theta_3$
The second step in the procedure is to read the angles of the master arm. Referring to Figure 2.1, the feedback from the right arm on the BAT does not come back to the central controller; the feedback stops at the joint/actuator cards. Therefore, the master arm positions are used to let the central controller know where the right arm of the BAT is. This is also covered in Appendix A. Reading the master arm lets the central controller know the initial angles of shoulder yaw, shoulder pitch, elbow, wrist motor A and wrist motor B. Discussed in Appendix A, Sect. 3, Part D.3, the program uses three functions to read and adjust the analog inputs from the ICS. The five master arm potentiometers are read along with the camera tilt and pan and the left arm roller motor. In this case, the program will read only the five master arm potentiometers. These functions will return integer values for $\psi_1, \psi_2, \psi_3, \psi_4$ and $\psi_5$. The integers come from an adjusted analog to digital conversion and represent 12-bit two's complement binary numbers. Using one of the options described in Appendix A, Sect. 3, Part D.5, the software was run with all power to the BAT shut-off. An option was used to show the hexadecimal number read by the central controller when the master arm was moved through each of the joint's limits. This hexadecimal number represented the integer from the adjusted analog to digital conversion. For example, the master arm's shoulder yaw was moved from $\psi_1^{\text{MIN}}$ to $\psi_1^{\text{MAX}}$ and the corresponding numbers were recorded. The three joint angle (shoulder yaw, shoulder pitch and elbow) measurements in radians and hexadecimal numbers are shown below.
In this figure, the operator has grasped two structural components with each arm and would like to make a connection. A supervisory option is selected and the right arm moves autonomously to point B. Point B was chosen to be next to the female end of the left arm's beam. The operator then switches back to master-slave control to make the connection. Point B was measured in cylindrical coordinates, in reference frame X₀,Y₀,Z₀ (Section 2.3), and has the following values:

\[ \phi = 2.26 \text{ radians (also this is shoulder yaw)} \]
\[ R = 0.6 \text{ meters} \]
\[ Z = -0.1 \text{ meters} \]

The final orientation angles \((\phi_4, \phi_5)\) were chosen to be:

\[ \text{wrist yaw, } \phi_4 = 1.57 \text{ radians} \]
\[ \text{wrist roll, } \phi_5 = 0.0 \text{ radians} \]

These values are stored as constants in an array at the beginning of the option. A new position can be measured and the values stored any time prior to the start-up procedure. In the future, an input function could be added to this option, so that the final point can be changed on-line. Presently, to speed up experimentation with this method of control, the final point is stored as a constant in the software.
b) At each time step, the radian measurement for each angle in step 4a) is converted to the appropriate integers for the control system.

c) At each time step, the integers are sent to the control system.

5) Control is returned to the main program.

The final position was chosen to be point B, shown in Figure 3.1.

FIGURE 3.1 Finish Point - Implementation I
CHAPTER 3
IMPLEMENTATION I:
UNKNOWN STARTING POSITION, KNOWN FINISH POINT

Using the straight line path generation program developed in the last chapter, supervisory control of the BAT's right arm can be added to the BAT software. With a specified finish point, the operator can select an option using the keyboard and the right arm will move to that point. The procedure is outlined below.

1) The final point $A$ is stored in the program in cylindrical coordinates $(\rho, R, Z)_f$ along with the final wrist orientation angles $(\psi_4, \psi_5)_f$.

2) The shoulder yaw, shoulder pitch and elbow angles $(\psi_1, \psi_2, \psi_3)_i$ along with the wrist motor A and B angles $(\psi_4, \psi_5)_i$ are read from the master arm potentiometers.

3) The angles read in step 2 are in integers and must be converted to the appropriate angular measurement (radians).

4) The straight line path generation program is called.

   a) At each time step, the program generates shoulder yaw, $\psi_1$, shoulder pitch, $\psi_2$, elbow, $\psi_3$, wrist motor A, $\psi_4$, and wrist motor B, $\psi_5$ angles in radians keeping point $A$ on a straight line path.
This program will be used to implement the supervisory control of the right arm. The first trajectory will be from an unknown starting position to a known finish point and will be discussed in the next chapter.
FIGURE 2.45 Z vs. X Position of Point A - Simulation III

FIGURE 2.46 Z vs. Y Position of Point A - Simulation III
FIGURE 2.43  Wrist Roll Position - Simulation III

FIGURE 2.44  Y vs. X Position of Point A - Simulation III
FIGURE 2.41 Arm Radius Acceleration - Simulation III

FIGURE 2.42 Wrist Yaw Position - Simulation III
FIGURE 2.39  Shoulder Yaw Position - Simulation III

FIGURE 2.40  Arm Radius Velocity - Simulation III
after the arm has completed the move, exit the supervisory mode.

The space bar is used as an emergency exit from both this option and the supervisory mode.

The program was tested in two steps:

step 1) select the option, enter the initial arm configuration in integers instead of using the master arm, write the joint angles \( \theta_1, \theta_2, \theta_3, \theta_4, \theta_5 \) to a file instead of sending them to the control system. This step checks the program flow and the integer/radian conversions.

In step one, the following integers (shown with their corresponding radian value) were entered:

Shoulder yaw, \( \theta_1 = 0.785 \) corresponds to 3373.

Shoulder pitch, \( \theta_2 = 1.57 \) radians corresponds to 110

Elbow, \( \theta_3 = 0.785 \) radians corresponds to 3130

Wrist yaw, \( \theta_4 = 0.0 \) radians

Wrist roll, \( \theta_5 = 1.57 \) radians

these correspond to:

Wrist motor A, \( \theta_4 = -1.46 \) radians corresponds to 4064

Wrist motor B, \( \theta_5 = -1.46 \) radians corresponds to 2550.
The arm was initially configured as shown in Figure 3.2. The resulting plots follow.

FIGURE 3.2 Arm Position - Step 1, Implementation 1
FIGURE 3.3 Shoulder Yaw Position, Step 1, Implementation I

FIGURE 3.4 Wrist Yaw Position, Step 1, Implementation I
FIGURE 3.5  Wrist Roll Position, Step 1, Implementation I

FIGURE 3.6  Y vs. X Position, Step 1, Implementation II
FIGURE 3.7  Z vs. X Position, Step 1, Implementation 1

FIGURE 3.8  Z vs. Y Position, Step 1, Implementation 1
As shown by the plots, there is a gradual acceleration and gradual braking of the shoulder yaw, wrist yaw and wrist roll angles. The last three plots show how the program does generate a straight line path for point A and it does finish at point B.

Step 2) Select the option, send the joint angles $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$ to a file instead of to the control system. This step checks everything except the actual movement of the BAT's right arm.

This step was run at the ICS and the values obtained are shown in the plots which follow.

![Shoulder Yaw Position, Step 2, Implementation I](image-url)
FIGURE 3.10  Wrist Yaw Position, Step 2, Implementation I

FIGURE 3.11  Wrist Roll Position, Step 2, Implementation I
FIGURE 3.12  Y vs. X Position, Step 2, Implementation I

FIGURE 3.13  Z vs. X Position, Step 2, Implementation I
The plots show how the angles (shoulder yaw, wrist yaw, and wrist roll) are gradually accelerated and gradually braked. The last three plots demonstrate the straight line generation of point A.

Actual tests done on the BAT with this method of control will be discussed in the last chapter. This chapter shows that the right arm control system can be upgraded to supervisory control. This now gives the operator the capability of moving the right arm to a specified position without using the master arm.
The last chapter demonstrated that the central controller could control the right arm of the BAT, by moving the right arm's end effector in a straight line path from any point to any other point, within the arm's range and flexibility limits. An improvement on this capability would be to enable the central controller to remember a sequence of points, concatenate their paths and move the right arm along that specified trajectory. This can be done easily with the move() function developed in the last chapter and the addition of three new options in the software.

The first new option enables the central controller to remember a sequence of points in space. These are the points that the operator would like the path of the right arm's end effector to move through. It can be done by reading the position of the master arm and storing that position in an array. This operation consists of four steps.

1) enter the supervisory mode
2) choose this option by hitting the 'x' on the keyboard
3) the central controller reads the five potentiometers of the master arm corresponding to the shoulder yaw, $\theta_1$, shoulder pitch, $\theta_2$, elbow, $\theta_3$, wrist motor A, $\psi_4$, and
wrist motor $\psi_5$. It stores these integers in a two-dimensional array (7 rows by 5 columns) with each column corresponding to $\theta_1, \theta_2, \theta_3, \psi_4, \psi_5$ (the integer value) in that order. The entire first row is skipped. This row is saved for the starting point. An operator can store as many as five points. The last row is saved for the finish point which is the same point as the starting point. Obviously, each row is a right arm position.

4) return to the calling function.

The maximum number of points stored can be increased easily by making the array have more rows. For this thesis it will remain at five.

The second new option allows the operator to clear the array of all points and start over. When the 'y' key is selected, the option simply re-initializes the array's elements to all zeroes. It then returns to the calling function.

Now the operator has the capability of storing any points in the right arm's range of motion. The operator simply moves the right arm, joins the master arm, to a point and selects 'x'. The point is a row of integers corresponding to the master arm's potentiometers of the joint angles. The operator then goes to the next point and selects 'x' again. This point is then stored in the row below the last stored point. The operator does this until all the points desired have been stored. The software joins these points in sequence to
make a desired trajectory. (See Figure 4.1.) The operator has the capability of clearing the trajectory and starting over. This is used when entering another trajectory or when a mistake has been made in entering one of the points.

FIGURE 4.1 Implementation II

The final option is the one which will move the right arm from wherever it is, through the sequence of points which were stored and then move back to where it originally started. The paths through all these points will be a straight line. The outlines of this new option is listed below.
1) enter the supervisory mode

2) choose to move through all the stored points by selecting 'z' on the keyboard

3) read the current position of the master arm and store this position in the array as the first row and directly after the last row stored (for example, if only three points have been stored when 'z' is selected the current position is in row number 1 and row number 5, row numbers 2,3,4 are the three points already stored by selecting 'x')

4) start a loop in which the program takes two rows of integers at a time, starting with the first and second row, and set these points (a row of integers) equal to the initial and final positions for the straight line path

5) convert the integers of the final position to cylindrical coordinates of point A and final wrist orientation angles ($\alpha, R, Z, \psi_4, \psi_5$). Of course, to do this it must first convert the five integers to shoulder yaw and pitch, elbow, and wrist motor angles to radians. It then uses the radian values of $\psi_1, \psi_2$ and $\psi_3$ to convert to $\alpha, R, Z$ of point A and $\psi_4, \psi_5$ to convert to $\psi_4, \psi_5$. This was covered in Chapter 2.

6) call the move() function passing it to the initial position, which is the first row of angles in integers, and
the final point A and orientation angles, found in step 5. The move() function moves the end effector of the right arm (point A) from the initial position to the final point in a straight line path.

7) When the move() function is completed, control goes back to step 4 and takes the next set of two points (in this case rows two and three, the next time it will take rows three and four and so on) and repeats steps 5, 6 and 7.

8) When the last pair is used the right arm will be back at the starting point (this is because the current position was also stored as the last point in the array).

9) Control is returned to the calling function.

In testing this method of concatenating points the same two step procedure used in testing the method of Chapter 3 was used here. First, all entries were made as integers by keyboard input and all angles were sent to a file. This tested program flow and integer/radian inversion. Second, the master arm was used to enter the points and all values were sent to a file. This tested all aspects of the program except the actual movement of the BAT's right arm. Actual tests done with the BAT will be discussed in Chapter 5.
Step 1) Keyboard entry, values sent to a file (see Figure 4.2).

<table>
<thead>
<tr>
<th>Joint</th>
<th>START/FINISH</th>
<th>POINT 1</th>
<th>POINT 2</th>
<th>POINT 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shoulder yaw</td>
<td>.785 rad</td>
<td>1.5707 rad</td>
<td>1.570 rad</td>
<td>1.22 rad</td>
</tr>
<tr>
<td></td>
<td>3373</td>
<td>4090</td>
<td>4090</td>
<td>3771</td>
</tr>
<tr>
<td>Shoulder pitch</td>
<td>1.570 rad</td>
<td>1.91986 rad</td>
<td>1.919 rad</td>
<td>2.18 rad</td>
</tr>
<tr>
<td></td>
<td>110</td>
<td>3662</td>
<td>3662</td>
<td>3428</td>
</tr>
<tr>
<td>Elbow</td>
<td>.785 rad.</td>
<td>.78539 rad</td>
<td>1.57 rad</td>
<td>1.047 rad</td>
</tr>
<tr>
<td></td>
<td>3130</td>
<td>3130</td>
<td>3894</td>
<td>3385</td>
</tr>
<tr>
<td>Wrist yaw</td>
<td>1.570</td>
<td>0.0 rad</td>
<td>.785 rad</td>
<td>2.35 rad</td>
</tr>
<tr>
<td>Wrist roll</td>
<td>0.0 rad</td>
<td>1.570 rad</td>
<td>.785 rad</td>
<td>0.0 rad</td>
</tr>
<tr>
<td>Wrist motor A</td>
<td>-1.57 rad</td>
<td>-1.462 rad</td>
<td>-1.516 rad</td>
<td>-2.35 rad</td>
</tr>
<tr>
<td></td>
<td>4002</td>
<td>4064</td>
<td>4033</td>
<td>3553</td>
</tr>
<tr>
<td>Wrist motor B</td>
<td>1.570 rad</td>
<td>-1.462 rad</td>
<td>.054 rad</td>
<td>2.35 rad</td>
</tr>
<tr>
<td></td>
<td>141</td>
<td>2550</td>
<td>3394</td>
<td>579</td>
</tr>
</tbody>
</table>
FIGURE 4.2  Arm Positions, Step 1, Implementation II
FIGURE 4.3 Shoulder Yaw Position, Step 1, Implementation II

FIGURE 4.4 Wrist Yaw Position, Step 1, Implementation II
FIGURE 4.5  Wrist Roll Position, Step 1, Implementation II

FIGURE 4.6  Y vs. X Position, Step 1, Implementation II
FIGURE 4.7 Z vs. X Position, Step 1, Implementation II

FIGURE 4.8 Z vs. Y Position, Step 1, Implementation II
The plots show the gradual acceleration and braking applied to the shoulder yaw, wrist yaw and wrist roll as the arm moves through all the points. The plots in X,Y,Z space show how the arm's end effector position (Point A) is kept on a straight line path through all the points. It also shows how the arm moves in a closed loop. It starts and finishes at the same point.

Step 2) Master arm entry, all values sent to a file.

FIGURE 4.9  Shoulder Yaw Position, Step 2, Implementation II
FIGURE 4.10  Wrist Yaw Position, Step 2, Implementation II

FIGURE 4.11  Wrist Roll Position, Step 2, Implementation II
FIGURE 4.12  $Y$ vs. $X$ Position, Step 2, Implementation II

FIGURE 4.13  $Z$ vs. $X$ Position, Step 2, Implementation II
Here again, the plots give the same type of results; the gradual acceleration and braking, the straight line paths and the closed loop trajectory.

The operator now has a great deal of supervisory control over the right arm. The only time the operator needs to use the master arm is to store points for a trajectory. He/she could then put the master arm in a set position and use the 'z' option to perform a movement task over and over. This will help free the operator's hands for other functions at the ICS. The extensions of this will be discussed in the next chapter.
CHAPTER 5
CONCLUSIONS

The two implementations discussed in Chapters 3 and 4 were incorporated into the existing BAT software. The operator can enter the supervisory control mode by either hitting the 'w' key or by pushing the black pushbutton between the joysticks on the ICS's bottom control panel. The code for the supervisory control mode is listed in Appendix B. While the BAT was in the laboratory (not in the neutral buoyancy environment), the software was tested. All of the BAT's right arm actuators were powered up except the shoulder yaw motor. It would place too much strain on the motor if it was driving the arm outside the neutral buoyancy environment. At the ICS the supervisory mode was selected and both implementations (unknown starting position to a known finish point; concatenating points) were run. Since the shoulder yaw motor was not powered up it could not be ascertained as to whether the path of point A (the end effector) was a straight line path. It was very apparent that the arm movement was not erratic and was very smooth. The joints that were powered did move with gradual acceleration and braking. When the second implementation was performed the arm did pass through all the points and returned to the finish point.

The supervisory control of the BAT was tested in the neutral buoyancy environment by the Space Systems Laboratory. While in master-slave control, one of the wrist motors was running intermittently. This caused a sudden jerk in the movement of the wrist since the two wrist
motors are coupled to give wrist yaw and roll. In the supervisory mode, when both implementations were tested, the same jerks also occurred in the wrist movement. Otherwise, the wrist moved to the points with a gradual acceleration and a gradual braking. The elbow moved smoothly throughout the test in both master-slave and supervisory mode. Shoulder yaw and pitch operated smoothly in master-slave control, but did not move at all when the supervisory options were selected. Due to an oversight in the development of the software, the shoulder pitch and yaw commands were not being sent to the control system. This was easily corrected but not in time to be tested again. Since only the elbow and wrist were being controlled, the end effector did not move in a straight line path.

Although the results of the test are inconclusive as to whether the supervisory control method will move the arm in a straight line path to a specified point, it does show that, in the neutral buoyancy environment, the elbow joint was controlled smoothly and gradually. The next test will be directed towards insuring all the joints are controlled properly. Upon completion of this step, the accuracy of the joint positions will be evaluated.

From the plots of the simulations in Chapter 2 and the plots of the implementations in Chapters 3 and 4, it is easy to see that for the given inputs the arm will move in a straight line path. With the test on the BAT in the laboratory, the results showed that the program flow was working properly and the arm was moving smoothly with gradual acceleration and braking. When the program is tested on the BAT again in the neutral
buoyancy environment, the only cause for error is if the relationship between the BAT's right arm joint angles and the joint angles which the central controller generates is not the same. This refers back to the assumption that the master arm's position gives the correct position of the BAT's right arm. This can be corrected by simply readjusting the integer/radian conversion between the master arm's analog to digital conversion and the angular position of the BAT's right arm joints. Once the integer/radian conversion is accurate, the previous results conclude that the arm's end effector will move in a straight line path smoothly while using this method of supervisory control.

With this method of controlling the BAT's right arm movements, many extensions can be incorporated into the BAT's software, so that the control relationship will be almost entirely supervisory. Other functions can be added to the software, so that the right arm can perform many other tasks by using keyboard entry at the central controller. For example, a function called rotate (arguments) can be developed. Here the orientation angles (wrist yaw and wrist roll) are changed according to the arguments of the function (i.e., rotate (π/2, 0) would change the wrist yaw π/2 radians and the wrist roll 0 radians). There could also be other functions like lift (arguments) where just the shoulder pitch and elbow angles would be changed, so that the arm's end effector would move in a straight vertical line. All these functions can be developed and stored in a library. The operator could then perform complete assembly tasks by simply typing in a list of these functions for the BAT to follow. For example, a list could be as follows:
small motor drive circuit can handle only 50 amps. This provides all eight drives (the camera tilt and pan, the five right arm motors and the left arm motor) necessary for the operation of the BAT. The control of the right arm is discussed in more detail in Appendix A, Section 3.

D) Propulsion Subsystem (see Figure A.8)

There are eight electric trolling motors pressurized for waterproofing. They combine the thrust to give the BAT six degrees-of-freedom. The main power subsystem supplies power to the power transistor modules which drive the motors. The modules contain pulse-width modulation drives for each of the eight motors. They are controlled from the control box propulsion controller through the relay box. The propulsion controller receives its commands from the ICS through the RRT. An inertial package provides attitude and rate feedback through the propulsion controller and the RRT to the ICS.

Aside from the video signals from the two cameras on the BAT, all of the BAT's input and output signals are passed through the RRT in the control box to the ICS. All the BAT's input signals originate at the ICS, and all the BAT's output is fed back to the ICS.

A.2 Hardware of the ICS

During the neutral buoyancy tests the BAT is controlled by the Integrated Control Station (ICS) (see Figure A.9). The ICS's frame is made of welded angle stock and mounted on wheels so it can be transported to test sites with ease. It has room to hold the operator, the controls and their displays, and video equipment.
FIGURE A.7  Right Arm Subsystem
FIGURE A.6 Left Arm Subsystem

DC MOTOR FOR ROLLER

END EFFECTOR

SMALL POWER MODULE

SOLENOID VALVE BOX

MAIN POWER

RRT

ICS
motors receive their input commands from the ICS through the RRT and two power modules. These power modules will be described in further detail in the right arm subsystem. Two optical encoders are used for tilt and pan position feedback to the interfaces in the control box. The cameras' video signals are connected to video hardware at the ICS through coax cables.

B) Left Arm Subsystem (see Figure A.6)

The left arm is a specialized actuator which can grasp a beam and move it laterally across the body of the BAT. It grasps the beam by means of a specialized end effector, which is opened and closed pneumatically. It is controlled by the ICS through the solenoid valve box and the RRT. At the base of the end effector is a roller. This roller can move a beam across the BAT. The roller is driven by a permanent magnet DC motor. The motor receives its power from the main power subsystem through a small power module (described in following section).

C) Right Arm Subsystem (see Figure A.7)

The right arm is a five degree-of-freedom manipulator with a grasping end effector. The end effector is controlled pneumatically from the ICS through the RRT. Each of the five degrees-of-freedom are driven by a permanent magnet DC motor. The motors are connected to the power modules, which are pulse-width modulation amplifiers. They are controlled from the ICS through the RRT. There are two power modules. Each power module contains four independent motor drive circuits; two large and two small. A large motor drive circuit can handle up to 100 amps, while a
FIGURE A.5  Video Subsystem
supplies low pressure air to keep each box in the BAT watertight. The other tank supplies high pressure air, which is connected to the solenoid valve box. The solenoid valve box uses this high pressure to control three components:
- the left arm end effector
- the right arm end effector
- the main power relay.

Both end effectors receive their open/close commands from the ICS through the RRT. The Main Power Relay is an electronically-controlled pneumatically-operated switch which controls the power output of the main battery box. Its normal operating position is closed. It may be activated from the ICS or from a switch on the BAT. In the event of a communications failure or upon activation it opens, disconnecting the main power from the BAT's actuators and thrusters.

II. Action Subsystems

A) Video Subsystem (see Figure A.5)

The video subsystem has two commercial cameras, each housed in an aluminum tube with a front plexiglas plate. The main camera is mounted on a tilt and pan unit on the top of the BAT. A second camera is mounted above the right arm pivot and tracks the position of the end effector. The cameras are connected to a camera support box, which powers the cameras with two gel cell lead acid batteries hooked to a voltage regulator. The platform is a tilt and pan unit with two permanent magnet DC motors. The
FIGURE A.4 Pneumatic Subsystem
FIGURE A.3  Control Subsystem
- the power amplifiers of the right arm, left arm and the tilt and pan unit of the main camera;
- the relay box of the propulsion subsystem;
- the battery monitor card in the control subsystem.

B) Control Subsystem (see Figure A.3)

The control subsystem has two components; the control battery box and the control box. The control battery box consists of 12 Vdc, 5 Vdc and ±6 Vdc power supplies which supply the power for the solenoid valves, the control logic and the communications link, respectively. Inside the control box is a battery monitor card, two encoder interfaces, a propulsion controller and communications hardware. The battery monitor card samples and digitizes the voltage levels of the main batteries and the air pressure transducer. The two optical encoder interfaces are used with each actuator in the action subsystem. They convert each encoder's output and transmit the outputs to the remote receiver/transmitter (RRT). The propulsion controller receives information from the inertial package of the propulsion subsystem. It also sends commands from the RRT to the propulsion subsystem. The control box communicates with the ICS through the telemetry uplink and downlink. This is implemented by the RRT which combines the uplink multiplexer with the downlink digital demultiplexer. It also contains the analog transceiver.

C) Pneumatic Subsystem (see Figure A.4)

The pneumatic subsystem consists of two separate air tanks. One
Main Power Subsystem

Propulsion Subsystem
Joint/Actuator Subsystem (Power Modules)
Control Subsystem (Battery Monitor Card)
The Beam Assembly Teleoperator (BAT) (Figure A.1) is a self-contained free-flying teleoperator. It was developed to be used underwater in neutral buoyancy structural assembly tests. The BAT's flotation panels provide most of its neutral buoyancy. It is controlled from an above-water control station, the ICS (described in Section A.3). The BAT can be broken down into four support subsystems and three action subsystems.

I. Support Subsystems
   A) Main Power Subsystem
   B) Control Subsystem
   C) Pneumatic Subsystem
   D) Propulsion Subsystem

II. Action Subsystems
   A) Video Subsystem
   B) Left Arm Subsystem
   C) Right Arm Subsystem

I. Support Subsystems
   A) Main Power Subsystem (see Figure A.2).

   The main battery box contains six battery packs connected in parallel. Each battery pack contains three gelled electrolyte lead acid cell batteries connected in series. Eighteen volts at 20 amp-hours is supplied by each battery pack. The use of the on-board battery box eliminates the need for a cumbersome umbilical and cuts down on power loss through transmission. The main battery box is connected through the main power relay (described in the pneumatic subsystem) to the bus-bar. The distribution bus-bar connects the main battery box to:
APPENDIX A

DESCRIPTION OF THE BAT

The Beam Assembly Teleoperator was developed and is used by the Space Systems Laboratory at M.I.T. to experiment with the man-machine interface for assembly tasks. This appendix describes the system in detail. The appendix is broken down into three sections:

A.1 Description of the BAT
A.2 Hardware of the Integrated Control Stations (ICS)
A.3 Software of the ICS.

A.1 Description of the BAT (see Figure A.1)
REFERENCES


- move(arguments)
- rotate(arguments)
- lift(arguments)
- move(arguments)

This list of functions can be stored as a macro, so that the operator can simply hit a function key (F1,F2, etc.) or type in the macro's name and the right arm of the BAT would perform the listed tasks. The operator can construct these lists, so that the arm would be performing a complete assembly task. All these functions (except the end effector's grasp() and release() functions) can be developed by simply modifying the supervisory method developed in this thesis.

The use of supervisory control on the BAT's right arm would free the operator's hands from the master arm and let the operator concentrate on other duties at the ICS. Supervisory control will increase the capabilities of the BAT and help improve the interface between man and machine during assembly operations.
CONTROL BOX

<table>
<thead>
<tr>
<th>CONTROL BOX</th>
<th>INERTIAL PACKAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROPULSION CONTROL</td>
<td></td>
</tr>
<tr>
<td>RRT</td>
<td>POWER TRANSISTOR MODULES</td>
</tr>
<tr>
<td>ICS</td>
<td>EIGHT THRUSTERS</td>
</tr>
<tr>
<td>RELAY BOX</td>
<td>MAIN POWER</td>
</tr>
</tbody>
</table>

FIGURE A.8 Propulsion Subsystem
The ICS consists of five subsystems:

A) ICS Controls and Displays
B) Central Controller
C) Joint/Actuator Control System
D) Propulsion Control System
E) Communication Link.

A) ICS Control and Displays

The ICS controls and displays system takes up most of the area of the ICS. It contains the helmet for video control, the master arm and the control/display panels. The helmet is located above the operator and is
mounted on a gimbal transducer system. This system converts head movements into tilt and pan commands used by the central controller to control the BAT's main video camera. This helmet may be swung out of the way and back-up camera controls on the control/display panels used instead. The master is a five degree-of-freedom kinematic duplicate of the right arm on the BAT. The operator straps his/her arm on the master arm and positions it to where he/she would like the BAT's right arm to go. The positions of each joint of the master arm are read by the central controller, and transmitted through the joint/actuator control system to the BAT's right arm. Both the master arm and helmet systems implement master-slave control for the right arm and video subsystems, respectively.

There are four control/display panels (see Figure A.10) situated in front of the operator. The functions of all the controls on these panels can be changed through the software of the central controller (described in the next section). The top panel contains back-up controls for the propulsion control of the BAT and the back-up joystick control for the tilt and pan of the video subsystem. The second panel from the top contains 5 knobs which can operate in two modes. In one mode, the knobs can tune the gains and positions of the joint/actuator control system. In the other mode, the knobs can control the five joints of the BAT's right arm. The modes can be switched using the software in the central controller. There is also a back-up switch that can control the opening and closing the the end effector of the BAT's right arm. A lever is situated on the right of the panel to control the motor for the left arm roller. The third panel
JOYSTICK BACKUP PROPULSION CONTROLS

CAMERA BACKUP CONTROL

PITCH ROLL YAW

X TRANS Y TRANS Z TRANS

BACKUP FOR LEFT ARM MOTOR

RIGHT END EFFECTOR TUNE GAIN LEFT ARM TUNE POSITION

MODES:
1) Joint Tuning
2) 5 DOF in Control Mode

JOYSTICK

FOUR INCH MONITOR FOUR INCH MONITOR FOUR INCH MONITOR FOUR INCH MONITOR LIGHTS

PRESSURE SWITCHES

LEFT ARM ON OFF
RIGHT ARM ON OFF
SAFETY VALVE ON OFF

NINE INCH MONITOR

NINE INCH MONITOR

BACKUP PRESSURE SWITCHES

ENABLE OPEN OPEN
DISASSEMBLE CLOSE CLOSE
SAFETY VALVE LEFT ARM RIGHT ARM

3 DOF JOYSTICK

PUSH BUTTONS

3 DOF JOYSTICK

FIGURE A.10 ICS Control Panels
from the top is the largest and contains the video monitors for the ICS. There are 2 nine inch monitors and 4 four inch monitors above them. On the panel is a dial which is used to switch the display around on the different monitors. These monitors can display images sent from the cameras on the BAT or graphics produced by the software of the central controller. The graphics display is used to show the system settings and the current status. Other cameras, such as payload bay cameras, may also be hooked into the system and shown on these monitors. Before being connected to the monitors, the cameras are hooked into video equipment located on a rack behind the panels. Above the monitors on the right side is a set of lights which monitor the duty cycle for each actuator on the BAT. To the right of the panel are the pressure switches which control the high pressure of the pneumatic subsystem of the BAT. The switches control the right arm end effector, the left arm end effector and the safety valve. They go directly to the communications link and bypass the central controller. Below the video monitors are three backup buttons for the three pressure switches. The bottom panel contains the primary controls for the BAT's propulsion subsystem. These two joysticks (each joystick has three degrees-of-freedom) are used to control the BAT's movements. These controls can also be changed by software to control the BAT's right arm. Four pushbuttons are located between these joysticks. These buttons are used as backups for selecting options in the software (these options are described in the next section). All the outputs from the master arm, the helmet gimbal system and the panel controls (with the exception of the pressure controls) go to an analog-to-digital converter for processing. The analog-to-digital converter digitizes
and multiplexes these inputs for use by the central controller software.

B) Central Controller

The central controller is located on a rack behind the ICS control/display panels. The central controller is an IBM Personal Computer with 256K RAM, two disk drives, a math 8087 coprocessor and special interface boards to communicate to the other subsystems (see Figure A.11). The central controller:

- interfaces with the controls and displays of the ICS
  the the analog-to-digital converter.
- commands and monitors the joint/actuator and propulsion control subsystem.
- provides integrated control functions.
- records test data.

System status (such as battery voltage levels) is sent from the control box on the BAT through the communications uplink to the central controller. The software which controls the BAT system is described in detail in Section A.4.

C) Joint/Actuator Control System

The joint/actuator control subsystem consists of two parts; a monitor/buffer card and the joint/actuator control cards (see Figure A.12). The monitor/buffer card is a standardized Z-80 microprocessor card which buffers the commands from the central controller and retransmits them to the joint/actuator control cards. It interfaces the central controller with the joint/actuator cards. There are four joint actuator cards, each of which uses a Z-80 microprocessor. Each card implements closed-loop
FIGURE A.11  Central Controller
FOUR JOINT/ACTUATOR CONTROL CARDS

ICS
MASTER ARM
BACKUP CONTROLS

ADC
CENTRAL CONTROLLER

BUFFER/MONITOR CARD

FOUR JOINT/ACTUATOR CONTROL CARDS

UPLINK
DOWNLINK
ENCODERS

JOINT/ACTUATOR SYSTEM

FIGURE A.12 Joint/Actuator Control System
control for two of the BAT's eight actuators. Control algorithms and parameters are downloaded to the cards from the central controller. They generate PWM drive signals and transmit them through the communications downlink to the BAT's five joint actuators of the right arm and the actuators for the left arm roller and for the camera tilt and pan. Each of the eight actuator positions are transmitted through the communications uplink back to the cards.

D) Propulsion Control System

The propulsion controller is implemented with a microprocessor card (see Figure A.13). The propulsion control card:

- accepts rotation and translation commands from the central controller
- performs the transformations necessary to convert rotation and translation commands to individual thruster commands.
- generates PWM drive signals for the power modules of the Propulsion subsystem.
- receives feedback from the BAT's inertial package through the communications uplink.

E) Communication Link

The communication link is a two megabits per second link which connects the control system to the BAT (see Figure A.14). The communication link consists of three parts; the downlink digital multiplexer, the uplink digital demultiplexer and the analog transmitter and receiver. The BAT has a similar link on-board. The speed of the communications link can
FIGURE A.13 Propulsion Control System
be upgraded if necessary. The digital multiplexer and demultiplexer are on separate cards in the control station whereas they are on the one card in the BAT. The analog receiver and transmitter are combined into a transceiver.

The hardware of the ICS and the BAT combine to make an effective teleoperator system. The interface between the operator and the system is in the software. The software can be used to change much of the way this interface occurs.

A.3 Software of the ICS

The central controller interfaces the ICS controls with the action and support subsystems of the BAT. The vast majority of the central controller software is written in the C programming language. The rest of the software is in assembly language. The modular aspect of C makes the software easier to follow and debug. Including the MAIN() function, the software is composed of 34 functions and the standard functions included in the C compiler. The 34 functions are all contained in three files; BAT1.C, SUPPORT1.C and SUPPORT2.C. These three files are linked together to make up the operating program for the BAT system.

The three files contain all the software to start up, run, adjust and shut-down the BAT system. It also displays all the necessary information on a monitor at the ICS. The file BAT1.C contains the MAIN() function which calls everything else (see Figure A.15).

BAT1.C calls:
A) initializebat()
B) calibratebat()
FIGURE A.15  Main() Function
C) screen() and screen routines
D) runbat()
E) shutdownbat()

SUPPORT1.C and SUPPORT2.C contain functions which are called from these five main functions.

A) Initializebat() (see Figure A.16)

This function configures the ICS interface and initializes the joint/actuator and propulsion control subsystem. It first calls batports() which configures the ports on the interface board in the central controller. Batports() does this by calling setport() for each of the five Intel 8255 Programmable Parallel Interface chips on the board. Setport() sets an 8255's port to the input or output mode. It calls outport() to send out the address and the control word to the chip. Outport() and inport() are functions which are written in Assembly language for speed and both do precisely what their names imply. They output information to a desired port from the central controller, and input information from a desired port to the central controller. After the ports are configured, initializebat() calls outport() to initialize the joint/actuator, video and propulsion control systems. It does this by either outputting zero to the port or strobing the port. The main power relay is then checked by inport() to see if the relay has been activated. If the main power relay is activated, the operator is notified and must deactivate the switch. The initialization process is then complete and control is returned to MAIN().

B) Calibratebat() (see Figure A.17)

This function calibrates the two 3 degrees-of-freedom joysticks used for primary control of the Propulsion subsystem. It informs the
FIGURE A.20  Shutdown BAT() Function
E) Shutdownbat() (see Figure A.20)

This function simply shuts down the BAT system by strobing the buffer/monitor and propulsion controller cards. It then returns to the MAIN() function. The MAIN() function then calls a screen routine and finishes.
actuator cards from a file. The screen() function is then called to restore the original display. Control is then returned to the beginning of the main loop.

i) 'i' - calls adjustpots(). This function uses the convert() and adjust() functions to adjust the potentiometers used for control inputs on the ICS. It can change the minimum or maximum limits, the offset or the interface between potentiometers and the ICS. It then calls the screen() function to restore the original display and returns control to the beginning of the main loop.

j) 'b' - this option is still being developed. It calls many functions such as testroutine(), readmast(), readjm(). Its purpose is to test the feedback of the joint/actuator control system. It attempts to bring the feedback to the central controller. Currently the feedback stops at the joint/actuator cards. It then restores the display using screen() and returns control to the beginning of the main loop.

k) '1,2,3,4' - these four options call shut cardoff(). This function shuts down the card which is selected. It sets the card's values to zero and calls transmast(). By shutting down the card, it also turns off that joint/actuator. It then returns control to the beginning of the main loop.

6) After these options are selected (or not selected as in the first case), synchronization is tested and the loop begins again. It does not exit the loop until carriage return is selected.
the propulsion card are reset using outport(). Initial values for address, data and command are sent to the buffer/monitor card via transmast(). Control is then returned to the beginning of the main loop.

f) 'c' - changes the camera control system. This is done by switching the value of the variable "headcontroller" (which was initialized as true). If the variable is true, the helmet gimbal system controls the video camera on the BAT. If it is false, the backup joystick on the ICS would then control the camera (Chapter 3, Figure 3.2). Control is then returned to the beginning of the main loop.

g) 'p' - switches the value of "puma". This variable determines whether the propulsion system is on or off. Control is then returned to the beginning of the main loop.

h) 't' - calls setupjoints(). Setupjoints() clears the screen, puts up a menu and retrieves a selection from the operator. The operator determines how he/she would like the joint control parameters set up. It gives the operator the capability of changing the gain of the controller, parameters of the controller and the type of control algorithm on the joint/actuator card (Proportional, Proportional-Derivative, State Feedback). It does this for each of the joint/actuators in the system by calling tunejoint(). Once the operator is satisfied with the values selected for each joint, the function setupjoints() writes the values to a file. The operator can also select the loadfromfile() function. This function loads the values (determined at some other time) of the joint/
putted from the keyboard and returns its value. If no character has been inputted, it returns zero. A menu is printed on the screen to help the operator select an option. The different options are outlined below.

- a) No input - if there is no input (i.e., the operator does nothing), the central controller calls inport() to check the status of five switches. These switches are located between the two joysticks on the bottom panel (see Chapter 3, Figure 3.2). Currently, since these switches have no function, the central controller simply checks if the switch settings were changed. If the settings were changed, the central controller prints a number corresponding to the switch settings and returns to the beginning of the main loop (checking synchronization). If the switches were not changed it just returns to the beginning of the main loop.

- b) Carriage return (value equals 13) - breaks out of the main loop and goes to shutdownbat(). It shuts down the system.

- c) 's' - switches the value of "joyflag". Switching is defined as whatever the variable was, it is now the opposite (i.e., if the variable was true it is changed to false). It controls whether the joysticks are used to control the propulsion subsystem or the BAT's right arm. Control is then returned to the beginning of the main loop.

- d) 'd' - switches the value of "draw". This determines whether or not the operator wants the actuator and thruster levels displayed on the screen. Control is then returned to the beginning of the main loop.

- e) 'r' - resets the control system. It reinitializes the system as it did prior to starting the main loop. The buffer/monitor card and
played on the screen using the values stored by either the master arm or the joysticks. If "draw" is false (the default) then this step is skipped. The values stored are then sent to the buffer/monitor card by transmast().

4) The propulsion subsystem then follows a similar procedure. If "joyflag" is true, the convert() function reads the analog inputs of the joysticks. The joystick offsets are then added and the values are adjusted. The values are stored in an array. If "joyflag: is false then the convert() function reads the backup controls on the ICS. After these values are adjusted they are stored in an array. As before, the variable "draw" is checked. If "draw" is true, the thruster levels are displayed using the values stored. If "draw" is false this step is skipped. The variable "puma" is then checked to see if the propulsion subsystem is on. If "puma" is true, the propulsion system is on and the values stored in the array are transmitted to the propulsion control card using transpuma(). Transpuma() is the analog of transmast(). It insures the propulsion control card is ready and performs all the handshaking necessary to transmit data. It then transmits the data to the propulsion control card. It utilizes the inport() and outport() functions. If "puma" is false (the default), it does not transmit the array values and goes to the next step. The outport function is then called to signal the end of this part of the main loop. The timing test is used to see how long the system is working in the action loop and how long it spends in the option loop described below.

5) The option loop is started by utilizing a screen routine supplied by the compiler. The routine (scrcsts()) reads a character in-
there are no hung joint cards and the propulsion card is ready. Upon a response from the stat() function, the action part of the loop begins. The outport() function is called to start a timing test. This test is to see how long the central controller is in this part of the main loop. The action part of the loop controls the right arm and the propulsion subsystem by converting the control inputs to commands and transmitting them to the appropriate subsystem.

3) The action part of the loop begins with the control of the right arm. The variable "joyflag" is first tested. It can be true or false (it is initially set to true). If "joyflag" is true, the master arm on the ICS controls the BAT's right arm and the two joysticks control the propulsion subsystem. If "joyflag" is false the two joysticks control the BAT's right arm and the backup controls on the ICS control the propulsion subsystem. If the master arm controls the BAT's arm, the central controller first uses the convert() function to read and convert the analog inputs from the master arm. It then uses the adjust() and align() functions to insure that maximum and minimum limits of the pots are not exceeded and that the pot offsets are included. These values, containing the address, data and commands of each joint/actuator are stored. If "joyflag" is false, the convert() function is used to read the analog inputs from the two 3 degrees-of-freedom joysticks. The joysticks' offsets, computed in calibratebat(), are added and the values are then adjusted. The values for the address, data and commands for each joint/actuator are stored. If the variable "draw" is true, the actuator levels are dis-
- the master arm to control the right arm
- the two 3 degrees-of-freedom joysticks to control the propulsion subsystem
- the propulsion system to off
- the helmet gimbal system to control the video subsystem
- the display of the actuator and thruster levels to off.

It initializes the buffer in the central controller for the control of the BAT's right arm. It does this by initializing three arrays for the address, data and command of each of the joint/actuators on the BAT. It sends these arrays to the buffer/monitor card by using the function transmast(). This function is used anytime any information must be passed from the central controller to the buffer/monitor card. It insures that the buffer is not full and that a joint card is not hung up. It performs all handshaking tasks needed to transmit the arrays. Inport() and output() are used throughout the function. After the buffer/monitor card is initialized, the main loop of runbat() is executed.

2) Controlling the main loop is a synchronization function (called synchronize()). It does precisely what its name implies and is tested at the start of each loop. If the system is synchronized it continues; if it is not synchronized it exits the loop and goes to shutdownbat(). After the synchronization test, the status of the buffer/monitor card and the propulsion control card are checked using the stat() function. This function insures that the buffer/monitor card is ready,
FIGURE A.19b  Runbat() Function
FIGURE A.19a  Runbat() Function
FIGURE A.18  Screen() Function
operator to position the joysticks in the center. After waiting for this to be done, it then uses convert() for each degree-of-freedom on each joystick. Convert() takes an analog input to the central controller's analog-to-digital converter and converts it to an integer value. With the joysticks at the center position and the integer value returned by convert(), the central controller computes and stores an offset for each of the joystick's degrees-of-freedom. Control is then returned to MAIN().

C) screen() (see Figure A.18)

This function sets up the video background for the displays created by the central controller. It calls screen routines and functions (such as printf() which are implemented in this version of the compiler. The screen() function is also called by runbat(). Runbat() uses some functions which change the display. After these functions are done, they call screen() to put up the original display. Control is returned to the calling function after the display is on the screen. Screen routines and input/output functions supplied by the compiler are called throughout the BAT system's software. These routines and the screen() function provide the operator with a continuous display of the BAT system operation. The screen routines also provide another method for the operator to interact with the software.

D) runbat() (see Figure A.19)

1) After the initialization and calibration functions are completed and the display is set up, the BAT system is ready for operation. The runbat() function first initializes the minimum and maximum positions of the potentiometers on the controls and configures the initial controls of the ICS. It sets:
FIGURE A.17  Calibrate Bat() Function
FIGURE A.16  Initialize BAT() Function
APPENDIX B

PROGRAM FOR SUPERVISORY CONTROL
SUPERVIS.C

/* This selects the options for supervisory control of the S47's right arm. It calls move(). */

Anthony J. Manganiello Spring 1988

#define THTMIN 0.0000001
#define THT2MIN 1.5089969
#define THT3MIN 0.5235987
#define THT4MIN 0.0600001
#define THT5MIN -1.7453292
#define THT1MAX 3.1415927
#define THT2MAX 2.6179938
#define THT3MAX 3.7524878
#define THT4MAX 3.1415927
#define THT5MAX 1.7453292
#define GAMMA 0.782
#define FI 3.1415927
#define L1 0.4364
#define L2 0.3356
#define AW 0.9310344

#include <math.h>

extern double value[5];

double *impost[5] = {2.2689230, -6, 1.1, 1.8797961, 0, 0};

int intang[5];

static int angle[5][5] = {
    0.0, 0.0, 0.0, 0.0,
    0.0, 0.0, 0.0, 0.0,
    0.0, 0.0, 0.0, 0.0,
    0.0, 0.0, 0.0, 0.0,
    0.0, 0.0, 0.0, 0.0
};

static int count = 1;

int angle[5] = {0.0, 0.0, 0.0, 0.0, 0.0};

#define F1 3.1415927
#define F2 0.4364
#define F3 0.3356
#define F4 0.9310344

define F5 0.782

define FI 3.1415927

define L1 0.4364

define L2 0.3356

define AW 0.9310344

define H1 0.782

define H2 3.1415927

define H3 0.4364

define H4 0.3356

define H5 0.9310344
double second;

FAINT MENU

/* */

scr_menu();

printf("nSUPERVISORY CONTROL\n\nOPTIONS:\n\nm - move to stored position\n\nx - store a point for concatenation\n\n(repeat max # of points - 5\n\n)y - clear all stored points\n\n2 - move thru all stored points\n\n")

printf("space bar - returns to calling function\n\nwithout selecting an option\n\n")

/* */

/* */

/* */

readyn = scr_menu();
while(readyn != 0)

readyn = (scr_menu() != 20)?

/* */

/* MAIN FINISH POINT; UNKNOWN STARTING POINT */

/* */

readyn = scr_menu();
while(readyn != 0)

readyn = (scr_menu() != 20)?

/* */

/* */

/* */

/* */

/* */
; move intang, func, osk, addr, idxld; 
scr circ();
print("\nTRAJECTORY COMPLETED\n");
return;
else{

/*******************************************************************************/
.CASCADE

/******************************************************************************/
; STORE POINTS TO BUILD A TRAJECTORY

*******************************************************************************/
if (ready == 'Y') {
  for (i=3, i=11, i++;
    test = convert(test), test2 = test, test2 = OFF, test = 1.2,
    test = Adjust(test, test);
    test = align(test, test2); & OFF;
    angle[count] = (test);
    count = count + 1;
    scr circ();
    print("POINT STORED\n");
    return;
}

/*******************************************************************************/
.CASCADE

*******************************************************************************/
CLEAN STORED POINTS

*******************************************************************************/
MOVE THRU ALL THE STORED POINTS

if (readyy == 'z') {
    count = count;
    for (j = 0; i = 0; i++, j++) {
        tdat = (convert iht[i][iht[i]] & 0xFFFF) + 102
    }
    tdat = adjust(tdat, i);
    tdat = align(tdat, i) & 0xFFFF;
    angle[0][j] = tdat;
    angle[count][j] = tdat;
}
for (i = 0; i < count; i++) {
    for (j = 0; j < count; j++) {
        intang[j] = angle[i][j];
        n,t = 1 + 1;
        nintang[j] = angle[next][j];
    }
}

CONVERT THE FINAL POSITION TO RADIANS

continue WITH THEIA'S CODE
TH14 = \text{-}0.5 \times \sin(\theta + \text{TH3})

\text{TH15} = \text{-}1.0 \times \cos(\theta + \text{TH3}) - (\text{TH4} \times \text{TH5}) \times \text{TH3}

* * *

/*

/* CHECK EACH THETA FOR LIMITS */

* *

* *

if (TH14 < TH1MAX)

TH14 = TH1MAX

else

1 + (TH14 \times TH1MIN)

TH14 = TH1MIN

1 *


if (TH15 < TH1MAX)

TH15 = TH1MAX

else

1 + (TH15 \times TH1MIN)

TH15 = TH1MIN

*)

if (TH24 < TH2MAX)

TH24 = TH2MAX

else

1 + (TH24 \times TH2MIN)

TH24 = TH2MIN

*)

if (TH34 < TH3MAX)

TH34 = TH3MAX

else

1 + (TH34 \times TH3MIN)

TH34 = TH3MIN

*)

if (TH44 < TH4MAX)

TH44 = TH4MAX

else

1 + (TH44 \times TH4MIN)

TH44 = TH4MIN

*)

if (TH54 < TH5MAX)

TH54 = TH5MAX

else

1 + (TH54 \times TH5MIN)

TH54 = TH5MIN

*)

/*
if (THTJF > PI)
    THTJF = THTJF;
GAMMAJ = sqrt((L1+L1)+(L2+L2)-(2*L1*L2*cos(THTJF)));

    if (GAMMAF > GAMMAJ)
        GAMMAF = GAMMAJ;
    arg1 = (L2*sin(THTJF))/GAMMAJ;
    PHIJ = arcsin(arg1);
    PSIJ = (PI/2)-(THTJF+PHIJ);
    IF = GAMMAJ*sin(PSIJ);
    RF = sqrt((GAMMAJ*GAMMAJ)-(2*IF*IF));
else:
    if (THTJF > PI)
        THTJF = THTJF;
GAMMAJ = sqrt((L1+L1)+(L2+L2)-(2*L1*L2*cos(THTJF)));

    if (GAMMAJ > GAMMAF)
        GAMMAJ = GAMMAF;
    arg2 = (L2*sin((2*PI)-THTJF))/GAMMAJ;
    PHIJ = arcsin(arg2);
    PSIJ = (PI/2)-(THTJF+PHIJ);
    IF = GAMMAJ*sin(PSIJ);
    RF = sqrt((GAMMAJ*GAMMAJ)-(2*IF*IF));
else:
    if (THTJF == PI)
        THTJF = THTJF;
GAMMAJ = L1+L2;
PSI = (PI/2)-THTJF;
    IF = GAMMAJ*sin(PSI);
    RF = GAMMAJ*cos(PSI);
    else:
        printf("ERROR IN ENTERING FINAL VALUE OF T
HEADING!");

if (GAMMAJ < 0)
    GAMMAJ = -GAMMAJ;
    THTJF = PI-THTJF;
    PHIJ = PHIJ;
    PSIJ = PSIJ;
    IF = IF;
    RF = RF;
    printf("ERROR IN ENTERING INITIAL VALUE OF T
HEADING!");

if (GAMMAJ > 0)
    GAMMAJ = -GAMMAJ;
    THTJF = PI-THTJF;
    PHIJ = PHIJ;
    PSIJ = PSIJ;
    IF = IF;
    RF = RF;
    printf("ERROR IN ENTERING INITIAL VALUE OF T
HEADING!");

if (GAMMAJ == 0)
    GAMMAJ = 0.00001;
    THTJF = PI-THTJF;
    PHIJ = PHIJ;
    PSIJ = PSIJ;
    IF = IF;
    RF = RF;
    printf("ERROR IN ENTERING INITIAL VALUE OF T
HEADING!");

if (THTJF < PI) && (THTJF > 0)
    GAMMAJ = GAMMAJ;
    THTJF = PI-THTJF;
    PHIJ = PHIJ;
    PSIJ = PSIJ;
    IF = IF;
    RF = RF;
    printf("ERROR IN ENTERING INITIAL VALUE OF T
HEADING!");

if (THTJF < 0) && (THTJF > PI)
    GAMMAJ = GAMMAJ;
    THTJF = PI-THTJF;
    PHIJ = PHIJ;
    PSIJ = PSIJ;
    IF = IF;
    RF = RF;
    printf("ERROR IN ENTERING INITIAL VALUE OF T
HEADING!");

if (GAMMAJ < 0) && (THTJF < PI) && (THTJF > 0)
    GAMMAJ = GAMMAJ;
    THTJF = PI-THTJF;
    PHIJ = PHIJ;
    PSIJ = PSIJ;
    IF = IF;
    RF = RF;
    printf("ERROR IN ENTERING INITIAL VALUE OF T
HEADING!");

if (GAMMAJ < 0) && (THTJF < 0) && (THTJF > PI)
    GAMMAJ = GAMMAJ;
    THTJF = PI-THTJF;
    PHIJ = PHIJ;
    PSIJ = PSIJ;
    IF = IF;
    RF = RF;
    printf("ERROR IN ENTERING INITIAL VALUE OF T
HEADING!");
funcall41 = THOST;
move inteng,finpos,rad,radm1;
)
end_routine

print("IN TRAJECTORY COMPLETED!");
return;

else

print("NO SELECTION MADE"); // Press space bar to exit ">
return;

}

)
The actual driving of the BAT's flight is under computer control.

Anthony J. Manganiello, Sprint
I. 

```c
THT11 = value[0];
THT21 = value[1];
THT31 = value[2];
THT41 = value[3];
THT51 = value[4];

// Compute theta 4 & theta 5

THT41 = -S.PSI1 + L.PSI1;
THT51 = S.PSI1 + R.PSI1; (LAW.) * PSI1;

// Check each theta for limits

THT11 = THT11 + THT11;
THT11 = MIN;
THT11 = MAX;
```

-156-


```c
56  int THT41, THTSMAX;
57  THT41 = THT4MIN;
58  else {
59    if (THT41 > THTSMAX)
60      THT41 = THTSMAX;
61    else if (THT41 < THTSMIN)
62      THT41 = THTSMIN;
63  }
64
65  /**************************************************************************/
66  /*
67  */
68  /* COMPUTE INITIAL POSITION IN CYLINDRICAL COORDINATES */
69  */
70  /**************************************************************************/
71  if (THT71 < PI1) {
72    a = 0;
73    THT1 = THT41;
74    GAMMA1 = sqrt((L1+L1)+(L2+L2)-(2*L1*L2*cos(THT31)));
75    if (GAMMA1 > GAMMAX)
76      GAMMA1 = GAMMAX;
77    arg1 = (L2+sin(THT31))/GAMMA1;
78    PHI1 = asin(arg1);
79    Z1 = GAMMA1*sin(PHI1);
80    R1 = sqrt((GAMMA1*GAMMA1)-(Z1*Z1));
81    else {
82      a = 1;
83      THT1 = THT41;
84      GAMMA1 = sqrt((L1+L1)+(L2+L2)-(2*L1*L2*cos((L1)*THT31)));
85      if (GAMMA1 > GAMMAX)
86        GAMMA1 = GAMMAX;
87      arg1 = (L2+sin(THT31))/GAMMA1;
88      PHI1 = asin(arg1);
89      Z1 = GAMMA1*sin(PHI1);
90      R1 = sqrt((GAMMA1*GAMMA1)-(Z1*Z1));
91      else {
92        if (a == 1) {
93          THT1 = THT41;
94          GAMMA1 = sqrt((L1+L1)+(L2+L2)-(2*L1*L2*cos(THT31)));
95        }
96        else {
97          THT1 = THT41;
98          GAMMA1 = sqrt((L1+L1)+(L2+L2)-(2*L1*L2*cos(THT31)));
99        }
100       }
```
131   PHI1 = PHI1 - 2*PI;
132   RI = GAMMA + SIN(PHI1);
133   R1 = GAMMA + COS(PHI1);
134   if (R1 > 0.0)
135     print("ERROR IN ENTERING INITIAL VALUE OF
136           PHI1")
137   end
138
139   \*
140   \* INPUT THE FINAL POSITION IN CYLINDRICAL COORDINATES \*
141   \*
142
143   \*
144   \* COMPUTE CONSTANTS \*
145   \*
146
compute delta theta and total time
double PH1, PH2, PH3, PH4, PH5;
double DELTHET, DELTHET2;
double PHI4, PHI5;
double sin4, sin5, sin6, sin7;  // for PH4 and PH5
double cos4, cos5, cos6, cos7;  // for PH4 and PH5
int duration1 = 10, duration2 = 10, duration3 = 10,
int duration4 = 10, duration5 = 10;

double sin6 = 0, cos6 = 1;
while duration1 > 0 {  // PH1
  duration1--;  // PH1
  t = dt;
  sin4 = sin6;
  cos4 = cos6;
  DELTHET = 0.000001;
  DELTHET2 = 0.000001;
  PHI4 = 2*PI;  // PHI4
  PHI5 = 2*PI;  // PHI5
  int i = 0
  while i < duration2 {  // PH2
    PHI4 = PHI4 - DELTHET;  // PHI4
    PHI5 = PHI5 - DELTHET;  // PHI5
    i++
  }
  while i < duration3 {  // PH3
    PHI4 = PHI4 - DELTHET2;  // PHI4
    PHI5 = PHI5 - DELTHET2;  // PHI5
    i++
  }
  while i < duration4 {  // PH4
    PHI4 = PHI4 - DELTHET;  // PHI4
    PHI5 = PHI5 - DELTHET;  // PHI5
    i++
  }
  while i < duration5 {  // PH5
    PHI4 = PHI4 - DELTHET2;  // PHI4
    PHI5 = PHI5 - DELTHET2;  // PHI5
    i++
  }
}
I = (cos(\theta - \phi)) / \sigma \tan(\theta - \phi)

if (THT < THT) THT = THT + DELTHT + 1

else:
    if (THT > THT)
        THT = THT - DELTHT - 1

    else:
        if (THT < .01)
            THT = .01
        else:
            if (THT > 1.5533333 (THT < NINE))
                THT = 1.55333
            else:
                if (THT < 1.6052977) THT = 1.6052977

    y = \sqrt{tan(THT)}

else:
    if (THT < .01)
        y = 0
    else:
        if (THT > 1.6052977 (THT < NINE))
            THT = 1.6052977
        else:
            if (THT < 1.6052977) THT = 1.6052977

y = y1;

if (THT < THT)
    THT = 1.5533333

else:
    THT = 1.6052977
DECLTH1 = TH1 + TH1;
TH1 = TH1 + (DECLTH1 * (1/TT)) - ((1/(2*PI)) * sin((2*PI) + (1/TT)));
else;
DECLTH1 = TH1 - TH1;
TH1 = TH1 + (DECLTH1 * (1/TT)) - ((1/(2*PI)) * sin((2*PI) + (1/TT)));

if (TH11 < TH11);
DECLTH1 = TH11 - TH11;
TH11 = TH11 + (DECLTH1 * (1/TT)) - ((1/(2*PI)) * sin((2*PI) + (1/TT)));
else;
DECLTH1 = TH11 - TH11;
TH11 = TH11 + (DECLTH1 * (1/TT)) - ((1/(2*PI)) * sin((2*PI) + (1/TT)));

// ---------------------------

j27 = 0;
CHECK EACH THETA FOR LIMITS
j29 = -1;

// ---------------------------

if (TH22 < TH22)
TH22 = TH22;
else;
TH22 = TH22;

if (TH22 < TH22)
TH22 = TH22;
else;
TH22 = TH22;

if (TH22 < TH22)
TH22 = TH22;
else;
TH22 = TH22;

if (TH22 < TH22)
TH22 = TH22;
else;
TH22 = TH22;
THETA = THRESH1;
easel
11 (THETA < THRESH1)
THETA = THRESH1;

/*

* COMPUTE MOTOR ANGLES FROM THETA 4 & 5
*
*
*
*/

/*
*
*
*
*/

/*

* CONVERT ANGLES TO INTEGERS
*
*
*
*/

/*

* /
mean[1] = (mean[1] + FOR loss in EIP);
add[1] = (add[1] + EIP);

return;

err慰定();

printf(" LOST SYNCHRONIZATION in ");
return;

c = intang[1];

 TH11

double TH11, TH21, TH31, PSI1, PSI2, TH32;

TH11

TH21

TH31

TH32
if (intang[21] <= 287a) & (intang[21] == 2148)
286
intang[21] = 287a;
286
if (intang[21] == 192a) & (intang[21] == 2047)
287
intang[21] = 192a;
288
if (intang[21] == 287a) & (intang[21] <= 4098)
289
TH13i = 5.235587 + ((intang[21] - 287a) * .0010279 - 17);
290
if (intang[21] <= 400) & (intang[21] <= 192a)
291
TH13i = 3.7524573 - ((192a - intang[21]) * .0010279 - 71);
292
/*-----------------------------------------------*/
293  PS14i
294
/*-----------------------------------------------*/
295
if (intang[01] >= 217a) & (intang[01] <= 2048)
296
intang[01] = 217a;
297
if (intang[01] >= 2047) & (intang[01] <= 1732)
298
intang[01] = 1732;
299
if (intang[01] <= 170a) & (intang[02] <= 4098)
300
PS14i = ((intang[01] - 217a) * .001749247) - 4.4757998;
301
if (intang[01] <= 000) & (intang[01] <= 1732)
302
PS14i = 1.524293402 - ((1732 - intang[01]) * .001749247);
303
/*-----------------------------------------------*/
304  PS15i
305
/*-----------------------------------------------*/
306
307
308
309
310
311
312
PS15i = 1.156749782 - (192a - intang[11]) * .0017
313
d100 = intang[11];
314
// ----------------------------------------------
double val1();

double val1();

double THT1, THT2, THT3, PSI4, PSI5;

THT1 = val1();
THT2 = val1();
THT3 = val1();
PSI4 = val1();
PSI5 = val1();

/***************************************************************************/

THT1

/***************************************************************************/

if (THT1 > 3.141592654)
THT1 = 3.141592654;

if (THT1 < 0.0)
THT1 = 0.0;

if (THT1 <= 0.000333333 & (THT1 < 1.5779999999999999))
cerrang[11] = 2.2553 * (THT1 * 913.8078232);

if (cerrang[11] > 40.0)
cerrang[11] = 40.0;

if (THT1 <= 1.5779999999999999 & (THT1 < 3.141592654))
cerrang[11] = (THT1 - 1.5779999999999999) * 913.8078232;

/***************************************************************************/

THT2

/***************************************************************************/

if (THT2 > 1.500000015)
THT2 = 1.500000015;

if (THT2 > 2.517999999)
THT2 = 2.517999999;

if (THT2 <= 2.517999999 & (THT2 < 1.577999999))
cerrang[11] = (1.157799999 - THT2) * 1272.602214;

if (THT2 <= 1.577999999 & (THT2 < 0.617999999))
cerrang[11] = 4.000 - THT2 - 1.577999999 * 1272.602214;
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>errang[1] = 2.400 + (-1.741 + 1.8276) * 772.78542</td>
</tr>
<tr>
<td>2.</td>
<td>errang[21] = 40.52</td>
</tr>
<tr>
<td>3.</td>
<td>errang[221] = 40.52</td>
</tr>
<tr>
<td>4.</td>
<td>errang[311] = 40.52</td>
</tr>
<tr>
<td>5.</td>
<td>errang[321] = 40.52</td>
</tr>
<tr>
<td>6.</td>
<td>errang[331] = 40.52</td>
</tr>
<tr>
<td>7.</td>
<td>errang[332] = 40.52</td>
</tr>
<tr>
<td>8.</td>
<td>errang[333] = 40.52</td>
</tr>
<tr>
<td>9.</td>
<td>errang[334] = 40.52</td>
</tr>
<tr>
<td>10.</td>
<td>errang[335] = 40.52</td>
</tr>
<tr>
<td>11.</td>
<td>errang[336] = 40.52</td>
</tr>
<tr>
<td>12.</td>
<td>errang[337] = 40.52</td>
</tr>
<tr>
<td>13.</td>
<td>errang[338] = 40.52</td>
</tr>
<tr>
<td>14.</td>
<td>errang[339] = 40.52</td>
</tr>
<tr>
<td>15.</td>
<td>errang[340] = 40.52</td>
</tr>
<tr>
<td>16.</td>
<td>errang[341] = 40.52</td>
</tr>
<tr>
<td>17.</td>
<td>errang[342] = 40.52</td>
</tr>
<tr>
<td>18.</td>
<td>errang[343] = 40.52</td>
</tr>
<tr>
<td>19.</td>
<td>errang[344] = 40.52</td>
</tr>
<tr>
<td>20.</td>
<td>errang[345] = 40.52</td>
</tr>
<tr>
<td>21.</td>
<td>errang[346] = 40.52</td>
</tr>
<tr>
<td>22.</td>
<td>errang[347] = 40.52</td>
</tr>
<tr>
<td>23.</td>
<td>errang[348] = 40.52</td>
</tr>
<tr>
<td>24.</td>
<td>errang[349] = 40.52</td>
</tr>
<tr>
<td>25.</td>
<td>errang[350] = 40.52</td>
</tr>
<tr>
<td>26.</td>
<td>errang[351] = 40.52</td>
</tr>
<tr>
<td>27.</td>
<td>errang[352] = 40.52</td>
</tr>
<tr>
<td>28.</td>
<td>errang[353] = 40.52</td>
</tr>
<tr>
<td>29.</td>
<td>errang[354] = 40.52</td>
</tr>
<tr>
<td>30.</td>
<td>errang[355] = 40.52</td>
</tr>
<tr>
<td>31.</td>
<td>errang[356] = 40.52</td>
</tr>
<tr>
<td>32.</td>
<td>errang[357] = 40.52</td>
</tr>
<tr>
<td>33.</td>
<td>errang[358] = 40.52</td>
</tr>
<tr>
<td>34.</td>
<td>errang[359] = 40.52</td>
</tr>
<tr>
<td>35.</td>
<td>errang[360] = 40.52</td>
</tr>
<tr>
<td>36.</td>
<td>errang[361] = 40.52</td>
</tr>
<tr>
<td>37.</td>
<td>errang[362] = 40.52</td>
</tr>
<tr>
<td>38.</td>
<td>errang[363] = 40.52</td>
</tr>
<tr>
<td>39.</td>
<td>errang[364] = 40.52</td>
</tr>
<tr>
<td>40.</td>
<td>errang[365] = 40.52</td>
</tr>
<tr>
<td>41.</td>
<td>errang[366] = 40.52</td>
</tr>
<tr>
<td>42.</td>
<td>errang[367] = 40.52</td>
</tr>
<tr>
<td>43.</td>
<td>errang[368] = 40.52</td>
</tr>
<tr>
<td>44.</td>
<td>errang[369] = 40.52</td>
</tr>
<tr>
<td>45.</td>
<td>errang[370] = 40.52</td>
</tr>
<tr>
<td>46.</td>
<td>errang[371] = 40.52</td>
</tr>
<tr>
<td>47.</td>
<td>errang[372] = 40.52</td>
</tr>
<tr>
<td>48.</td>
<td>errang[373] = 40.52</td>
</tr>
<tr>
<td>49.</td>
<td>errang[374] = 40.52</td>
</tr>
<tr>
<td>50.</td>
<td>errang[375] = 40.52</td>
</tr>
<tr>
<td>51.</td>
<td>errang[376] = 40.52</td>
</tr>
<tr>
<td>52.</td>
<td>errang[377] = 40.52</td>
</tr>
<tr>
<td>53.</td>
<td>errang[378] = 40.52</td>
</tr>
<tr>
<td>54.</td>
<td>errang[379] = 40.52</td>
</tr>
<tr>
<td>55.</td>
<td>errang[380] = 40.52</td>
</tr>
<tr>
<td>56.</td>
<td>errang[381] = 40.52</td>
</tr>
<tr>
<td>57.</td>
<td>errang[382] = 40.52</td>
</tr>
<tr>
<td>58.</td>
<td>errang[383] = 40.52</td>
</tr>
<tr>
<td>59.</td>
<td>errang[384] = 40.52</td>
</tr>
<tr>
<td>60.</td>
<td>errang[385] = 40.52</td>
</tr>
<tr>
<td>61.</td>
<td>errang[386] = 40.52</td>
</tr>
<tr>
<td>62.</td>
<td>errang[387] = 40.52</td>
</tr>
<tr>
<td>63.</td>
<td>errang[388] = 40.52</td>
</tr>
<tr>
<td>64.</td>
<td>errang[389] = 40.52</td>
</tr>
<tr>
<td>65.</td>
<td>errang[390] = 40.52</td>
</tr>
<tr>
<td>66.</td>
<td>errang[391] = 40.52</td>
</tr>
<tr>
<td>67.</td>
<td>errang[392] = 40.52</td>
</tr>
<tr>
<td>68.</td>
<td>errang[393] = 40.52</td>
</tr>
<tr>
<td>69.</td>
<td>errang[394] = 40.52</td>
</tr>
<tr>
<td>70.</td>
<td>errang[395] = 40.52</td>
</tr>
<tr>
<td>71.</td>
<td>errang[396] = 40.52</td>
</tr>
<tr>
<td>72.</td>
<td>errang[397] = 40.52</td>
</tr>
<tr>
<td>73.</td>
<td>errang[398] = 40.52</td>
</tr>
<tr>
<td>74.</td>
<td>errang[399] = 40.52</td>
</tr>
<tr>
<td>75.</td>
<td>errang[400] = 40.52</td>
</tr>
</tbody>
</table>

**Note:** The above table represents the steps involved in a calculation process, where each step increments the errang value by a factor of 40.52.
double arcsine

double a;

double s, m, ang, l, l1, ang1, ang2, ang3, t1t;

if (fabs(a) >= 1.)

printf("ERROR IN COMPUTING ARGUMENT FOR ARCSINE r")

ang = 0.0;

else:

if (fabs(a) <= 0.707107)

a = a;

ang1 = ang2 = ((a*s-1.0)*s+1.0)*s;

ang3 = ang2 + (1.0-s)*s;

else:

ang = tan(0.14159265);

else:

w = 0;

th = 0;

y = 5*((y/w)-w);

while (fabs(y)) > 0.00000001

w = w+y;

th = y;

y = 5*(y/w)-w;

k = w;

ang2 = k + ((m*s)/k) + l1*t - l1;

ang3 = ang2 + (l1*t)*s + l1;

ang1 = ang3 + (m*s)*s + th;

ang = th + l1*t979d-ang1;

return(ang1);

}
double arctan(double x)

double s;

double tz, tz1, az, b1, b2, ang;

tz = (x > 0) ? 1 : -1;

g = fabs(x);

cz = 0;

if (g > 1) {
    cz = 1;
    s = 1.0;
}

s = g * s;

b1 = 1.00238929 * az - (161.85) * az + 0.429038 * az;

b2 = f(1.00238929 * az + 0.035025 * az - 142.09) * az + 0.0000082;

ang = s * (1.00238929 * az + 0.035025 * az - 142.09);

return ang;