

**FLEXIBLE MANUFACTURING OF RIBBED ROOFING PANELS:  
PROCESS LAYOUT AND RIB ORIENTATION**

by

Susan D. Ward

S.B., Mechanical Engineering  
Massachusetts Institute of Technology  
1992


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 1994

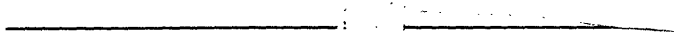
1994 Massachusetts Institute of Technology  
All rights reserved

Signature of Author



Department of Mechanical Engineering  
May, 1994

Certified by

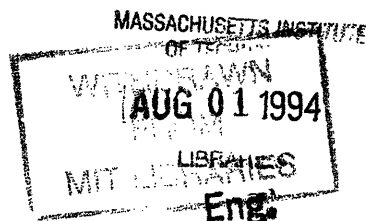


Dr. Andre Snaoron  
Thesis Supervisor

Accepted by



Professor Ain A. Sonin  
Chairman, Graduate Thesis Committee  
Department of Mechanical Engineering





**FLEXIBLE MANUFACTURING OF RIBBED ROOFING PANELS:  
PROCESS LAYOUT AND RIB ORIENTATION**

by  
**SUSAN DIANE WARD**

Submitted to the Department of Mechanical Engineering  
on May 6, 1994 in partial fulfillment of the  
requirements for the Degree of Master of Science in  
Mechanical Engineering

**ABSTRACT**

Traditional truss and rafter methods of roof construction struggle to provide the flexibility of design currently demanded by consumers. In doing so, they also sacrifice livable attic space and often encounter difficulties with insulation installation and ventilation. A panelized roofing system developed at MIT promises to relieve these problems, as well as to replace old-growth materials with composites and to reduce on-site assembly time, material usage, and labor costs. However, before these benefits can be realized, the system has to be coupled with a cost-effective manufacturing system.

Described in this document is the development of a complete flexible panel assembly layout capable of up to 175 panels/day throughput with minimal material and energy waste. Since rib orientation was central to rib preparation and assembly and required one of the most critical and unique applications of technology, a prototype of the station was designed, built, and tested.

With this prototype, continuous ranges of complex geometries from 2" to 24' long were proved to be orientable without extensive vision or sensor systems, despite significant initial mis-rotation and displacement. The flexible algorithm generated to determine the necessary manipulation steps was based on rib geometry and the final desired configuration alone.

These results showed that the methodology applied by the rib orienting prototype was an affective means of meeting flexible orientation requirements for the panel assembly system as well as other related applications.

Thesis Supervisor: Dr. Andre Sharon

Title: Executive Officer of The Manufacturing Institute at MIT.



## ACKNOWLEDGMENTS

So many people,  
So many faces I care about,  
And no way to tell them  
How much I appreciate them.

In particular, I would like to express my thanks to those people which made this project possible and much more fun in the process:

To my parents for their unfailing, invaluable support in every aspect of my life along the way.

To Andre Sharon, who took me on despite the "risk" and gave considerate and valuable advice and encouragement as to my drawing, design, writing, and vacationing.

To all my office-mates for their tolerance, conversation, advice and humor, and especially to Dave Phillips who was an excellent and thorough partner in line research and general curiosity.

To Fred, the ever helpful, patient, reliable and smiling one man rescue team who made my fabrication work in and out of the shop (except occasionally the kitchen), much more efficient, affective, and safe.

To everyone in the Building Technology Group who contributed endless smiles, advice and information.

To Tiny and Company, (Norm and Bob) for their continuing friendliness and helpfulness when it came to cutting OSB into smaller and smaller pieces.

To Phil Marshall and Glen Dlugosz of Servo Systems, who answered every last question and need with wonderful service and consideration far above and beyond the call of duty.

To all those who cheerfully helped to untangle my understanding of the motor wiring and programming process, but most of all, Dave Parish of Omnitech Robotics, and Eric Schrier.

To and Thomas Lozano-Perez who gave their interest, advice, and great sources information.

To of Kevin Kordis of Accel Linear, for interested consultation and advice on motors, conveyors, and inexpensive machine shops.

To every last person I know or who is involved in my extra-curricular playing, for joining me in running around, laughing, chatting, and enjoying all the other fun things in life aside from this work.



## **TABLE OF CONTENTS**

|                                |               |
|--------------------------------|---------------|
| <b>ABSTRACT</b> .....          | <b>Pg. 3</b>  |
| <b>ACKNOWLEDGMENTS</b> .....   | <b>Pg. 5</b>  |
| <b>TABLE OF CONTENTS</b> ..... | <b>Pg. 7</b>  |
| <b>LIST OF FIGURES</b> .....   | <b>Pg. 10</b> |

### **Chapter 1: INTRODUCTION**

|   |               |
|---|---------------|
| <b>1.1 Housing Research Trends: History</b> | <b>Pg. 15</b> |
| <b>1.2 The MIT Panelized Roofing System</b> | <b>Pg. 19</b> |
| 1.2.1 Evolution                             |               |
| 1.2.2 System and Panel Design               |               |
| 1.2.3 Expectations                          |               |

### **Chapter 2: ROOF PANEL MANUFACTURE**

|  |               |
|--|---------------|
| <b>2.1 Introduction</b>                                    | <b>Pg. 27</b> |
| <b>2.2 Redesign for Manufacture</b>                        | <b>Pg. 27</b> |
| <b>2.3 Individual Process Considerations</b>               | <b>Pg. 31</b> |
| 2.3.1 OSB Supply Cutting, Joining and Conveying            |               |
| 2.3.2 Assembly Motion Tradeoffs                            |               |
| 2.3.3 Insulation Methods                                   |               |
| 2.3.4 End Cap Attachment                                   |               |
| 2.3.5 Rib Orientation and Gluing                           |               |
| <b>2.4 Complete Conceptual Line Layout</b>                 | <b>Pg. 40</b> |
| <b>2.5 Specific Process and Machinery Design Evolution</b> | <b>Pg. 43</b> |

# TABLE OF CONTENTS

Cont.

## Chapter 3: RIB ORIENTATION, GLUING AND TRANSPORTATION: SYSTEM DESIGN

|   |        |
|---|--------|
| 3.1 Original Task Description                               | Pg. 45 |
| 3.2 Partial Concepts and Design Evolution                   | Pg. 46 |
| 3.3 System Cost, Complexity and Motion Constraints          | Pg. 50 |
| 3.4 Rib Geometry Handling Mechanisms, Methods and Materials | Pg. 52 |
| 3.5 Final Concept   | Pg. 55 |
| 3.6 Orienter Prototype                                      | Pg. 62 |
| 3.6.1 Introduction  |        |
| 3.6.2 Fabrication   |        |

## Chapter 4: RIB ORIENTER: CONTROL

|   |        |
|---|--------|
| 4.1 Prototype Control System Setup                      | Pg. 67 |
| 4.2 Rib Orientation Approach: Evolution                 | Pg. 69 |
| 4.3 Rib Orientation Approach: Detailed Task Description | Pg. 73 |
| 4.3.1 Conceptualization and Organization                |        |
| 4.3.2 Effects of Geometry Variations                    |        |
| 4.3.3 Ranges and Restrictions of Geometry Variations    |        |
| 4.4 Control Code  | Pg. 81 |
| 4.4.1 Development and Basic Organization                |        |
| 4.4.2 Program Flow and User Interface                   |        |
| 4.5 Orienting Strategies                                | Pg. 86 |
| 4.5.1 Universal Preparatory Motions and Assignments     |        |
| 4.5.2 Symmetric and Virtually Symmetric Ribs            |        |
| 4.5.3 Non-Symmetric Ribs                                |        |



# **TABLE OF CONTENTS**

Cont.

## **Chapter 5: DISCUSSION AND CONCLUSION**

|   |                |
|---|----------------|
| <b>5.1 Design Notes</b>                             | <b>Pg. 105</b> |
| <b>5.1.1 Degrees of Freedom</b>                     |                |
| <b>5.1.2 Total Estimated Task Completion Times</b>  |                |
| <b>5.2 Future Work</b>                              | <b>Pg. 108</b> |
| <b>5.2.1 Orientation Completion/Success Sensing</b> |                |
| <b>5.2.2 Testing</b>                                |                |
| <b>5.2.3 Orientation Algorithms</b>                 |                |
| <b>5.3 Conclusion</b>                               | <b>Pg. 113</b> |
| <b>List of References</b>                           | <b>Pg. 115</b> |

## **APPENDICES**

|  |                |
|--|----------------|
| <b>Appendix A: Alternative Manufacturing Line Layouts</b>                          | <b>Pg. 117</b> |
| <b>Appendix B: Alternative Rib Orienting Designs</b>                               | <b>Pg. 123</b> |
| <b>Appendix C: Motor Force and Velocity Requirement Calculations</b>               | <b>Pg. 125</b> |
| <b>Appendix D: Orienter Prototype Parts List</b>                                   | <b>Pg. 127</b> |
| <b>Appendix E: Orienter Prototype Assembly and Fabrication Drawings</b>            | <b>Pg. 116</b> |
| <b>Appendix F: Initial Control Programs</b>  | <b>Pg. 147</b> |
| <b>Appendix G: Documented Control Code</b>   | <b>Pg. 149</b> |
| <b>Appendix H: Rib Angle Assignments</b>   | <b>Pg. 203</b> |
| <b>Appendix I: Calculation of Required Orientation Block Positions</b>             | <b>Pg. 205</b> |
| <b>Appendix J: Mis-rotation Capabilities: Excel Spreadsheet</b>                    | <b>Pg. 209</b> |
| <b>Appendix K: Demonstration Number Four Orientation<br/>Algorithm Explanation</b> | <b>Pg. 213</b> |

## **LIST OF FIGURES**

### **Chapter 1: INTRODUCTION**

|  |               |
|--|---------------|
| <b>Figure 1.21: Panelized Roofing System Components.</b>             | <b>Pg. 21</b> |
| <b>Figure 1.22: Panel Components.</b>                                | <b>Pg. 22</b> |
| <b>Figure 1.23: Standard Roof Design and Resulting Panel Shapes.</b> | <b>Pg. 23</b> |
| <b>Figure 1.24: Panel End Geometries as a Result of Roof Pitch.</b>  | <b>Pg. 24</b> |

### **Chapter 2: ROOF PANEL MANUFACTURE**

|   |               |
|---|---------------|
| <b>Figure 2.21: Illustration of Reinforcement Strips and Replacement Strapping.</b>         | <b>Pg. 29</b> |
| <b>Figure 2.22: Illustration of OSB Blocking and Polyurethane Foam Block Replacement.</b>   | <b>Pg. 31</b> |
| <b>Figure 2.31: Illustration of Waste Potential for Panels with Extreme End Geometries.</b> | <b>Pg. 32</b> |
| <b>Figure 2.32: Illustration of Face Shuffling Conveyor.</b>                                | <b>Pg. 34</b> |
| <b>Figure 2.41: Final Panel Assembly Process Layout.</b>                                    | <b>Pg. 42</b> |
| <b>Figure 2.51: Panel Assembly Layout Sub-Section for Development.</b>                      | <b>Pg. 44</b> |

### **Chapter 3: RIB ORIENTATION, GLUING AND TRANSPORTATION SYSTEM DESIGN**

|  |               |
|--|---------------|
| <b>Figure 3.21: Preliminary Orienter Design Concept.</b>                               | <b>Pg. 47</b> |
| <b>Figure 3.22: Second Orienter Design Concept.</b>                                    | <b>Pg. 48</b> |
| <b>Figure 3.41: Range of Possible Rib Geometries.</b>                                  | <b>Pg. 52</b> |
| <b>Figure 3.42: Illustration of Rib Mis-Orientation.</b>                               | <b>Pg. 53</b> |
| <b>Figure 3.43: Rotation of Rib Geometries From Their Unstable Vertical Positions.</b> | <b>Pg. 53</b> |

## **LIST OF FIGURES**

Cont.

|  |               |
|--|---------------|
| <b>Figure 3.51: First Rib Orientation Step.</b>                      | <b>Pg. 56</b> |
| <b>Figure 3.52: Second Rib Orientation Step.</b>                     | <b>Pg. 57</b> |
| <b>Figure 3.53: Third Rib Orientation Step.</b>                      | <b>Pg. 58</b> |
| <b>Figure 3.54: Fourth Rib Orientation Step.</b>                     | <b>Pg. 59</b> |
| <b>Figure 3.55: Fifth Rib Orientation Step.</b>                      | <b>Pg. 60</b> |
| <b>Figure 3.56: Sixth Rib Orientation Step.</b>                      | <b>Pg. 61</b> |
| <b>Figure 3.61: Hand-Driven Proof-of-Concept Orienter Prototype.</b> | <b>Pg. 63</b> |
| <b>Figure 3.62: Photograph of the Working Orienter Prototype.</b>    | <b>Pg. 66</b> |

### **Chapter 4: RIB ORIENTER: CONTROL**

|  |               |
|--|---------------|
| <b>Figure 4.11: System Schematic: One of Two Identical Halves.</b>   | <b>Pg. 68</b> |
| <b>Figure 4.31: Conceptual Organization of Rib Types.</b>  | <b>Pg. 74</b> |
| <b>Figure 4.32: Illustration of Symmetric Rib Created by Metal Strapping.</b>  | <b>Pg. 76</b> |
| <b>Figure 4.33: Effects on Panel End Geometry from Merging<br/>Different Roof Pitches.</b>                                     | <b>Pg. 77</b> |
| <b>Figure 4.34: Coordination of Roof Pitch, Ridge Beam Shape,<br/>Available Living Space, and Material Consumption.</b>        | <b>Pg. 79</b> |
| <b>Figure 4.41: Final Code Flow and User Interface.</b>  | <b>Pg. 85</b> |
| <b>Figure 4.51: Illustration of Orientation Preparatory Motion.</b>  | <b>Pg. 86</b> |
| <b>Figure 4.52: Illustration of Arm Assignments for Ribs With Identical Geometries<br/>After Flipping About Vertical Axis.</b> | <b>Pg. 88</b> |
| <b>Figure 4.53: First Symmetric Rib Geometries Oriented.</b>   | <b>Pg. 88</b> |

## **LIST OF FIGURES**

Cont.

|  |                |
|--|----------------|
| <b>Figure 4.54: Illustration of Orientable Position and Arm Assignment for Symmetric Ribs.</b> | <b>Pg. 89</b>  |
| <b>Figure 4.55: Illustration of First Symmetric Rib Orientation Motion.</b>                    | <b>Pg. 91</b>  |
| <b>Figure 4.56: Illustration of Second Symmetric Rib Orientation Motion.</b>                   | <b>Pg. 92</b>  |
| <b>Figure 4.57: First Non-Symmetric Rib Geometries Oriented.</b>                               | <b>Pg. 93</b>  |
| <b>Figure 4.58: Contact Position Transitions for Non-Symmetric Rib Orientation.</b>            | <b>Pg. 95</b>  |
| <b>Figure 4.59: Illustration of Arm Assignment for Non-Symmetric Ribs.</b>                     | <b>Pg. 95</b>  |
| <b>Figure 4.60: Illustration of Non-Symmetric Rib Orientation Motion 1.</b>                    | <b>Pg. 96</b>  |
| <b>Figure 4.61: Illustration of Non-Symmetric Rib Orientation Motion 2.</b>                    | <b>Pg. 97</b>  |
| <b>Figure 4.62: Illustration of Non-Symmetric Rib Orientation Motion 4a.</b>                   | <b>Pg. 98</b>  |
| <b>Figure 4.63: Illustration of Non-Symmetric Rib Orientation Motion 5a.</b>                   | <b>Pg. 99</b>  |
| <b>Figure 4.64: Illustration of Non-Symmetric Rib Orientation Motion 3b.</b>                   | <b>Pg. 100</b> |
| <b>Figure 4.65: Illustration of Non-Symmetric Rib Orientation Motion 4b.</b>                   | <b>Pg. 101</b> |
| <b>Figure 4.66: Illustration of Non-Symmetric Rib Orientation Motion 6c.</b>                   | <b>Pg. 103</b> |

## **APPENDICES**

|  |                     |
|--|---------------------|
| <b>Figures A1 - A5: Alternative Manufacturing Line Layouts, #1 thru #5</b> | <b>Pgs. 117-121</b> |
| <b>Figure D1: Orienter Prototype Parts List</b>                            | <b>Pg. 127</b>      |
| <b>Figures E1: Orienter Prototype Assembly Drawing</b>                     | <b>Pg. 129</b>      |
| <b>Figure E2 - E17: Orienter Prototype Fabrication Drawings</b>            | <b>Pg. 130-145</b>  |
| <b>Figure H1: Illustration of Rib Angle Assignments</b>                    | <b>Pg. 203</b>      |

## **LIST OF FIGURES**

Cont.

|  |                      |
|--|----------------------|
| <b>Figure I1: Illustration of Required Block Position Calculation Dimensions</b>   | <b>Pg. 206</b>       |
| <b>Figure I2: Cosine Law and Longest Rib Length Dimensions</b>   | <b>Pg. 208</b>       |
| <b>Figures J1: Illustration of Rib Centroid Position and and Mis-Rotation Capability Calculations: Illustrations and Spreadsheet</b> | <b>Pg. 209</b>       |
| <b>Figures J2, J3: Centroid and Mis-Rotation Calculation Spreadsheet</b>   | <b>Pgs. 210, 211</b> |



## Chapter 1: INTRODUCTION

### 1.1 HOUSING RESEARCH TRENDS: HISTORY

In The Evolving House, a three-volume work on the history of human shelter, Albert Farwell Bemis states, "... the chief factor of the modern housing problem is physical structure. A new conception of the structure of our modern houses is needed, better adapted... to modern means of production: factories, machinery, technology, and research."<sup>1</sup> Mr. Bemis' theory, thus stated in 1938, always has been and will continue to be a major driving force in the housing industry. One of the most visible manifestations of this fact is the continuously progressing effort to produce a pre-fabricated house.

At first, houses were basically "stick built," or assembled piece by piece. This standard and methodology, born in a non-technical era, resulted in relatively low quality construction which was difficult and time consuming. For these reasons, when the industrial era arrived, the benefits of factory production were quickly recognized and explored. Even as early as the colonial days, ready-framed houses were shipped from New England and Louisiana to the French and English Islands. Later, "row houses" evolved. Then, by 1932, Walter Gropius erected a modular house at a "Growing Homes" exhibition in Germany, with "no small satisfaction at last to be able to put into practice the doctrine of pre-fabrication which he had so consistently advocated since 1910, that doctrine which sought to unite the advantages of maximum standardization and with the

---

<sup>1</sup> Bemis, Albert Farwell. The Evolving House, p.viii. The Technology Press at MIT, Cambridge, MA. 1938.

desired goal of variability."<sup>2</sup> However, as despite avid proponents such as Gropius, this idea did not succeed commercially early on, (with some exceptions such as the Sears Roebuck & Co. "catalog house"), and was virtually abandoned during the Depression.

However, World War II brought a drastic change to the United States and its housing industry. Along with a large pent-up demand based on new families and the American Dream of owning a home, the large post-war production capacities and regenerated economy opened the way for pre-fabrication.

The first major developments aimed at the low end housing market. In order to make both factory and on-site construction efficient, feasible and inexpensive, merchant builders created designs which were standard and simple. This, of course, made them rather plain; box-like in shape (for fewer wall breaks), with simple roofs (to avoid hips), and with standard window and door configurations. Even though these houses were not architecturally impressive or particularly esthetically pleasing, they were immensely popular. Pre-fabrication was beginning to alleviate old problems such as poor planning and weather, or insufficient time, labor, and materials, which had sometimes resulted in unfinished projects. The "specialization, material control, precutting, and pre-assembly" technology and organization developed and implemented by the merchant builders at this time "brought a degree of speed and predictability that had previously been deemed impossible."<sup>3</sup> By the 1950's, merchant builders were well established across the US and housing starts had reached a post-war high of almost 2 million.

---

<sup>2</sup> Gropius, Walter and Wachsmann, Konrad. The Dream of the Factory-Made House, p.147. The MIT Press, Cambridge, MA. 1984.

<sup>3</sup> Eichler, Ned. The Merchant Builders, p. 70. The MIT Press, Cambridge, MA. 1982.



In the 1960's the industry began to slow down. The previous demand was beginning to wane, there were fewer births, the cost of building was rising, and home resales began to dominate. However, prefabricated housing companies did not disappear. Instead, the general enthusiasm and tenacity of merchant builders remained and they simply shifted their focus to other mass-producible modes of building, including community or clustered housing such as townhouses, condominiums, "quads" and "patio houses." At this time merchant builders also began to either go bankrupt, merge, or go public. The companies that survived this transition then had better access to increased capital and stability which helped them face the market swings to come.

The housing industry boomed from 1975 to 1979, but then declined to a virtual "crisis" state in the 1980's. Throughout this roller coaster, pre-fabrication methods, materials and organization were forced to grow and constantly improve; as in earlier words, to become "better adapted... to modern means of production: factories, machinery, technology, and research." For example, automated assembly of structural components such as floor and roof trusses and panelized walls became increasingly commonplace. Outside parties such as the Forest Products Laboratory developed new materials such as "Arrowood" and "Oriented Strand Board." Sub-contractors began to provide the means for vast componentization of items including staircases, doorways, chimneys, cabinets, etc. Plus, several aspects of interior components including toilets, lights, etc. were becoming more advanced. All of these developments helped to reduce cost, material, labor, and time consumption, and to improve the general quality of homes across the country and abroad.

However, at the same time, a different, previously unaddressed demand began to arise. Along with low cost and efficiency, consumers were beginning to ask for a degree of complexity and individuality not available with standard townhouses or more simply designed structures like those built in the 50's and 60's. This new demand influenced the design and manufacture of the entire house, but the one aspect that it affected most was the roof.

Initially, production lines for roof trusses were employed, but they couldn't handle the necessary range of up to 50 different sizes per job, or the innate complexity of aesthetic details like steep roof pitches or multiple gables, hips and valleys. While the available technology slowly began to catch up with the demand by employing flexible automation of software and machinery, there was still room for much improvement. For example, insulating methods and floor space utilization were far from optimal. Late in the 80's, most companies in the truss industry were experimenting with possible new methods of roof prefabrication.

## **1.2 THE MIT PANELIZED ROOFING SYSTEM**

### **1.2.1 EVOLUTION**

In 1986, led by the efforts of John Crowley, originally Research Director at Ryan Homes, MIT joined the general housing industry efforts by forming the Innovative Housing Construction Technologies Program (IHCTP). With funding from six different companies, research began with a broad look at several individual components of the standard American home, and the materials, structures and methods used for building them. Several issues were deemed worthy of further in-depth pursuit, but it was soon decided that future research should focus on a pre-fabricated roofing system.

Roofs were chosen for several reasons. As discussed earlier, builders were finding it increasingly difficult to meet consumer demands for complexity and individuality with conventional truss and rafter construction methods without encountering substantial insulation installation and ventilation problems. This would become increasingly problematic as energy efficiency codes, especially in the Northeast, became more stringent. Plus, while escalating land costs and housing density were demanding more efficient use of floor space, these traditional roofing methods were leaving most of the attic as dead, unusable space.

Another issue that needed to be addressed was a projected decline in available qualified workers. Since roof installation heavily employed skilled labor, this decline promised to substantially increase future on-site labor costs. One method of reducing this cost, which was already proving itself effective, was the use of pre-fabricated components.

For example, for a standard site-built staircase, cost was fairly evenly divided between materials and labor. However, a pre-fabricated staircase would cost approximately 30% less over all, and require almost 60% less labor. Plus, pre-fabricated components would also help to reduce installation time as well as material consumption and waste.<sup>4</sup>

With these issues in mind, research continued by looking at several different possible roofing panel materials and designs. After comparing the relative structural strength, material content, and ease of manufacture for prospective designs including foam core panels, circular, triangular, and vertically ribbed panels, and wave, folded and corrugated panels, vertically ribbed panels were found to be the optimal solution.<sup>5</sup> Plus, the preferred material was Oriented Strand Board (OSB). If panels were made entirely of OSB, material consumption could be shifted from expensive old growth timber to a more ecologically appropriate young growth composite. Research as to optimal OSB thicknesses, overall dimensions, insulation standards, construction requirements and mounting methods, etc. continued until an entirely roofing system was designed. By June of 1991, a full scale, hand built proof-of-concept structure was erected.

---

<sup>4</sup> Crowley, John S. and Parent, Michel. "Engineered Roof Products: A Flexible Product Concept for Net Shaped House Components." Unpublished Report. IHCTP at MIT, Cambridge, MA. February, 1991.

<sup>5</sup> Sharma, Vikas. "Roof Panels and OSB: Issues in Design and Manufacturing." Mechanical Engineering M.S. Thesis at MIT, Cambridge, MA. September, 1991.

## 1.2.2 SYSTEM AND PANEL DESIGN

The panelized roofing system is composed of two key components: ridge beams and panels. A triangular beam runs the length of all roof ridges and provides the highest support and assembly reference point for the entire system. Individual panels are secured and registered against a 2" x 4" on the ridge beam and span the distance to the eaves. They are then attached to each other by splines, strapped to the eaves by metal strips, and finished with soffits, facias, and finish cladding, including tar paper, shingles, tile, or metal. Figure 1.21 Illustrates this system and its components.

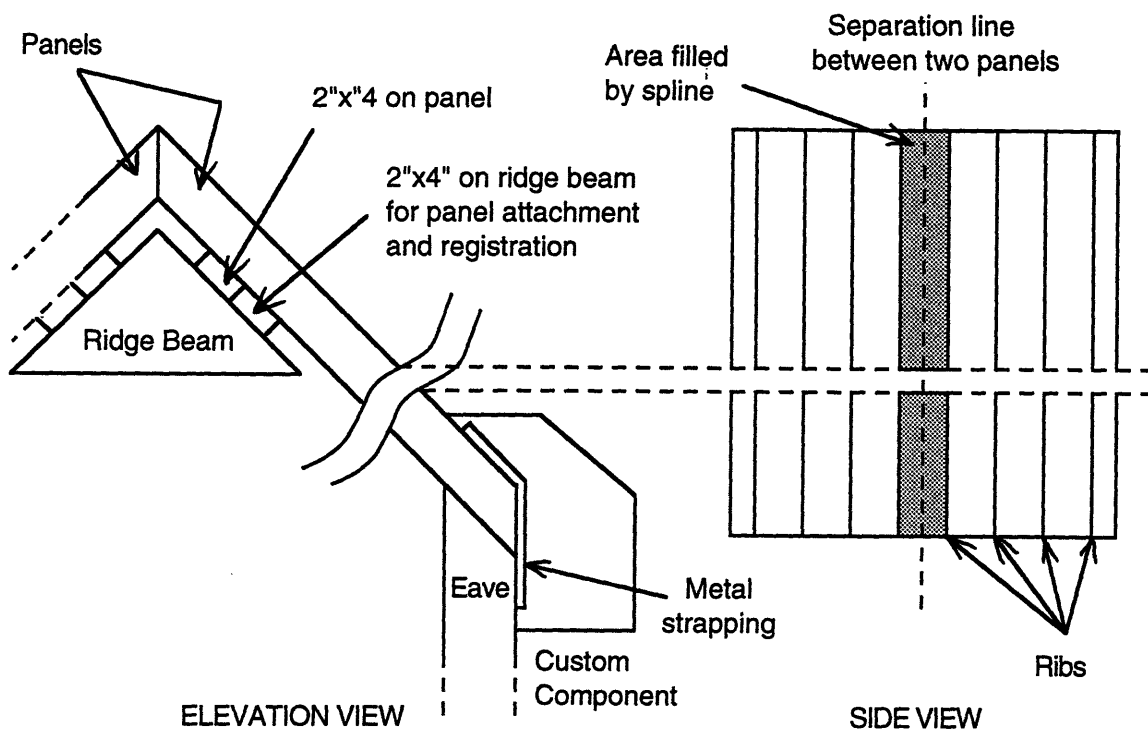


Figure 1.21: Panelized Roofing System Components

As shown in figure 1.22, the panels themselves are composed of a top face, two to four ribs, a bottom face, two end caps, insulation and blocking.

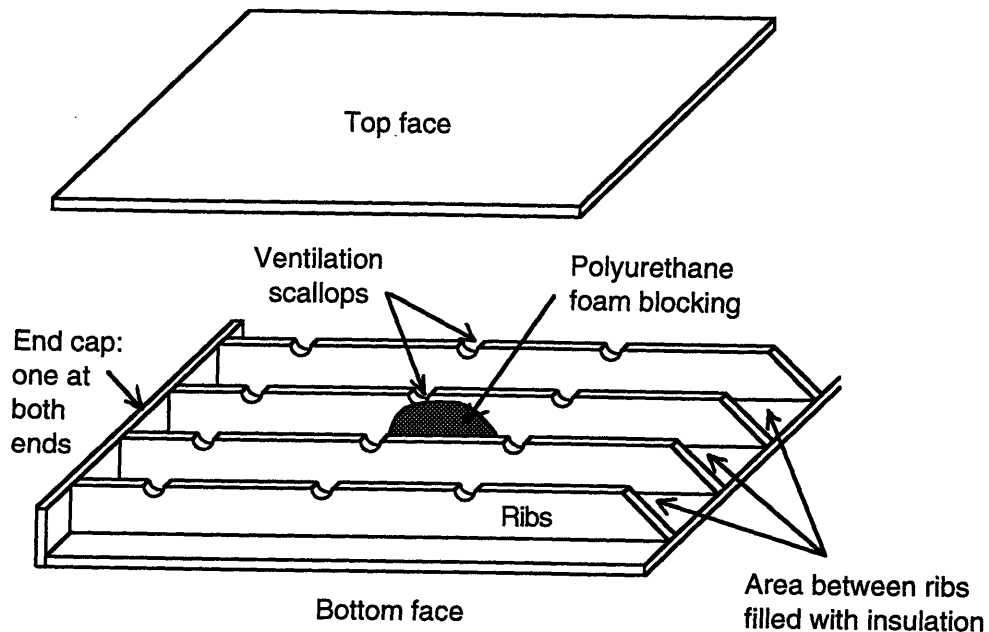


Figure 1.22: Panel Components

All components are made of Oriented Strand Board (OSB) with the exception of the mineral wool, fiberglass or cellulose insulation and the polyurethane foam blocking. Ventilation for the insulation is provided by 4" diameter scallops which are cut into the tops of the ribs. The ribs support the longitudinal shear inflicted upon the panel by snow and other elements, and the end caps on the ribs work together with a single polyurethane foam block to prevent racking of the panel during transport and assembly. The end caps also assist roof assembly by providing a smooth panel interface surface at the ridge, hips, and valleys, and serve to retain the insulation as it is added during manufacture. All structural joints are secured by a phenolic resin adhesive and staples or nails.

Individual panels may have several different geometries. Figure 1.23 shows a standard roof design and the resulting panel shapes.

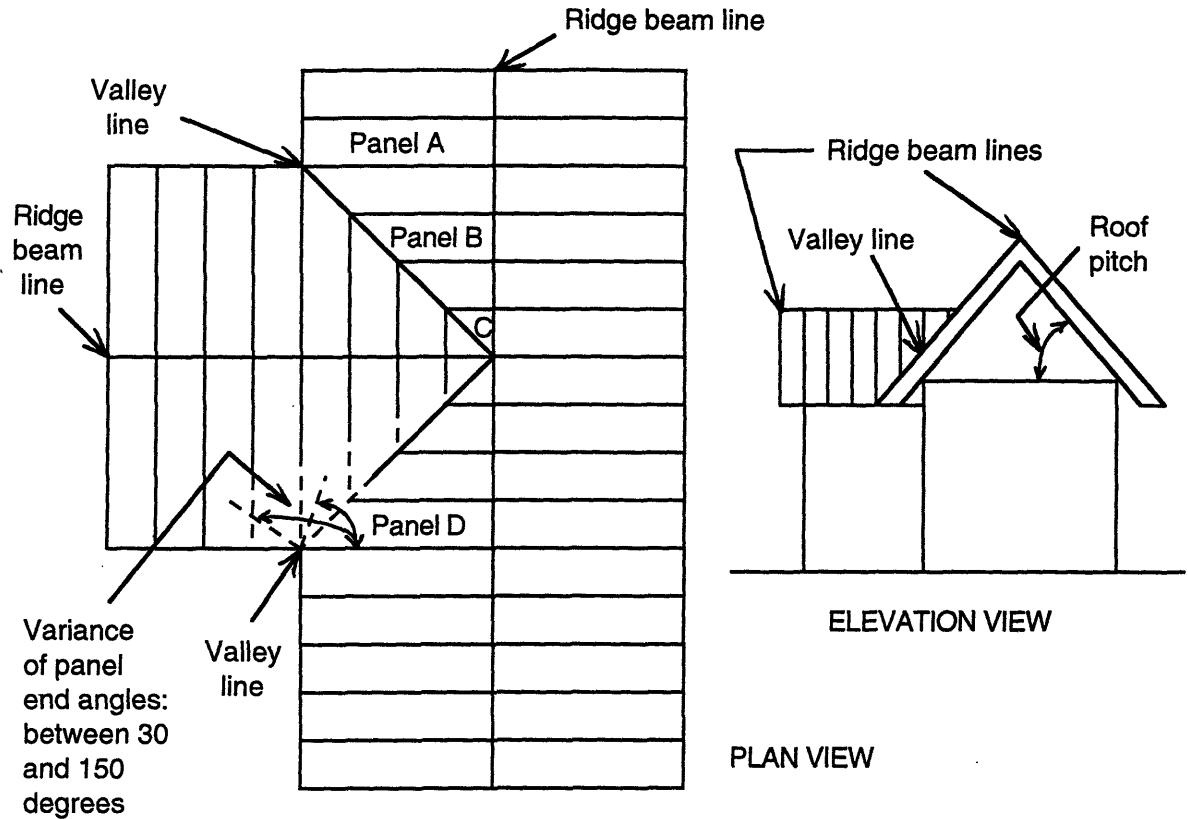


Figure 1.23: Standard Roof Design and Resulting Panel Shapes

The largest panels will be rectangular and up to 24' long, like panel A. Some will also have triangular ends like panel B, or be entirely triangular in shape, like panel C. As illustrated on panel D, their angles will vary between 30 and 150 degrees according to roof pitch.

The pitch will also determine the panel end geometries and rib shapes. Ribs will always be 9 1/4" tall, 7/16" thick, and spaced 14 1/2" apart, but as shown in figure 1.24, their end geometries will vary between 40 and 140 degrees.

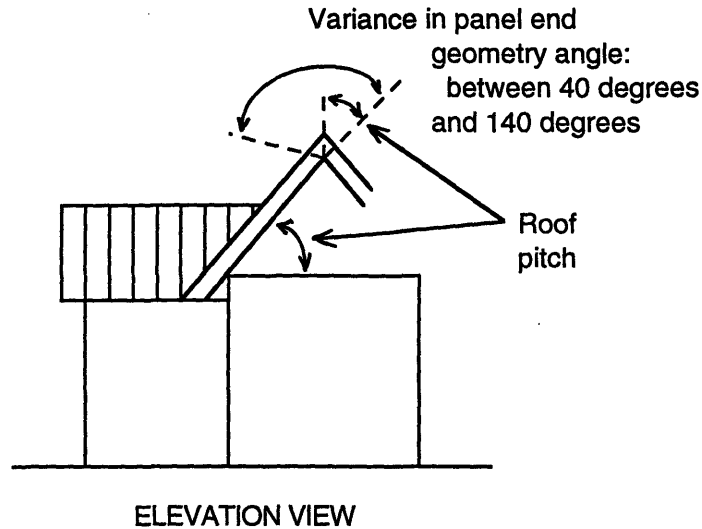


Figure 1.24: Panel End Geometries as a Result of Roof Pitch

### 1.2.3 EXPECTATIONS

The benefits promised by the MIT panelized roofing system are as follows:

#### Complex roof geometry capabilities

#### Reduced cost:

- Available attic living space at 1/3 cost of expanding the rest of the house
- On-site roof assembly time reduction of 83%
- Reduced use of skilled labor

#### Improved insulation:

- R-values exceeding Northeast code requirements
- Installation eliminated as a post-process
- Improved and isolated ventilation

#### Improved maintenance capabilities

- Old-growth materials replaced by new plantation and lesser value species composites.



Researchers involved in the development of this system knew that before the prospective product could fulfill its maximum potential and reach the intended markets at a competitive price, it had to be coupled with a cost effective manufacturing system. Such a system would also provide additional benefits such as increased tolerance control, greater design flexibility, improved customer perception, and maximized throughput with minimal material and energy waste. For this reason, production constraints were considered during the development of the panels, several line layouts were conceptualized early in the project, and methods for developing and employing a CAD system which could both generate a roof design and determine an affective arrangement of panels were explored.

The next chapter describes the more in-depth development of a complete flexible panel manufacturing system, including panel redesign for manufacture and the incorporation of several important individual process considerations. Chapter 3 discusses the continuing research which involved further design of the rib orientation, gluing, and transport sub-system, as well as prototype fabrication for the rib orientation station. Chapter 4 describes the control of the prototype. This involved the development of algorithms that determined rib manipulation steps based on geometry and required final position alone. In the final chapter, orienter design notes are made, and rib preparation time is estimated. Also, future work with and testing of process limits, completion and success sensing, reliability, and algorithm improvement is suggested. Finally, a general concluding statement is given.



## **Chapter 2: ROOF PANEL MANUFACTURE**

### **2.1 INTRODUCTION**

Once the design of the MIT panelized roofing system was complete, research was more heavily directed toward the fundamentally important issue of affective manufacturing processes. Of all the components in the system, the panels themselves were undoubtedly the most critical; they were the basis of all energy efficiency, structural soundness, and inter-connections between elements. Plus, they would present the most complexity and require the most accuracy of all manufacturing solutions. For these reasons, it was decided that the development of an automated assembly line for panels should be addressed first. In October of 1992, with partial funding from the Department of Energy, the Manufacturing Institute at MIT began the task of designing a panel manufacturing line.

### **2.2 REDESIGN FOR MANUFACTURE**

The first issue to be addressed was design for manufacture. The panel design had been partially geared toward mass production, but in order to verify and improve its design for assembly, all panel components were re-evaluated according to their function and necessity. As a result of this process, several aspects of the panel were eliminated or redesigned, thereby simplifying and improving the panel as well as simplifying and broadening the range of possible manufacturing solutions.

The first major design change was the elimination of what were known as reinforcement strips. Originally, separate 2" x 4" pieces were secured between each rib at both panel ends to provide substantial material for power screw attachment to the ridge beam and eaves. Since these strips were the only structural components made of a material other than OSB, their removal nullified the need for an entirely separate material supply, cutting, transport, orientation and attachment subsystem of the manufacturing line. However, in order to maintain the structural soundness which the original reinforcement strips provided, an alternative method of securing the panel at the ridge beam and eaves had to be developed.

Of all the possible options, metal strapping was identified as the most favorable replacement for screw attachment. First of all, metal strapping looked to provide an even more reliable joint than the original screws. Second, in the initial panel design, eave end rib geometries were to have compound angles in order to facilitate the addition of detailed eave finishing ornamentation. This would call for a complex dual saw for cutting ribs on the manufacturing line. However, employing metal strapping would make it necessary to have a flat edge parallel and flush with the wall at the eave. This flat, single-angled surface, like that at the ridge beam end of the panel, would require only a single saw for fabrication. Thus, the change in attachment methods simplified manufacturing requirements by eliminating an entire saw and sensor system from the line. Custom eave components could still be attached, and with the panel now ending at the wall of the house, insulation would no longer be wasted on filling the otherwise overhanging portion of the panel.

Figure 2.21 illustrates the original reinforcement strips and rib end angles and the final flat edged metal strapping solution.

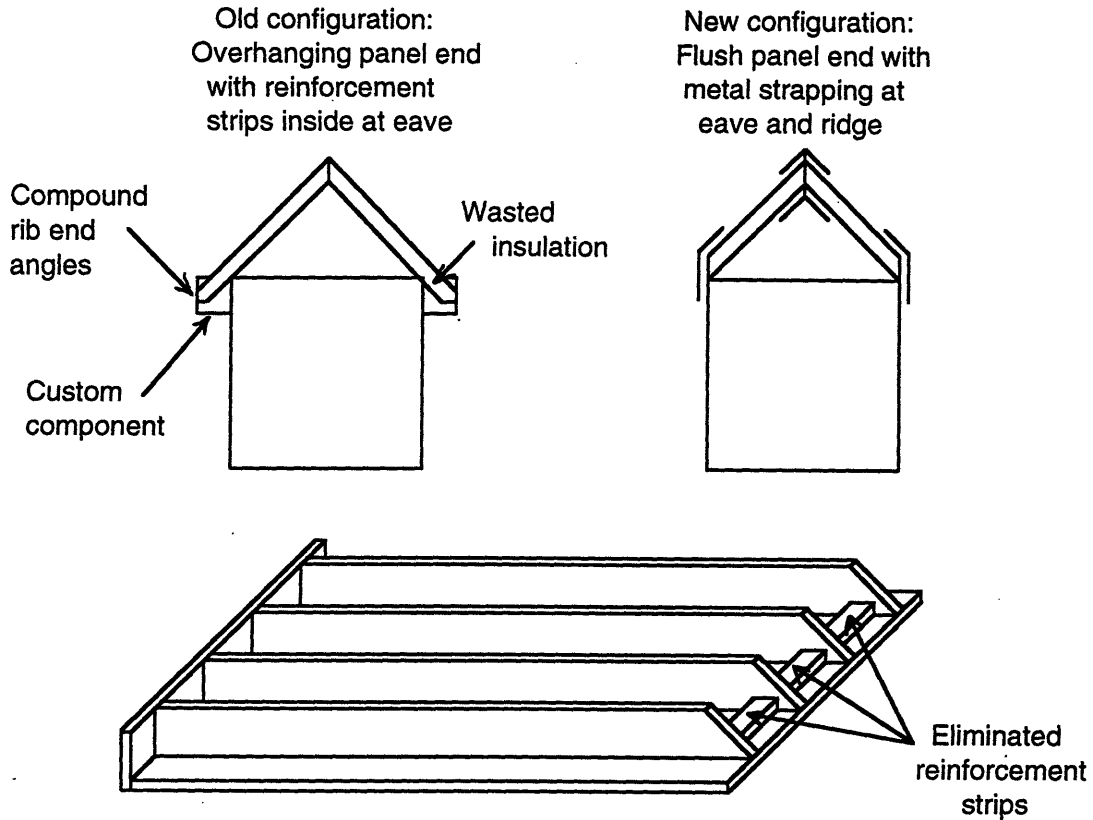


Figure 2.21: Illustration of Reinforcement Strips and Replacement Strapping

Another component eliminated was solid OSB blocking. These small pieces, individually placed between each rib near both panel ends, presented both functional and manufacturing limitations. The presence of this type of blocking near the panel ends created small, difficult cavities to insulate. If, for simplicity of manufacturing, these spaces were left poorly insulated or uninsulated, thermal performance in the most critical roof areas near the ridge beam would be jeopardized. Also, much like the reinforcement strips,

removal of this type of blocking would eliminate the need for an additional dedicated cutting, transport and assembly sub-system.

Preliminary investigation of the panel design suggested that the blocking could be removed without replacement, as its only original function was to assist accurate hand assembly. However, consequent structural tests disproved this hypothesis. The presence of end caps was found to be sufficient for supporting the panels once they were assembled on the roof, but end caps alone could not withstand the extreme lateral loading conditions likely to be induced during transport or storage. At this point, polyurethane foam blocking was considered as a possible replacement. Standard systems for storage and transport of polyurethane foam already existed, so no new technological advancements would not be necessary in this area. Also, the foam could be simply poured or sprayed into the panel such that the need for sophisticated manipulation systems could be eliminated. Plus, incorporation with the insulation process would be possible.

Before the polyurethane foam could be adopted as a substitute, its properties needed to be tested. Test results implied that a single block of foam placed at the center of the panel would indeed provide the necessary initial lateral structural stability without sacrificing thermal properties of the panel. The performance of polyurethane beyond a few months was unpredictable due to its potential to creep. However, in the light of its necessity only during the early periods of storage and shipping, and the fact that its structural functionality could not be completely lost or its thermal properties degraded in any way, it was considered an acceptable solution. Figure 2.22 illustrates the original

OSB blocking and the polyurethane foam block which replaced it.

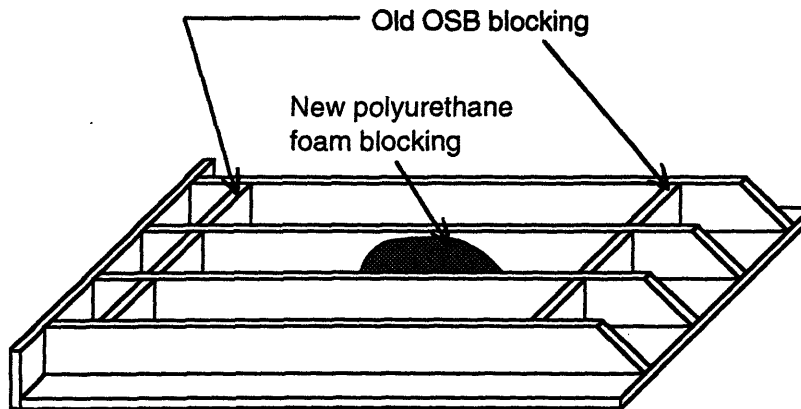


Figure 2.22: Illustration of OSB Blocking and Polyurethane Foam Block Replacement.

### 2.3 INDIVIDUAL PROCESS CONSIDERATIONS

After arriving upon a panel design which was more suitable for manufacture, line design began. At first, in order to better understand each aspect of the system, panel assembly was broken down into isolated processes. For example, ribs, faces and end caps all had to be individually cut, conveyed, glued and stapled, and the assembled panel had to be insulated, weatherproofed, and presented ready for stacking and transport to the job site. Each of these processes would affect each other as they were incorporated together in a line, but initially addressing them as a separate tasks brought greater familiarity and attention to their respective features, limits, requirements, and compatibility with other processes. Also, in this way, their individual solutions could be initially generated and

molded by their own respective capital investment, throughput, energy and material consumption capabilities. Then, additional options, considerations and/or restrictions would be developed as they were brought together. Some of the process-specific constraints and several consequent global considerations which were identified in this phase are discussed in this section.

### 2.3.1 OSB SUPPLY CUTTING, JOINING AND CONVEYING

The least expensive form of OSB which could be input to the system at the time was 4' x 8' sheet. However, with panel sizes ranging up to 24', either 24' sheets had to be available, or the 8' sheets needed to be joined end to end. The 8' sheets were finally chosen to be the only OSB supply size for three reasons: Multiple supply sources would unnecessarily complicate feeding systems, 8' sheets were inherently more manageable than 24' sheets, and the virtually continuous strip of OSB produced by joining would allow for substantial reductions in material waste. Figure 2.31 illustrates how the ability to cut faces sequentially eliminated the large waste potential for panels with extremely angled end geometries.

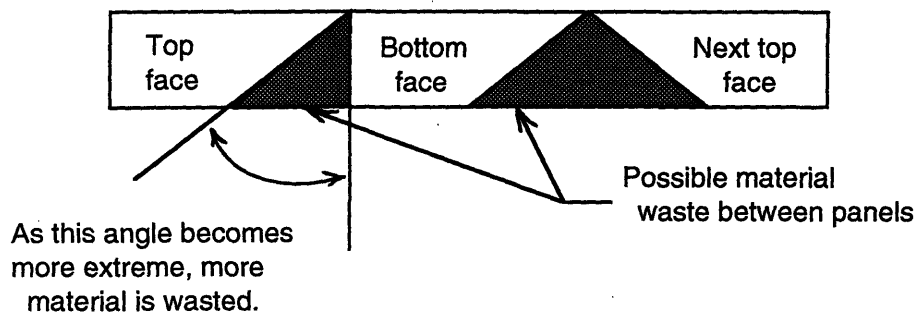


Figure 2.31: Illustration of Waste Potential for Panels With Extreme End Geometries



Following this decision, investigation of joining methods found that only scarf joining with radio frequency (RF) curing could meet system requirements. Compared to the 30 seconds required for RF curing of a single joint, other joining methods required glue cure times of over two minutes per joint, or 4 minutes for each 24' face. This large difference in cure times completely ruled out other alternatives on the basis of throughput alone, since for a target fabrication time of 175 panels a day, a panel needs to be completed every 2.74 minutes. One major problem with this solution was that the cost of RF curing was well above that of other methods. However, in order to ensure line functionality and acceptable throughput, RF curing had to be included in the line layout.

It would have been preferable to assign separate supply and joining stations to ribs, top faces, and bottom faces, but the escalated cost of RF curing systems made this out of the question. In order to meet the throughput goal of 175 panels a day, at least one independent station had to be dedicated to ribs and another to faces. Therefore, two joining stations were employed. Dedicating a single joining and cutting station to supplying both top and bottom faces required a more sophisticated conveying system, (consecutively cut faces needed to be shuffled between buffers and stations in order to reduce scrap and present the desired geometry at the right time), but this conveyor system looked to be less expensive and to produce less waste than adding another cutting and joining station. Figure 2.32 illustrates the necessary face shuffling.

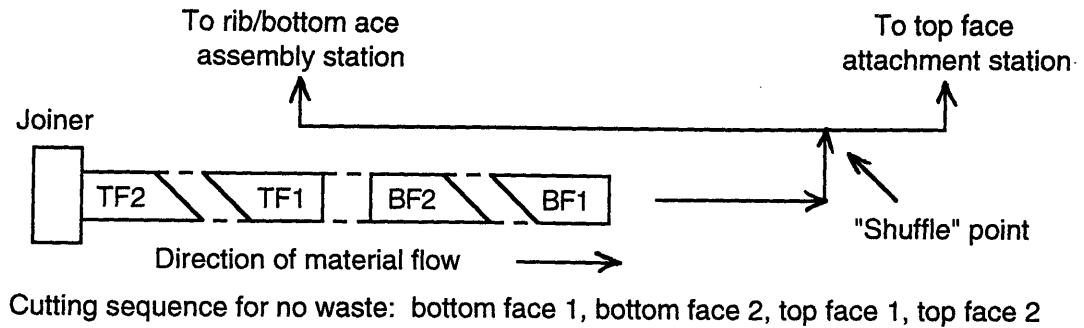


Figure 2.32: Illustration of Face Shuffling Conveyor

It should be mentioned here that new, affective and less expensive gluing methods were being developed concurrently with this design. Should one of these options become available, the tradeoffs between joining, cutting and conveying solutions should be reconsidered.

### 2.3.2 ASSEMBLY MOTION TRADEOFFS

In general, throughput and the use of floor space are heavily affected by the distance and direction that components move. Conceptually, this means that all movements should be performed across the shortest dimension possible. For this layout, it meant that ribs and faces should be moved across their 9 1/2" or 4' dimension as opposed to their 24' dimension, since the relative floor space coverage and time for travel would otherwise be up to six times as long. However, operations such as joining and gluing could only be performed by lengthwise motion, and the optimal presentation and assembly of a single component such as end caps often called for two or more orthogonal motions. Combining these constraints into a compact layout which included the greatest

ease of assembly called for several compromises between preferred motions and overall organization.

With large panel sizes and the extensive processes required for preparing ribs or insulating the panel, redundant machines and simultaneous process group execution needed to be implemented. Theoretically, the more places this could be done, the better the throughput would become. In practice, this concept was limited by the tradeoffs between relative process complexity, floor space utilization, capital cost, and cycle time. For example, no matter how efficiently other stations performed, the slowest station would determine the entire line throughput. It might be possible to dedicate extra floor space or technology to improve the performance of this station, but there would be a point at which the gains in cycle time would not exceed the capital cost expense. Also, for distribution purposes, there would be a point at which it would be more reasonable to employ multiple manufacturing lines at separate locations instead of a single, large production facility.

### 2.3.3 INSULATION METHODS

Two options were considered as possible means for insulating the panel: individually installing batts or continuously blowing in materials such as mineral wool, cellulose, or fiberglass. Batts were a functionally attractive option because they would easily retain their installed position without potential for creep, shift or settling during the entire panel life span, and they relieved the close end cap tolerance requirements necessary to retain spray insulation. However, from a manufacturing point of view, batts

were rejected immediately. In order to use them on the assembly line, substantial material storage space would have to be dedicated to large rolls of uncut insulation. Next, installing the batts would call for an extra, complicated cutting and handling system which would be awkward and produce generous amounts of scrap. Finally, the rolls would have to be changed frequently, sometimes in the middle of insulating a single panel, thereby interrupting the flow of the entire line on a regular basis.

Spray insulation offered a much more viable solution in every way. Not only did it provide greater implementation flexibility, but the insulation components required much less storage space and could be mixed on demand with minimal waste. Plus, a spray station would only need attention at the beginning and end of each work shift. There was only one potential drawback of material creep and shift, but this was considered resolvable by an affective combination of a binder with the insulating fibers.

Initially, the insulating process posed the threat of becoming a bottleneck. However, even though this process promised to be extensive, if developed affectively and employed independently of all other processes, the potentially large throughput constraints could be avoided.

#### 2.3.4 END CAP ATTACHMENT

The variables governing end cap attachment were numerous. Structural functionality of the end caps was critical, tight tolerance control was necessary, and the process of orienting and attaching end caps was identified as a secondary bottleneck. These facts placed great emphasis on the effectiveness and time consumption of all required motions. Along with these constraints, there were also several procedural options to be considered. End caps could be attached to the ribs before or after they were fixed to the bottom face. Attaching end caps to the ribs before the bottom face was introduced would make the rib ends more accessible, allow more space for the required end cap mechanisms, and avoid possible interference between the end caps and bottom face. However, there were other important factors. In order to eliminate the expense of redundant technology, only one end cap station could be included on the line. This meant that ribs would have to be moved in relation to the end cap station. Longer ribs are incredibly unstable and prone to collapse under their own weight if supported only at their ends. Without the comparatively substantial support provided by the bottom face or an overhead mechanism, any such motion presented potential for damage and mis-orientation of the ribs. The consequent threat of defective joints between the ribs and the end caps or bottom face was unacceptable and led to the final configuration where attachment of end caps follows the assembly of the ribs with the bottom face.

The conveying and introduction of end caps for assembly also presented several questions as to possible operator interfaces and the incorporation of expensive robotic

manipulation. Before an overall final solution could be generated, the tradeoffs between joint and rib integrity, relative motion, capital cost, and time and space consumption had to be considered.<sup>6</sup>

### 2.3.5 RIB ORIENTATION AND GLUING

The combined processes of receiving ribs from the cutting station, rotating them to an upright position, applying glue to their bottom and side edges, and transporting them for assembly with the bottom face was identified early on, not only as a critical station, but as the primary bottleneck. This was true for several reasons. First of all, in order to optimize throughput, the final layout needed to employ assembly processes in parallel wherever possible. The result of this type of organization was that for each panel, in the same amount of time as a single-step process such as insulating, or a four-step process such as attaching end caps was performed, this particular combination of four rib preparation processes would have to be completed once for each rib, for a total of sixteen steps.

The time required to complete these processes would also be heavily influenced by the complexity of each step and the difficulty of incorporating them. The final rib position had to be accurate to maintain the necessary close panel assembly tolerances. This standard design constraint was compounded by the unusual 2" to 24' range of ribs to be oriented, the potential for ribs to be substantially mis-oriented by the conveyor, and the

---

<sup>6</sup> Phillips, David. "Flexible Manufacturing of Roof Panels: Process Layout and Assembly Equipment Development." Mechanical Engineering M.S. Thesis at MIT, Cambridge, MA. 1994.

fact that a vision system was not an environmentally viable or economically feasible solution. Finally, incorporating gluing with any process would be extremely difficult since glue would be inherently detrimental to any machine upon contact. In this case however, glue had to be applied to the three most critical edges for determining rib orientation, and they had to be thoroughly coated to ensure sound joints and critical panel stability. Also, the options for implementing this process were limited by the fact that once the ribs were adequately secured in an overhead mechanism, their edges would most likely be less accessible.

This long list of constraints was accompanied by a relatively endless yet limited range of approaches. Ribs could be oriented directly at the sawing stations by several independent mechanisms, or, they could be placed on a buffer and moved through a single orientation station, or, the overhead mechanism could include the necessary components for both transport and orientation. Each of these different options included several tradeoffs between redundant motions, capital cost, and throughput which needed to be carefully considered. Further elaboration on this task, its constraints, possible solutions, and a final working prototype of the rib orientation system are included in Chapter 3.

## **2.4 COMPLETE CONCEPTUAL LINE LAYOUT**

Once all the processes and constraints were considered and developed individually, they were combined into several different complete line layouts. The operation logistics and combination principles discussed in the previous section and all the possible applications thereof made for several distinct approaches to the solution. After an in depth analysis of the comparative tradeoffs between throughput, material and energy efficiency, capital cost and logic simplicity involved in each layout, a final line concept was determined. Other alternative concepts are given in Appendix A.

The entire system is driven by a CAD file generated when the vendor and customer meet to design the roof. From this file, the appropriate on-roof panel organization and resulting component geometries, as well as the most material-efficient on-line panel assembly order is determined. Then, this information is translated into commands which coordinate the preparation and presentation of the ribs and faces as they are needed.

As illustrated in figure 2.41, the complete manufacturing layout, if divided into rib, face and assembly sub-systems, is described as follows.

### **Rib subsystem:**

- (1R) Standard 4' x 8' OSB supply sheets pass through rip saws which create 5, 9.25" x 8' strips.
- (2R) The strips are sorted and joined end-to-end, thereby creating a continuous supply of rib material.
- (3R) Ventilation scallops are drilled into the top of each rib by a stationary press as the ribs are moved out of the joiner, cut to length, and placed on a buffer/conveyor.



- (4R) Ribs are received from the buffer/conveyor, oriented, rotated to their upright position, and grasped by a guidance clamp.
- (5R) Ribs edges are glued as they are translated lengthwise into the overhead mechanism.
- (6R) Ribs are secured in the overhead mechanism and transported to the rib/bottom face assembly station.

**Face subsystem:**

- (1F) Standard 4' x 8' OSB supply sheets are joined end-to-end, thereby creating a continuous supply of face and end cap material.
- (2F) Two adjustable saws cut faces and end caps to size.
- (3F) End caps and faces are sorted and conveyed to their respective assembly stations.

**Assembly subsystem:**

- (1A) Ribs with glue on the bottom and side edges are held in contact with the bottom face and stapled from below.
- (2A) As the rib/bottom face assembly leaves the first assembly station, it pauses for one end cap to be nailed into place. Before reaching the second assembly station, it pauses again for attachment of the opposite end cap.
- (3A) Insulation and the polyurethane foam are sprayed into the open panel structure. As the panel leaves this station, glue is applied to the top of the ribs.
- (4A) The top face is stapled onto the panel. As the panel leaves this station, it is weather-proofed.
- (5A) The finished panel waits in a buffer until stored or loaded onto a truck for delivery.

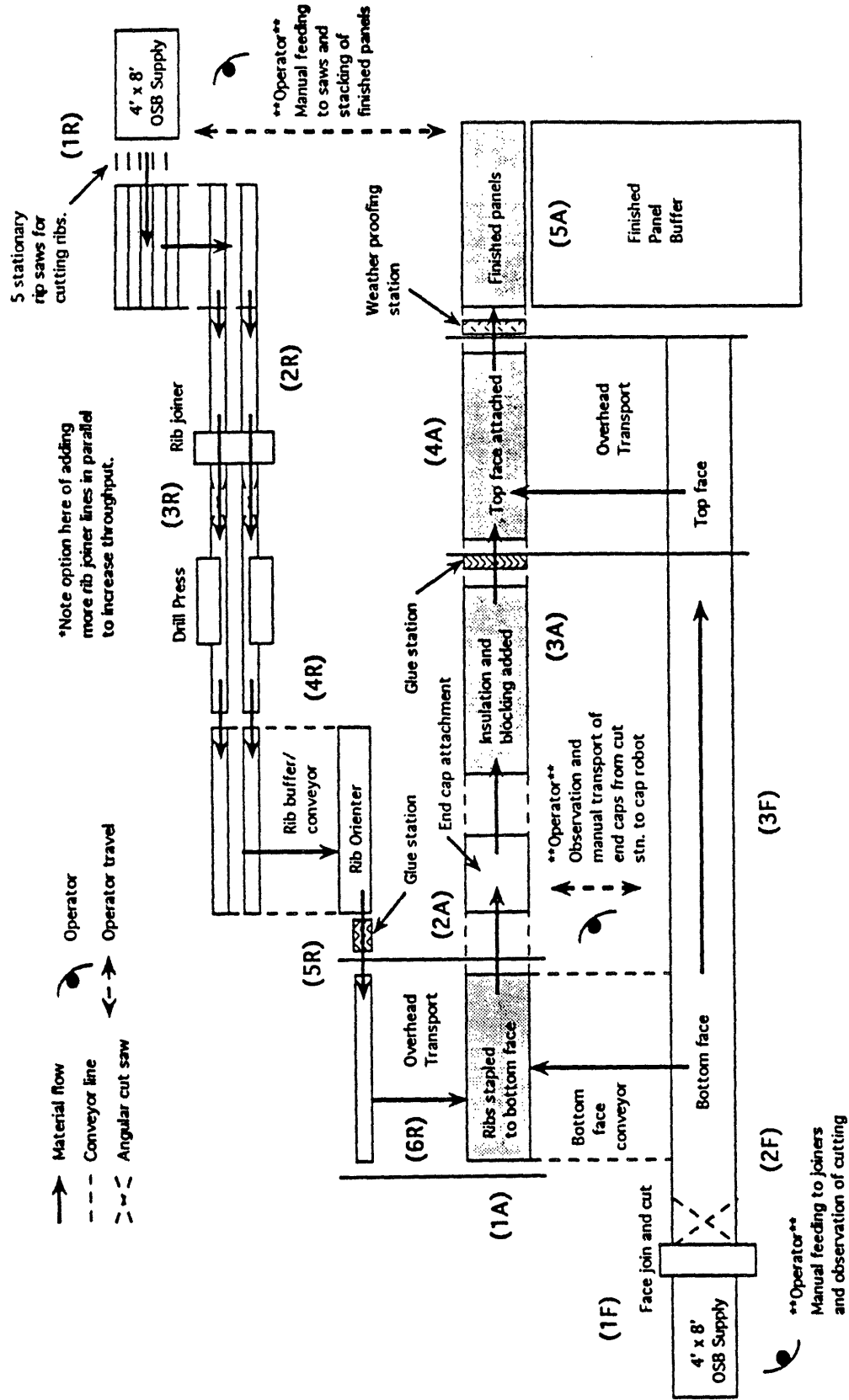


Figure 2.41: Final Panel Assembly Process Layout

## **2.5 SPECIFIC PROCESS AND MACHINERY DESIGN EVOLUTION**

Once the overall line organization was determined, some of the critical operations and mechanisms along the line needed to be designed in detail. This continuing development was necessary to ensure that the basic proposed uses and combinations of new and/or existing technology were actually feasible. For example, glue mixing and dispensing systems were common, but one capable of applying a steady bead to the bottom of a narrow strip of wood was currently unavailable and would be inevitably complex. Automated stapling systems were largely employed, but not upside down, in a space constrained or adjustable configuration. Robotic orientation of objects was a common research topic, but the wide range of rib geometries and the unique combination of orienter design requirements would be more complicated. Generating a detailed design of the entire line was unnecessary since several components such as the joining stations were still subject to change, and most of the conveying problems could be solved by standard technology. However, developing a smaller section of the line which employed the most critical processes in their most difficult forms would be necessary to prove the technology. If appropriately selected, it could also provide the added benefit of having a working system that could assist the assembly of sample panels for testing.

With these things in mind, a small sub-section of the total layout was chosen for further design. As shown in figure 2.51, this section included the rib orientation and gluing station, the overhead transport mechanism, the rib/bottom face assembly station, the end cap attachment mechanisms, and the insulation and polyurethane blocking station. Chapter 3 describes the conceptual design of the rib orientation station, the gluing station

and the overhead transport mechanism, and Chapters 3 and 4 describe the prototyping process for the most critical of these: the rib orienting mechanism.

Detailed design of the end cap attachment station and the insulation/blocking station, as well as a dynamic analysis of the proposed end cap attachment mechanism, is described in [Phillips, '94].

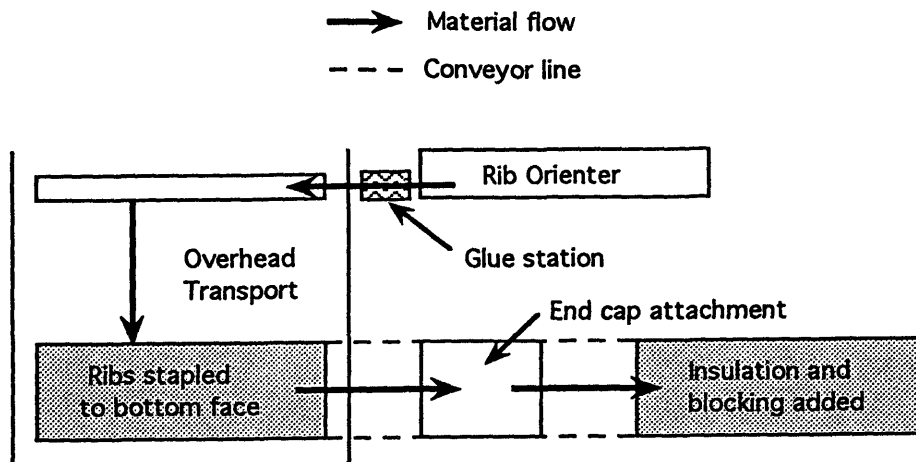


Figure 2.51: Panel Assembly Layout Sub-section for Development

## **Chapter 3: RIB ORIENTATION, GLUING, AND TRANSPORTATION: SYSTEM DESIGN**

### **3.1 ORIGINAL TASK DESCRIPTION**

During the early stages of manufacturing research, the combination of rib orientation, gluing and transport processes was found to be one of the most critical aspects of the entire line. Section 2.3.5 discusses in more detail how the performance of this station heavily influenced the desired throughput and flexibility of the manufacturing line as well as the resulting panel tolerances and quality. Included here is a summary of the most important general manufacturing constraints (MC) and the consequent design criterion (DC).

1. MC: Pre-cut ribs must arrive at the rib/bottom face assembly station such that they may be affectively joined to maintain panel tolerances and stability.  
DC: Design a system which takes ribs from the cutting station, applies glue to three edges, and precisely positions (orients) and presents them for assembly with the bottom face.
2. MC: Rib orientation, gluing and transport is the primary manufacturing line bottleneck.  
DC: All motions and processes involved in these tasks must be time efficient and run in parallel if possible.
3. MC: The entire manufacturing line must be flexible.  
DC: Operating by CAD file geometry and final rib configuration information alone, the rib orientation station must be able to handle all ribs, which will be 9 1/4" tall and 7/16" thick universally, but range between 2" and 24' in length.
4. MC: Capital expense must be limited.  
DC: Avoid excess redundancy and complexity wherever possible, minimize and standardize parts, avoid vision systems or extensive sensor systems.

### 3.2 PARTIAL CONCEPTS AND DESIGN EVOLUTION

The design of the rib orienting and gluing mechanism was a gradual process. What began as an apparently standard task became increasingly challenging as basic manufacturing issues and physical constraints began to present more obstacles and require more complexity than was originally expected. The nature of these constraints and the reasoning behind the final design are best described and understood by tracing their evolution. It should be noted that not all preliminary concepts are discussed here, and that those presented are included for the purpose of explaining and illustrating the task and its various aspects.

At first, the orienting, gluing and transportation processes were considered individually. As a result, the original task definition was relatively simple: The orienting mechanism should receive a rib (assumed to be reasonably manageable) from a conveyor belt, rotate it to an upright position, and deliver it to the overhead mechanism. Once all ribs are secured in the overhead mechanism, it should carry them to a point where glue is applied to their edges. Figure 3.21 shows a rough sketch of a possible solution governed by this scenario.

Here, the orienter is a toothed clamping mechanism. In the horizontal loading position, rollers slightly protruding up between the bottom teeth move a rib from the conveyor into the orienter. The upper teeth then clamp down on the rib, and the orienter rotates ninety degrees to the upright vertical position. Next, the orienter moves along a track system leading to the desired location for rib transfer to the overhead transport mechanism. The overhead mechanism grasps the rib in the free space between the teeth,

thereby allowing the orienter to release the rib and return to its original position at the conveyor. No specific gluing solution was generated at this point since it was considered a secondary operation, probably to be performed by a simple linearly mounted gluing head once the ribs were secured in the overhead transport mechanism.

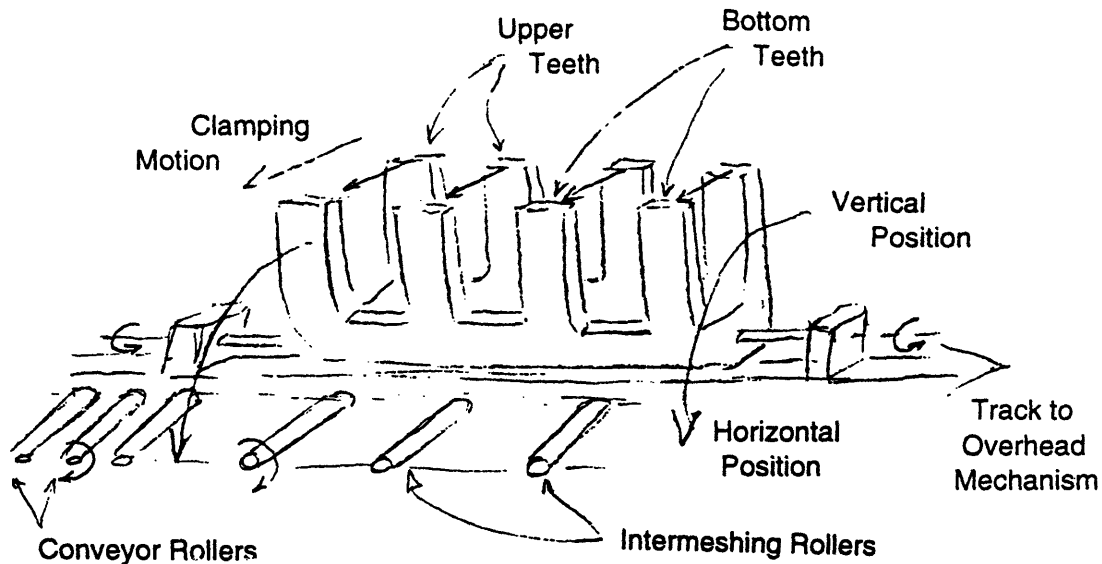


Figure 3.21: Preliminary Orienter Design Concept

There were several aspects of this concept which needed improvement. First of all, in order to meet the original manufacturing restrictions, much greater tolerance control was needed for the relative positioning of the rib within the orienter. The addition of some sort of end stop was seen as a potential solution, and was employed in various different forms in every design hereafter. The second problem was the assumption that all ribs would be long enough to span the distance of several teeth. This would have made them stable enough to behave somewhat predictably while being conveyed, rolled into the

orienter, and rotated into their upright position. However, this assumption was eventually invalidated even though it was sustained through the next few design concepts.

One of the next designs considered is shown in figure 3.22.

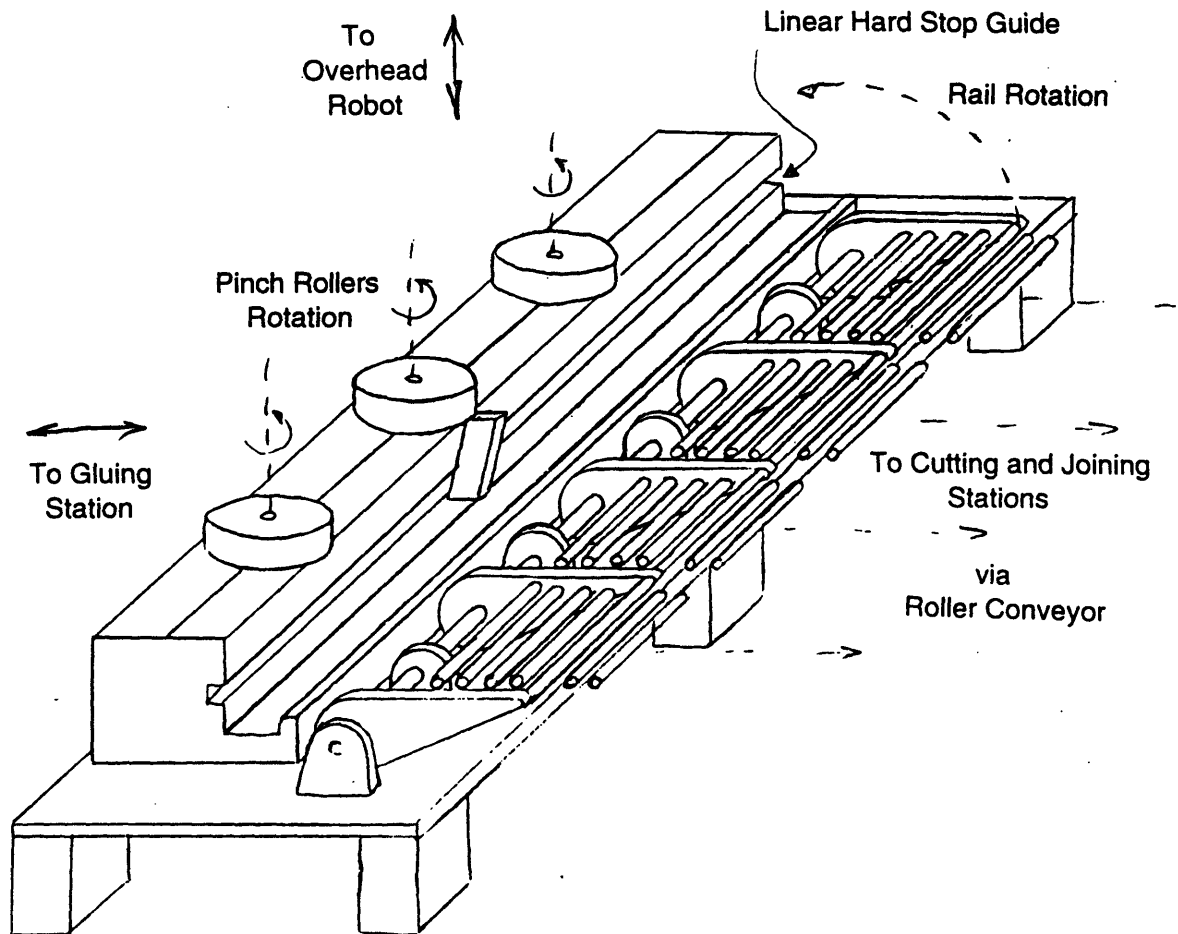


Figure 3.22: Second Orienter Design Concept

First, the idea of intermeshing rollers employed by the previous design was slightly modified. Here, a single orienter rail lies between several conveyor rollers. Once the bottom edge of the rib contacts the orienter, the rail rotates pneumatically, lifting the top edge and causing the rib to slide back and rotate into an upright position. Extra pressure from the rail holds the rib against pinch rollers in order to slide it lengthwise until it is



justified by a precision hard stop. At this point an overhead mechanism grasps the top edge of the rib, moves vertically to clear the orienter, translates laterally 14 1/2 inches (the standard distance between ribs), and moves down again such that glue may be applied to the bottom edge of the first rib while a second is obtained from the orienter.

This proposed design was closer to an acceptable solution, but several modifications were considered. For example, in order to reduce the wear caused by sliding the rib toward the hard stop, a passive roller conveyor system could have replaced the rib support surface. Then, if driven, this conveyor could create the necessary longitudinal orienting motion for the ribs and eliminate the need for pinch rollers. While the locational accuracy of this system was better than that of the earlier design, it was still insufficient. Unpredictable combinations of vibration, friction and bounce induced by the rail pressure and roller rotation would limit precision. A possible means of improving this situation would be to constrain the trailing rib edge with a second hard stop. Then, if this hard stop was powerful enough to push the rib weight alone, the conveyor could be changed back into a passive system.

At this point, a final solution was beginning to evolve. In order to ensure that they were in fact feasible, while conceptual evolution and modification continued, research as to fabrication options and costs began. As a result of this simultaneous revision and research process, several new restrictions emerged which greatly increased task complexity, yet clarified its definition. The next two sections include a discussion of the newly discovered design factors and their implications for system cost, complexity, materials, motions, and control.

### **3.3 SYSTEM COST, COMPLEXITY AND MOTION CONSTRAINTS**

A primary goal of the orienter design was to minimize capital cost in order to maintain process practicality and competitive panel prices. As noted in the original task description, this meant that all unnecessary degrees of freedom had to be eliminated or incorporated with other systems wherever possible, overall part counts had to be minimized, and stock components needed to be employed where custom fabrication would consume excess time and/or money. Certain aspects of the earlier designs emphasized the implications of violating these constraints. For example, even after the second design mentioned here was modified to the point of utilizing two hard stops and a passive roller system, it was still far from optimal. The redundancy of the independent, precision linear hard stops and conveyor was undesirable. Plus, and the high part count of the conveyor and the unconventional size of each of these components made them difficult and time consuming to fabricate. Dealing with these components and the complications they introduced made it clear that any future design should not employ redundant and/or non-standardized parts if possible.

During fabrication research, gluing systems were found to be inherently expensive, but the innovations proposed by the earliest solutions were relatively too complex and costly to be truly feasible. Currently available gluing nozzle technology could precisely lay down beads of glue to moving surfaces, but the process of applying a bead to the bottom edge of OSB with an inverted, moving nozzle that ensured sticking and minimal dripping was deemed impossible by several glue dispensing vendors. One robotics company

affirmed that it was possible, but that the necessary linear system would cost an additional \$10,000 or more, requiring a more precise mechanical configuration and additional sophisticated sensors and damping systems, thereby becoming less reliable and possibly less accurate than any stationary setup. In light of these facts, and in order to limit the cost of the already expensive glue mixing and pumping system, it was decided that the preferable solution would one which employed a simply mounted and/or stationary bottom rib edge gluing head.

If the glue head was required to be stationary, then the rib would have to be moved over it. The overhead transport mechanism could be used to create this motion, but this would not be a desirable solution. Moving the entire overhead mechanism over the gluing head each time an individual rib was picked up would be neither time nor energy efficient, especially if the next rib was ready before this process was complete. Plus, in order to maintain a grasp capable of maintaining the rib position during a stapling operation that involved substantial impact forces, more than half the rib height would have to be secured by clamps generously interspersed along the length of the rib. Maneuvering a nozzle around the clamps posed a substantial risk of smearing glue on the mechanism components. In order to avoid these obstructions and keep glue away from both the overhead mechanism and orienter, the best time for applying glue would be as the rib was transferred between the two stations. Lengthwise motion was already necessary to achieve the accurate relative rib position within the overhead mechanism, and it could be generated in two ways. The most straightforward method would have been to provide a separate linear guide and drive system at each rib location within the overhead mechanism,

but this would introduce excessive redundancy. Therefore the preferable and cost effective solution was identified as one with a single independent linear motion control system that could engage and position all four ribs.

### 3.4 RIB GEOMETRY: HANDLING MECHANISMS, METHODS AND MATERIALS

The most critical physical constraint was the fact that ribs might be as small as 2" wide. Rollers could not support such a small rib, linear hard stops would knock this type of rib over, and if a solid conveyor was employed, vibration would inevitably topple the smallest ribs from their upright position. These facts made orientation and transfer to the overhead robot impossible for all previous designs. Figure 3.41 shows the actual range of geometries to be handled.

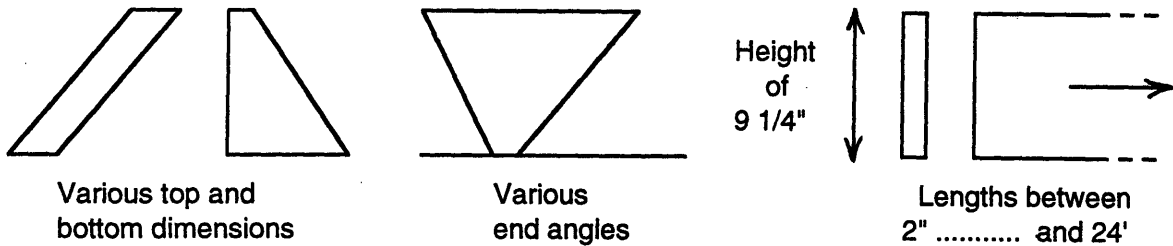


Figure 3.41: Range of Possible Rib Geometries

Large ribs presented no relatively significant design challenges, but the range of small ribs presented two major complications. First, the behavior of these ribs on a conveyor would be unpredictable. As shown in figure 3.42, aside from displacing the ribs along their length, the vibration of a conveyor would inevitably cause them to become mis-rotated. Therefore the orienter would have to be able to handle any possible

combination of these displacements. Robotic manipulation and/or extensive vision and sensor systems were out of the question due to cost and the harsh environment. Thus the final design would have to be one where the manipulation schemes were complex, but the necessary mechanical means were simple.

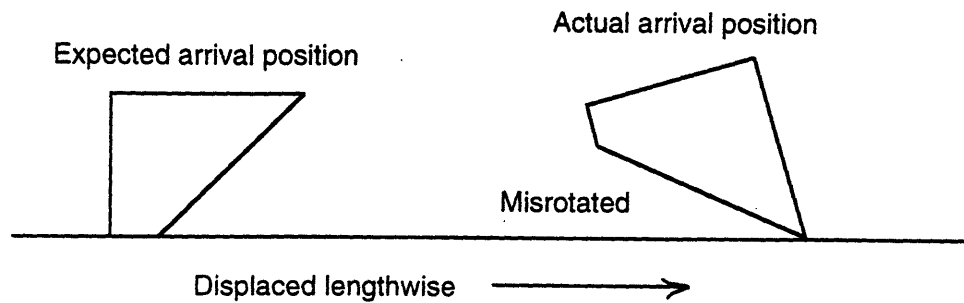


Figure 3.42: Illustration of Rib Mis-Orientation

The second complication was the varying "stability" of rib geometries. As shown in figure 3.43, ribs with parallel angled edges or a top dimension greater than that of the bottom, when rotated to a vertical position, would topple under the influence of gravity. For this reason, any successful system would have to include a means for supporting the ribs throughout the orientation and hand-off process. However, gripping methods would have to be simple. If the number of actuators, controllers and sophisticated joints was not kept to a minimum, the system would become more expensive, less robust, more prone to failure, and difficult to service.

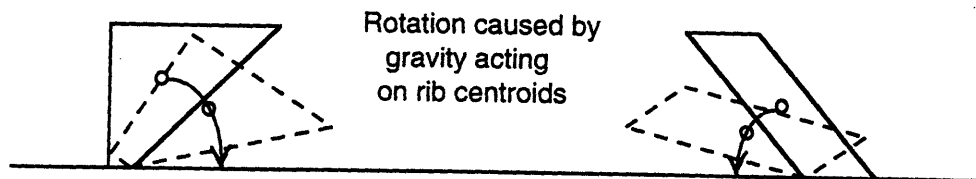


Figure 3.43: Rotation of Rib Geometries From Their Unstable Vertical Positions

Vacuum suction cups were considered as a possible gripping method, but were rejected for both the orienter and the overhead mechanism. In order to affectively grip the smallest ribs, suction cup diameters would have to be limited to 1 1/2". While smaller ribs might still be manageable under these conditions, handling larger ribs might be unreliable due to the unpredictable surface roughness of the OSB which could limit the ability to provide a consistent holding force with suction cups of this size. Also, mounting suction cups on components that would move 24' would be awkward as this would involve extensive valves and tubing. Finally, incorporating suction as a part of the overhead mechanism created the threat of glue contact which could damage the rubber suction element functionality. Any clamping device operating in the presence of glue would need to be removable and made of a robust, cleanable material.

The materials of any component contacting the ribs needed to be carefully considered. For example, the initially ambiguous location of the rib after arriving on the conveyor presented situations where a machine component would move while the rib remained stationary or vice versa. Due to the abrasive nature of freshly cut rib edges, the resulting sliding motion would cause wear on the contact surface unless it was made of an appropriate material. In order to accurately predict, follow or model any motion of the rib, there must also be minimal friction between the rib and all components it contacts. The resulting material requirements of maximum wear resistance and minimal friction are sometimes inherently contradictory. Since some plastics would provide lower coefficients of friction, metals such as steel were ruled out. The material that best satisfied wear,

friction and cost restrictions was found to be Ultra High Molecular Weight plastic. Hence it was the material of choice for the primary rib contact surfaces.

### **3.5 FINAL CONCEPT**

As a result of the considerable design evolution and research described in the previous sections, a final list of design specifications and guides emerged, as described below.

The orienting mechanism should be able to handle 2" to 24' ribs, the smallest of which will arrive on a conveyor substantially displaced and misrotated, and will be unstable once rotated to an upright position.

In order to maintain a minimal amount of friction and wear, all rib/orienter contact surfaces should be made of Ultra High Molecular Weight (UHMW) plastic.

The bottom edge gluing process should be performed by a single stationary head, preferably during the linear transfer of ribs from the orientation station to the overhead transport mechanism.

In order to avoid redundancy, a single servo system should be used to both create the precision gluing motion and orient the ribs longitudinally within in the overhead mechanism, and independent orienting mechanisms should not be dedicated to each rib.

All clamping methods should be as simple as possible, since any complication will lead to an expensive, less robust, and redundant system.

Under these constraints, the final orienting, gluing and transport design evolved. Some examples of rejected approaches are discussed in Appendix B. Figures 3.51-3.57 illustrate the final concept and follow the steps involved in receiving a rib from the conveyor, maintaining its stability while rotating it to the upright position, applying glue to its edges, locating it longitudinally, and securing it for transport to the rib/bottom face assembly station. (Note: Figures are conceptual only.)

Step 1: Ribs arriving from the conveyor roll onto an orienting platform made of UHMW.

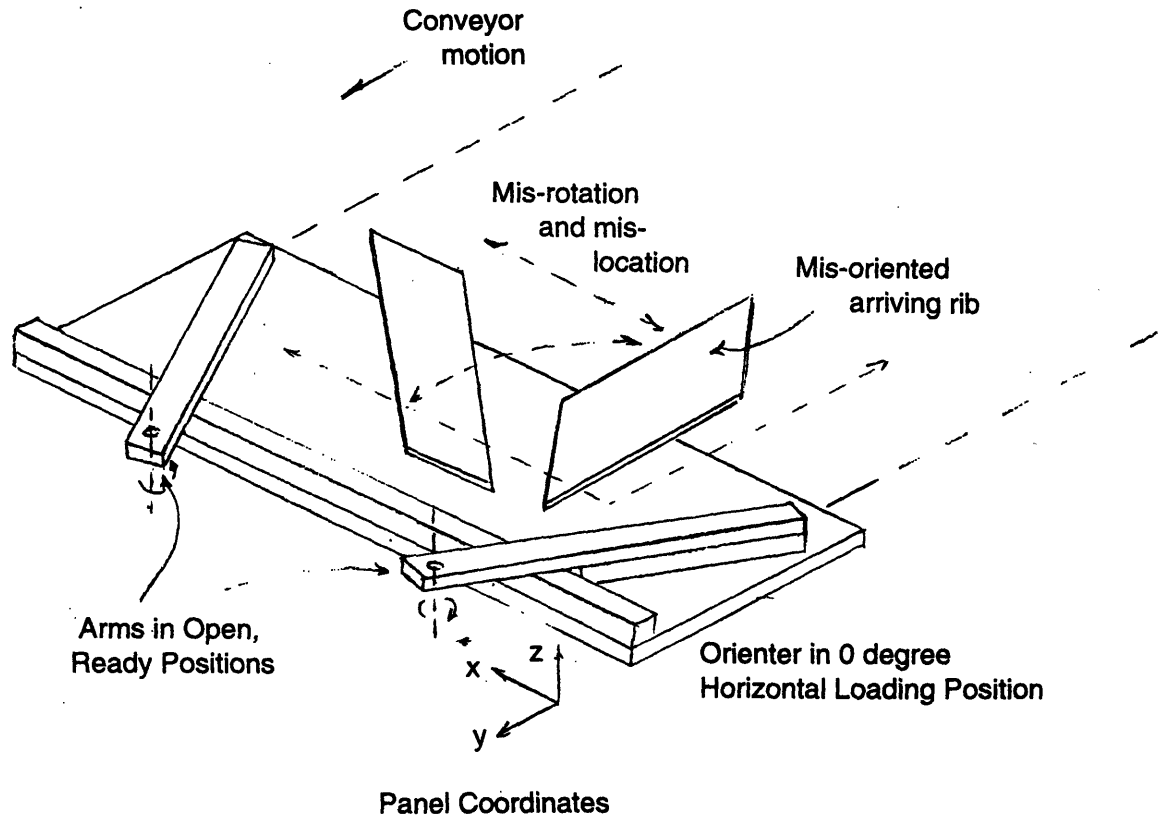


Figure 3.51: First Rib Orientation Step



Step 2: The platform rotates pneumatically from the horizontal to about 45 degrees, or half way to the upright assembly position. This causes large ribs to easily slide to rest on the supporting rail, and they will only be out of position lengthwise. Smaller ribs, which may have been mis-rotated by the conveyor will slide and/or rotate on the low friction platform from their original indeterminate, unstable, and longitudinally and rotationally unknown position, to a stable and semi-determinate position in the orienter.

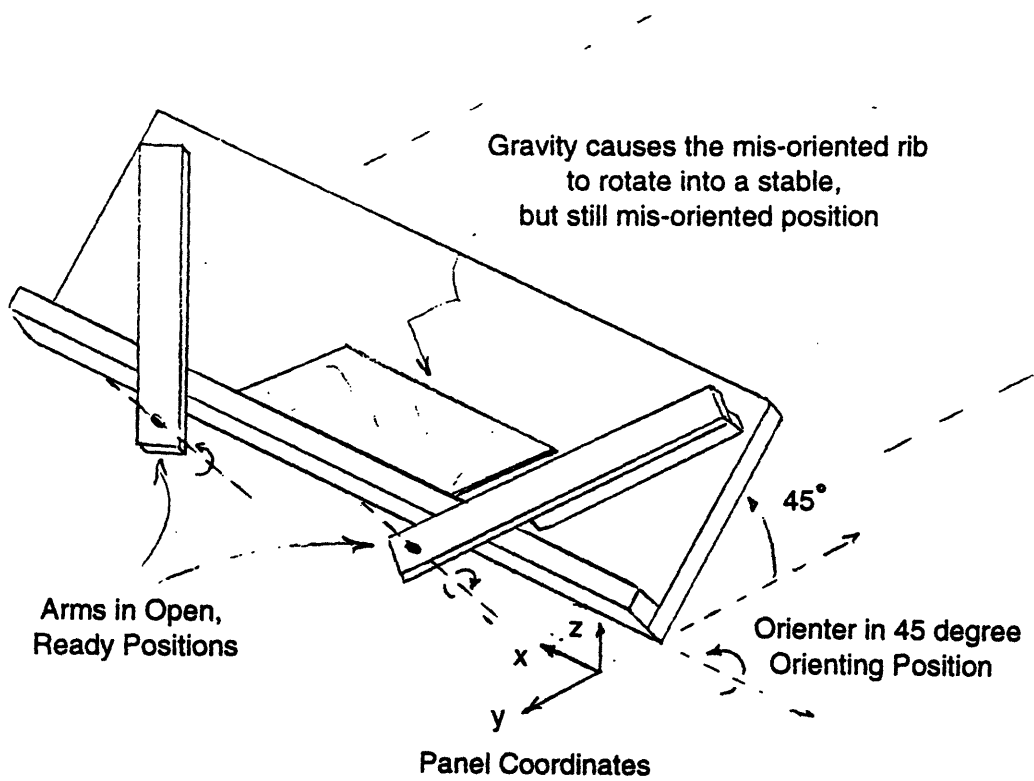


Figure 3.52: Second Rib Orientation Step

Step 3: The rib is approached from the right by an angled arm which slides it into a more determinate position along the length of the platform. For larger ribs, the angle of the right arm will match that of the rib, and guide it into contact with the left arm. This will complete the orientation of large ribs. However, if the rib is small enough to have been mis-oriented, a series of coordinated steps is performed by the left and right arms. These steps, based entirely on the known rib geometry and desired final configuration alone, will rotate and slide the rib into its final assembly position on the platform. Chapter 4 contains a detailed description of this manipulation process and development.

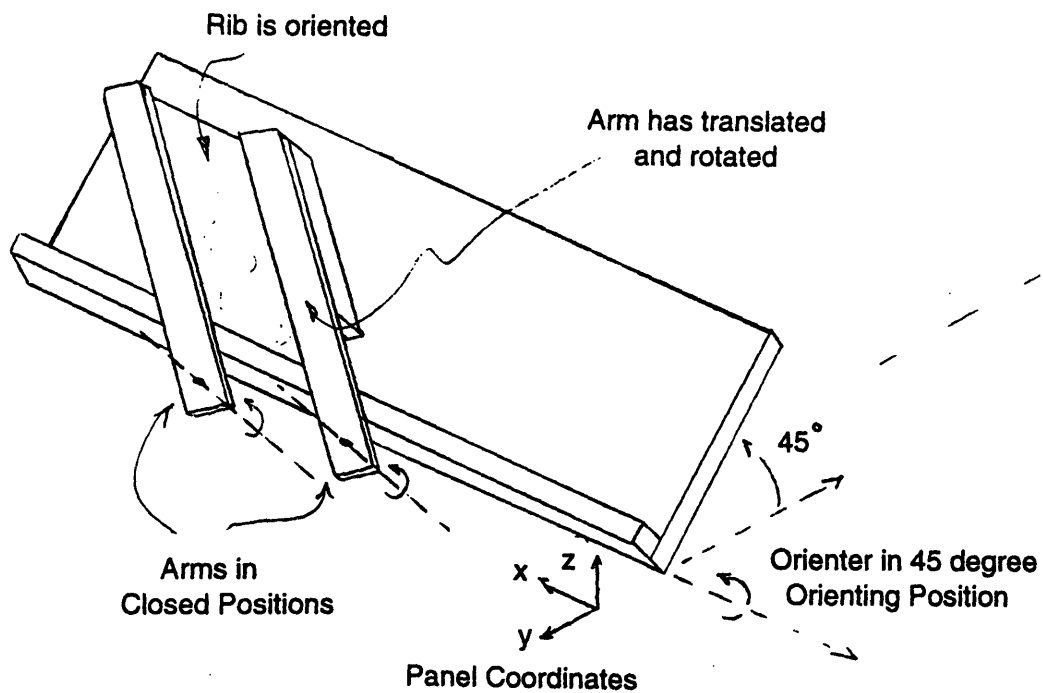


Figure 3.53: Third Rib Orientation Step

Step 4: The platform rotates to the 90 degree or vertical assembly position. At this point, a part of the platform retracts and a clamp from the overhead mechanism grasps the top edge of the rib.

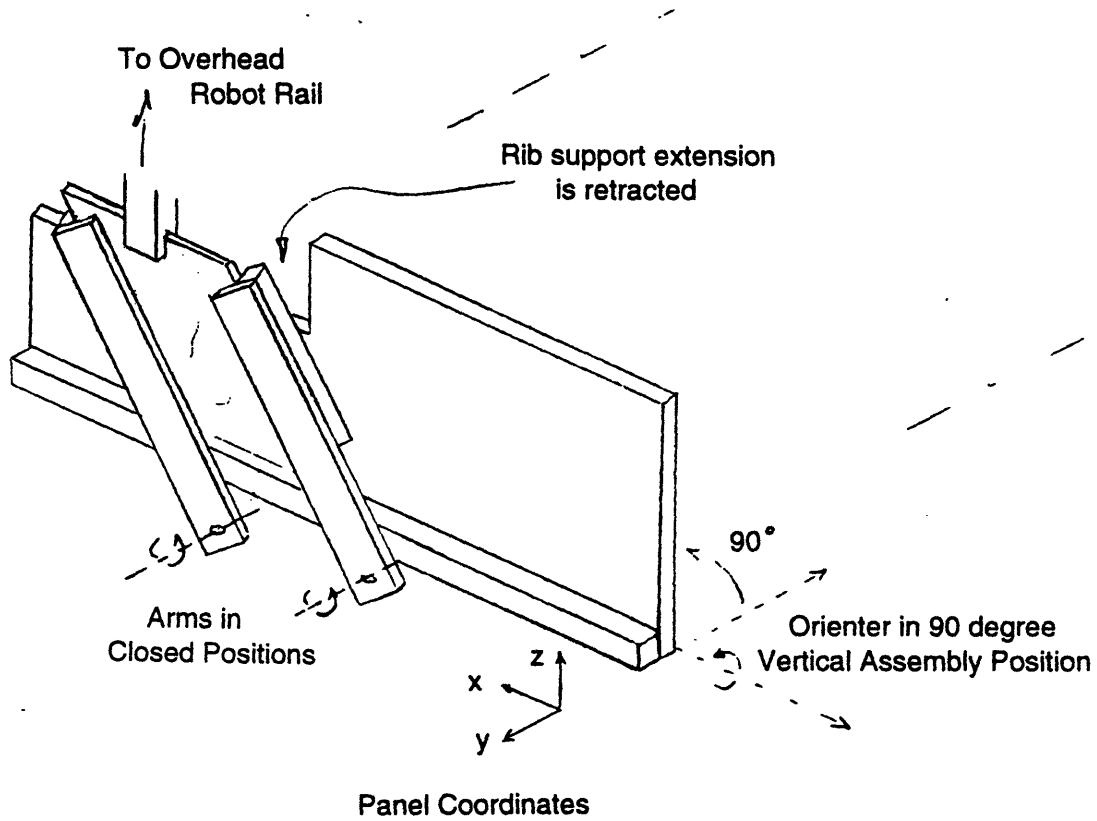


Figure 3.54: Fourth Rib Orientation Step

Step 5: The left arm rotates out of the way and the guide clamp also engages the rib. This servo driven guide slides the rib out of the orienting mechanism and past another mechanism which applies glue from top to bottom of the first emerging rib edge.

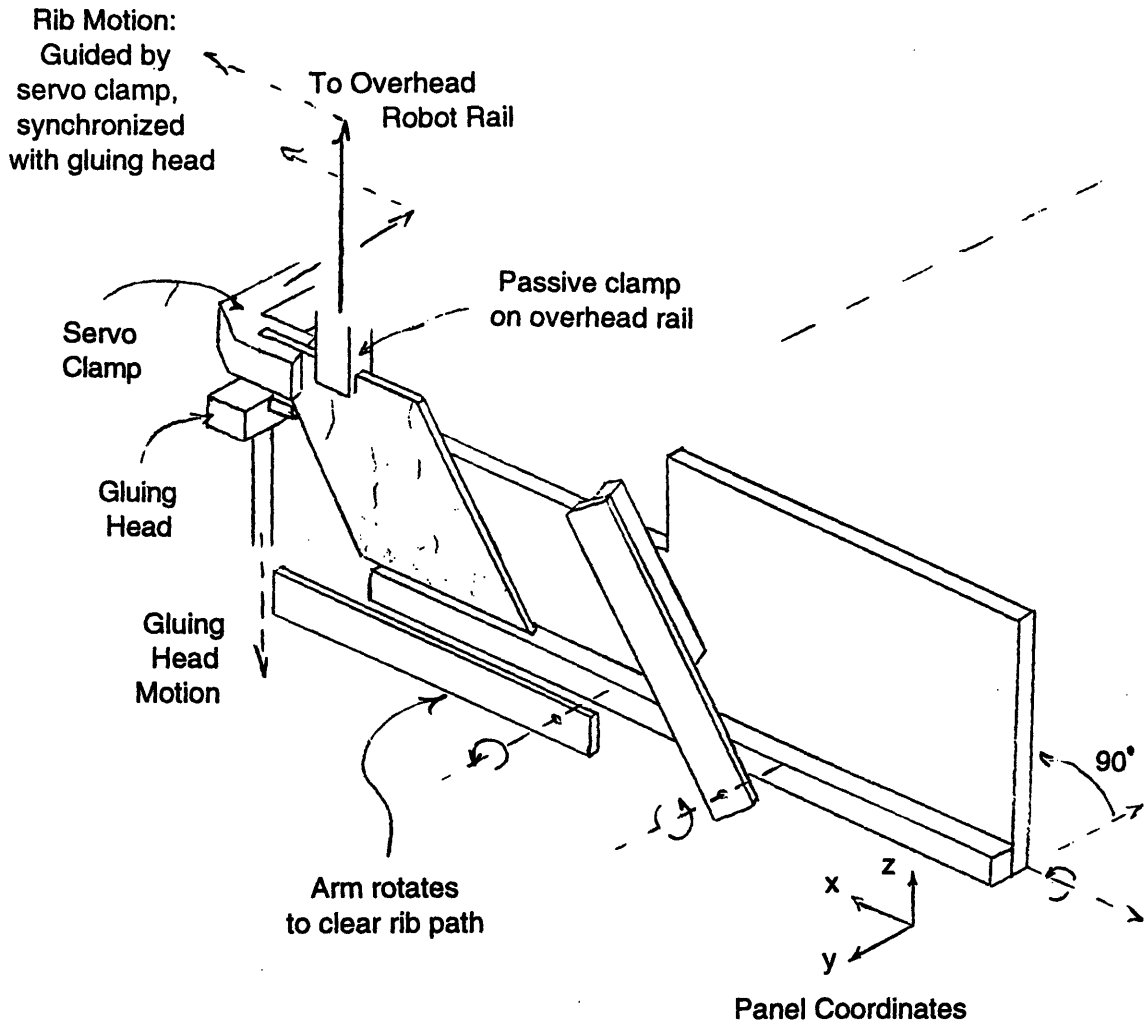


Figure 3.55: Fifth Rib Orientation Step

Step 6: Longitudinal motion pauses after the first rib edge is completely glued. The gluing mechanism then rotates and locks into a second position, such that glue may be applied to the bottom rib edge when motion resumes. Additional passive clamps engage the top edge of the rib to support it during travel into the overhead mechanism. Just before the rib completely leaves the orienting station, the servo assist pauses again for the gluing mechanism to rotate to its third and final position for gluing the trailing edge of the rib.

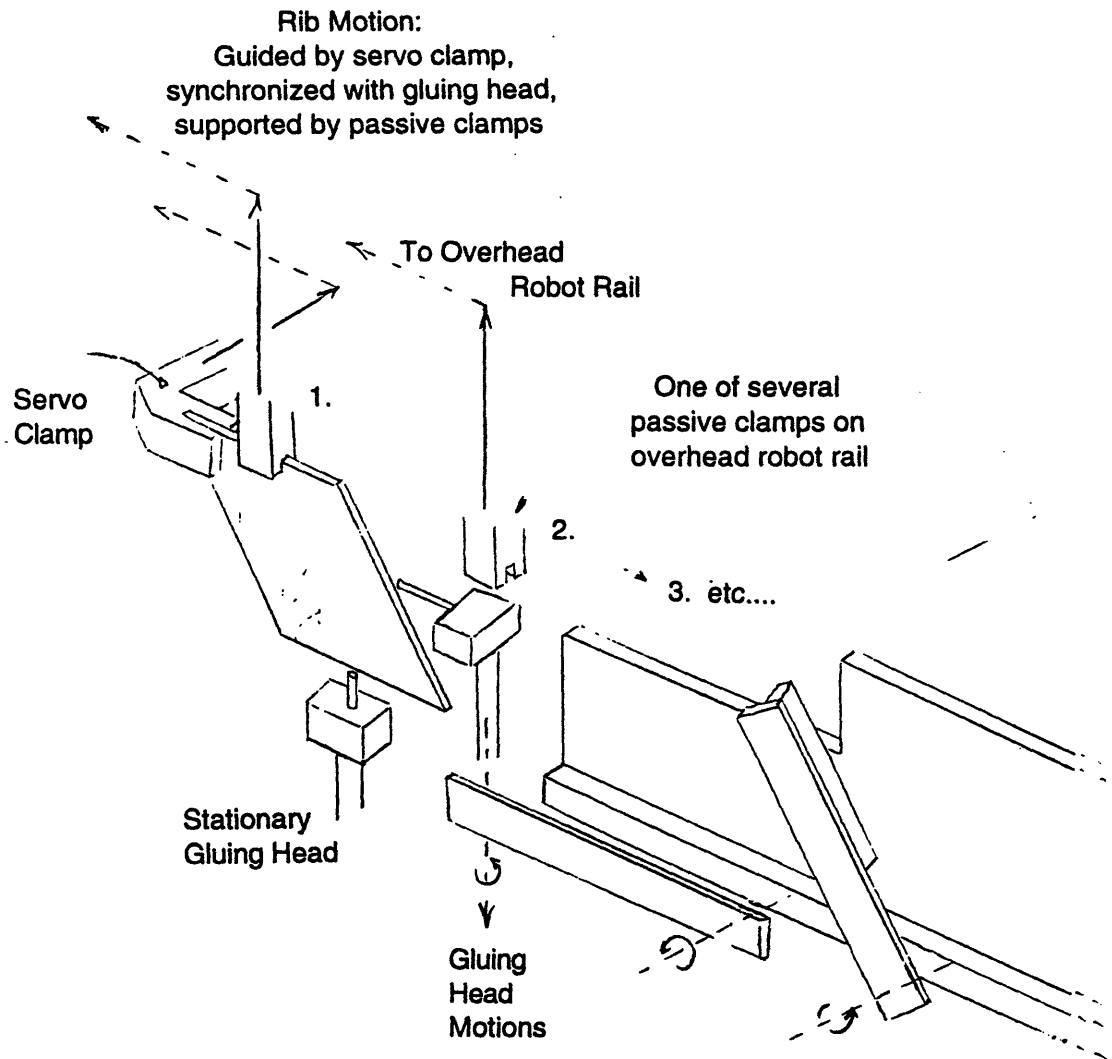


Figure 3.56: Sixth Rib Orientation Step

Step 7: The servo guide precisely positions the rib within the overhead mechanism, and stationary clamps within the mechanism secure the rib in place. The servo mechanism then releases the rib and returns to the orienter to grasp the next rib. Once all necessary ribs have been processed, the overhead mechanism transports them to the assembly station for stapling to the bottom face. Once the assembly process is complete, the overhead clamps move the partly assembled panel forward into the end cap attachment station. After the first end cap is attached, the overhead mechanism returns to collect the next set of ribs.

### **3.6 ORIENTER PROTOTYPE**

#### **3.6.1 INTRODUCTION**

The next step in developing the proposed orienting, gluing and transportation technology was to fabricate and test the most critical stations and processes involved. Since the orienting mechanism was of primary importance, it was decided that prototyping efforts would be best spent on this station.

Before any real time or money was dedicated to developing a working system, a rough mock-up was constructed to test the concept and feasibility of manipulating ribs with the proposed arms. Of the entire range of possible rib types, four different representative geometries were cut from OSB and tested. The small, hand-driven wooden prototype used is illustrated in figure 3.61.

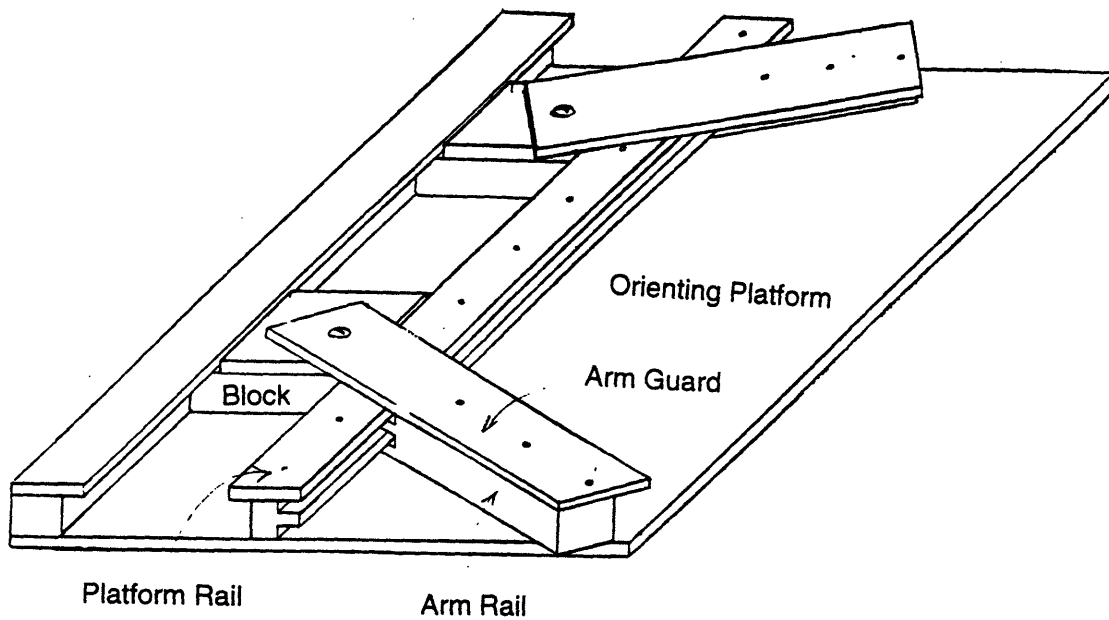


Figure 3.61: Hand-driven Proof-of-Concept Orienter Prototype

After several trials with each rib, they were in fact affectively handled by the arms, and there seemed to be a correlation between the rib geometry and the series of steps required to achieve the final desired position. The correlation was not obvious, but was sufficient to warrant the construction of a second, automated prototype.

Building a prototype capable of handling the largest 24' rib was considered unnecessary at this stage. Longer ribs did not look to present a substantial orientation problem, and future primary design considerations and inevitable modifications would most likely revolve around the orientation of small ribs. In addition, the costs of non-standard length linear guides and pulley systems and machining and handling 25' metal stock were substantially greater than that required to construct a smaller system. Therefore, the preliminary plan was to construct a prototype of 1/8 the total length to develop the control system for orienting small ribs alone. However, in anticipation of the

future production of a scale model, it was decided that every part of the system, with the exception of the length, should be made to scale and that the motors and mounts should be designed such that they could be interchanged with a larger system.

### 3.6.2 FABRICATION

First, the stock, long-lead components were determined and ordered. The linear guide system needed to be able to withstand the moment induced by the motor weight at all orientation angles, so a double rail system was chosen over a single rail system, and attained from THK. The pulley system components were attained from Browning, along with the necessary pillow blocks and retaining rings. Before purchasing motors, the required forces and velocities for manipulating the entire range of ribs were calculated as shown in Appendix C. Four PMI ServoDisc DC motors were available from Servo Systems which were of a higher quality and more powerful than absolutely necessary, but their price made them well suited for the situation. Next, controllers were needed. In order to reduce cost, instead of purchasing a more expensive four axis controller, one Omnitech Robotics single axis (MC1000) and one triple axis (MC3000) controller were also attained from Servo Systems, along with two 36V power supplies, two 15V power supplies, four Advanced Motion Controls 25A8 amplifiers, and sufficient cabling. A parts list of all components is included in Appendix D.

Next, the mechanical drawings of all other components were finished. All assembly and part drawings are included in Appendix E.



Once the components were ordered and the drawings were complete, fabrication began. The limited amount of time and money which could be spent on the prototype affected the design and construction in several ways. Instead of using steel to construct an open truss base and frame, a high quality 3/4" oak plywood was employed. Instead of designing a pneumatic rotational system for the joint between the base and frame, this degree of freedom was designed to be manually driven, and the simple steel-in-plywood shaft/base interface was deemed a sufficient bearing. Instead of using solid spacer blocks with incorporated bearing mounting holes, aluminum C channel separated standard pillow blocks from the plywood frame. The mounting support blocks for all motors were precision machined in steel such that they could be employed in a final, full scale system, but instead of giving them an oxide finish, they were coated with rust resistant paint.

Once all the parts were fabricated, the prototype was assembled. Figure 3.62 is a photograph of the actual prototype.

The UHMW sheet used for the platform surface had self-inflicted ripples at its edges, caused by its tendency to bow around its longitudinal axis. This called for additional fastening points to improve surface flatness and avoid hindering rib and arm motions. With a few other minor adjustments such as this, everything fit together.

The next step in the prototyping process was to develop the control system, as described in the following chapter.

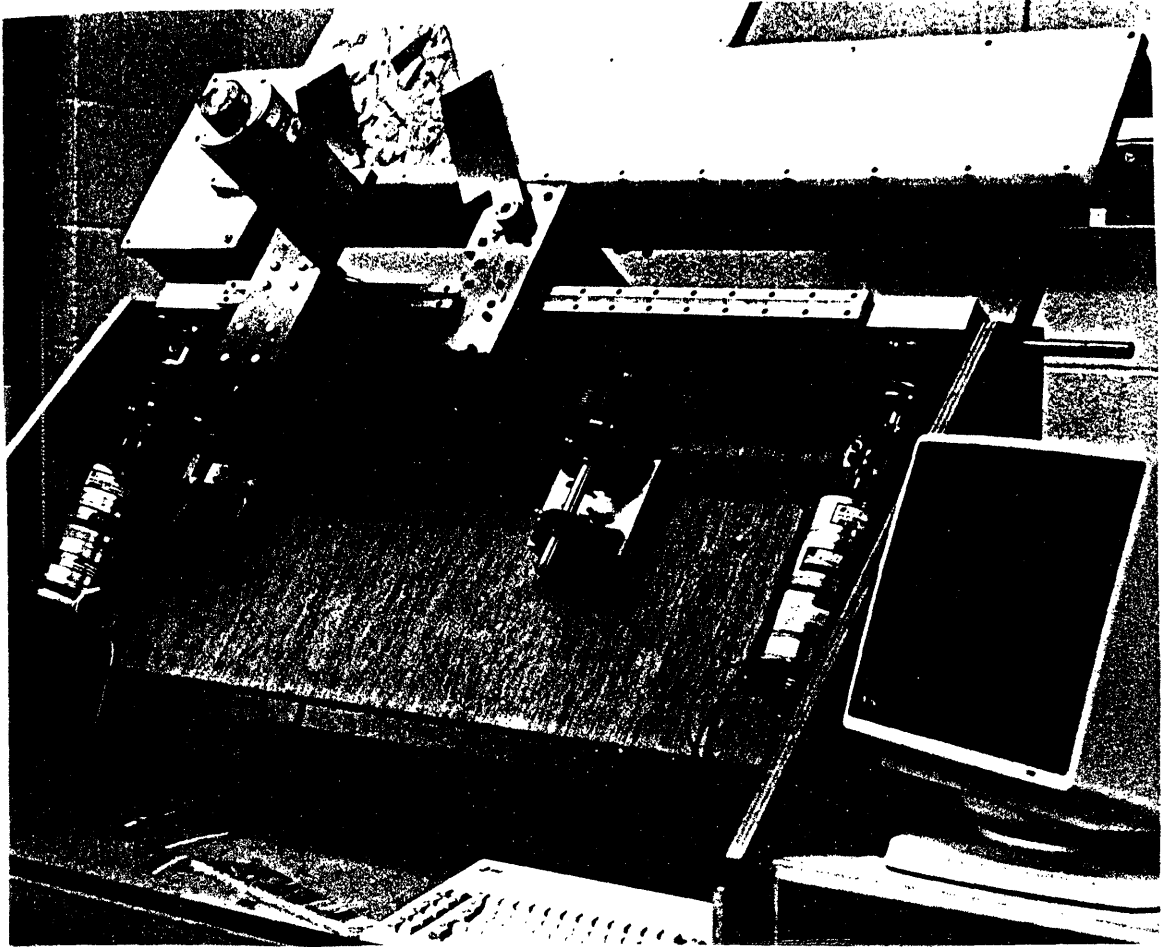


Figure 3.62: Photograph of the Working Orienter Prototype

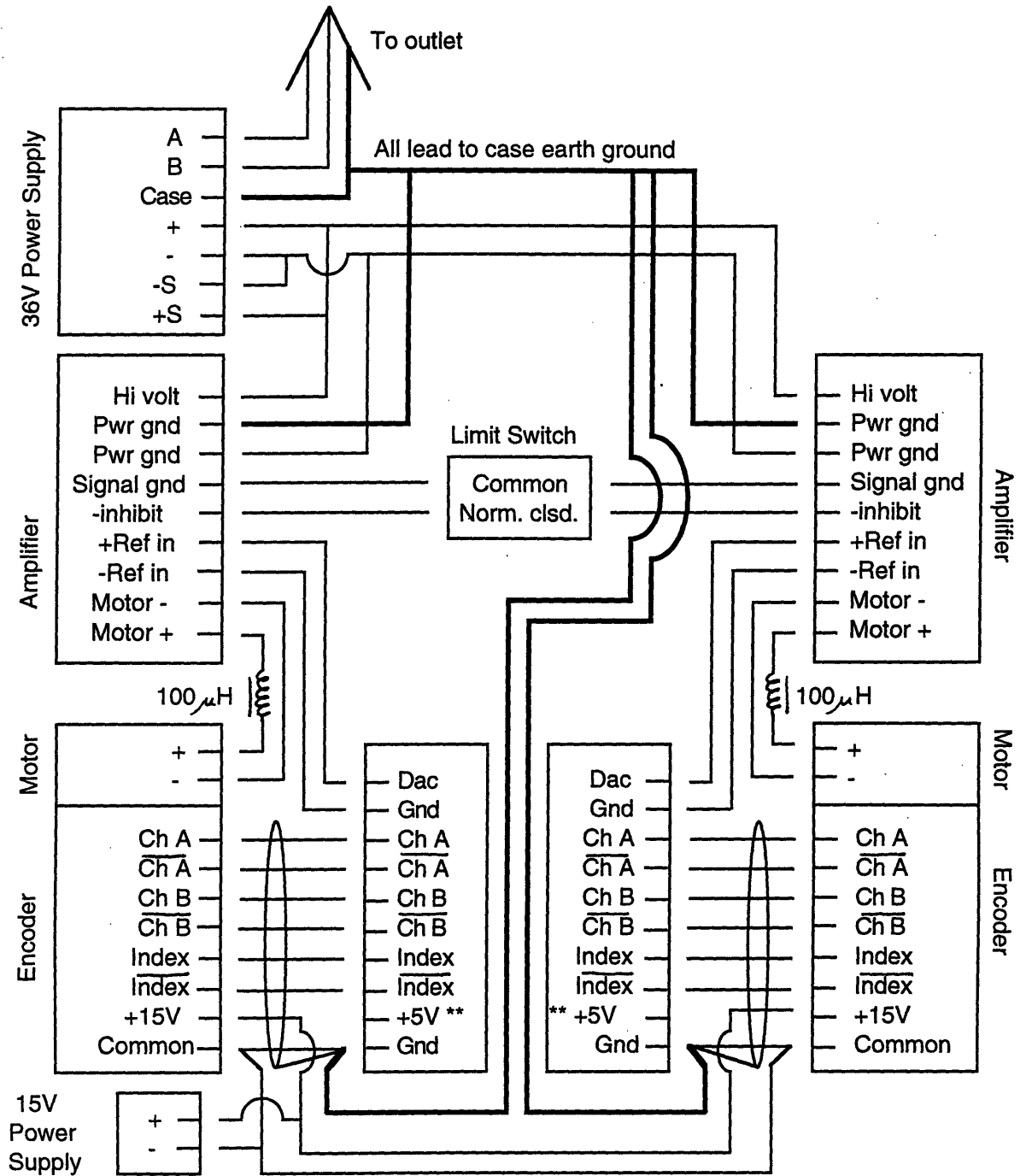
## **Chapter 4: RIB ORIENTER: CONTROL**

### **4.1 PROTOTYPE CONTROL SYSTEM SETUP**

The prototype control system utilized a Omnitech MC1000 single-axis and MC3000 triple-axis control card installed in a DOS-based personal computer, along with four 25A8 Motion Controls amplifiers, two 36V power supplies, two 15V power supplies, and four PMI ServoDisc motors. Figure 4.11 shows the schematic for one of the two identical halves of the system.

Motor axis assignments were decided as follows: The motion of the two blocks in the system was semi-redundant, and it was hoped that the control of all arm separation might be provided by the motion of a single block. In order to test this possibility, the left block would initially be motionless. Therefore this motor was assigned to the MC1000. This left the MC3000 X-Y-Z axis coordination capabilities for the remaining motors. The rotation of the right arm was designated as the X axis, the linear motion of the right block was the Y axis, and the left arm rotation was the Z axis.

Once the system was completely hard wired, it could be tuned. Internal amplifier gains were set, motor drifts were centered, and current limits were set to allow unrestricted motion at 20 qd/st (quadrature counts per sample time) for the unloaded motors. Once all bugs were eliminated from the system, full attention was turned toward the control algorithms.



\*\*Note: Pull-up resistors between +5V and all channel inputs

Figure 4.11: System Schematic: One of Two Identical Halves

## 4.2 RIB ORIENTATION APPROACH: EVOLUTION

The original plan for algorithm development was to analyze each possible rib geometry, predict and model the frictional and gravitational forces involved in different manipulations, and thereby derive a series of steps which could move any rib into its desired position no matter how it arrived at the orienter. First, in order to prove that an orientation strategy exists, possible failure conditions need to be identified and proved impossible due to corresponding rib shapes or motion strategies.<sup>7</sup> This is effectively the inverse of a preimage or "definition of orientations in state space from which goal attainment is both guaranteed and recognizable..."<sup>8</sup>

Next, the motion of the ribs needs to be modeled. The orienter strategy may be described as a pushing operation. This positioning method has been identified as preferable where, like the ribs on the orienter platform, "the initial position and the goal position share a common support surface."<sup>9</sup> In order to determine the motion of the ribs during orientation, the motion and force constraints may be represented in several ways.

Kevin Lynch presents kinematic contact modes and slider motions, as well as force representations like support friction, which identify a "set of stable pushing operations for a given contact configuration."<sup>10</sup> However, the conditions under which this methodology

---

<sup>7</sup> Idea discussed in a personal meeting with Thomas Lozano-Perez.

<sup>8</sup> Lozano-Perez, T., Mason, Matthew T., and Taylor, R. H. "Automatic Synthesis of Fine-Motion Strategies for Robots." *International Journal of Robotics Research*. 1984. Vol. 3, No. 1, p. 3-24.

<sup>9</sup> Mason, Matthew T. "Mechanics and Planning of Manipulator Pushing Operations." *International Journal of Robotics Research*. Fall, 1986. Vol. 5, No. 3, p. 53-57.

<sup>10</sup> Lynch, Kevin M. "The Mechanics of Fine Manipulation by Pushing." *Proceedings of the International Conference on Robotics and Automation*. May 12-14, 1992. Nice, France.

may actually be applied are limited.<sup>11</sup> Also, some rib motions will be unstable. In this case, forces and motions may be modeled graphically. The concepts of moment labeling and friction cones<sup>12</sup> and the appropriate mapping thereof<sup>13</sup> for representing the forces generated by one or more frictional contacts are presented by Mason in several different papers.

Regardless of how the operation is modeled, the purpose of any orienting process is to "act on the part in a manner that reduces uncertainty. Uncertainty is reduced whenever an action decreases the set of possible configurations of the part."<sup>14</sup> Mason and Erdmann suggest the method of dropping an object on a plane and forcing it to slide along a wall and/or move into a corner under the influence of gravity.<sup>15</sup> This idea correlates very well to moving a rib onto the orienting platform, tilting the platform 45 degrees to invoke gravitational properties, using an arm to slide it along the rail, and then forcing it into a corner created by the second arm. However, Mason and Erdmann only discuss the modeling and predicting of motions and forces for objects where geometry was ignored, and this limited the direct application to rib orienting, where geometry was known and

---

<sup>11</sup> Lynch, Kevin M. "Planning Pushing Paths." International Conference on Advanced Mechatronics. August 2-4, 1993. Tokyo, Japan.

<sup>12</sup> Mason, Matthew T. "Two Graphical Methods for Planar Contact Problems." IEEE/RSJ International Workshop on Intelligent Robots and Systems. November 3-5, 1991. Osaka, Japan.

<sup>13</sup> Brost, Randy C., and Mason, Matthew T. "Graphical Analysis of Planar Rigid-Body Dynamics With Multiple Frictional Contacts." Fifth International Symposium on Robotics Research. 1989. Tokyo, Japan.

<sup>14</sup> Erdmann, Michael, Mason, Matthew T., and Vanecek, George Jr. "Mechanical Parts Orienting: The Case of a Polyhedron on a Table." *Algorithmica*. 1993. Vol. 10, p. 226-247.

<sup>15</sup> Erdmann, Michael A., Mason, Matthew T. "An Exploration of Sensorless Manipulation. *IEEE Journal of Robotics and Automation*." August 1988. Vol. 4, No. 4, p. 369-379.

would be the basis of all manipulation strategies. However, their ideas of a contact analysis phase and a search phase helped to guide the formulation of a rib orientation scheme.

In a slightly modified version of their approach, each rib orientation algorithm could be developed by first determining the number of configurations or contacts presented by a particular rib geometry, and then modeling different arm motions and predicting their effects on each rib position. From this information, a series of steps could be generated such that all possible contacts could be resolved into one determinate position. Then, from this position, the final rib orientation could be achieved.<sup>16</sup>

Before this theory was applied, an "exerciser" or interpreter program provided by Omnitech Robotics was used to create some basic programs for orienter control. The first of these were for system startup, calibration, motion demonstration, and shutdown. The majority of these programs were also modified and improved later to be implemented in the control code. Appendix F explains the initial programs, and Appendix G contains the commented code that employed them in their final form.

Next, the general behavior of ribs within the orienter was tested. Basic arm motions and the corresponding rib reactions from the wooden prototype were studied and then roughly translated in to control commands for the working prototype. This process furthered task and system familiarity, and successful experimental programs for orienting the most simple rib geometries were created.

---

<sup>16</sup> Also similar to a method discussed in a personal meeting with Thomas Lozano-Perez.

System measurements were also attained by observing these programs. The arm and block ranges of motion were determined, the conversion between commanded motor quadrature counts and linear or angular displacement was recorded, and a suitable combination of maximum velocity and acceleration for defining trapezoidal velocity controlled motion was found.

At this point, earlier ideas as to the optimal approach for finding a universal rib orientation scheme were beginning to change. Considering that orientation steps for one specific rib geometry could be modified and applied to ribs with similar dimensions, certain "families" of ribs could be identified and addressed by a single general algorithm. The slight difference in required block and arm angles or displacements could be correlated to and calculated from their individual geometries. Then, each rib type would only be separated from the next by differences in rib end angles. If the angle at which one rib family and corresponding rib orienting algorithm merged into the next could be determined, then all ribs within and between them could be oriented. If enough representative ribs, orientation schemes, and merging points were found, then the entire orientation control would be complete. For this reason, orientation algorithms for more complex geometries were developed through significant physical hypothesizing and testing.



## **4.3 RIB ORIENTATION APPROACH: DETAILED TASK DEFINITION**

### **4.3.1 CONCEPTUALIZATION AND ORGANIZATION**

The first step was to organize and define the boundaries between discrete rib types present within the wide range of possible geometries. At this point, several had already been identified and were simply separated from each other by variations in end angles. Each rib type was assumed to have its own orientation method, and was approached accordingly. It was known that the variations in end angles would affect the top and bottom dimensions and the gravitational behavior of each rib, and the goal was to identify the points at which these variations became critical and separated one rib type from the next. While some conceptualizations successfully identified the discrete rib types and provided accurate pictorial descriptions of the angle-induced geometry changes and relations, not all lead toward a more universal algorithm. Also, early orientation tests uncovered some less obvious similarities and affective rib classification and manipulation methods that sometimes depended on pre-orientation mis-rotation.

Figure 4.31 illustrates the final conceptual organization. Here, the flow of geometry changes is linear and roughly separated into two main types. One type is defined as symmetric, as it presents the same geometry if rotated 180 degrees around the vector extending from its centroid and perpendicular to its face. This makes it predictable and simple to orient. The other type is more bulky and asymmetrical by this definition, with protruding angles which dominate its gravitational behavior. This makes it more difficult to orient, but as described in more detail in Section 4.5.3, if subjected to

consecutive rotations, the rib will finally come to rest in a known contact position which is most stable due to its geometry.

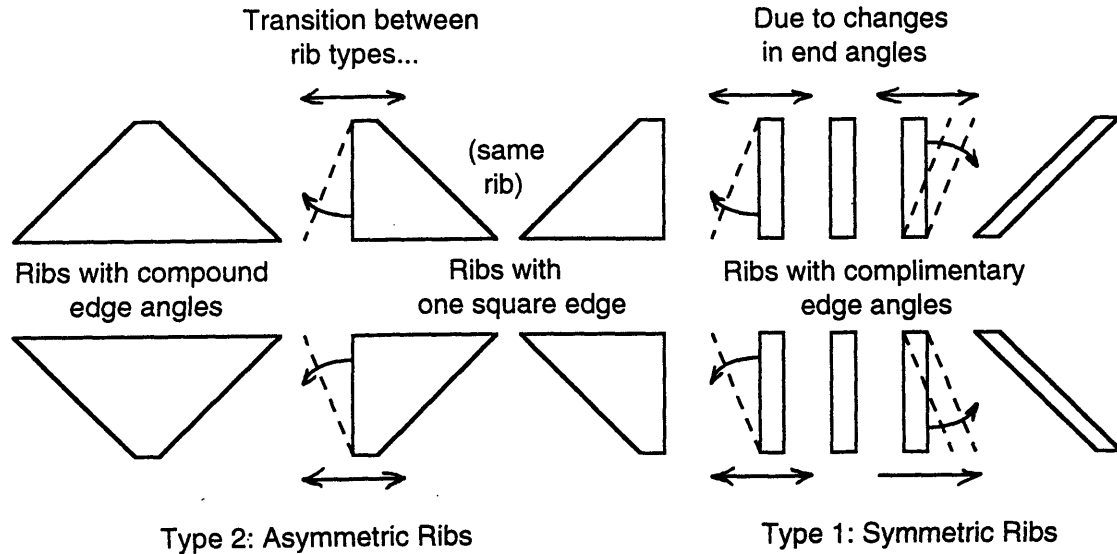


Figure 4.31: Conceptual Organization of Rib Types

#### 4.3.2 EFFECTS OF GEOMETRY VARIATIONS

The most critical aspect of the orientation problem was to determine which and how angle variations affected orientation steps within and between rib types. Ideally, small angle differences would only affect the final block and arm positions. However, in practice, they drastically affect the general rib orientation and pre-orientation mis-rotation capabilities.

The two worst case scenarios induced by geometry variations both occur with a square-ended rib. A 5", square-ended, symmetric rib may be mis-rotated up to 360 degrees before orientation without affecting the success of its corresponding algorithm. However, a 5 degree change in only one of its edge angles makes the difference between a

rib that can and cannot be handled. Mis-rotation of 10 degrees in either direction from the intended upright position causes a double square edged rib to fall to that side. Since a vision system is not present to supply this mis-rotation information, and the 0.8" difference between the top and bottom rib dimensions is probably not great enough to discernibly affect its behavior during manipulation, its orientation is nearly impossible. The second difficulty arises when the length of a double square-ended rib approaches 8.5" or 9". If the rib is mis-rotated by more than 45 degrees in either direction, the closer the length dimension gets to the 9.25" height, the more difficult it becomes to differentiate the top and sides from each other. In short, these two examples show that orientation becomes impossible without sufficient pre-orientation arrival information or the ability to discern rib geometry by its rotational behavior in the presence of gravity.

While double square-ended ribs present the worst case scenarios, small variations in angles also affect arrival requirements and subsequent orientation steps for all ribs. In order to ensure that ribs can in fact be oriented, constraints need to be imposed on either the geometry variations, the initial rib mis-rotations, or both.

#### 4.3.3 RANGES AND RESTRICTIONS OF GEOMETRY VARIATIONS

Considering the actual range of geometric possibilities provided helpful information, relieved some system requirements, and pointed toward a means of imposing realistic restrictions on rib mis-rotation. First, no matter what kind of roof is to be constructed, it will never include a panel that requires a small rib with two square ends. Only panels for flat roofs require double square-ended ribs, and in this case they will all be

considerably over 2' long. This will keep them from becoming significantly mis-oriented by a conveyor and will thereby simplify their positioning.

The next significant fact is that, apart from the longest ribs whose final configurations are not yet determined, all ribs arriving at the orienter will be symmetric or nearly symmetric. This means that for small rib geometries defined by a rib lying horizontally (not in elevation), the right and left angles measured from the outside horizontals to each separate rib edge will be complimentary. If the metal strapping discussed in Section 2.2 is employed, then even the longest ribs will be symmetric in the same way. Figure 4.32 illustrates this fact with an elevation view of a standard roof.

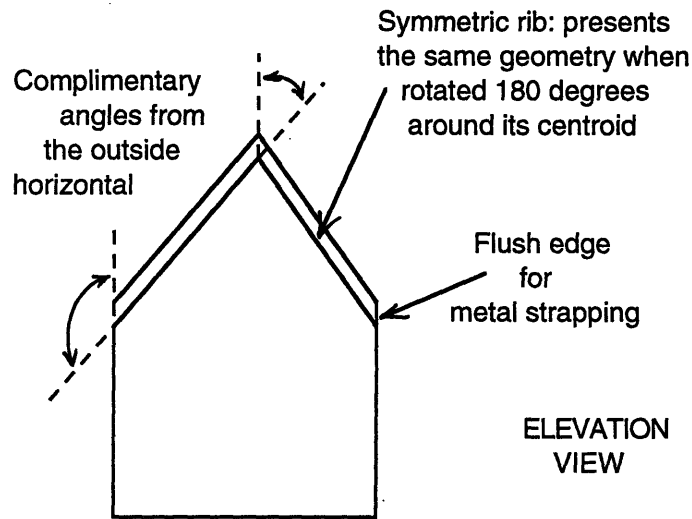


Figure 4.32: Illustration of Symmetric Rib Created by Metal Strapping

The end geometry shown does not change as two equally pitched ridge lines come together in a hip or valley. This is because, regardless of the plan angle at which two ridge beams meet, the plane at which their respective panels meet must be perpendicular to the ground in order to ensure that the panel thicknesses are equal at the point of intersection. If the panels were not equally thick at the point of intersection, this would

result in an unacceptably uneven roof or ceiling seam. The requirement of equal panel thickness also governs the rib end angles of panels between two unequally pitched ridges. For example, the greater the roof pitch, the greater the cross section created by an intersecting perpendicular plane, and vice versa for low-pitched roofs. So, when a panel from a high-pitched roof section meets a panel from a low-pitched roof section at a hip or valley, the low-pitch panel must be slightly angled under the high-pitch panel. As illustrated in figure 4.33, this decreases the high-pitch cross-section, increases the low-pitch cross-section, and matches the panel thicknesses and angles at the intersection.

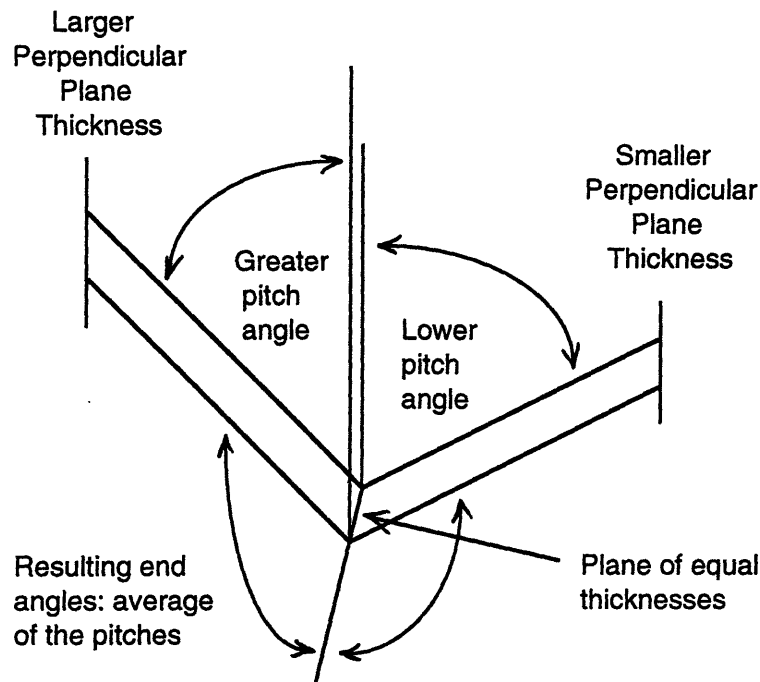


Figure 4.33: Effects on Panel End Geometry From Merging Two Different Roof Pitches

If the design of roofs is limited to one pitch, all ribs will be perfectly symmetric and therefore simple to orient. Regardless of the final eave end configurations, this simplicity will not be affected, since any rib that is long enough to reach the eave will be long enough such that end geometries will not affect orientation capabilities. Thus, ribs with just one

square end or compounded end angles are also eliminated from the list of small ribs that must be oriented. However, limiting roof designs to a single pitch is self defeating since the MIT roofing system was primarily developed to allow for creativity and diversity in roof design. Therefore, the possible range of roof pitches and corresponding symmetric or virtually symmetric rib geometries needs to be determined.

The variables determined by roof pitch include living space size, structural requirements, and material consumption. Because of the desire to create livable attic space, and the fact that as the roof pitch decreases, the ridge beam approaches a structurally undesirable (flat) configuration, the minimum roof pitch is around 7/12 or approximately 30 degrees. This results in rib end angles of 60 and 120 degrees from the outside horizontal. Roof pitches will most likely never exceed 13/12 or 50 degrees, since making the roof steeper gains little living space relative to the increased material consumption. A 50 degree roof pitch results in rib end angles of 40 and 140 degrees from the outside horizontal. Figure 4.34 illustrates the coordination of roof pitch, ridge beam shape, living space and material consumption.

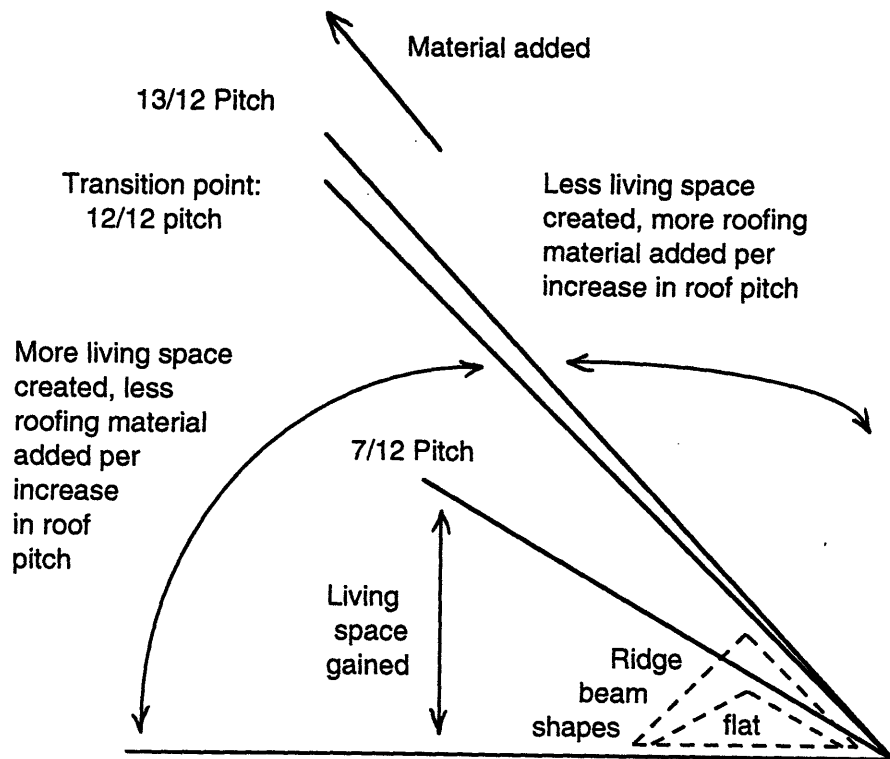


Figure 4.34: Coordination of Roof Pitch, Ridge Beam Shape, Available Living Space and Material Consumption

Thus, the entire range of ribs is defined. End angles range from either 40 to 60 degrees or 120 to 140 degrees from the outside horizontal. All ribs to be oriented by this application are symmetric unless a roof design included two different pitches. If this is the case, ribs required for hip and valley panels would be virtually symmetric: the edge facing the ridge beam will have the angle corresponding to the original pitch, and the other edge, facing the hip or valley, will have an angle which is the average of the two mating pitches.

All ribs created by this range of roof pitches were found to be within the currently established orienter capabilities. Symmetric ribs presented no difficulty whatsoever. Ribs with one square end or compounded end angles of 60 or 120 degrees from their outside horizontal, though never addressed by this application, were also handled successfully.

Their orientation was more difficult and time consuming, but considering that the algorithm used to test this rib type was not completely refined, this geometry limit was considered reasonable. The same ribs, with end angles less than 40 or greater than 140 degrees from the horizontal could also be handled since their more characteristic angles facilitate orientation. As illustrated earlier by the double square-ended rib scenarios, it was most difficult to orient small ribs which were virtually symmetric. However, it was still possible to orient them. Their geometric instability created sufficiently characteristic behavior in the orienter, and while the entire 360 degrees of mis-rotation could not be handled, their maximum mis-rotation in both the clockwise and counter-clockwise directions could be calculated.

For this reason, a simple spreadsheet was developed such that the mis-rotation possibilities for all ribs could be determined. First, the rib centroid location was calculated. Then the maximum mis-rotation in the clockwise direction was calculated by balancing the centroid over the bottom right rib corner, and vice versa for the counter-clockwise direction. If the resulting mis-rotation angle was negative, (thereby implying that the rib would topple from its upright position under the influence of gravity) then the rib mis-rotation constraint was conservatively set at 90 degrees in that direction. Appendix J includes a sample spreadsheet along with a figure which explains these calculations. In addition, the commented code in Appendix G contains the centroid and balance equations used to notify the user of rib mis-rotation capabilities.



## 4.4 CONTROL CODE

### 4.4.1 DEVELOPMENT AND BASIC ORGANIZATION

Along with the exerciser program discussed earlier, a C code library of lower-level register commands was provided with the controllers. To maintain compatibility for their incorporation, the main orientation code was developed with a Borland Turbo C and C++ programming language compiler.

In order to make the code easy to modify and understand, it was generated in a hierarchical and modular fashion. First, a (ORIENT.C) file which controlled the main program flow was generated. Then, the control loops for the procedures called by ORIENT.C were developed (with motor command series omitted) and placed in CLOOPIF.H.

Next, the necessary basic controller level command procedures were generated. The early startup and shutdown programs were simply modified, translated into the C format, and placed in BCMDIF.H along with the controller command library and several smaller procedures written to simplify the program by incorporating the most commonly shared command patterns. This file was then included by CLOOPIF.H.

Also included in CLOOPIF.H was OALGIF.H, which contained the orientation algorithms themselves. Like the early startup and shutdown programs, most commands employed by the early orientation schemes were roughly modified and converted into code. However, the calculations of independent block positions and respective arm

motions also had to be incorporated. Section 4.5 contains more detailed descriptions and explanations of these calculations and orientation algorithms.

#### 4.4.2 PROGRAM FLOW AND USER INTERFACE

Figure 4.41 illustrates the code flow and organization, and Appendix G contains the actual commented code.

The program starts (`startup()`) with a title page that appears on the screen until the user presses return. Then, the user is asked whether s/he would like the orienter to run in standard or demonstration mode. If the user selects the demonstration mode, all functions are performed automatically, and the user is only allowed to modify motor speed and input numbers correlating to pre-selected rib geometries. Otherwise, in standard mode, the original code used to develop the orientation algorithms is executed, where any geometry may be entered, and several loop and command options may be designated by the user. Once the orientation mode has been chosen successfully, the user is consulted as to whether the motors are in their preliminary, closed startup positions. If the system does not pass the corresponding position check, the program will insist upon a 'no' answer from the user which will cause the system to enter its last shutdown loop. However, if the motors are in their closed positions and the user is ready to continue, a basic motor initialization is performed: all motor constants and positions are set to prepare the system for the first control commands.

The program then enters a calibration (`calib()`) loop. If the orienter is in demonstration mode, the user may adjust the motor velocities. Once this option is

completed or declined, the motors are automatically moved to their ready positions. Within standard orientation mode, the user may check motor calibration, reset motor constants and velocities, move the motors to their ready positions, or quit the program. Once the user is satisfied with all motor settings and has moved the motors to their startup positions, the geometry input loop may be entered. However, if the motors are not in their startup positions, the user is notified, and the loop will repeat until this has been done. Similar appropriate checks exist for all options within this loop.

In demonstration mode, the geometry input loop (`get_geom()`) presents a list of six options, including five different rib types that can be selected for orientation, or program exit. The user is prompted until an acceptable answer is selected, and then either the appropriate "hard-wired" dimensions are assigned to their respective variables or the system enters its last shutdown loop. In the standard orientation mode, the user is prompted for the bottom edge dimension in inches, and the two rib end angles in degrees. Appendix H contains an explanation of rib angle assignments. Once values for these three variables are entered and the user acknowledges that they are correct, if they lie within the acceptable range of rib dimensions, the loop terminates and the orientation loop is entered. Otherwise the user is forced to acknowledge that the dimensions are incorrect, at which point s/he may reenter the dimensions or quit the orientation program.

No matter the orientation mode, the orientation (`orient()`) loop determines the rib type by comparing the top and bottom dimensions, notifies the user of the geometric characteristics and mis-rotation capabilities, waits for an acknowledgment, and then performs the appropriate orientation algorithm. Within the orientation loops themselves,

if geometry that causes arm interference or other system errors has been entered, the orientation is aborted and an error message is given.

After every orientation, a wrapup (wrapup()) is performed. If the orienter is in demonstration mode, the user is asked if s/he would like to continue. If so, the arms are opened, the user is requested to remove the previous rib from the orienter, and the calibration loop is reentered. If not, the final shutdown loop is entered. In standard orientation mode, the user may choose between opening the orienter arms to release a trapped rib (i.e., one with a smaller top dimension), continuing with calibration for orienting the next rib, or quitting the program.

All the procedures described above, excluding Startup, are included in a loop which continues until the user opts to exit the program. At this point, a final (lastchk()) procedure is executed. Both demonstration and standard modes are virtually the same; standard mode just offers more options. Here, the user is prompted as to whether there is a rib in the orienter, and mode-appropriate options are presented and executed. Once this step is complete, or if there was originally no rib in the orienter, the user is prompted as to the possibility of a standard shutdown. An affirmative response sends the motors back to their original closed positions and exits the program. Otherwise, the program quits without moving the motors and refers the demonstration mode user to an error guide sheet.

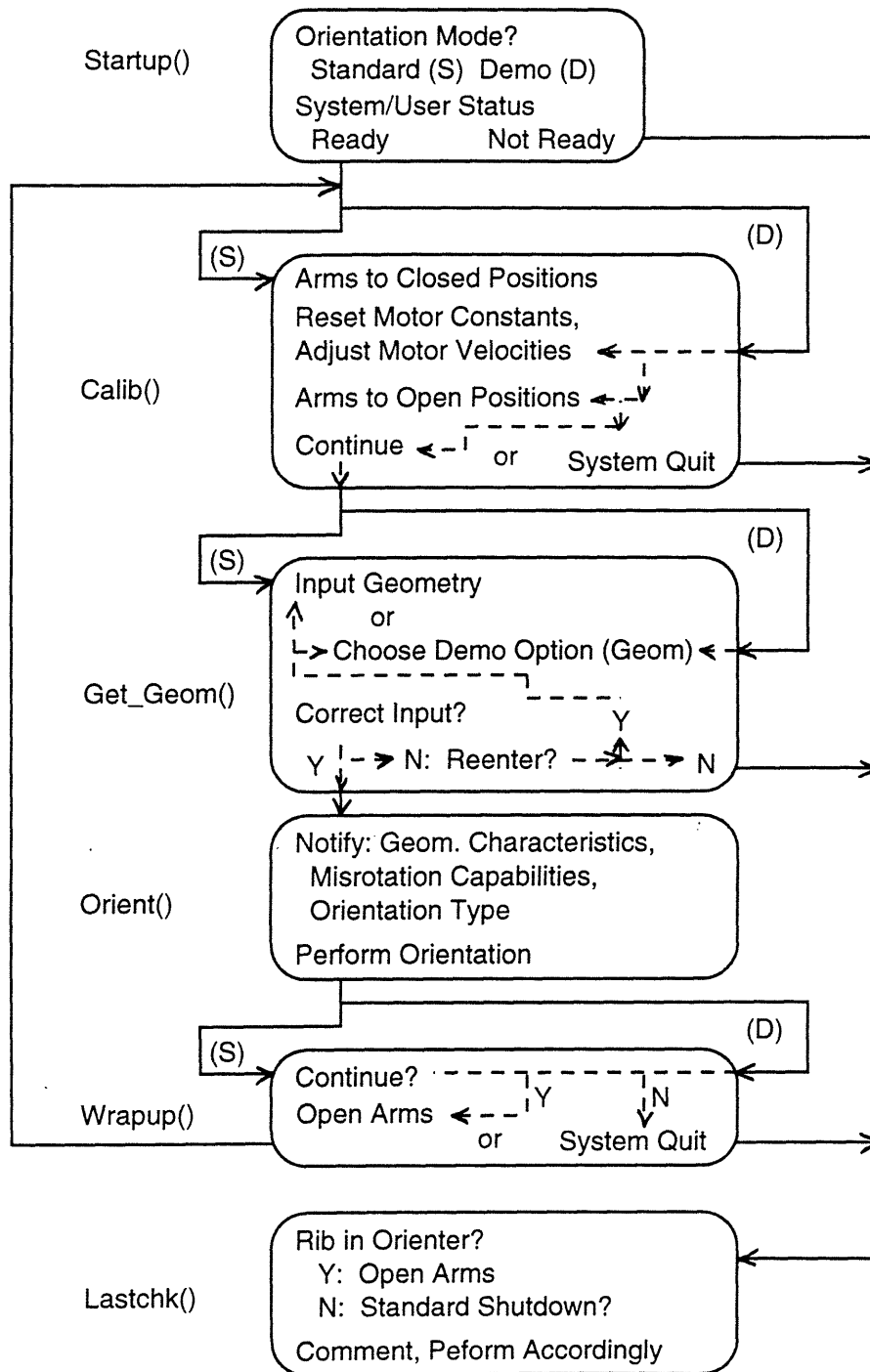


Figure 4.41: Final Code Flow and User Interface

## 4.5 ORIENTING STRATEGIES

### 4.5.1 UNIVERSAL PREPARATORY MOTIONS AND ASSIGNMENTS

All ribs presented by the conveyor may be displaced along the length of the orienter. Therefore, before executing the appropriate algorithm, the rib must be moved into the orienting area of the platform. Figure 4.51 illustrates the series of arm and block motions that serve this purpose.

Motion 0: The right arm rotates (i) to its upright position and then its block moves (ii) toward the left arm which is in its lowest position. Once the block reaches the clear position, it retracts slightly and waits for the next motion.

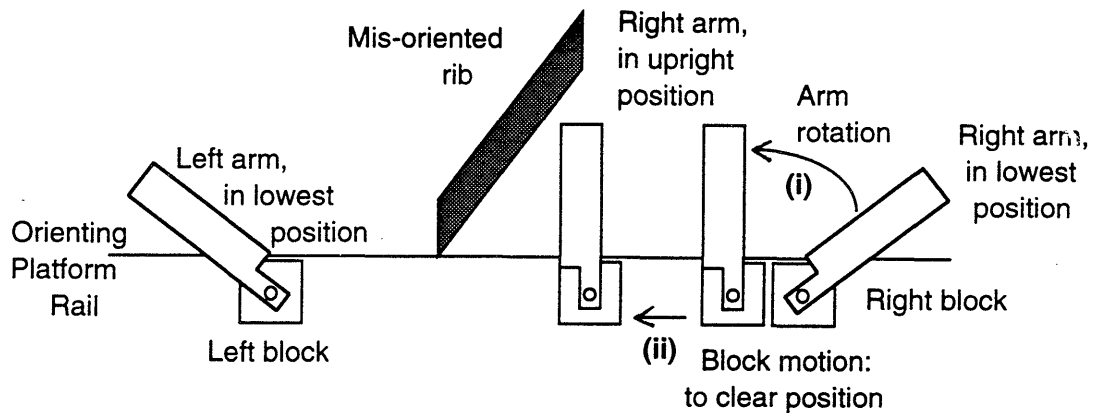


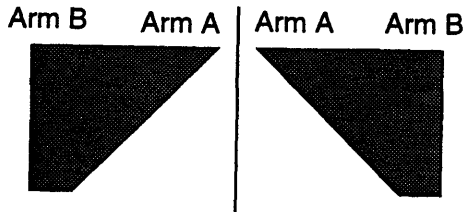
Figure 4.51: Illustration of Orientation Preparatory Motion

This combination of motions is referred to as the preparatory motion from now on. No matter the initial rib location, moving the right arm to the clear position (where the arms are as close together as possible without engaging the rib) brings the arm in contact with the rib and slides it into the orienting area. It is necessary for the right arm to be upright during this motion for situations where the rib might not be resting in a stable

position or entirely contacting the rail. For example, symmetric ribs, even if they are only 2" wide, may be over 15" long from tip to tip, depending on the end angles involved. If this rib arrives at the orienting platform such that it ends up standing on its acute point, any arm contact lower than 7" will topple the rib over the arm and out of the mechanism. This step must be performed before every orientation process since variations of this situation may occur with all rib types. This series of steps is particularly important for the full scale orienter since the rib may arrive mis-located by several feet. However, since this condition can be controlled for the prototype, once this strategy was proven to be affective, it was not implemented in the final prototype algorithms.

At this point, the longitudinal position of the rib as well as the rib geometry and final desired orientation are known. In order to employ only three degrees of freedom, all arm separation is controlled by the left block alone. However, motions must be performed by the individual arms according to the rib geometry and desired final configuration. This is because two ribs which are identical after flipping about their vertical axis can be oriented by the same series of steps if the arm motions are simply switched or traded. This fact is addressed and illustrated in figure 4.52 and in the following sections by designating arm A as the arm which addresses the smallest rib angle from the outside horizontal when the rib is resting in its final destination position.

These ribs can be handled by the same orientation algorithm by assigning and switching arm motions



Arm A motions are assigned to that arm which addresses the rib edge with the smallest angle from the outside horizontal when the rib is in its final oriented position

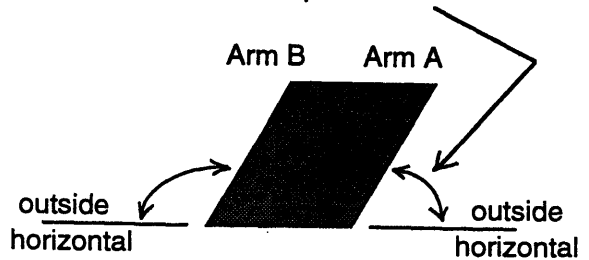


Figure 4.52: Illustration of Arm Assignments For Ribs With Identical Geometries After Flipping About Vertical Axis

#### 4.5.2 SYMMETRIC AND VIRTUALLY SYMMETRIC RIBS

The first attempts at orienting ribs briefly focused on one with equal bottom and top dimensions and two square ends, followed by a symmetric rib (one that presents the same geometry when rotated 180 degrees about its centroid) with angled ends. Both ribs are shown in figure 4.53.

Double square-ended rib



2" base thicknesses

Complimentary angled rib

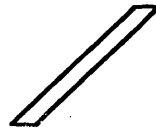


Figure 4.53: First Symmetric Rib Geometry Oriented

These two ribs were initially treated as independent rib types until their algorithms were observed and the possibility of combining them was realized. When the general goal was broadened further to include virtually symmetric ribs and those with thicker bottom dimensions, this algorithm was revised into its final form.



The virtual symmetry of these ribs makes them the easiest to orient. A brief analysis of all possible initial configurations for symmetric ribs reveals the fact that no matter where the rib arrives or to what degree it is mis-rotated, it can only present two possible contacts for manipulation: one where it is upright and oriented, or one where it has fallen over and is lying on one of its edges. The same is true for virtually symmetric ribs which arrive within their mis-rotation limits. Therefore the orientation algorithm must be able to move both contacts into an orientable position. As illustrated in figure 4.54, this position is one where the rib was slightly resting on the arm corresponding to the smallest angle from the horizontal. (As described in the previous section, this arm is universally designated as arm A.) From this contact, if arm A is raised to its final position while arm B is within a range that prohibits the rib from sliding out of contact but avoids engaging the bottom edge of the rib, final orientation is then simply achieved by moving the block and arm B to their final positions.

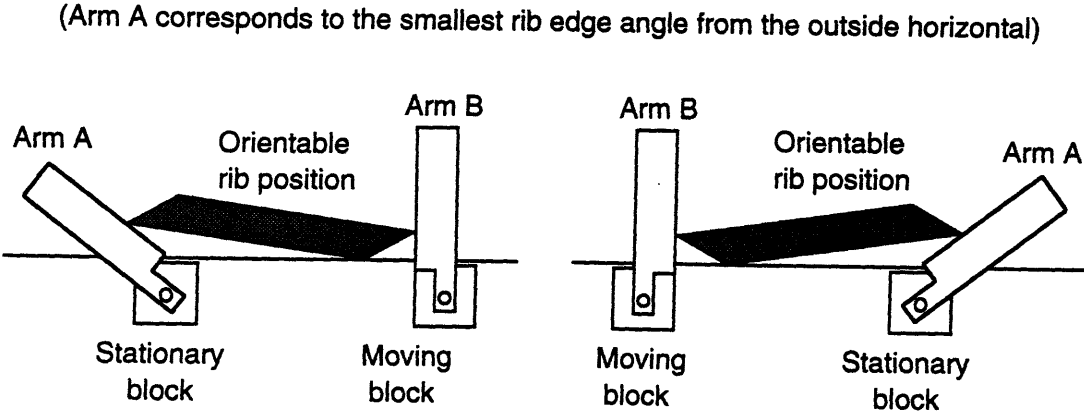


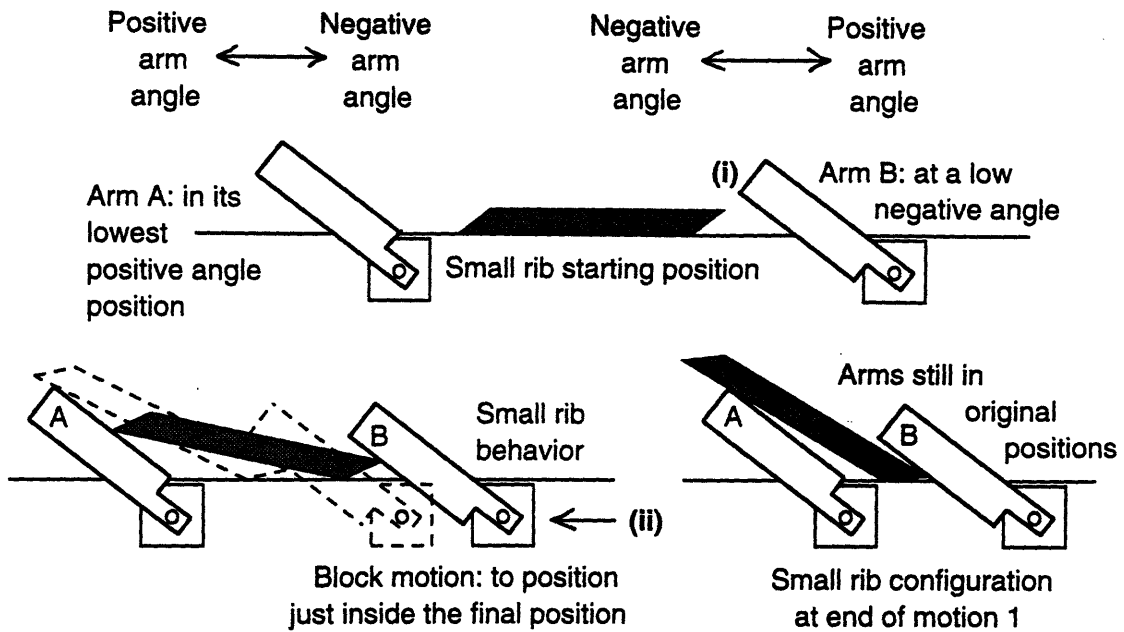
Figure 4.54: Illustration of Orientable Position and Arm Assignment for Symmetric Ribs

The necessary orientation steps are described below.

**Motion 0:** Preparatory motion, as described in section 4.5.1.

**Motion 1:** Arm B rotates **(i)** to a negative angle, and the block moves **(ii)** to a position just inside the final position.

Figure 4.55 illustrates the effect and reasoning behind this group of motions for both large and small ribs. As noted earlier, before this motion, the rib will either be oriented or lying on its long edge. In either case, the rib needs to be moved into the desired final contact with arm A. If the arms are simply moved closer together while in their lowest startup positions, the rib can slide up either arm rail. If arm B is in an upright position during the block motion, two things can happen: the rib can slide up arm A, or the rib can unfavorably rotate around the point where arm A meets the rail. In order to constrain this rotation, arm B needs to be in a low, negative angle. This either causes the rib to slide over, or as the rib begins to rotate, the low contact point with arm B eventually lifts it over the arm/rail corner, and causes it to fall into the final desired contact position.



Large rib starting positions: one is the possibly instable final position, the other occurs when the rib has fallen over and is lying on its side

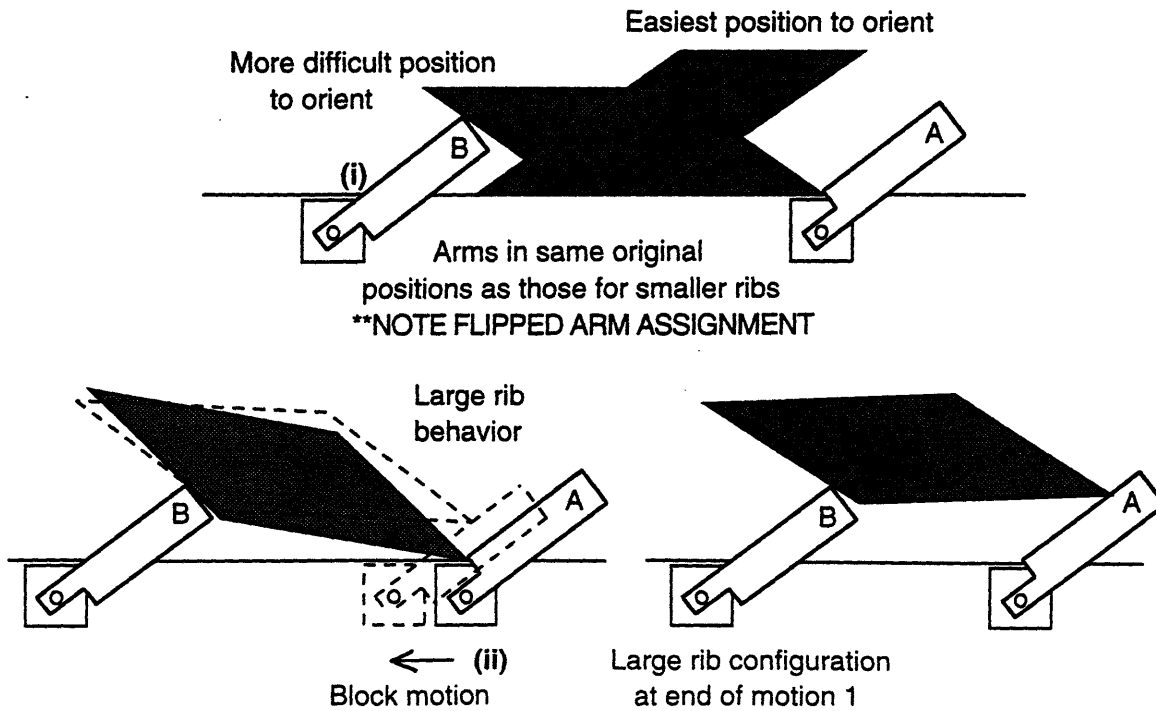


Figure 4.55: Illustration of First Symmetric Rib Orientation Motion

Motion 2: Arm B rotates (i) to an upright position while the block moves (ii) outward to its last position, and then (iii) arm A rotates into its final position, followed by arm B.

This is the final series of motions described in the introduction. As shown in figure 4.56, the desired orientable contact position is attained by (i) and (ii). Plus, separating the arms and moving arm B into its upright position clears arm B from interference. Then, raising arm A further rotates the rib such that arm B and the block may be successfully moved to their final positions.

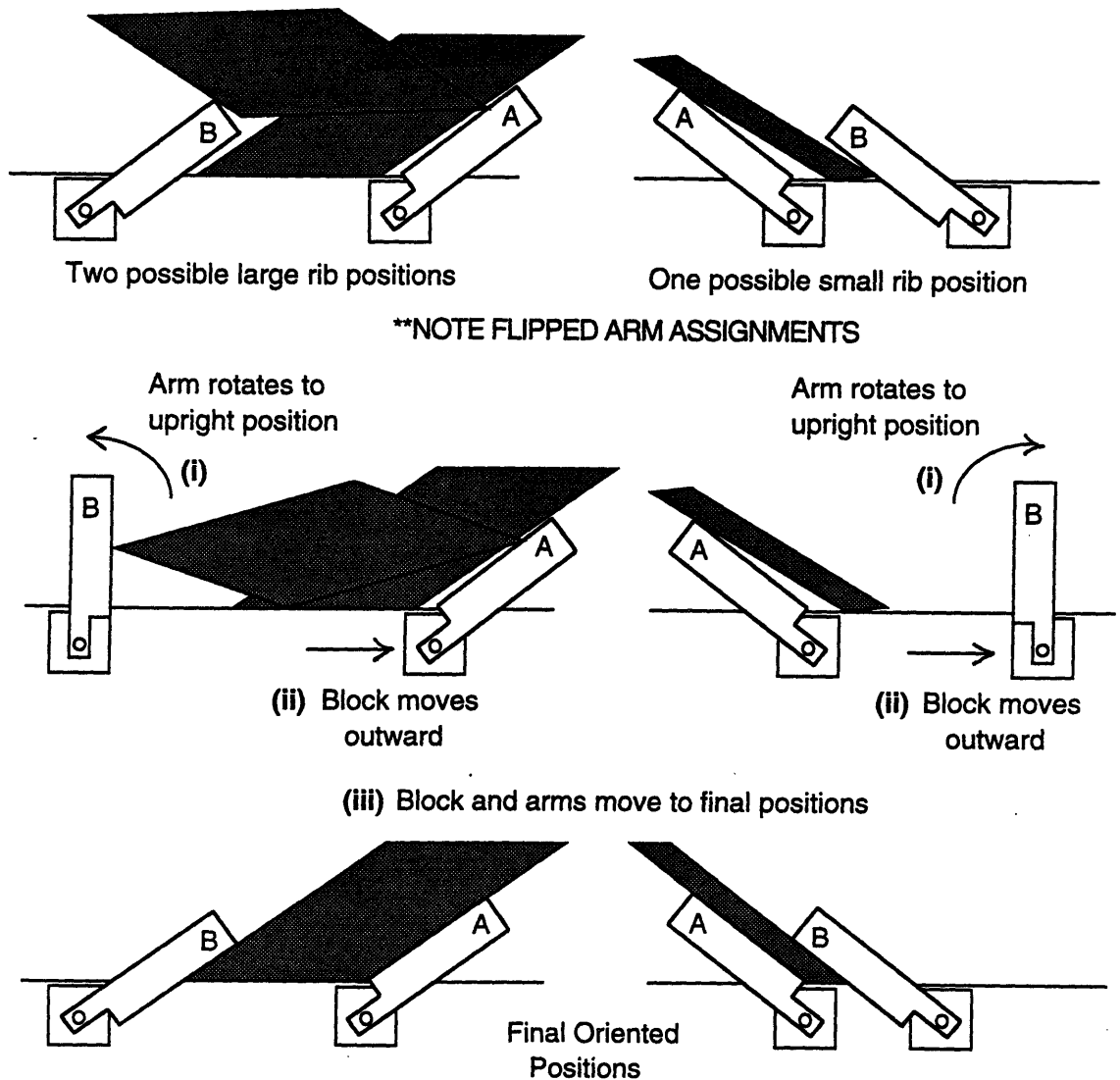


Figure 4.56: Illustration of Second Symmetric Rib Orientation Motion

### 4.5.3 NON-SYMMETRIC RIBS

During the early stages of algorithm generation, two different non-symmetric square-ended ribs were considered. As illustrated in figure 4.57, one had a top dimension greater than the bottom, and the other was the same rib turned over, with a top dimension less than the bottom.

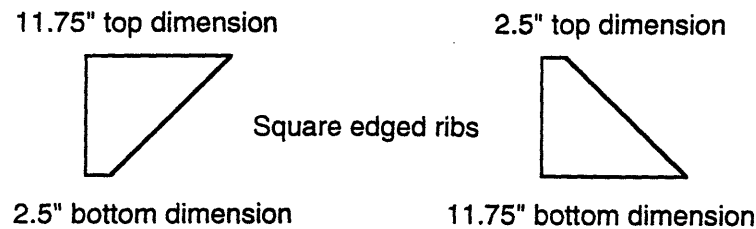


Figure 4.57: First Non-Symmetric Rib Geometries Oriented

At first, orientation schemes were developed independently for each of these ribs. However, the original algorithm for the rib with the smaller bottom dimension assumed a mis-rotation constraint of 90 degrees in either direction. Also, if the mis-rotation of the rib with the larger bottom dimension could be held to under 40 degrees in either direction, then no orientation steps would be necessary. In order to expand both of these mis-rotation capabilities to 360 degrees and incorporate the ability to handle ribs with compound edge angles, a more standard orientation approach was developed.

Theoretically and practically speaking, non-symmetric ribs are more difficult to orient than symmetric ribs because they present more possible initial configurations. Each edge presents a different contact, but since ribs resting on their smallest dimension can be tipped over, the contact position count is effectively reduced to three instead of four.

Several different series of steps were tested as possible orientation methods. For example, since the most stable contact position is the one where the rib is resting on its

longest dimension (top or bottom edge), the quickest approach is to generate a series of steps that make the other two side edge contact configurations rotate into the third stable position. However, this sort of algorithm has to induce rib rotation in two directions. Since this is more difficult and less reliable, the search phase looked instead for a series of steps that affected only two of the contacts, and returned the third to its original position. This way, as long as the series of steps does cause the rib to toggle between two positions, the orientation of the rib can be determined. Then, from this intermediate known position, a concluding series of steps can be employed to move the rib into the desired final position.

The resulting algorithm consists of a series of arm and block motions, repeated twice, which rotates the rib in a single direction. When the rib comes to rest on its longest edge, it will no longer be affected by the motions. As illustrated in figure 4.58, this means that if the rib is initially resting on its square end, it rotates to its angled end, and then from its angled end to its longest edge. If the rib is initially resting on its angled end, it rotates to its longest edge, and then remains in that position even after the steps are repeated. In other words, a rib that arrives on its square edge rotates twice, a rib that arrives on its angled end rotates once, and a rib that arrives on its bottom edge does not rotate at all.

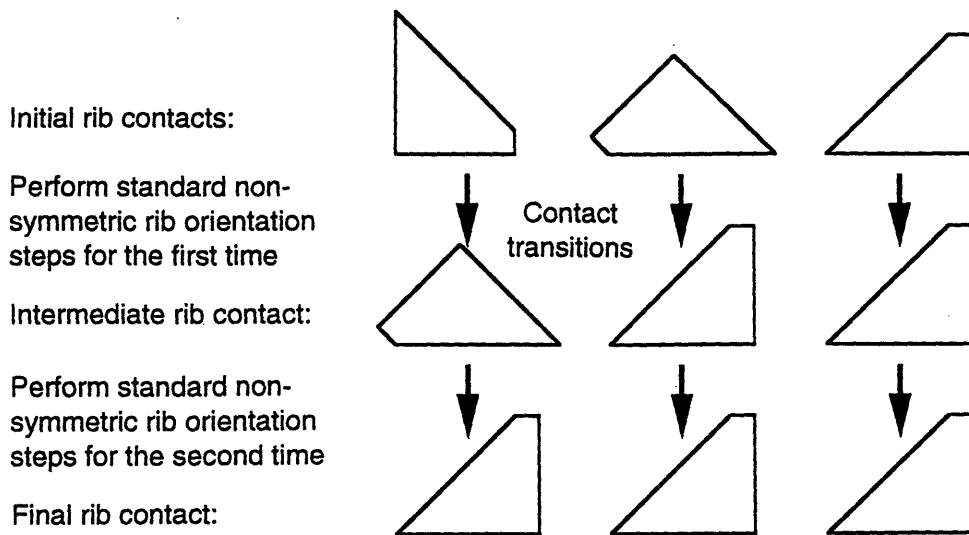


Figure 4.58: Contact Position Transitions for Non-Symmetric Rib Orientation

The block motions required for this algorithm have more marked effects on rib behavior than they do in the algorithm described previously. For symmetric and virtually symmetric ribs, block motions effectively serve to control the distance between arms and make no other substantial contributions. However, in this case some steps are determined according to which arm addresses the square rib edge and/or is located on the moving block. As illustrated in figure 4.59, in order to better describe the orientation process, arm A is denoted as the arm corresponding to the square edge (again the smallest angle from the outside horizontal in the final rib configuration) when the rib is resting on its long edge.

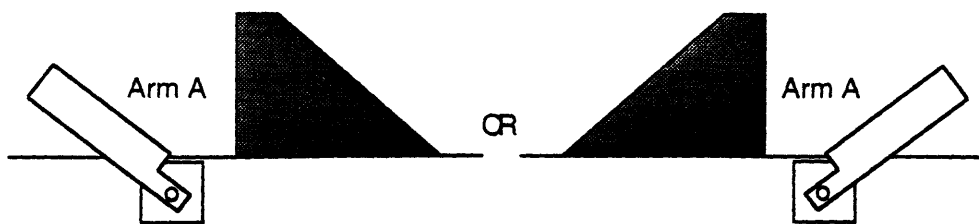


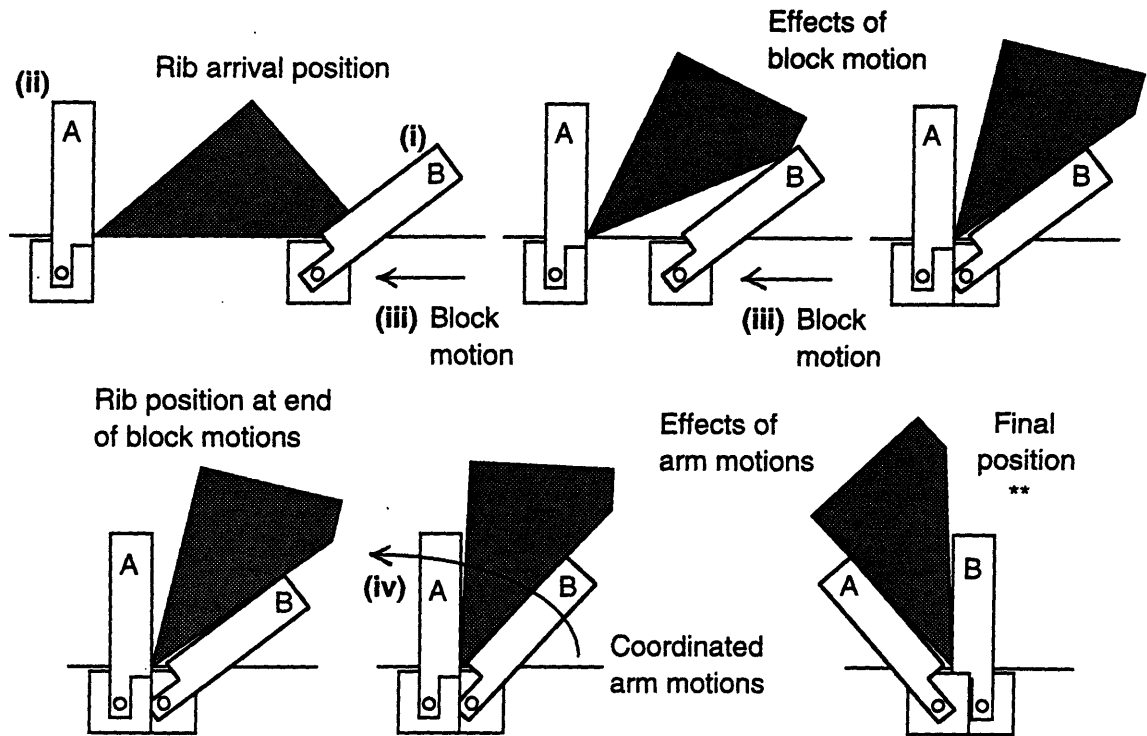
Figure 4.59: Illustration of Arm Assignment for Non-Symmetric Ribs

The required orientation process is described as follows.

Motion 0: Preparatory motion, as discussed in the previous section.

Motion 1: After arm B has rotated (i) to its lowest position and arm A has rotated (ii) to its upright position, the block moves (iii) the arms close together. Then they rotate (iv) slowly, at the same rate, toward the square edge until arm A doesn't quite reach its lowest position.

Figure 4.60 illustrates Motion 1. No matter the initial rib position, this series of steps creates the necessary rotating motion described earlier. In some cases, depending on the rib size and geometry, this step squeezes the rib considerably away from the grasp of the arms, but the following step resolves this situation.



\*\* Note: this sequence has the same affect on all three rib arrival positions. The two final configurations not shown here may be conceptualized by simply rotating this final position such that the other two "corners" are facing the platform rail.

Figure 4.60: Illustration of Non-symmetric Rib Orientation Motion 1



**Motion 2:** After a pause, both arms move into their lowest positions.

As illustrated in figure 4.61, as arm A moves into its lowest position, the rib slides down the arm. If the rib is resting on a particularly acute edge and/or its center of gravity is very close to the tip of arm A, there is a possibility that the rib will slide off the arm rail and out of the orienter. However, the initial proximity of arm B reduces this risk by catching the rib tip inside its guard and forces it to slide down to contact the orienting platform rail. Nonetheless, because of this situation, this step is the limiting factor in the size of orientable ribs.

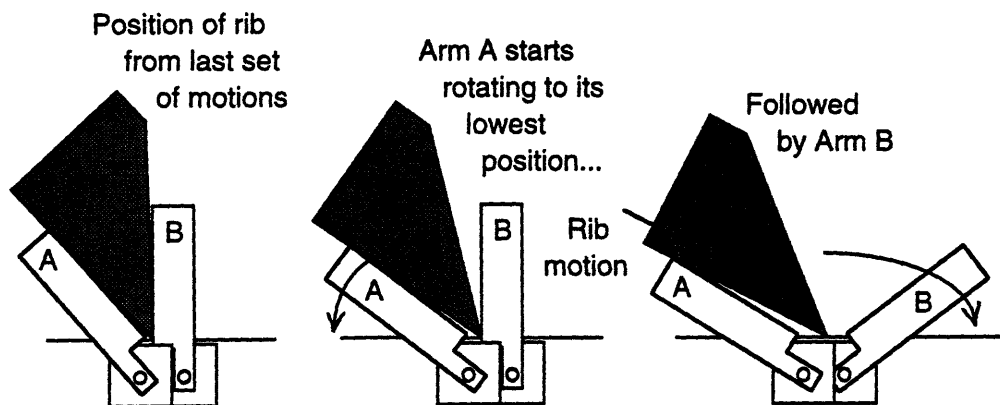


Figure 4.61: Illustration of Non-symmetric Rib Orientation Motion 2

At this point, as discussed earlier, there are two different approaches for continuing rib orientation. If the orienter utilizes four degrees of freedom, separate approaches are not needed and block motions can be assigned by the method described in section 4.5.1 for arm motions. In order to employ three degrees of freedom, two separate branches are needed. One branch is necessary for handling ribs whose square edges (in their final configurations) face the stationary block, and one is necessary for ribs whose square edges face the driven block.

The following steps are for ribs whose square edges face the stationary block.

Motion 3a: The block separates the two arms.

Arm B needs to be out of the way as arm A performs the next series of steps.

Motion 4a: Arm A rotates (i) up slightly and back down. It then rotates (ii) up again to a second position just under half way between the lowest and upright positions, and then back to its lowest position. Finally, it rotates (iii) up past the upright position and over to the opposite midway point and back.

As illustrated in figure 4.62, this series of motions slowly urges the rib down arm A, onto the orienting platform, and into its new contact. If these steps are performed quickly, or the arm moves directly to the final position described here, the rib might rotate back into its previous position. The configuration most sensitive to this problem is the one illustrated here where the rib will end up resting on its longest edge. If the rib is rotated out of this critical position by this series of steps, the entire algorithm will fail. For these reasons, the required slow speed of these steps is hard-wired into the prototype code such that it cannot be affected by user inputs. Once the last position is reached, the motor speed is returned to its original value.

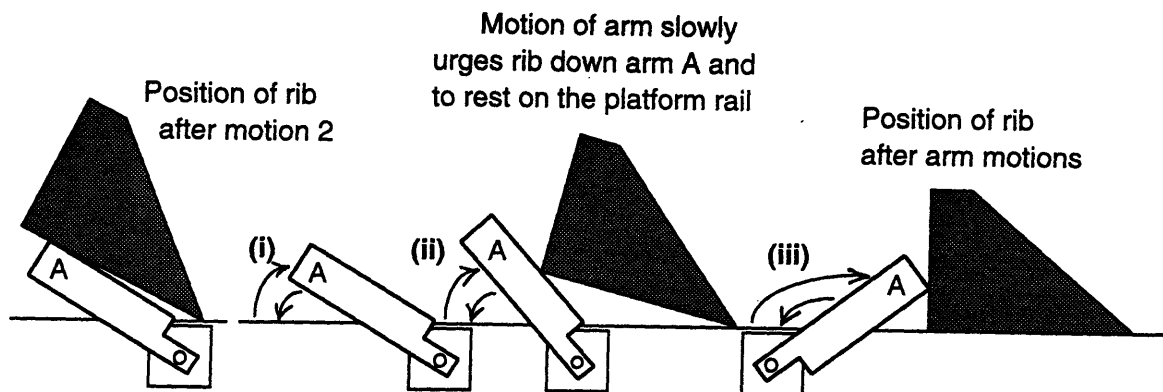


Figure 4.62: Illustration of Non-symmetric Rib Orientation Motion 4a

Motion 5a: Arm B rotates (i) somewhat past the upright position and stays there as the block moves (ii) it toward the rib. Arm B then retracts as the block moves it away from the rib.

As illustrated in figure 4.63, this series of motions is necessary to tip over any rib that might be standing on its shortest edge. Otherwise the rib rotation from the square edge to the angled edge could not be completed. This series of motions also concludes the branch for a rib with its square edge facing the stationary block.

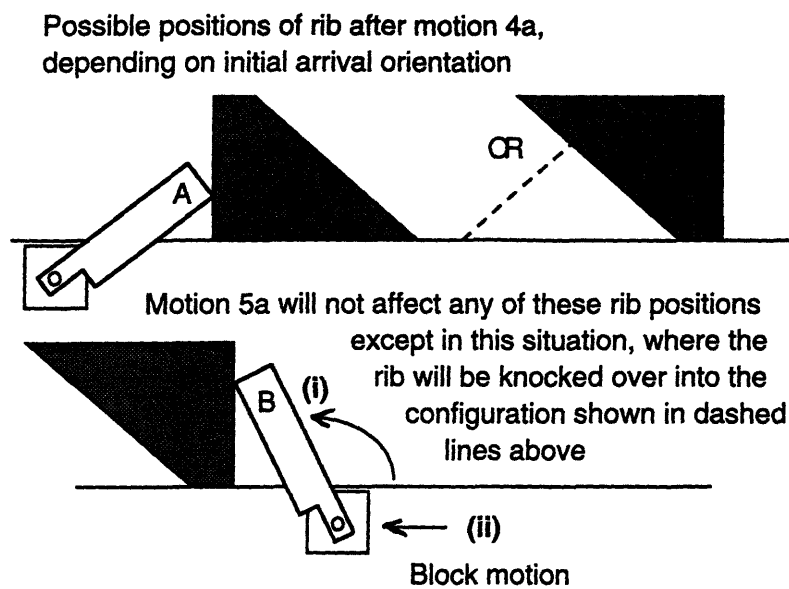
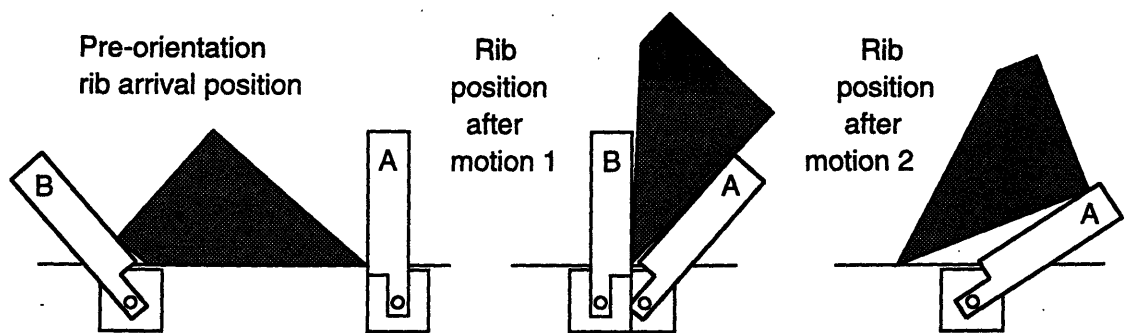


Figure 4.63: Illustration of Non-symmetric Rib Orientation Motion 5a

The following steps are for a rib that has its square edge facing the driven block. At this point the rib has just been rotated and is resting partially on arm A.

Motion 3b: As the block (i) separates the two arms, arm A rotates (ii) to a point just below its upright position and back down to its lowest position.

As shown in figure 4.64, this motion ensures that the rib slides off arm A and into its new contact. Like corresponding motion 4a, the most critical situation is the one where the rib starts with its square edge down. In this case, the arm motions performed previously cause the rib to come to rest with its square edge against arm A, and this motion ensures that it will rotate back to its long edge.



\*\* Note: this sequence has the same affect on all three rib arrival positions. The two final configurations not shown here may be conceptualized by simply rotating this final position such that the other two "corners" are facing the platform rail.

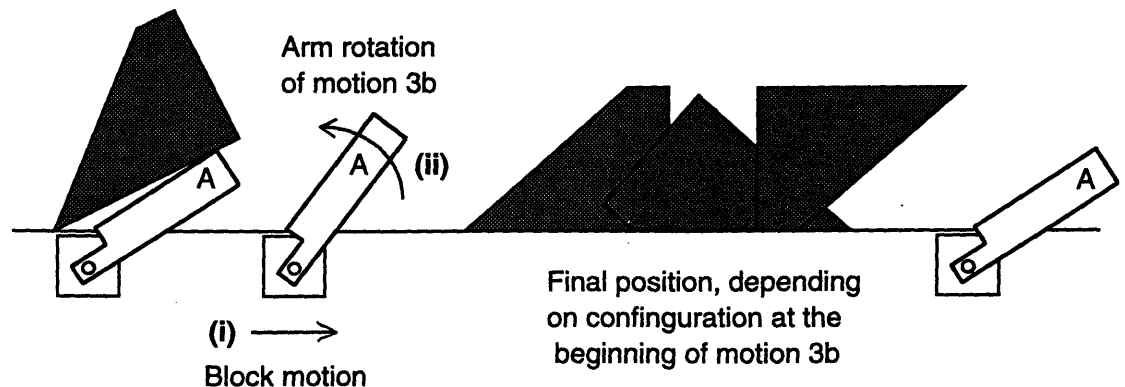


Figure 4.64: Illustration of Non-symmetric Rib Orientation Motion 3b

Motion 4b: Arm B rotates (i) slowly up and back down and then rotates (ii) just past its upright position and back down.

This series of motions addresses two of three possible scenarios. At this point the rib may be either resting on arm B, standing upright on its smallest edge, or fallen and resting on its angled edge. As illustrated in figure 4.65, motion (i) of arm B urges any rib resting on it forward such that motion (ii) will not change its configuration. If the rib is standing upright on its smallest edge, it will remain unaffected until motion (ii) knocks it over. This series of motions concludes the branch for a rib that has its square edge facing the driven block.

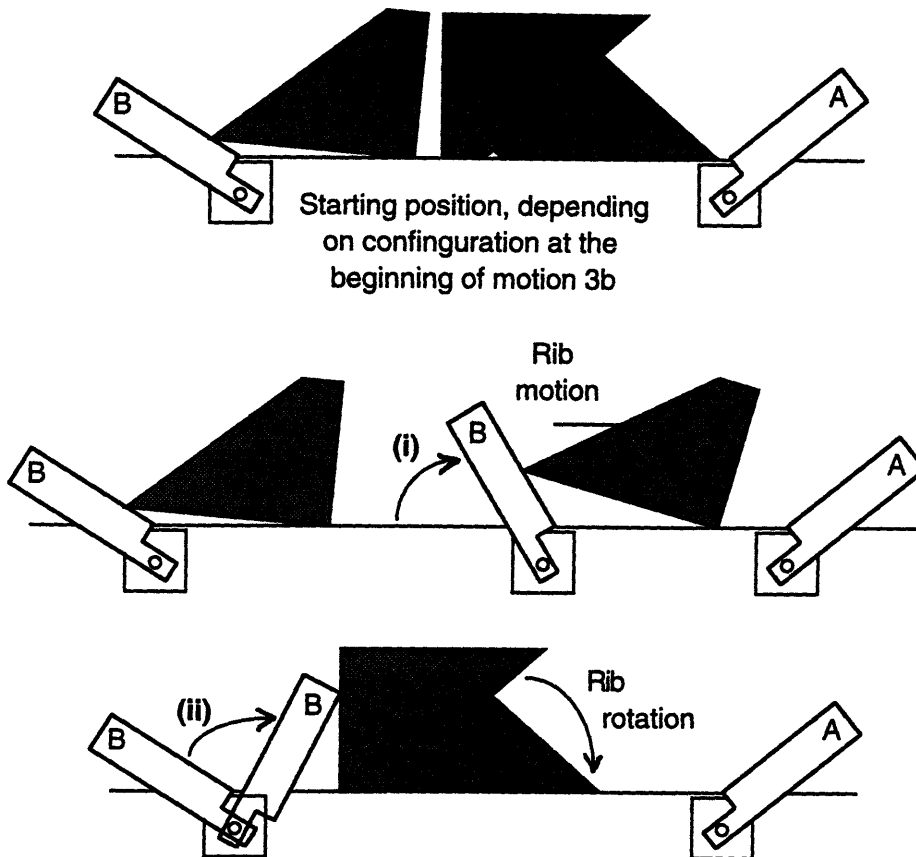


Figure 4.65: Illustration of Non-symmetric Rib Orientation Motion 4b

At this point, both branches are complete, and the rib has been rotated once from its original contact position. As discussed earlier, at this point it will either be resting on its longest edge or the angled edge. In order to address the rib lying on its angled edge (but leave the rib lying on its longest edge unaffected), starting from motion 1, the series of steps are now repeated.

After this second time through the orientation steps, the rib will always be resting on its longest edge, and a second branch is reached. If the rib geometry in its final configuration has a longer bottom edge, orientation is virtually complete; the block and arms must simply move to their final positions. In order to avoid interference, the block must move first, followed by the arms. However, if the rib geometry in its final configuration has a shorter bottom edge, more orientation steps must be performed. The orientable position for this sort of rib is one where it is resting on its angled edge, partially on arm A. The standard set of motions just described cause the rib to rotate from its angled edge to its longest edge. In order to rotate the rib from its longest edge back to the angled edge, these steps are executed in reverse, but slightly modified to leave the tip of the rib resting on arm A. Then, one last set of motions is executed.

**Motion 6:** After arm B rotates (i) to its upright position, while arm A remains in its lowest position, the block moves (ii) to its final position, followed by arm A (iii).

As illustrated in figure 4.66, arm B must be in its upright position to prevent the rib from rotating or sliding up arm B as the block moves the arms closer together. This in turn causes the rib to slide further up arm A, such that when arm A finally rotates to its upright position, the rib is completely oriented.

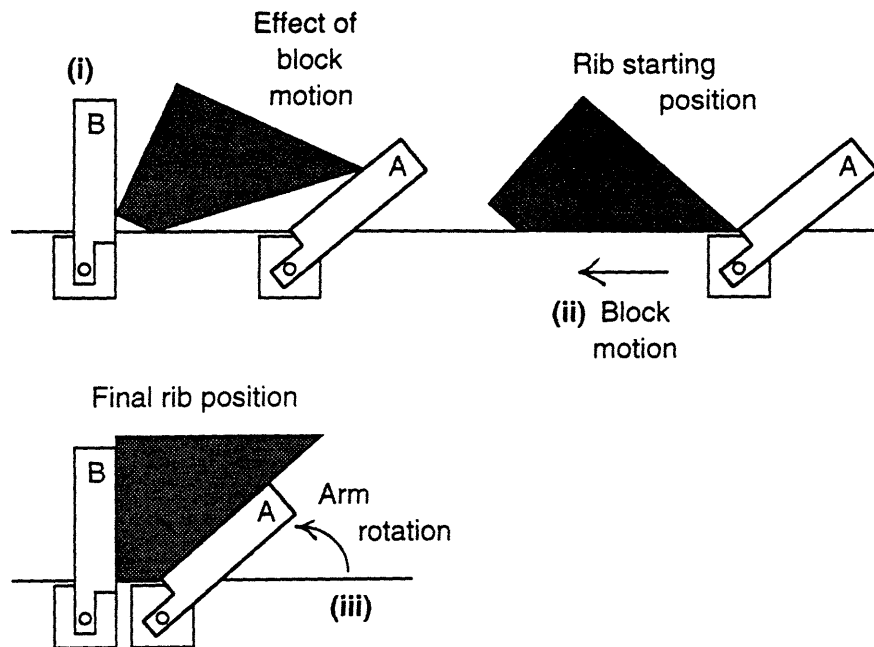


Figure 4.66: Illustration of Non-symmetric Rib Orientation Motion 6c

This completes the non-symmetric rib orientation. While this method does allow for 360 degrees of mis-rotation, it is relatively inefficient. It would not be necessary to perform these steps for ribs with a bottom edge less than their top edge if pre-orientation mis-rotation was constrained to 90 degrees in either direction. In this case, the rib could be handled in less than 5 seconds with the orientation method described in appendix K and employed in orientation demonstration number four. As calculated in Appendix J, if mis-rotation was constrained to 40 degrees in either direction, the ribs with a bottom edge greater than their top edge could be handled in roughly 2 seconds, as they would not have to be oriented at all.





## **Chapter 5: DISCUSSION AND CONCLUSION**

### **5.1 DESIGN NOTES**

#### **5.1.1 DEGREES OF FREEDOM**

In order to test the possibility of employing only three degrees of freedom, all rib orientation algorithms were generated without the use of the left motor block. If only symmetric and virtually symmetric rib geometries are to be oriented, then the motion provided by the arms and the right block alone is sufficient for this application. However, the use of a fourth degree of freedom is recommended if this methodology is to be applied to other situations where more complex and non-symmetric geometries must be handled. The additional cost would be justified by the increased reliability it introduced. Tests showed that some geometries react more favorably and predictably when contacted by an arm that is translating as well as rotating. Comparing the two branches in the non-symmetric rib algorithm illustrates this point. The series of up and down rotations necessary for orienting a rib that has its square end facing the stationary block were the least reliable set of motions of all orientation algorithms. On the other hand, the simpler combined rotation and translation of the right arm that performed the identical task in the other branch was relatively infallible compared to all other sets of motions, and especially its equivalent.

### 5.1.2 TOTAL ESTIMATED TASK COMPLETION TIMES

Assuming that the speeds achieved by the prototype were the maximum possible for orientation, the amount of time necessary to orient four ribs, glue them, secure them in the overhead mechanism, and transport them to the rib/bottom face assembly station may be estimated. Starting at the time of the first rib arrival, it should take 1 second for the platform to reach its 45 degree orientation position. Then, if the rib is 24' long, it will take about 2 seconds for it to be moved into its final position in the orienter. If the rib is smaller and requires orientation, this time will increase. First, traveling at 2 ft/sec, the right block will take approximately 2 seconds to move the rib from its original lengthwise position on the orienting platform rail into the orienting area. Then, if the rib is symmetric, orientation will take about 3 seconds. If it is virtually symmetric and arrives within the mis-orientation ranges calculated in this paper, this is still a good estimation. (Longer non-symmetric rib orientation times are excluded from this estimate since they are not be handled on the manufacturing line.) After adding 1 more second for orientation verification while the platform rotates to reach its upright position, the maximum orientation time becomes 7 seconds.

Now, it will take approximately 3 seconds for overhead clamp engagement, arm release, and guidance clamp engagement, and 2 more seconds for the first rib edge to be glued. The time necessary to glue the bottom rib edge as it moves into the overhead mechanism will vary roughly according to rib length. If the rib is 24' long and moves at approximately 2 ft/sec, this will take up to 15 seconds, and the rib will be virtually positioned. The shortest ribs will take about 3 seconds to glue, but might also take

another 10 to travel from the gluing position to their final position within the orienter. After adding 2 seconds for gluing the last rib edge and 3 seconds for final rib positioning and clamping, the maximum total time required for rib gluing, orienting and securement within the overhead mechanism orientation is 25 seconds. If these times are combined and a 10% error factor is added, the processing time for each rib is roughly 35 seconds, and thus the maximum time required to process a standard four-rib panel will be 2 minutes, 20 seconds.

After the overhead mechanism is loaded, it should take about 10 seconds to position the ribs above the bottom face, and another 15 for the gang of staplers to travel the 24' panel length. Moving the ribs out of the mechanism and should take 2 seconds, and returning the mechanism to the orientation station should take a final 10 seconds. After adding another 10% error factor to these three motion times, they combine to be about 41 seconds. However, before the mechanism can return to the orientation station, it must wait for the end cap to be attached. This time will vary from 15 to 30 seconds, depending on the type of panel to be stapled.<sup>17</sup>

The estimated maximum processing time required to take four ribs off the conveyor, orient them, glue three of their edges, secure them in the overhead mechanism, transport them to the assembly station, and staple them to the bottom face is approximately 3 minutes, plus end cap attachment time. With all the error factors taken out, and a more efficient estimation of the undefined and combinable movements, this time is reduced to around 2.5 minutes.

---

<sup>17</sup> Phillips, David.

Considering that this combination of processes is the main bottleneck, they will determine the maximum rate at which panels can be discharged at the end of the line. If the line is to produce 175 panels a day, panels must move off the line every 2.74 minutes. If throughput is set at 160 panels a day, this time increases to 3 minutes. The additional time required to attach end caps should not extend the 2.5 minute estimate beyond these goals. Therefore the system developed here is right on target.

## **5.2 FUTURE WORK**

### **5.2.1 ORIENTATION COMPLETION/SUCCESS SENSING**

One aspect of orientation, integral to the success of a full scale system but left unaddressed by this prototype, is a method for determining whether the rib has been oriented correctly. This could be done in several ways. A vision system could be employed, but initial design constraints already identified this solution as unacceptable. Another approach might be to rely on the positional feedback of the motors. While this would be an effective means of checking all block and arm positions, the presence of the rib within the orienter could not be ascertained.

A feasible solution is to mount a pair of relatively inexpensive infrared sensors on a linear guide at the back of the orientation platform. After orientation, as the platform rotates to its upright position and the orienting support retracts, these sensors can quickly scan part of the section where the rib should be in the orienter. If the sensors are placed

just above and below the top rib edge, their combined reports will assess the orientation status. If the rib is oriented correctly, its top edge will be level and extend exactly two inches above the top of the orientation platform. This condition will be verified if the top sensor never detects an obstruction and the bottom one always does. An unsuccessful orientation which leaves the rib lying on the rail, or only partially or unevenly extending above the platform will also be detected by the combined sensor reports. The presence of the arms will not affect the results since the sensors operate at 9" from the bottom rail, while the arms extend a maximum of 7" in their upright positions.

### 5.2.2 TESTING

Some further testing is recommended before the prototype design is extended to a full-scale system. In the situation where two panels come together at a hip or valley, their ends will be slanted. In order for end caps to be effectively attached to these panels, their rib edges will have to be slanted across their 7/16" dimension. The effect of this slant upon rib behavior within the orienter needs to be assessed. While the gravitational properties of the rib will not be affected, the rib/arm interaction might be slightly altered. The worst case scenario would be that a rib could get jammed between the arms and the orientation platform. A flat, virtually frictionless orientation surface will probably make jamming less likely, but this should be determined.

The upper motor speed limits for orientation also need to be explored. For virtually symmetric ribs, speed is not particularly critical considering the small number of steps employed by their algorithms. The possible reduction in cycle time is insignificant

compared to the time consumption of other processes involved in rib preparation. However, due to the greater number of steps required to orient non-symmetric ribs, this experimentation could significantly improve their orientation time.

Finally, the reliability and repeatability of this orientation process need to be evaluated. While the system was predominantly successful during testing, success and failure depended on several variables, especially including the condition of the rib corners and the resulting friction between these corners and the orienter rail. In order to ascertain whether and/or at what speed these variables affect the reliability of the algorithms developed, the repeated orientation success rate for several different rib geometries and motor speeds needs to be observed. As described earlier, the reliability of non-symmetric rib orientation may be improved by employing a fourth degree of freedom. As discussed in the next section, the algorithms themselves may also be further tailored to improve reliability.

### 5.2.3 ORIENTATION ALGORITHMS

At this point, the orienting algorithms can still be effectively refined. Some, but not all angles and positions hard wired in the orienting algorithms were optimized. Many were determined by observing of the effects of varying their values, but only a single rib geometry was used in this process. Since the resulting positions and angles were affective for all other rib types, no further tests were performed. Therefore the correlation between rib geometry and block and arm position was not completely known. There is a great

possibility that their relationship can be determined and tailored to improve all orientation algorithms' performance and reliability.

Block and arm speeds can also be related specific geometries. The effect of this process is seen in the series of up and down motions necessary for orienting a non-symmetric rib which has its square end facing the stationary block. If the speed of these motions is not hard wired in the prototype orientation algorithms, the probability of their failure drastically increases as the speed increases. Therefore their speed must be limited. However, in order to optimize cycle time, the motors need to run at their highest speeds whenever possible. In this case, the simpler and more reliable steps can and should be greatly accelerated without affecting their performance. In short, different rib types and configurations react differently to arm and block speeds according to their respective gravitational and inertial properties. Streamlining and increasing orientation algorithm efficiency will require a better understanding of this relationship.

Finally, the possibility of an orientation algorithm for virtually symmetric ribs, capable of handling the entire 360 degrees of mis-rotation, should be investigated. Standard (if there is such a thing it would depend on aesthetics which by nature is not standard) roof pitch combinations should be determined, and the corresponding and most difficult ribs should be assessed. It is possible that if the two rib end angles are different enough, the resulting geometry will have a characteristic behavior in the orienter. If this is the case, it is possible to eliminate the current restrictions on virtually symmetric rib pre-orientation mis-rotation, all small ribs are in fact orientable over the entire 360 degrees of misrotation, and the manufacturing line algorithm is truly and completely flexible.





### **5.3 CONCLUSION**

While standard truss and rafter methods of roof construction are struggling to meet the current market demand for complex roof design, efficient use of living space, and increasing insulation installation and ventilation standards, a panelized roofing system developed at MIT promises to alleviate these problems. The research described here brought the system one step closer to application in the field, by developing a manufacturing line concept for the ribbed panels, as well as prototyping one of the most critical stations for ensuring the necessary flexibility and throughput. In the process, a method for handling complex and widely varying geometries was developed which can be applied to panel assembly as well as other related manipulation tasks. Should this line evolution continue successfully, the goal of providing a flexible roofing system with minimal material, labor, and energy usage will be met.



## LIST OF REFERENCES

1. Bemis, Albert Farwell. The Evolving House. The Technology Press at MIT, 1938.
2. Brost, Randy C., and Mason, Matthew T. "Graphical Analysis of Planar Rigid-Body Dynamics With Multiple Frictional Contacts. Fifth International Symposium on Robotics Research. 1989. Tokyo, Japan.
3. Crowley, John S., and Parent, Michel. "Engineered Roof Products: A Flexible Product Concept for Net Shaped House Components. Unpublished Report. IHCTP at MIT, Cambridge, MA. February, 1991.
4. Eichler, Ned. The Merchant Builders. The MIT Press, Cambridge, MA. 1982.
5. Erdmann, Michael A., Mason, Matthew T. "An Exploration of Sensorless Manipulation." IEEE Journal of Robotics and Automation. August 1988. Vol. 4, No.4, p. 369-379.
6. Erdmann, Michael, Mason, Matthew T., and Vanecek, George Jr. "Mechanical Parts Orienting: The Case of a Polyhedron on a Table." Algorithmica. 1993. Vol. 10, p. 226-247.
7. Gropius, Walter and Wachsmann, Konrad. The Dream of the Factory-Made House. The MIT Press, Cambridge, MA. 1984.
8. Lozano-Perez, T., Mason, Matthew T., and Taylor, R. H. "Automatic Synthesis of Fine Motion Strategies for Robots." International Journal of Robotics Research. 1984. Vol. 3, No. 1, p. 3-24.
9. Lynch, Kevin M. "The Mechanics of Fine Manipulation by Pushing." Proceedings of the International Conference on Robotics and Automation. May 12-14, 1992. Nice, France.
10. Lynch, Kevin M. "Planning Pushing Paths." International Conference on Advanced Mechatronics. August 2-4, 1993. Tokyo, Japan.
11. Mason, Matthew T. "Two Graphical Methods for Planar Contact Problems. IEEE/RSJ International Workshop on Intelligent Robots and Systems. November 3-5, 1991. Osaka, Japan.
12. Mason, Matthew T. "Mechanics and Planning of Manipulator Pushing Operations." International Journal of Robotics Research. Fall, 1986. Vol. 5, No. 3, p. 53-57.
13. Phillips, David. "Flexible Manufacturing of Roof Panels: Process Layout and Assembly Equipment Development. Mechanical Engineering M.S. Thesis at MIT, Cambridge, MA. 1994.
14. Sharma, Vikas. "Roof Panels and OSB: Issues in Design and Manufacturing." Mechanical Engineering M.S. Thesis at MIT, Cambridge, MA. September, 1991.



## Appendix A: ALTERNATIVE MANUFACTURING LINE LAYOUTS

Presented here, in the order of their development, are several alternative manufacturing line layouts. Each concept approaches in different ways the individual and collective process goals of limited capital investment and floor usage, minimal material and energy waste, and efficient throughput. As discussed in Chapter 2, after considering the relative success and restrictions realized by each of these progressing solutions, the final layout was evolved.

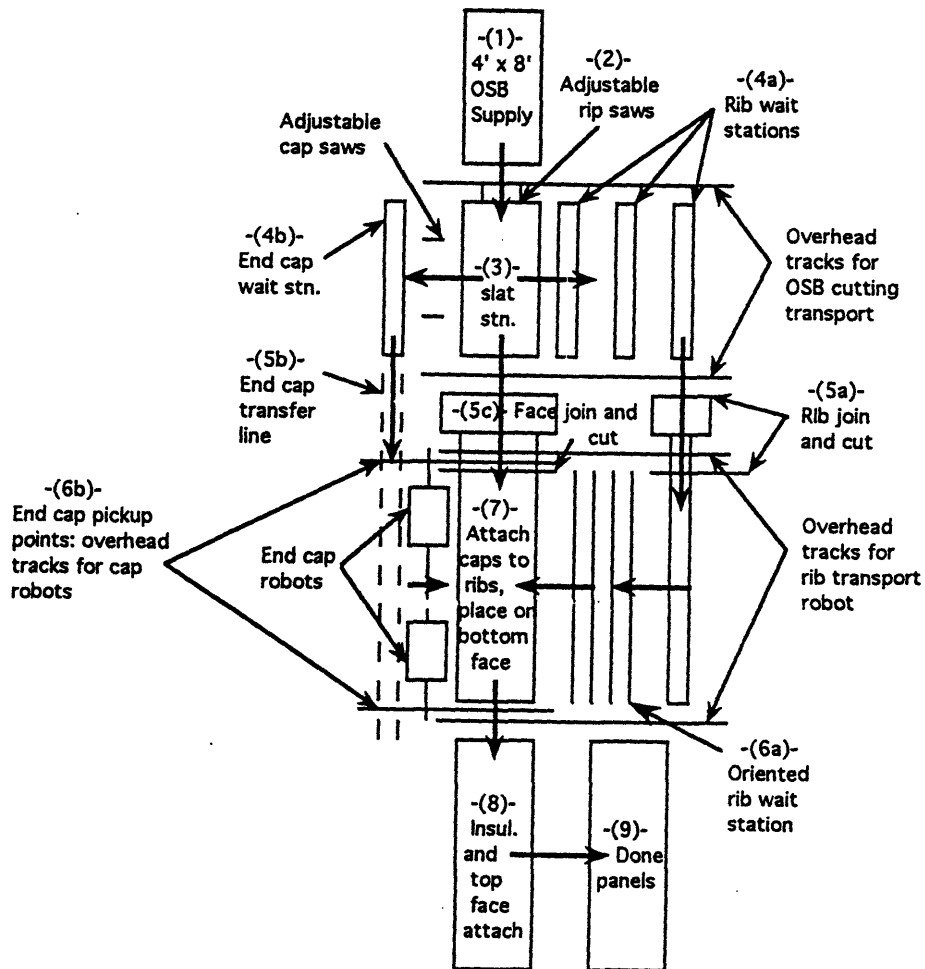


Figure A1: Alternative Manufacturing Line Layout #1

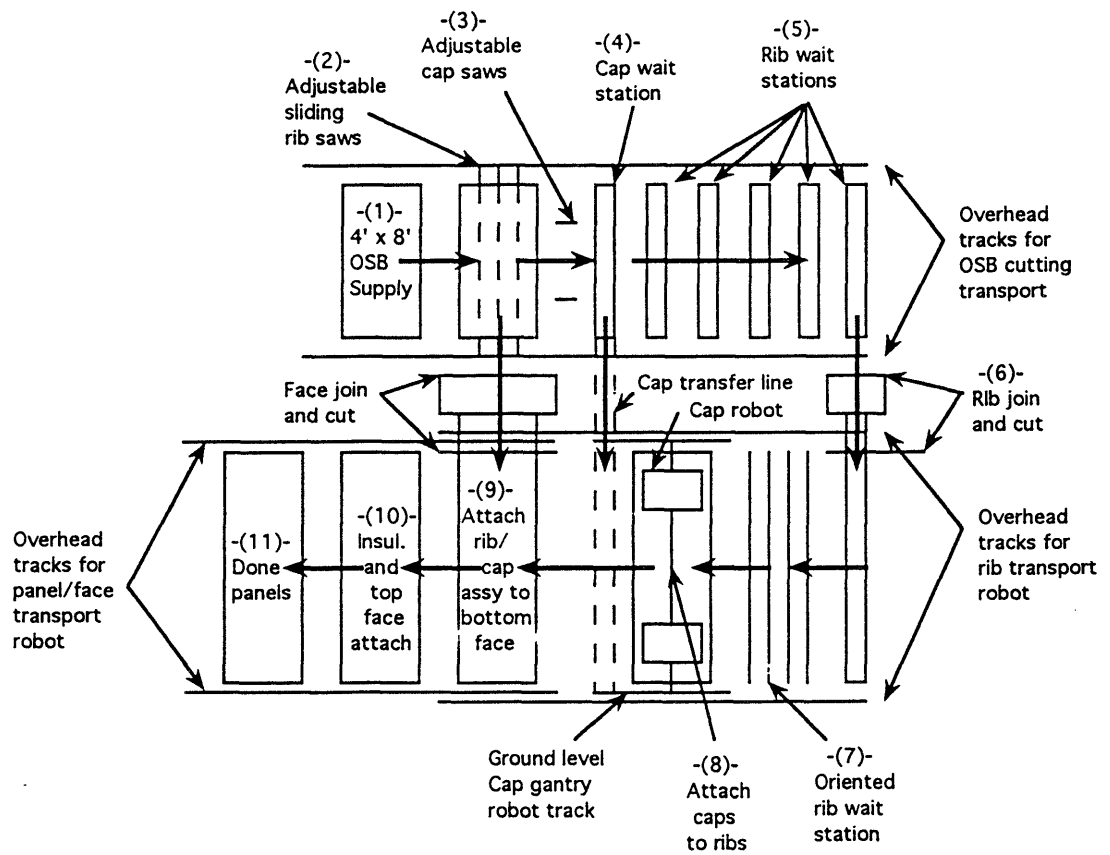


Figure A2: Alternative Manufacturing Line Layout #2

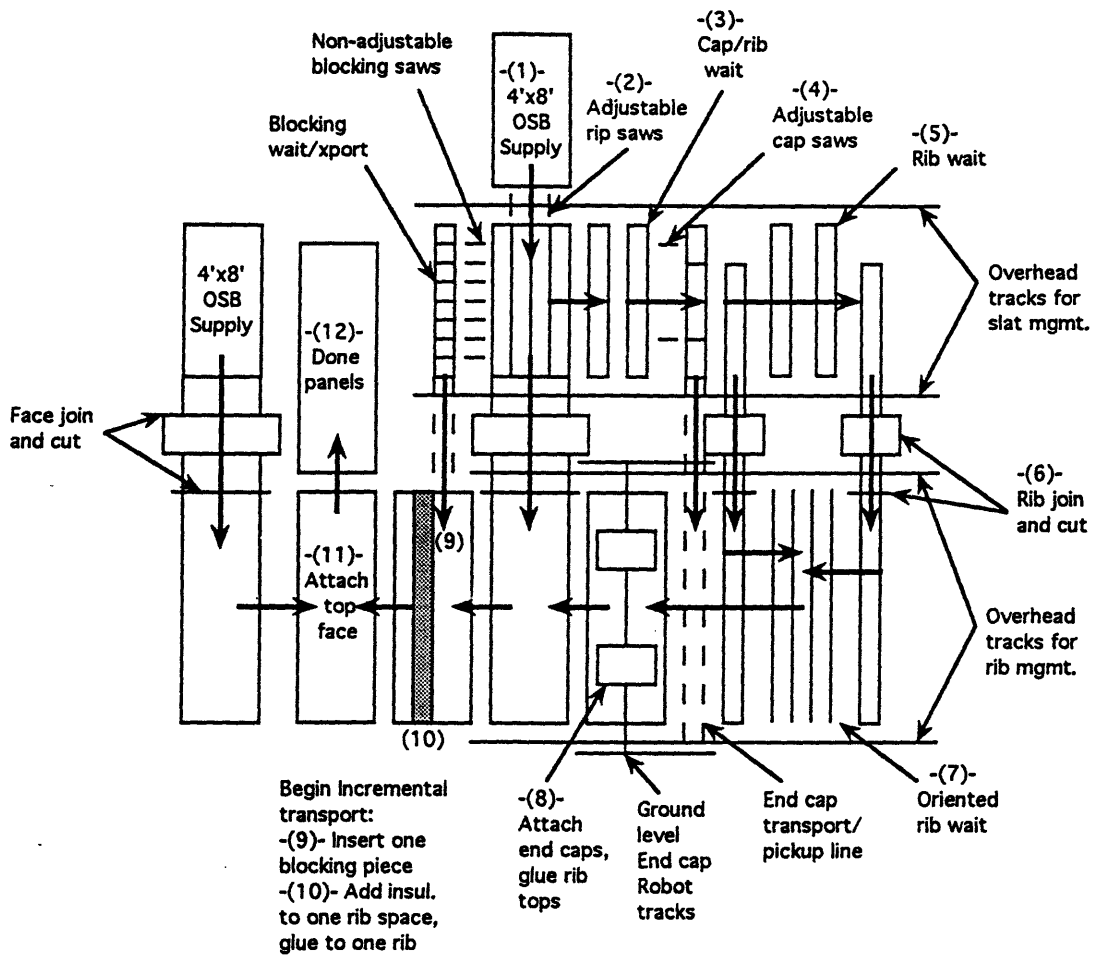


Figure A3: Alternative Manufacturing Line Layout #3

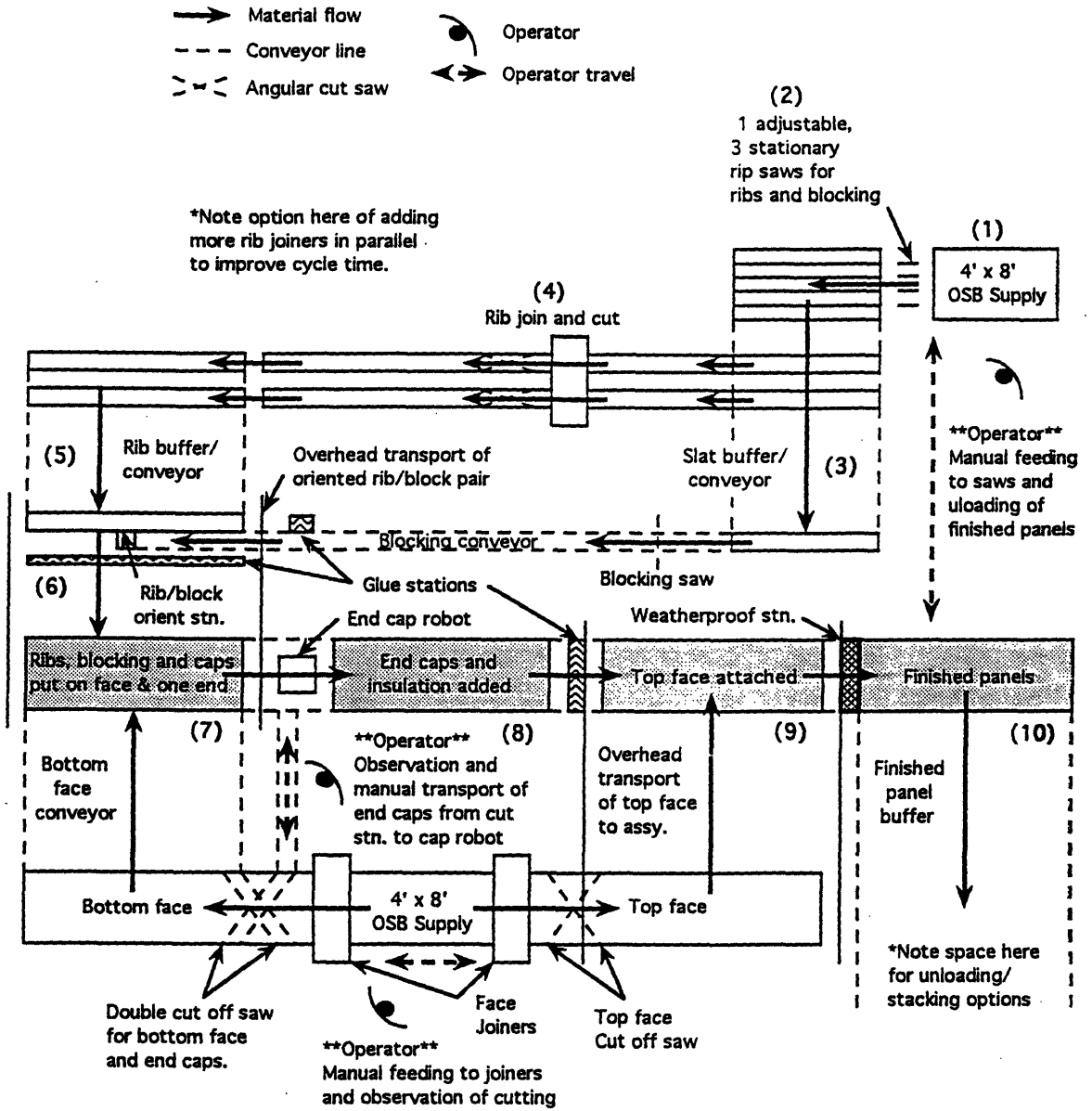


Figure A4: Alternative Manufacturing Line Layout #4



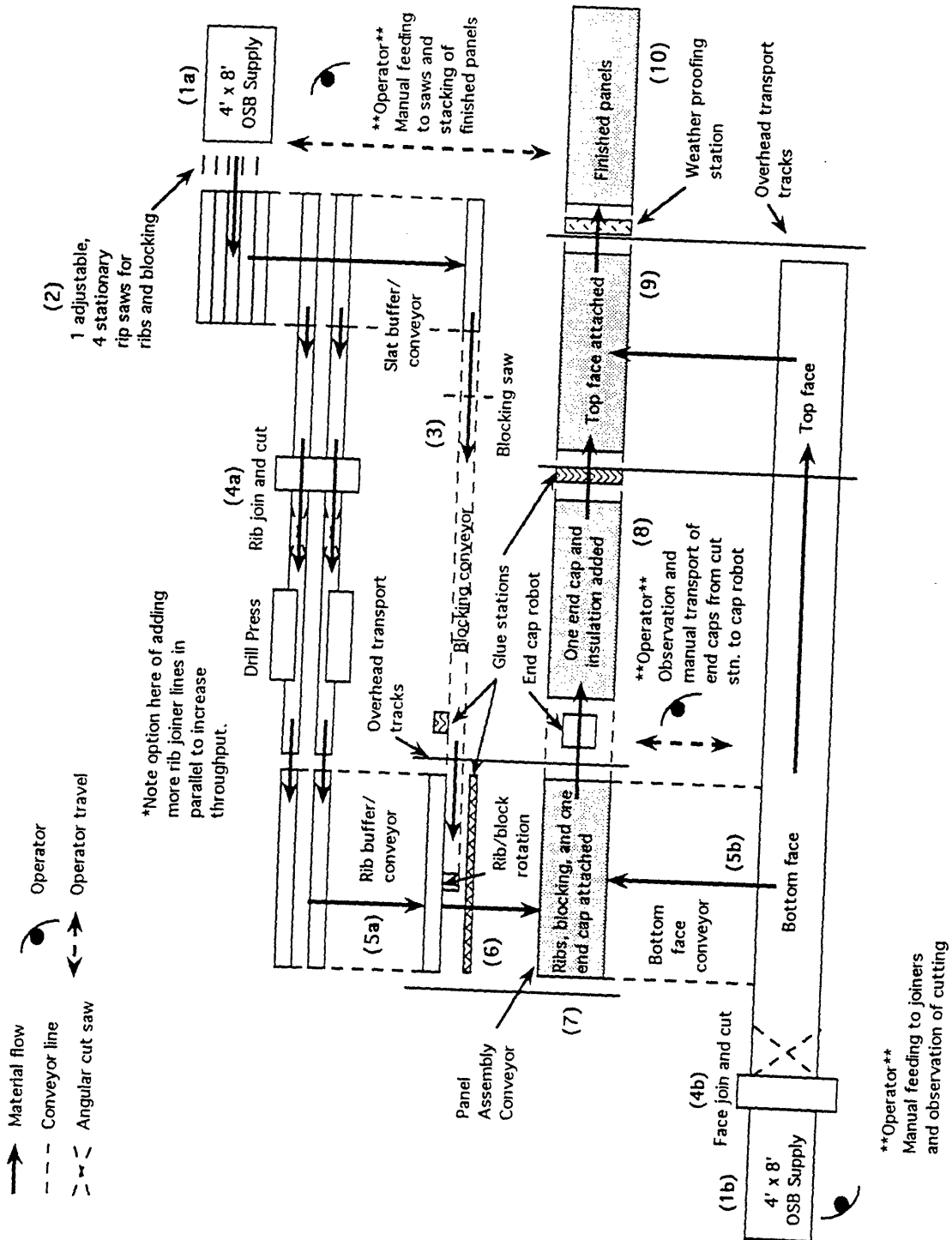


Figure A5: Alternative Manufacturing Line Layout #5



## **Appendix B: ALTERNATIVE RIB ORIENTER DESIGNS**

During the exploration of possible approaches to receiving a mis-oriented rib from a conveyor, it was considered whether the rib should even be placed on a conveyor at all. When the rib is at the cutting station, its orientation is completely known and it is substantially secured to ensure precise cutting. If the rib could be grasped directly from this position, the need for conveyors and a complex orienting mechanism could be eliminated. However, as this solution was investigated, its negative aspects were quickly discovered.

First of all, the final manufacturing line layout may include up to five independent, parallel rib cutting stations. If this is the case, and each station is to individually orient ribs, five redundant rotational and linear servo systems will have to be employed, which is relatively expensive. It then seems that the best solution to this problem is to make the clamps stationary, leaving the overhead mechanism responsible for both rotational and lengthwise positioning capabilities. However, the mere task of installing rotational systems in an overhead system without substantial interference is difficult and undesirable due to design and maintenance space restrictions. In addition, a large number of clamps are necessary to accommodate for small ribs and to support the longer ribs which will otherwise buckle under their own weight during rotation. This issue aside, there are several problems involved with relying on the overhead mechanism for lengthwise positioning. One solution would employ redundant and thereby expensive servo systems at each of the four rib positions. The other solution would require the mechanism to travel approximately 48' to be able to position the ribs anywhere within its 24' length. The

presence of multiple cutting stations also further increases the necessary range of motion, thereby increasing cost and complexity.

The one possible benefit from taking the ribs directly from the cutting station might be a reduction in cycle time. Putting cost aside, if the cutting clamps included the rotational degree of freedom and four redundant overhead positioning systems are incorporated with a system of four cutting stations spaced 14.5" apart, (standard distance between ribs for assembly), the overhead mechanism can be loaded with all four ribs simultaneously. However, the next question is how to apply glue to the rib edges. It is too expensive to employ four gluing stations, and if a single station is employed, the system must pause while glue is applied to each rib. Hence, the initial cycle time advantage is reduced.

## Appendix C: MOTOR FORCE AND VELOCITY REQUIREMENT CALCULATIONS

### VELOCITY:

The estimated **linear velocity** was 1 ft/sec = 720 in/min.

Pulleys with a 1" radius create 6.28 in/rev.

Combined result: 115 rpm. With a safety factor of two: 230 rpm.

The estimated **arm rotational velocity** was 90 deg/sec,

or 0.25 rev/sec = 15 rpm. With a safety factor of two: 30 rpm.

### FORCE:

OSB, 7/16" thick and 9 1/4" tall, weighs 1.18 lb/ft.

Therefore the **maximum board weight** will be:

$(24 \text{ ft})(1.18 \text{ lb/ft}) = 28.32 \text{ lb}$ . Round up for variances: 30 lb.

The static coefficient of friction for OSB/UHMW is under 0.25.

Therefore the resulting **maximum tangential force** will be:

$(30 \text{ lb})(0.25) = 7.5 \text{ lb}$ . Round up for abnormalities: 8 lb.

The arm length, from motor center to tip is under 11".

If all rib weight is pushed at this point and moved by the arm alone, the maximum required **arm motor torque** will be:

$(8 \text{ lb})(11 \text{ in}) = 88 \text{ in-lb}$ . With a safety factor of 1.5: 132 in-lb.

Combined motor, linear guide block and mounting plate wt: 15 lb.

When friction and damping are ignored and this mass is multiplied by a linear acceleration of 6 in/sec<sup>2</sup>, added to the tangential force and applied at a pulley radius of 1", the resulting required **block motor torque** will be:

$((15/386.4)(6) + 8)(1) = 0.23 \text{ in-lb}$ . With safety factor: 0.35 in-lb.



# Appendix D: ORIENTER PROTOTYPE PARTS LIST

| Mechanical Parts                            | Dwg/Part #  | Material                  | Basic Dimensions                    | Amount/type reqd. fasteners  | Cost    | Notes           |
|---|---|---------------------------|-------------------------------------|------------------------------|---------|-----------------|
| orienter frame (bottom piece) (1)           | SW100   | Plywood (a)               | 1" x 46" x 25"                      | 0                            | 0.00    | (incl. in a)    |
| linear actuation motor mounts (2)           | SW101, SW102  | Steel                     | 1" x 4.31" x 4.81"                  | 4: 3/8-16 x 1.5"             | 280.00  |                 |
| linear actuation motors (2)                 | OP100, OP101  |                           | 4.31" d x 8.5"                      | 8: 10-32 x 1.5"              | 962.00  |                 |
| motor/pulley shaft couplings (2)            | OP110, OP111  | Iron, Hytrel              | 1.75" d x 2.125" l x 0.625" bore    | 0                            | 30.64   |                 |
| pulley blocks (8)                           | OP120 - OP127   |                           | 0.625" bore                         | 16: 3/8-16 x 0.75"           | 109.36  |                 |
| pulley block spacers (4)                    | SW103-106   | Aluminum (b)              | lg. 1.467" x 4.75" x 4.68"          | 8: 1/4-20x3", 8: 1/4-20x1.5" | 0.00    | (incl. in b)    |
| pulley shafts (4)                           | OP130   | Steel (c)                 | 0.6245" d, total 32.25" l           | 0                            | 0.00    | (incl. in c)    |
| pulley shaft retaining rings (8)            | OP131 - OP136   |                           | 0.625" bore                         | 0                            | 2.00    |                 |
| pulleys (2)                                 | OP140 - OP143   |                           | lg. 2.875" od, 2" w                 | 0                            | 77.90   |                 |
| belt (2)                                    | OP144, OP145  | Glass cord                | 1" w, 37.5 and 36.5 centers         | 0                            | 52.52   |                 |
| ball/linear guide block connectors (2)      | SW122, SW123  | Aluminum (f)              | 2.5" x 2.25" x 0.899"               | 0                            | 0.00    | (incl. in f)    |
| linear guide and two blocks (+1replem)      | OP150   |                           | 930mm rail                          | 16: 10-32 x 1"               | 496.00  |                 |
| rotary actuation motor mounts (2)           | SW120, SW121  | Steel                     | 0.625" x 4.31" x (9.54" and 11.42") | 12: M4                       | 360.00  |                 |
| rotary actuation motors (2)                 | OP102, OP103  |                           | 4.31" d x 8.5" l                    | 8: 10-32 x 1"                | 962.00  |                 |
| rotary actuation motor shaft collars (2)    | OP112, OP113  | Aluminum                  | 0.625" id, 1.3125" od, 0.437" l     | 4: 4-40 x 0.75"              | 5.80    |                 |
| arm rails (2)                               | SW124, SW125  | UHMW                      | 7.26" x 0.75" x 0.5"                | 6: 10-32 x 0.75"             | 170.00  |                 |
| arm guards (2)                              | SW126, SW127  | Steel                     | 0.25" x 2.9" x 11.09"               | 0                            | 130.00  |                 |
| rib support rail (1 part made of 2 pieces)  | SW131, SW132  | UHMW                      | (2) 46.6" x 1" x 3.75"              | 12: 8-32 x 0.75"             | 185.00  |                 |
| orienting platform surface (1)              | SW130   | UHMW (d)                  | 1" x 11.37" x 53.5"                 | 11: 10-32 x 1.5"             | 0.00    | (incl. in d)    |
| orienting platform support (1)              | SW140   | Plywood (e)               | 1" x 11.37" x 53.3"                 | 0                            | 0.00    | (incl. in e)    |
| platform spacers (8)                        | SW150 - SW158   | Aluminum (g)              | lg. 1" x 6.19" x 2.357"             | 20: 10-32x1.5", 2: 1/4-20x3" | 0.00    | (incl. in g)    |
| orienter frame (top piece) (1)              | SW160   | Wood (a)                  | 1" x 46" x 14.75"                   | 6: 1/4-20 x 3"               | 0.00    | (incl. in a)    |
| linear guide and rotational shaft mount (1) | SW110   | Aluminum                  | 1.5" x 3" x 46" 0.188" l            | 14: 1/4-20 x 3"              | 140.00  |                 |
| rotational shaft and connection blocks (2)  | SW111   | Steel                     | 1.125" x 1.125" x 4"                | 0                            | 90.00   |                 |
| rotational shaft retaining rings (2)        | OP137, OP138  |                           | 0.625" bore                         | 0                            | 0.50    |                 |
| orienter base sides (2)                     |   | Plywood (a)               | 1" x 22" x 33.125"                  | 0                            | 0.00    | (incl. in a)    |
| orienter base assembly rails (2)            |   | Steel                     |                                     | 32: 10-32 x 1.5"             | 200.00  | (incl. in e)    |
| orienter base connection pieces (6)         |   | Aluminum (e)              | 1" x 5" x 5"                        | 0                            | 0.00    | (incl. in e)    |
| miscellaneous                               |   |                           |                                     |                              |         |                 |
| Materials:                                  | OP - ordered part #<br>SW - part drawing #<br>- fabricated out-of-house | (k) - material supply pc. |                                     |                              |         |                 |
| oak plywood                                 |   | (a)                       | 4" x 8" x 3/4"                      |                              | 44.99   |                 |
| aluminum C channel                          |   | (b)                       | 1.5" x 5" x 25"                     |                              | 30.00   |                 |
| steel shaft                                 |   | (c)                       | 0.6245" bore, 32.125"               |                              | 0.00    |                 |
| UHMW sheet                                  |   | (d)                       | 0.25" x 54" x 12"                   |                              | 52.00   |                 |
| aluminum L Channel                          |   | (e)                       | 5" x 5" x 8"                        |                              | 10.00   |                 |
| aluminum block                              |   | (f)                       | 2.25" x 2.5" x (1.75" and 2.25")    |                              | 0.00    |                 |
| aluminum tubing                             |   | (g)                       | 1" x 3" x 77"                       |                              | 42.00   |                 |
| Control Components:                         |   |                           |                                     |                              |         |                 |
| Servo amps (4)                              | OP104 - OP107   |                           |                                     |                              | 1180.00 |                 |
| Power supplies (2)                          | OP108, OP109  |                           |                                     |                              | 598.00  |                 |
| Controller cards (2)                        | OP140, OP141  |                           |                                     |                              | 0.00    |                 |
| Connector boards (2)                        | OP142, OP143  |                           |                                     |                              | 0.00    |                 |
| Cables                                      | OP144   |                           |                                     |                              | 1275.00 | (incl. 3 items) |
| Software                                    | OP145   |                           |                                     |                              | 250.00  |                 |
| Total Spent:                                |   |                           |                                     |                              | 7735.81 |                 |





**Appendix E: ORIENTER PROTOTYPE ASSEMBLY AND FABRICATION DRAWINGS**

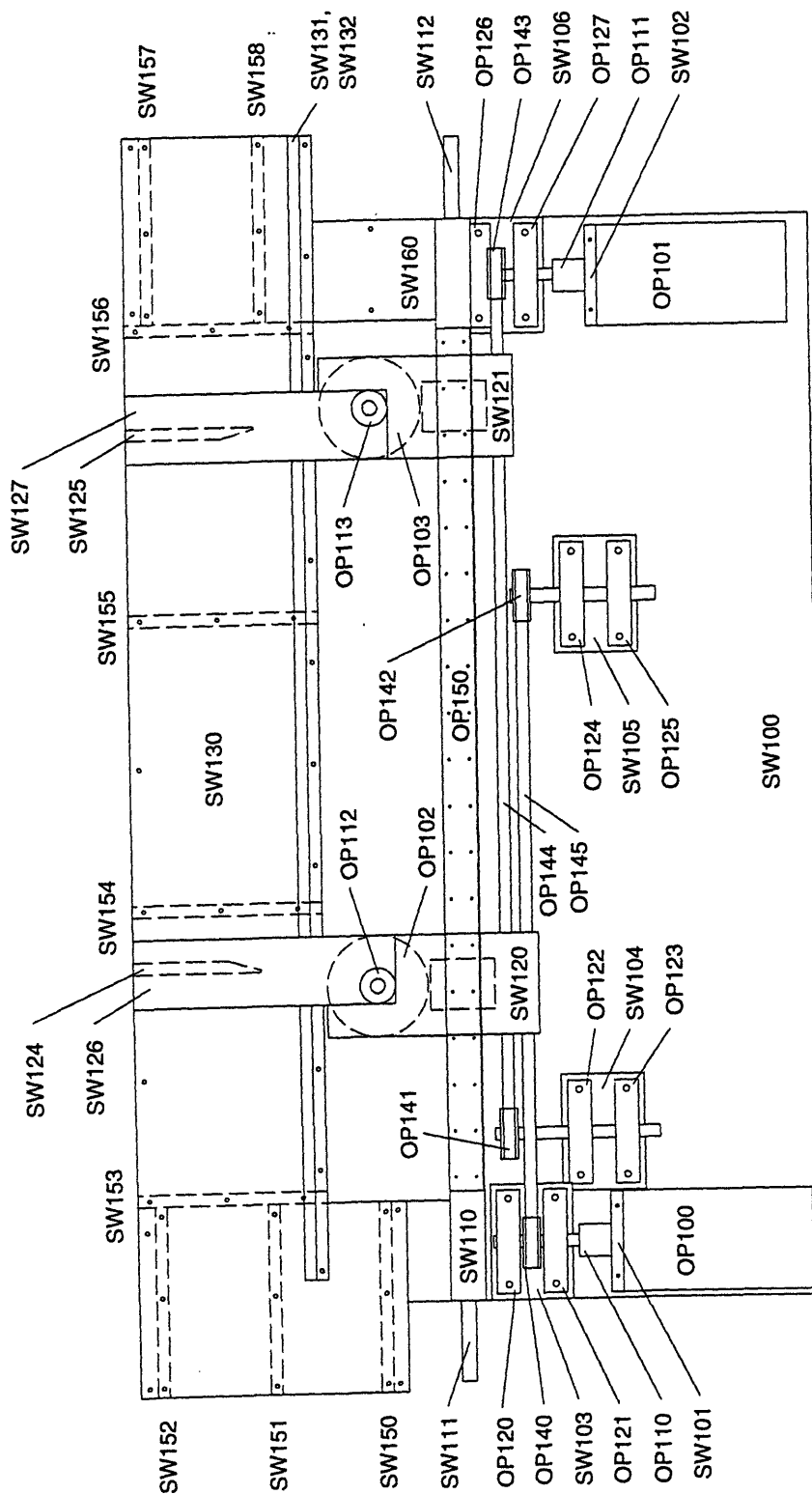


Figure E1: Prototype Assembly Drawing

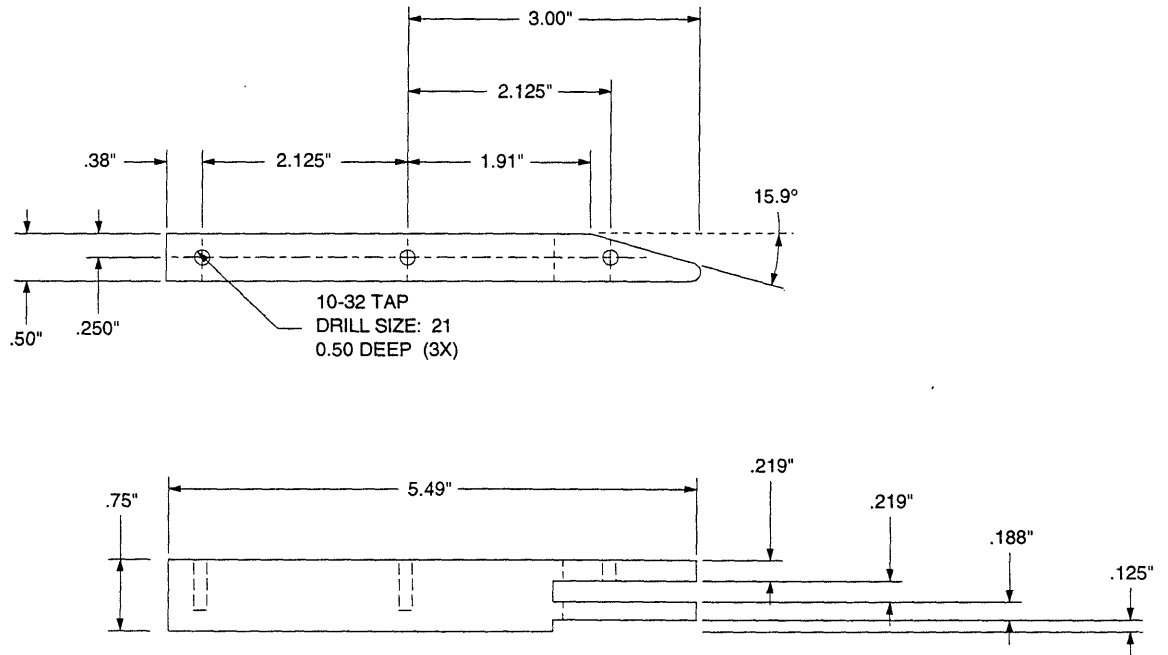


Figure E2:  
 LEFT ARM RAIL  
 DWG/PART#: SW124  
 MTRL: UHMW

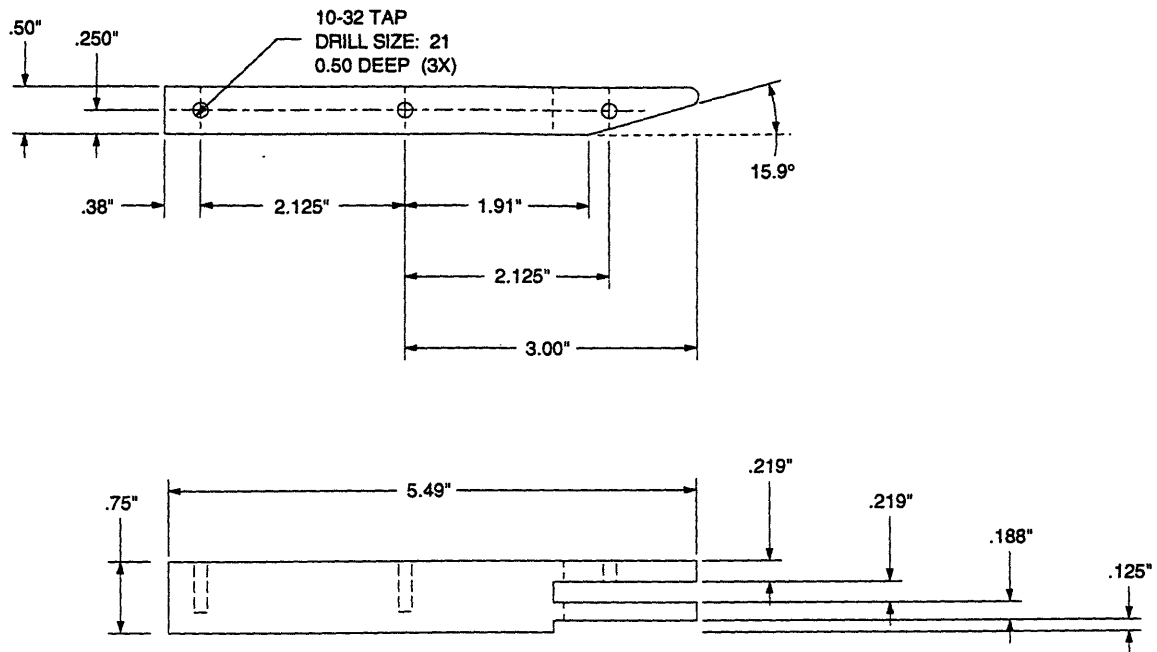


Figure E3:  
 RIGHT ARM RAIL  
 DWG/PART #SW125  
 MTRL: UHMW

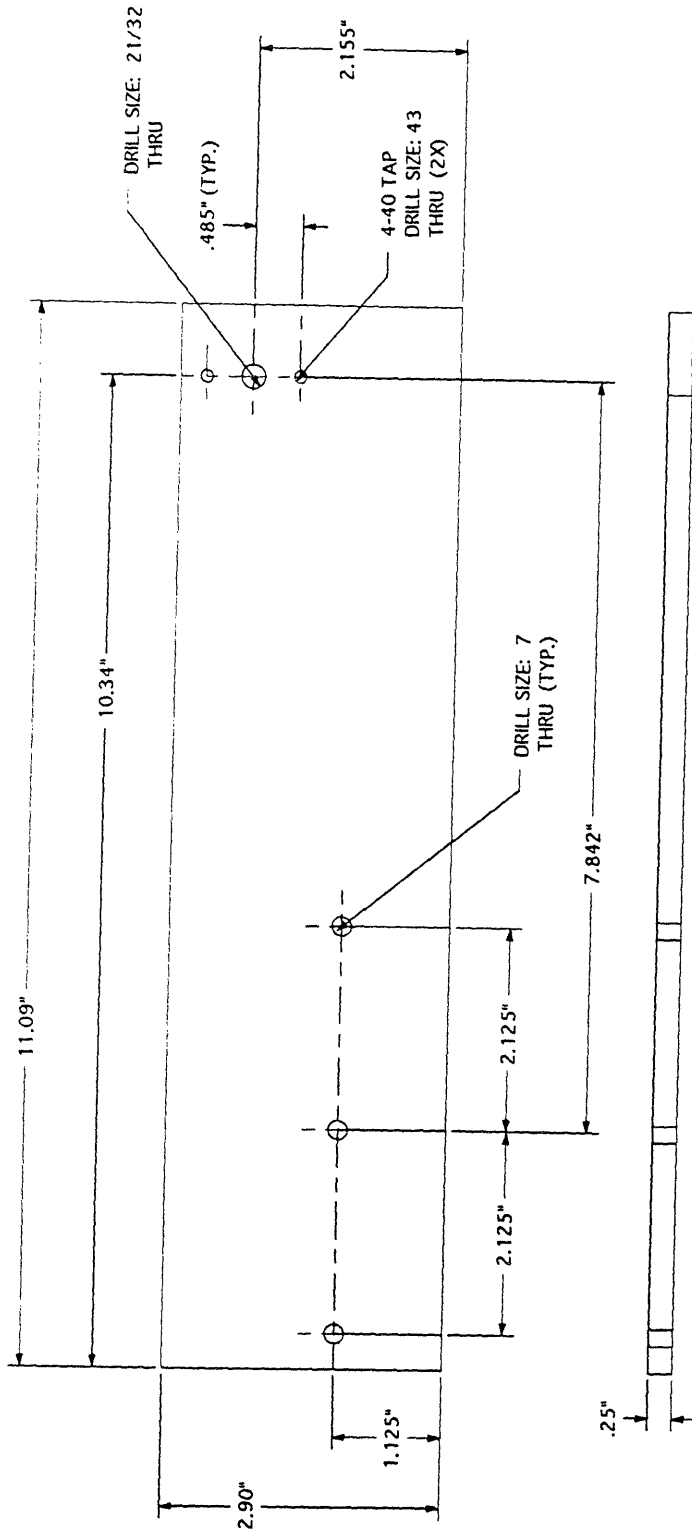


Figure E4:  
 ARM GUARD  
 DWG/PART#: SW126, SW127  
 MTRL: STEEL

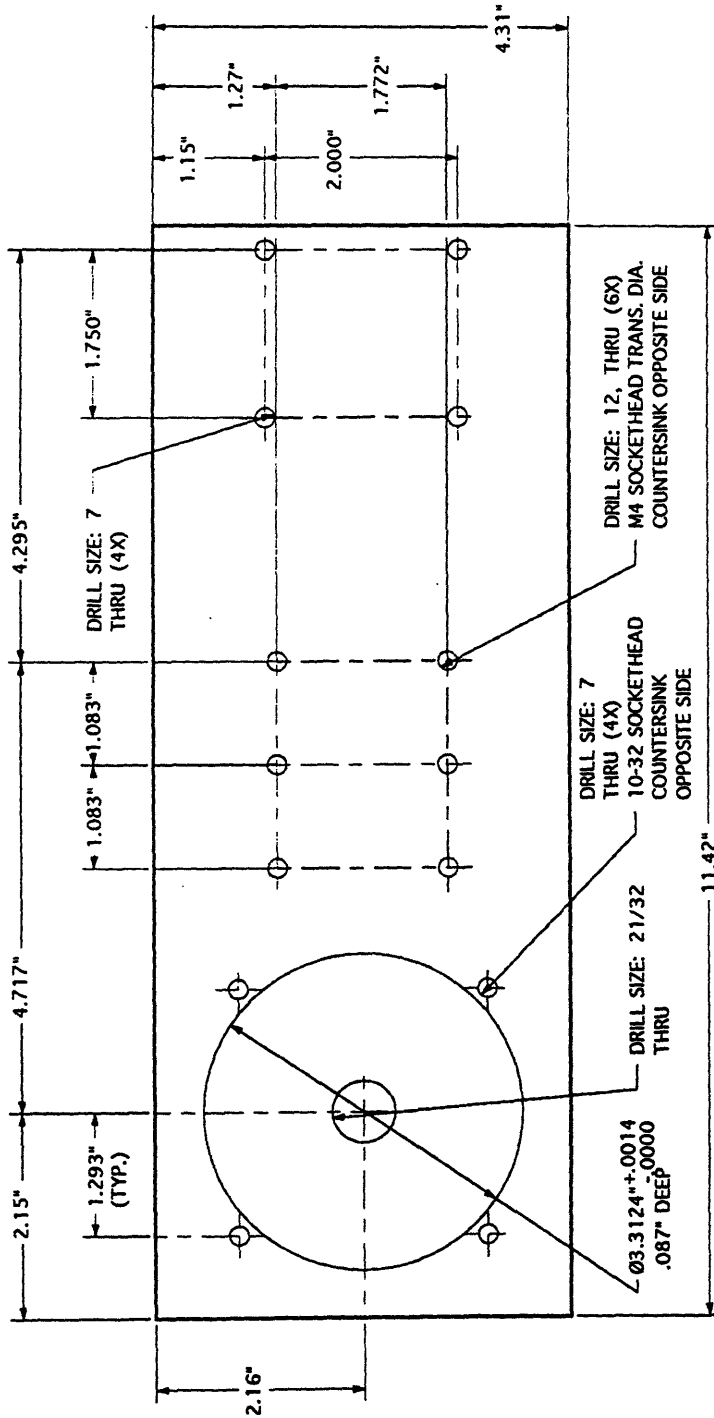


Figure E5:

LEFT ROTARY MOTOR MOUNT  
 DWG/PART #: SW121  
 MTRL: STEEL

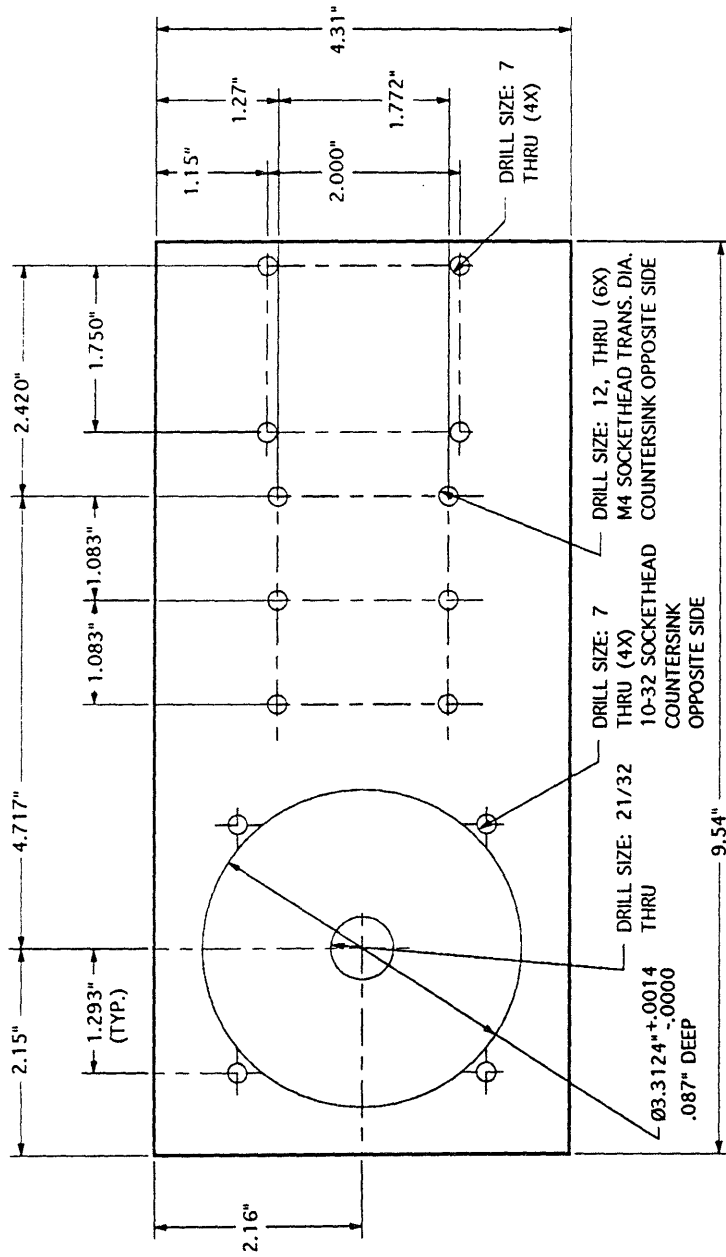


Figure E6:

RIGHT ROTARY MOTOR MOUNT  
 DWG/PART#: SW120  
 MTRL: STEEL

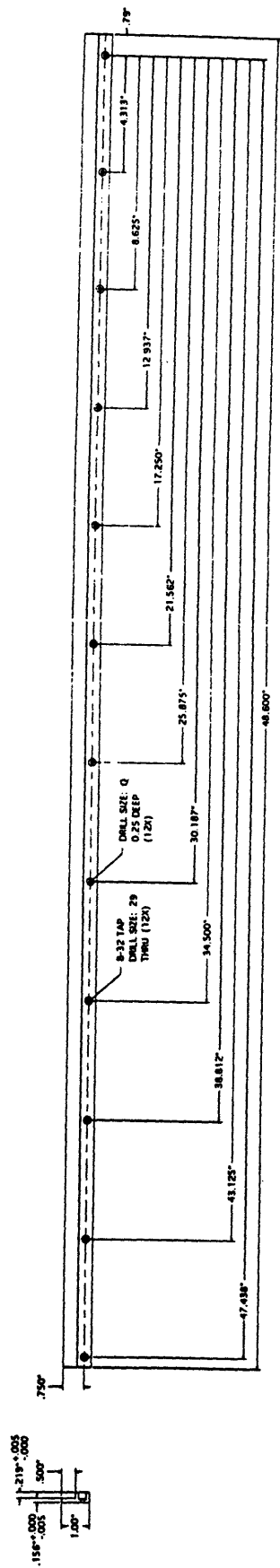


Figure E7:

RB SUPPORT RAIL (TOP PC)  
 DWG/PART#: SW131  
 INTL: UNWIN

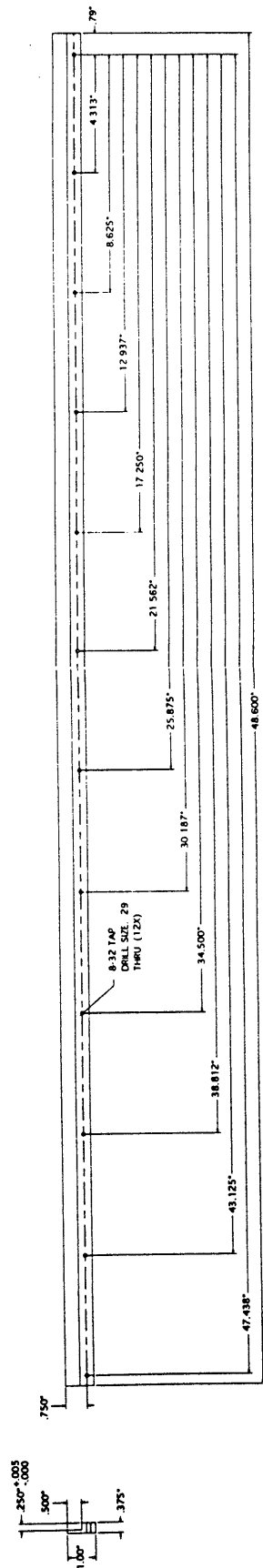


Figure E8:  
 RB SUPPORT RAIL (BOTTOM PC)  
 DWG/PART# SW132  
 MTRC.UHMY



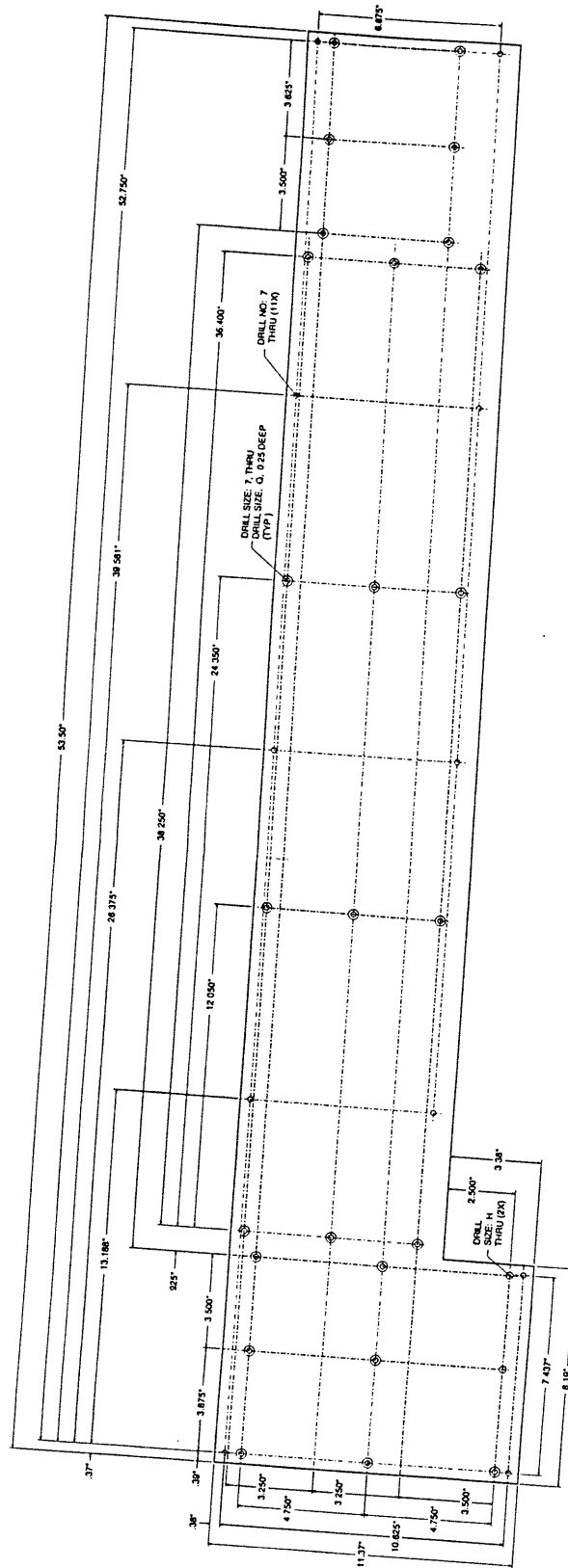


Figure E9:

ORIENTING PLATFORM SUPPORT  
 DWG PART: 1140  
 MFLC.DWG PLYWOOD

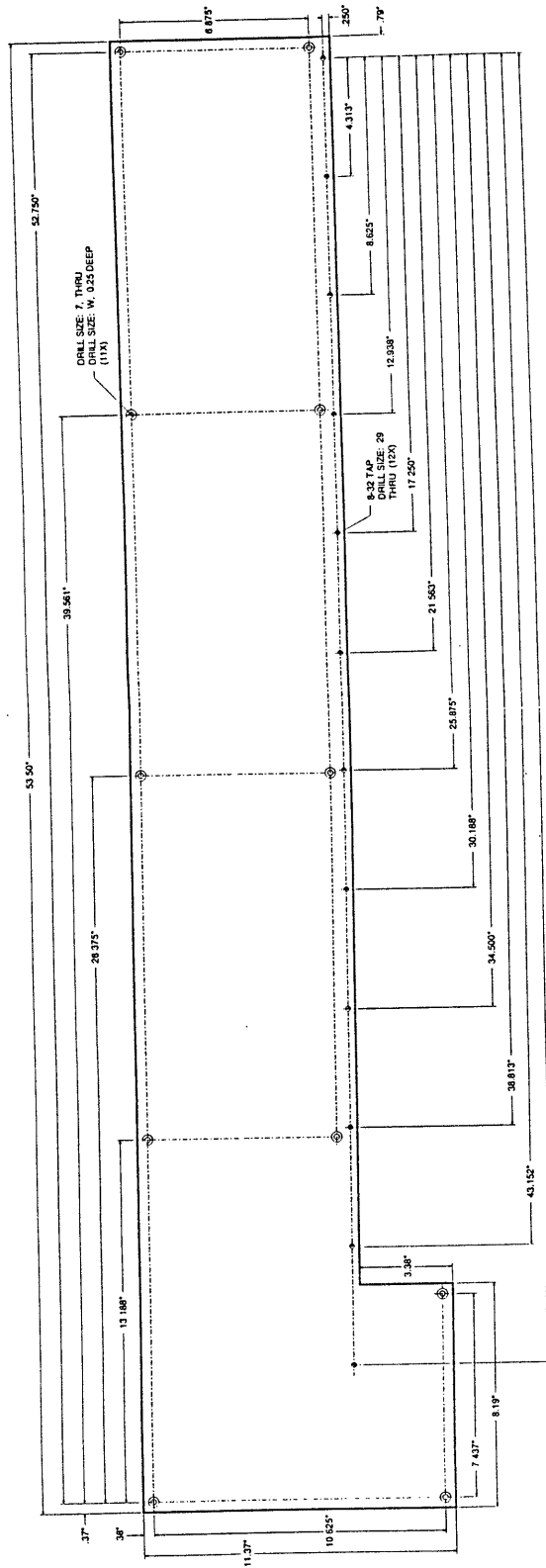
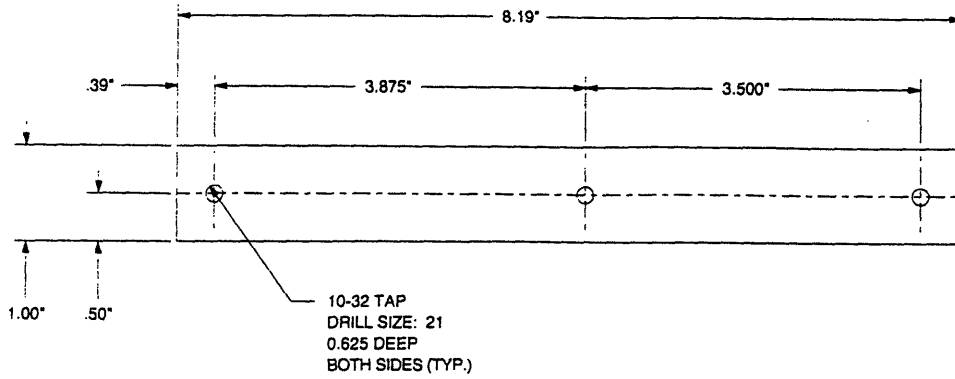


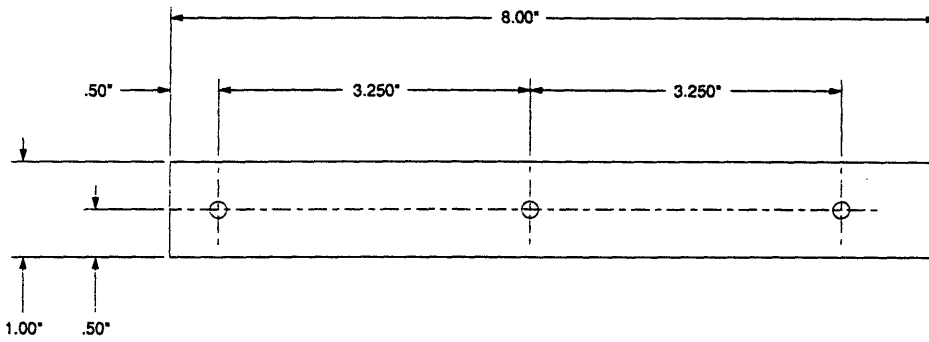
Figure E10:

ORIGINS PLANTWORK SURFACE  
DIM PARTS SW130  
MTRL: UHMW

NOTE: BUILD THREE 2.357" THICK PIECES TO THESE DIMENSIONS



NOTE: BUILD FOUR 2.357" THICK PIECES TO THESE DIMENSIONS



NOTE: BUILD TWO 2.357" THICK PIECES TO THESE DIMENSIONS

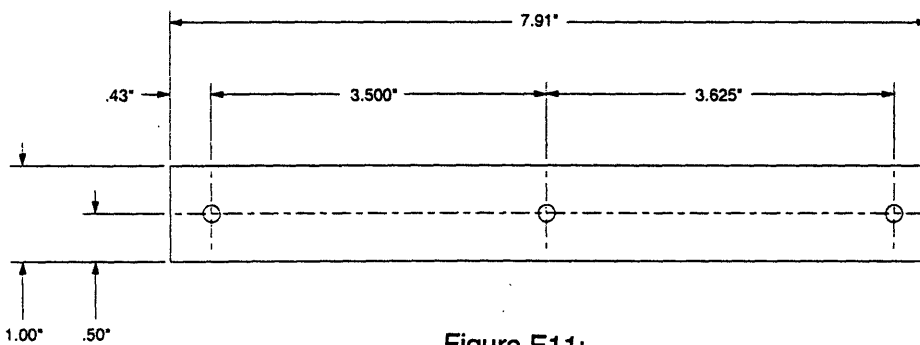


Figure E11:

PLATFORM SPACERS  
DWG/PART#: SW150-158  
MTRL: 1"X3"X.25"ALUMINUM TUBING



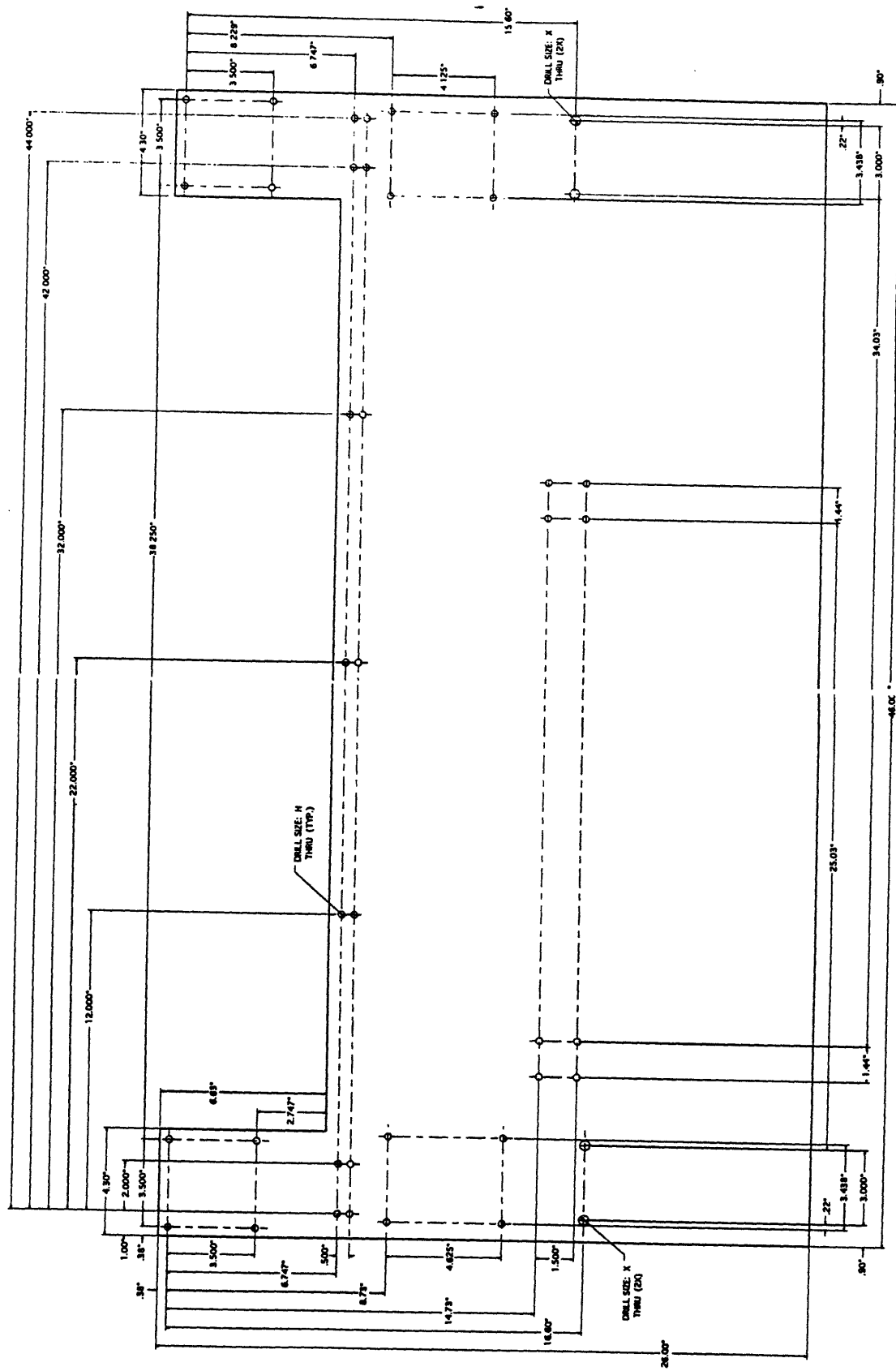
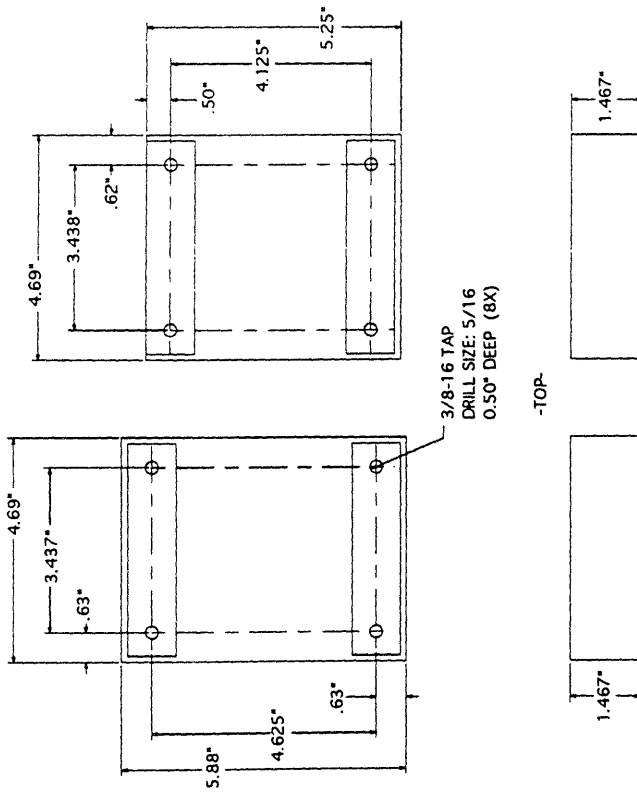


Figure E13:

ORIENTED FRAME (BOTTOM FC)  
 DIMENSIONS: AS SHOWN  
 MTRL: 3/4" OAK PLYWOOD

NOTE: SAME HOLE LOCATIONS ON TOP AND BOTTOM.  
BEGIN WITH SIZE 7 DRILL THRU



NOTE: SAME BLOCK DIMENSIONS FOR EACH PART  
BUT WITH MIRROR IMAGE SLOT LOCATIONS.

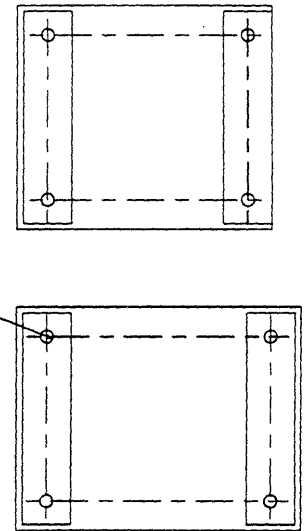
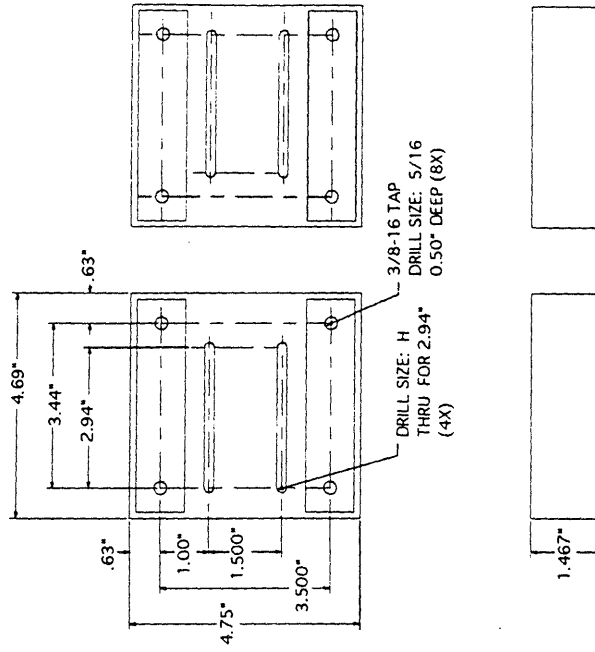


Figure E14:

PILLOW BLOCK SPACERS  
DWG./PART#: SW103-SW106  
MTRL: ALUMINUM C CHANNEL

NOTE: ALL 4.69" DIMENSIONS WERE  
EXTENDED TO 5" TO FIT C CHANNEL

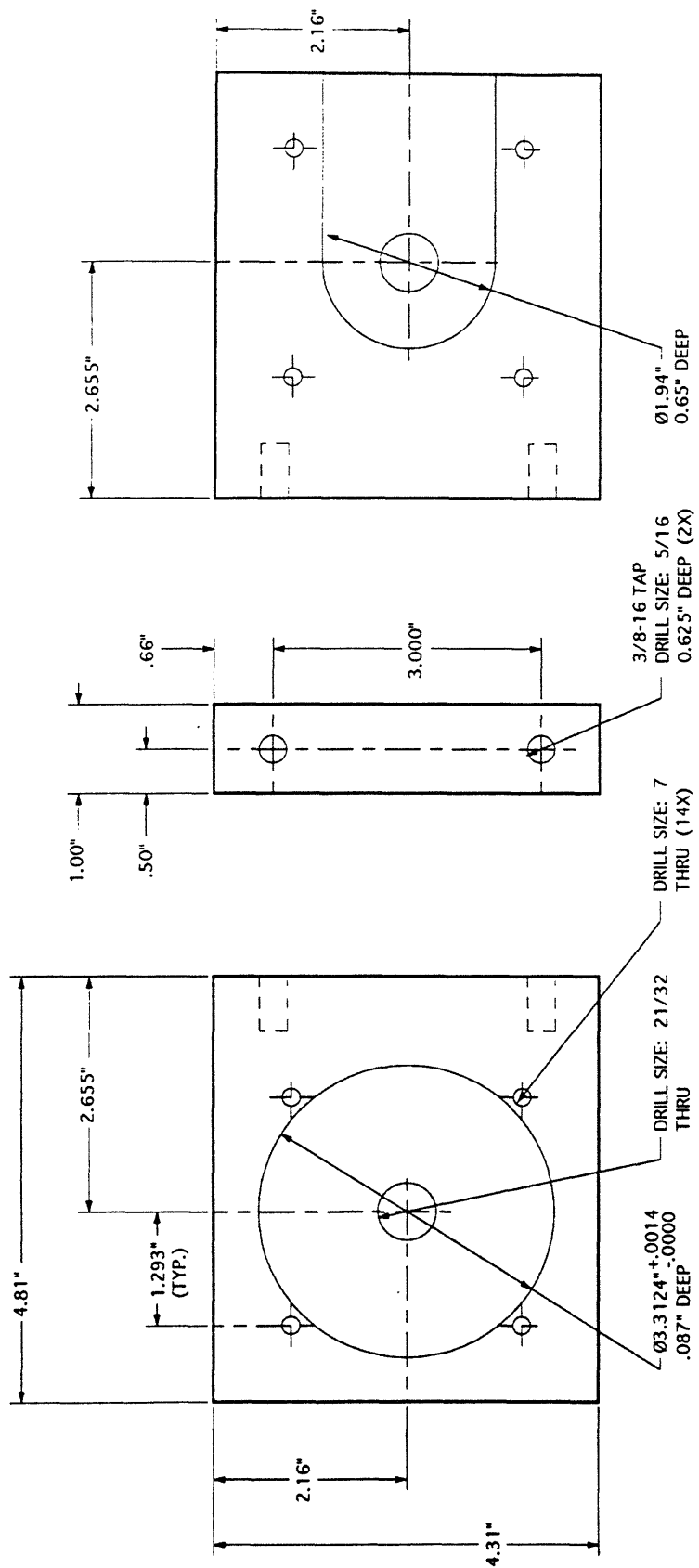


Figure E15:

LINEAR MOTOR MOUNT  
 DWG/PART #: SW101, SW102  
 MTRL: STEEL

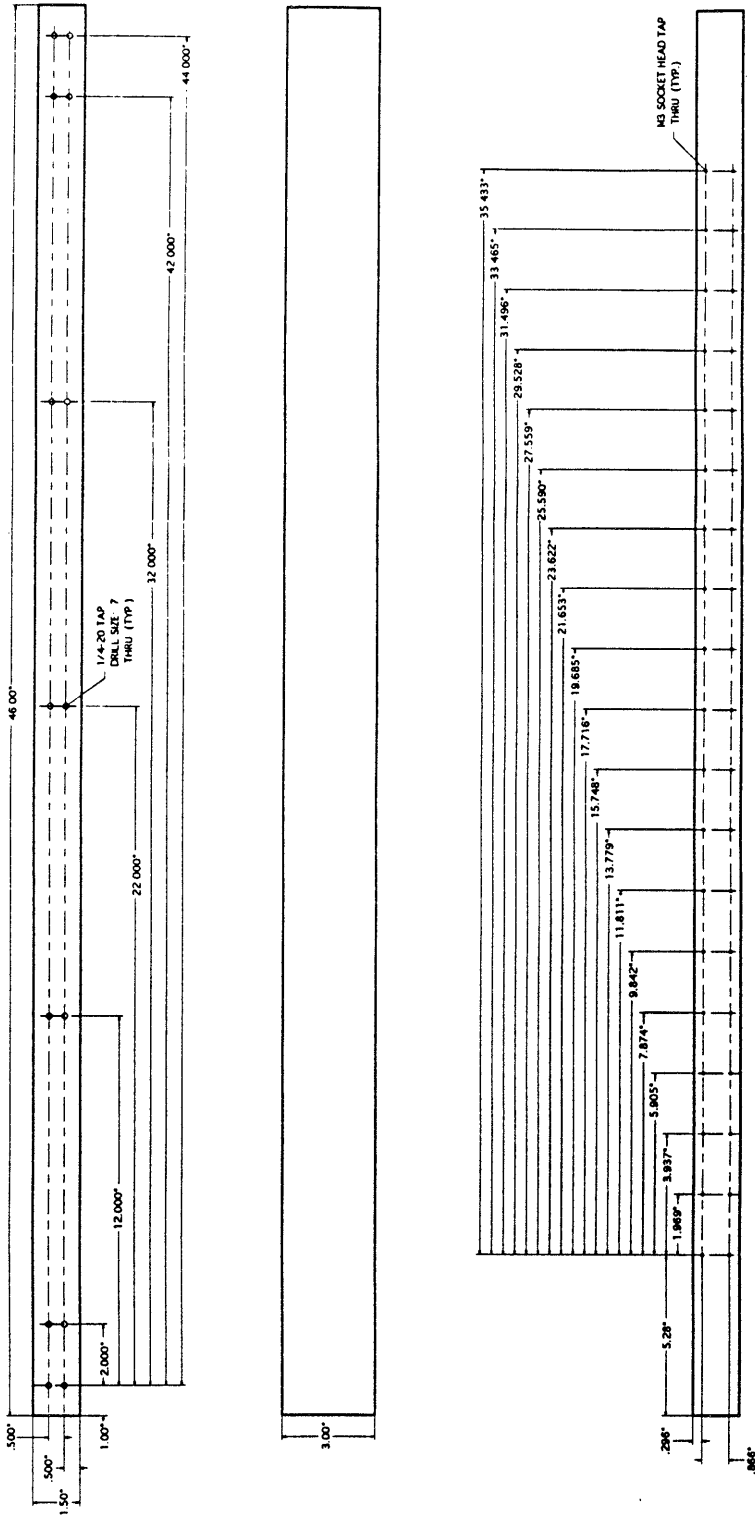


Figure E16:

LINEAR GUIDE AND ROTATIONAL SHAFT MOUNT  
 MATERIAL: 3010  
 MTL: ALUMINUM TUBING



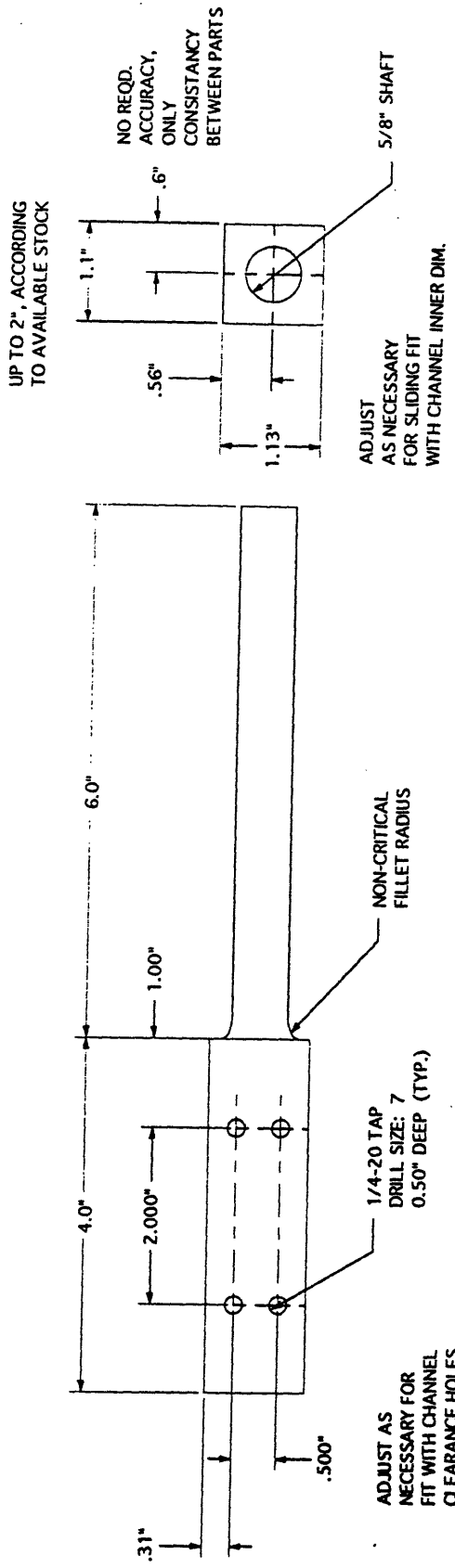


Figure E17:

ROTATIONAL SHAFT AND CONNECTION BLOCKS  
 DWG/PART#: SW111, SW112  
 MTRL: STEEL



## **Appendix F: INITIAL CONTROL PROGRAMS**

Four initial control programs were written to be used with the exerciser program provided with the control hardware. If "exer" was typed at the prompt, generic commands which in turn actuated a corresponding series of register setting commands could be entered line-by-line. The programs described in this appendix, consisting of simple lists of these same commands, were accessed by including their names along with the "exer" command, i.e., "exer startup.cmd."

The "startup.cmd" program was written to be executed before every orientation process. This program assumed that the user had moved the arms into their upright positions, and the blocks into their closed positions. First, the encoders were set to zero, and then motor constants including the poles, zeroes, gains, maximum velocities, accelerations, and timers were designated. This effectively calibrated the position of the motors. Then, the arms were moved into their open ready positions for receiving ribs. At this point, any orientation program could be executed. When the main control code was written, this program was split into the two smaller sections--init() and tostart()--such that they could be employed separately.

To complement the "startup.cmd", a "shutdwn.cmd" program was written. This program initially moved the blocks and rotated the arms away from each other to eliminate interference potential, and then returned them all to their zero positions. This program was embellished and then translated into shtdwnmt() for the main code.

For situations where an orientation process was completed successfully, the "restart.cmd" program was written. This program simply performed the "shutdwn.cmd"

followed by the "startup.cmd" in order to reinitiate the motors for subsequent orientation without user interface.

If a rib was stuck between or covered by the arms, it could be released by executing the "release.cmd" which rotated the arms into their open positions without moving the block.

## **Appendix G: DOCUMENTED CONTROL CODE**

### **ORIENT.C**

**Orienter Control Code**

**by**

**Susie Ward**

**April 1994**

**This is the main code for operating the rib orienter.**

**\*\*\*\* Note: \*\*\*\***

The entire code for the orienter is organized such that it may be followed and understood in increasing detail. For example, **ORIENT.C** only contains the **main()**, which is a simple outline of the entire code operation. The procedures called within the **main()** are contained in **CLOOPIF.H**, alias the "Control Loop Include File." These procedures are also simple outlines, created to control the logic of all individual phases of orientation, including startup, calibration, geometry input, orientation, wrapup, and shutdown. Within these control loops, further calls are made to more detailed procedures contained in **BCMDIF.H**, alias the "Base Command Include File." These procedures, necessary for setting motor constants, calculating and moving to set positions, delaying between commands, etc, are incorporated in this file along with the MC controller level C macros (provided by Omnitech Robotics) that they call. Finally, the orientation algorithms are contained in a file separate from the others, called **OALGIF.H**, alias "Orientation Algorithm Include File." (This file, along with **BCMDIF.H** and all standard libraries is included at the beginning of **CLOOPIF.H**.) For more information, see the documentation within the individual files.

All **/\* \*/** have been removed for the purposes of organized file conversion and presentation, but all intermeshed comments have been converted to a smaller font such that they are more easily recognized as separate from the main code.

---

Include the necessary command loop procedures

```
#include "cloopif.h"
```

Begin main program

```
void main()  
{  
  startup();
```

Start up the system: Check if motors are in their closed positions. If they are, set motor control positions and constants, and determine the orientation mode: standard or demonstration.

\*\*\*\*\* Main Program Loop \*\*\*\*\*

```
  while (mainprog != QUIT)  
  flag is set  
  {
```

While the main program continuation

```
    calib();
```

1.) Calibrate motors

```
    if (mainprog != QUIT)  
  calibration...  
    {  
      get_geom();  
    }
```

...If no error was incurred during

2.) Get rib geometries

```
  geometry  
  if (mainprog != QUIT)  
  {  
    orient();  
    wrapup();  
  }
```

...If no error was incurred during

input...

3.) Orient the rib

4.) Evaluate the orientation

...If no error was incurred during  
orientation...

REPEAT

\*\*\*\*\* Main Program Loop \*\*\*\*\*

```
  lastchk();
```

Execute a final system check and shutdown.

```
} End: Main Program
```

## **CLOOPIF.H**

**Control Loop Include File**

**by  
Susie Ward**

**April 1994**

This header file contains all procedures called in the ORIENT.C "main()". These include the control loops for system startup, calibration, geometry input, orientation, wrapup, and final system check and shutdown. The actual controller commands are included in the procedures called from this file, which are either contained in BCMDIF.H, the Base Command Include File, or OALGIF.H, the Orientation Algorithm Include File.

**\*\*Note:** all /\* \*/ have been taken out for the purposes of organized file conversion and presentation, but all intermeshed comments have been converted to a smaller font such that they are more easily recognized as separate from the main code.

---

**Include standard libraries necessary for calculation and register manipulation.**

```
#include <stdio.h>  
#include <conio.h>  
#include <math.h>  
#include <stdlib.h>  
#include <string.h>
```

**Include header files containing global variables, procedures, orientation algorithms, and register commands.**

```
#include "bcmdif.h"  
#include "oalgif.h"
```

---

**Begin regular code...**

## STARTUP()

Verify that the motors are in their closed positions and that the user is ready to continue. If so, initialize the motors (set positions to zero) and set motor constants.

```
void startup()
{
    Define variables

    char sinput1, sinput2;    startup ready, mode inputs
    char sinput1f, sinput2f;  startup input status flags

    Print introductory credits

    clrscr();

    printf("\n\n\n");
    printf("          FLEXIBLE GEOMETRY (RIB) ORIENTER");
    printf("\n\n\n\n");
    printf("          Design and Control Algorithms");
    printf("\n\n");
    printf("                by");
    printf("\n\n");
    printf("                Susan D. Ward");
    printf("\n\n\n\n");
    printf("          Under the Advisement of Andre Sharon");
    printf("\n\n\n");
    printf("          The Manufacturing Institute at MIT");
    printf("\n\n");
    printf("                1993 - 1994");
    printf("\n\n\n\n");
    printf("Press enter to continue...");
    scanf("%c",&garbage);

    clrscr();
}
```



### Begin user interface

```
printf("\n\nStartup: \n");
```

```
    Repeat until valid mode input is entered
for (sinput1f = NEEDED; sinput1f != VALID;)
{
```

```
    Notify user of orientation mode options
printf("Enter the desired orientation mode: ");
printf("\n 1.) Demonstration");
printf("\n 2.) Standard");
```

```
    Get user orientation mode preference
printf("\nYour choice: ");
scanf("%c%c",&sinput1,&garbage);
```

```
    If demonstration mode is selected
if (sinput1 == '1')
```

```
{
    sinput1f = VALID;           Set input flag; input
    omode = DEMO;             Set orientat. mode flag
}
```

```
    If standard mode is desired
else if (sinput1 == '2')
```

```
{
    sinput1f = VALID;         Set input flag; valid
    omode = STANDARD;       Set orientat. mode flag
}
```

```
    Otherwise there was an input error
else
```

```
{
    sinput1f = INVALID;      Set input flag; invalid
    printf("Incorrect entry...");
}
```

```
}           End: Mode input loop
```

```
printf("\n");
```

```

Repeat until valid ready input is entered
for (sinput2f = NEEDED; sinput2f != VALID;)
{
    Get input as to user and system status
    printf("Are you and the system ready to continue? ");
    printf("\n i.e. Motors are in their closed positions (y or n)? ");
    scanf("%c%c",&sinput2,&garbage);

    If user and system are ready
    if (sinput2 == 'y')
    {
        If motors ARE reasonably near their starting (closed, "ready") positions.
        if (mposchk(768,0,40) == 'y' && mposchk(992,0,150) == 'y' &&
            mposchk(993,0,600) == 'y' && mposchk(994,0,150) == 'y')
        {
            sinput2f = VALID;                set input flag; valid
            startmt();                       execute startup procedure
        }

        Otherwise response was invalid
    else
    {
        sinput2f = INVALID;                set input flag; invalid
        printf("NOTE: Motors are NOT near their closed positions. \n");
    }
    }

    If system or user are not ready
    else if (sinput2 == 'n')
    {
        sinput2f = VALID;                set input flag; valid
        mainprog = QUIT;                set main program exit flag
        printf("\nExiting program...");
    }

    Otherwise there was an input error
    else
    {
        sinput2f = INVALID;                set input flag; invalid
        printf("Incorrect entry...");
    }
}
End: User/system status input loop
End: Startup procedure

```

---

## CALIB()

Attain user input for, and execute calibration mode options including moving to the closed calibration check point, calibrating motors and setting motor speeds, moving to the open start position, continuing to geometry input, or quitting the orientation program.

```
void calib()
```

```
{
```

```
    Define Variables
```

```
double stconv;           sample time conversion
double blksped;         arm block speed, inches/second
double speedval;       velocity adjustment value
double timerval;       new timer value
int x;                 timer adjustment counter

char calibproc;        calibration process flag
char cstep = 'x';     calib. step var, set to avoid error from possible
                    "valid" but garbage value possibly present in
                    demo mode

char spadjust;         speed adjustment flag
char spadjustl;       speed adjustment loop flag
```

```
    Begin User Interface
```

```
printf("\nCalibration: \n");
```

```
        Repeat while calib. options may be executed
```

```
for (calibproc = BEGIN; calibproc != QUIT;)
```

```
{
```

```
        If in standard orientation mode
```

```
if (omode == STANDARD)
```

```
{
```

```
        Present calibration options
```

```
printf("Select from the following: \n");
printf(" 1.) Move motors to calibration home positions \n");
printf(" 2.) Motors are in (closed) home positions, calibrate \n");
printf(" 3.) Motors are calibrated, move to start positions \n");
printf(" 4.) Motors in start positions, input rib geometries \n");
printf(" 5.) Quit orientation program \n");
```

```
        Get user option input
```

```
printf("Enter your choice by number: ");
scanf("%c%c",&cstep,&garbage);
```

```

        If invalid input was given
if (cstep != '1' && cstep != '2' && cstep != '3' && cstep != '4'
    && cstep != '5')
    printf("Incorrect entry...");
}

        If the motor homing opt. has been selected
if (cstep == '1')
{
    printf("\nMoving to calibration check positions... \n\n");
    shtdwnmt();                                Move motors to closed pos
}

        If the motor calibration option has been selected or the system is in demo mode
if (cstep == '2' || omode == DEMO)
{
    If motors are in closed positions or the system is running in demo mode
if (mposchk(768,0,60) == 'y' && mposchk(992,0,60) == 'y' &&
    mposchk(993,0,60) == 'y' && mposchk(994,0,60) == 'y' ||
    omode == DEMO)
{
    ONLY if the system is in standard mode
if (omode == STANDARD)
    startmt();                                Initialize motors, set constants

    Begin a speed adjustment option loop
for (spadjustl = BEGIN; spadjustl != DONE;)
{
    Calculate current block speed
stconv = 8.0*(timers[2] + 1.0)*0.000001;      Eqn for sample time conversion
blksped = ((mvelos[2])*(1.0/INTOQD))/(stconv);  from Omnitech manual,
                                                block speed calc. by unit conv.

    Present block speed and adjustment option
printf("Block speed is currently %5.2lf in/sec; Adjust (y or n)? ",blksped);
scanf("%c%c",&spadjust,&garbage);

    If the user doesn't want to adjust the speed
if (spadjust == 'n')
{
    spadjustl = DONE;                            Set loop flag; done
    if (omode == STANDARD) printf("\n");
}

    If there was an incorrect entry
else if (spadjust != 'y')
    printf("Incorrect entry...");
}
}

```

```

        If speed is to be adjusted
else
    {
        Get new speed from the user
        printf("Enter new value for speed: ");
        scanf("%lf%c",&speedval,&garbage);

        Calculate corresponding timer value, maximum velocity isn't changed
        since arm and block velocities must stay porportional
        stconv = (mvelos[2]/speedval)*(1.0/(INTOQD));
        timerval = (stconv/0.000008) - 1.0;
        printf("New timer value: %d",(int) timerval);

        If the timer value is within a reasonable/acceptable range
        if (timerval > 30 && timerval < 100)
            {
                Set all timers to the new value
                for (x = 0; x < 4; x++)
                    {
                        timers[x] = (int) timerval;
                        set_base(bases[x]);
                        set_timer(timers[x]);
                    }
                Notify of the change
                printf("\nSpeed adjusted.\n\n");
            }

        Otherwise the speed is too fast or too slow
        else
            {
                printf("\nThe desired speed is not allowed ");
                printf("with this prototype\n\n");
            }
    }
    End: Speed adjustment loop
}
    End: Speed adj. option present loop
}
    End: Motors in correct position loop

        If motors are NOT in their closed positions
else
    printf("NOTE: Motors are NOT near their home positions. \n");
}
    End: Option 2

```

```

        If the motors start pos. option has been selected or the system is in demo mode
if (cstep == '3' || omode == DEMO)
{
    printf("\nMoving to startup positions... \n\n");
    tostart();                Move motors to start positions
}

        If the procedure exit option has been selected or the system is in demo mode
if (cstep == '4' || omode == DEMO)
{
    If motors are in start positions
if (mposchk(768,0,70) == 'y' && mposchk(992,2800,70) == 'y' &&
    mposchk(993,-32000,70) == 'y' && mposchk(994,2800,70) == 'y')
    calibproc = QUIT;                Set flag to quit option prompts

    If motors are not in their start positions
else
    {
        printf("NOTE: Motors are NOT in start positions. \n");
        if (omode == DEMO)
            {
                printf("Exiting program...\n");
                cstep = '5';
            }
    }

}

        If the main program exit option has been selected
if (cstep == '5')
{
    calibproc = QUIT;                Set flag to quit option prompts
    mainprog = QUIT;                Set main program exit flag
}

}                End: Calibration option entry/execution loop

}                End: Calibration procedure

```

---

## GET\_GEOM()

Get and verify rib geometries from the user.

```
void get_geom()
```

```
{
```

Define Variables (Note: actual geometry variables are global)

```
char geomstat;           Geometry status flag
char ginput1, ginput2;   Geom. continuation inputs
char ginput1f, ginput2f; Corresponding input error flags
```

Begin User Interface

```
printf("Geometry Input: ");
```

Repeat until geometry input is valid or done

```
for (geomstat = NEEDED; geomstat != VALID && geomstat != DONE;)
```

```
{
```

If the system is running in standard mode

```
if (omode == STANDARD)
```

```
{
```

Prompt user for bottom dim and side angles

```
printf("\nEnter the bottom edge length (inches): ");
scanf("%lf",&bott);
printf("From the outside horizontal, enter (degrees):\n");
printf(" 1.) Left edge angle (positive CW): ");
scanf("%lf",&langle);
printf(" 2.) Right edge angle (positive CCW): ");
scanf("%lf%c",&rangle,&garbage);
```

Calculate top dimension

```
topp = bott + 9.25/tan(langle*DEGTORAD) + 9.25/tan(rangle*DEGTORAD);
```

Notify user of entered and calculated dim.

```
printf("\nThe dimensions entered are as follows: ");
printf("\n Bottom edge length:%6.2lf inches",bott);
printf("\n Left edge angle:  %5.1lf degrees",langle);
printf("\n Right edge angle:  %5.1lf degrees",rangle);
printf("\n Top edge length:  %6.2lf inches\n",topp);
```

```
}
```

```

        Otherwise the system is in demo mode
else
{
    Present and get user input as to rib type demonstration numbers
    printf("\nOrientation options: ");
    printf("\n 1.) Small square (symmetric) rib");
    printf("\n 2.) Larger trapezoidal (symmetric) rib");
    printf("\n 3.) Square ended, shorter top edge rib");
    printf("\n 4.) Square ended, longer top edge rib");
    printf("\n 5.) Small non-symmetric rib");
    printf("\n 6.) Quit orientation program");
    printf("\nYour choice: ");
    scanf("%c%c",&demonum,&garbage);
}

    Repeat until the appropriate input has been verified
for (ginput1f = INVALID; ginput1f != VALID;)
{
    Prompt user
    printf("Is this correct (y or n)? ");
    scanf("%c%c",&ginput1,&garbage);

    If user signifies correct input and it actually IS within the expected range
if (ginput1 == 'y' && (omode == STANDARD && (bott > 0 &&
    langle > 44 && rangle > 44 && langle < 136 && rangle < 136))
    || (omode == DEMO && (demonum == '1' || demonum == '2' ||
    demonum == '3' || demonum == '4' || demonum == '5' ||
    demonum == '6')) ) )
    {
        ginput1f = VALID;                Set input flag; valid
        geomstat = VALID;                Set geometry flag; valid

        If the system is in demo mode, send values to global geometry variables
        corresponding to the demo choice number.
if (omode == DEMO)
    {
        if (demonum == '1')
        {
            bott = 2;
            langle = 90;
            rangle = 90;
        }
        else if (demonum == '2')
        {
            bott = 8;

```



```

        langle = 45;
        rangle = 135;
    }
else if (demonum == '3')
    {
        bott = 11.75;
        langle = 135;
        rangle = 90;
    }
else if (demonum == '4')
    {
        bott = 4;
        langle = 90;
        rangle = 45;
    }
else if (demonum == '5')
    {
        bott = 3;
        langle = 135;
        rangle = 50;
    }
else
    {
        mainprog = QUIT;                               Set main program exit flag
        printf("\nExiting program...\n");
    }

        Calculate the top dimension
topp = bott + 9.25/tan(langle*DEGTORAD) + 9.25/tan(rangle*DEGTORAD);
    }
}      End: geometry assignments by demonstration number

        If user signifies incorrect input
else if (ginput1 == 'n')
    {
        ginput1f = VALID;                               Set input flag; valid

        Repeat until valid reentry option has been chosen
for (ginput2f = INVALID; ginput2f != VALID;)
    {
        Prompt user for reentry
        printf("Do you wish to reenter the ");
        if (omode == STANDARD) printf("dimensions ");
        else printf("orientation option ");
        printf("(y or n)? ");
        scanf("%c%c",&ginput2,&garbage);
    }

```

```

                                If user wishes to reenter data
if (ginput2 == 'y')
{
    ginput2f = VALID;                Set input flag; valid
    geomstat = NEEDED;              Geometry is needed
}

                                If user doesn't wish to reenter
else if (ginput2 == 'n')
{
    ginput2f = VALID;                Set input flag; valid
    geomstat = DONE;                Geometry input is done
    mainprog = QUIT;                Set main program exit flag
    printf("\nExiting program...\n");
}

                                If reply (input2) is invalid
else
{
    ginput2f = INVALID;              Set input flag; invalid
    printf("Incorrect entry...");
}

}                                End: invalid reply (input2) loop

}                                End: if incorrect input signified loop

                                If reply (input1) is invalid
else
{
    ginput1f = INVALID;              Set input flag; invalid
    printf("Incorrect entry...");
}

}                                End: invalid reply (input1) loop

}                                End: geometry input continuation loop

}                                End: geometry input procedure

```

---

## ORIENT()

From the geometry entered and calculated, specify and execute appropriate type of orientation algorithm.

First, some procedures that aren't really within orient are included because they are necessary to calculate and notify the user of pre-orientation mis-rotation capabilities. These values and functions were originally generated in Excel, and a sample spread sheet is included in Appendix J.

Procedure for returning the sign of a number

```
double thesign(double x)
{
  if (x < 0) return(-1.0);
  else return(1.0);
}
```

Procedure for calculating the pre-orientation mis-rotation capabilities. For equations and explanation, again, see Appendix J.

```
int misrot(char direction)
{
```

Define variables: Center, Left, Right, Total and Centroid Xbar, Ybar, Area, Xbar\*Area, and Ybar\*Area, and left and right mis-rotation angles.

```
double cxbar, lxbar, rxbar;
double cybar, lybar, rybar;
double carea, larea, rarea, totarea;
double cxbara, lxbara, rxbara, totxbara;
double cybara, lybara, rybara, totybara;
double croidxbar, croidybar;
double rmisrot, lmisrot;
```

Begin calculations

Center, left, right and total areas

```
carea = 9.25*bott;
larea = (1.0/2.0)*9.25*(9.25/tan(langle*DEGTORAD));
rarea = (1.0/2.0)*9.25*(9.25/tan(rangle*DEGTORAD));
totarea = carea + larea + rarea;
```

Center, left and right Xbar locations

```
cxbar = 0;  
lxbar = -((bott/2.0) + (1.0/3.0)*( 9.25/(tan(langle*DEGTORAD)) ));  
rxbar = (bott/2.0) + (1.0/3.0)*( 9.25/(tan(rangle*DEGTORAD)) );
```

Center, left and right Ybar locations

```
cybar = 9.25/2.0;
```

If the rib is virtually symmetric, both right and left Ybar, whether negative or positive, will be at (2

```
if (bott < (topp + 3.5) && bott > (topp - 3.5))
```

```
{  
  lybar = 6.16;  
  rybar = 6.16;  
}
```

Otherwise the combination of these signs determines if Ybar is (1 or 2

```
else
```

```
{  
  lybar = 9.25*( 1.0/2.0 - (1.0/6.0)*thesign(larea)*thesign(bott-topp) );  
  rybar = 9.25*( 1.0/2.0 - (1.0/6.0)*thesign(rarea)*thesign(bott-topp) );  
}
```

Center, left, right and total Xbar\*area

```
cxbara = cxbar*carea;  
lxbara = lxbar*larea;  
rxbara = rxbar*rarea;  
totxbara = cxbara + lxbara + rxbara;
```

Center, left, right and total Ybar\*area

```
cybara = cybar*carea;  
lybara = lybar*larea;  
rybara = rybar*rarea;  
totybara = cybara + lybara + rybara;
```

Centroid Xbar and Ybar are both equal to the sum of all bars\*areas divided by the total area

```
croidxbar = totxbara/totarea;  
croidybar = totybara/totarea;
```

Left and right mis-rotation angles

```
lmisrot = (1.0/DEGTORAD)*atan(( bott/2.0) + croidxbar )/croidybar);  
rmisrot = (1.0/DEGTORAD)*atan(( bott/2.0) - croidxbar )/croidybar);
```

If mis-rotation value is less than 10, the rib is probably symmetric or square ended with a smaller base. This value corresponds to the fact that the rib will just fall over on its own or when lightly tapped. By inspection it becomes clear that the actual mis-rotation capability is greater than 90.

```
if (lmisrot < 10.0) lmisrot = 95.0;  
if (rmisrot < 10.0) rmisrot = 95.0;
```

```
Return appropriate (safe ~ val - 5) mis-rotation value  
if (direction == CLOCKW) return((int) (rmisrot-5.0));  
else if (direction == CTCLOCKW) return((int) (lmisrot-5.0));  
else return(0);
```

```
} End: mis-rotation capability calculations
```

Beginning of actual orientation control loop

```
void orient()
```

```
{  
printf("\nOrientation: \n");
```

If applicable, note first rib attribute

```
if (langle == 90.0 || rangle == 90.0) printf("Square edge\n");
```

If rib is symmetric within the largest possible range of variance, note rib attributes and execute type 1 orientation. Note: this is the only type of rib that will arrive on the manufacturing line.

```
if (topp < (bott+3.5) && topp > (bott-3.5))
```

```
{
```

If the rib is truly symmetric

```
if (bott > (topp - 0.5) && bott < (topp + 0.5))
```

```
{
```

Note rib attributes and mis-rotation capabilities

```
printf("Symmetric rib\n");  
printf("Misrotation capability: 360 degrees\n");
```

```
}
```

Otherwise rib is virtually symmetric

```
else
```

```
{
```

Note rib attributes and mis-rotation capabilities

```
printf("Virtually symmetric rib\n");  
printf("Misrotation capability: \n");  
printf(" %d degrees clockwise\n",misrot(CLOCKW));  
printf(" %d degrees counter-clockwise\n",misrot(CTRCLOCKW));
```

```
}
```

If the orienter is in demonstration mode, the first and simplest of algorithms is to be performed.

```
if (omode == DEMO && demonum == '1')
{
    printf("alias ...Algorithm type #1a\n");
    printf("Press return to orient...");
    scanf("%c",&garbage);
    type1a();
}

    Otherwise note and execute first rib type
else
{
    printf("alias ...Algorithm type #1\n");
    printf("Press return to orient...");
    scanf("%c",&garbage);
    type1();
}
}
```

If rib top dimension is longer than bottom

```
else if (topp > bott + 3.5)
{
    Note rib attributes and mis-rotation capabilities
    if (topp > bott + 11.0) printf("Compound angled edges\n");
    printf("Top dimension greater than bottom dimension\n");
    printf("Misrotation capability: ");
```

If the fourth demonstration has been chosen, note misrotation capabilities and execute the original algorithm for this type of rib

```
if (omode == DEMO && demonum == '4')
{
    printf("\nFor complete contact knowledge:");
    printf("\n %d degrees clockwise\n",misrot(CLOCKW));
    printf(" %d degrees counter-clockwise\n",misrot(CTRCLOCKW));
    printf("For this demonstration:");
    printf("\n 90 degrees clockwise");
    printf("\n 90 degrees counter-clockwise");
    printf("\nalias ...Algorithm type #2a");
    printf("\nPress return to orient...");
    scanf("%c",&garbage);
    type2a();
}
```

```

        Otherwise execute the final algorithm, capable of handling 360 deg. of mis-rot.
else
    {
    printf("360 degrees\n");
    printf("alias ...Algorithm type #2\n");
    printf("Press return to orient...");
    scanf("%c",&garbage);
    type2();
    }
}

        If rib has a greater bottom dim. than top dimension, execute type3 orientation.
else if (topp < bott-3.5)
    {
    if (topp < bott - 11)
        printf("Compound angled edges\n");
    printf("Top dimension less than bottom dimension\n");
    printf("Misrotation capability: 360 degrees\n");
    printf("alias ...Algorithm type #3\n");
    printf("Press return to orient...");
    scanf("%c",&garbage);
    type3();
    }
}

        End: orientation procedure

```

---

## WRAPUP()

Verify success of orientation process, provide necessary consequent options: continue to next orientation, release surrounded or jammed ribs, override motor positions, or quit orientation program.

```
void wrapup()
{
    Define Variables

    char wrapstat;           Wrapup status flag
    char wopt;              Wrapup option number

    Begin User Interface

    printf("\nWrapup: \n");

        Begin wrapup loop
    for (wrapstat = BEGIN; wrapstat != DONE;)
    {
        If the orienter is in standard mode
    if (omode == STANDARD)
    {
        Get successful orient. wrapup option choice
        printf("Choose one of the following options:\n");
        printf(" 1.) Open arms to release rib\n");
        printf(" 2.) Rib is clear of orienter; prepare for next rib\n");
        printf(" 3.) Quit orientation program\n");
        printf("Your choice: ");
    }

        Otherwise ask if the demo mode should cont.
    else
        printf("Continue orientation demonstration (y or n)? ");

        Get the corresponding reply.
    scanf("%c%c",&wopt,&garbage);

        If an invalid reply was given
    if (wopt != '1' && wopt != '2' && wopt != '3' && wopt != 'y' && wopt != 'n')
        printf("Incorrect entry...");
    }
}
```



```

                If the orienter is in demonstration mode or the user has chosen to open the arms
if (wopt == '1' || omode == DEMO)
{
    printf("\nOpening arms...");
    release();           Open arms to release rib
    printf("\nPlease remove rib from orienter\n");
    if (omode == STANDARD) printf("\n");
}

                If the orienter is in demo mode or the user designates, prepare for next rib
if (wopt == '2' || omode == DEMO)
    wrapstat = DONE;

                Option: Exit orientation program
if (wopt == '3' || wopt == 'n')
{
    printf("\nExiting program...\n");
    wrapstat = DONE;           Set option repeat flag; done
    mainprog = QUIT;          Set main program exit flag
}

}           End: Orientation wrapup loop
}           End: Orientation wrapup procedure

```

---

## LASTCHK()

Check orienter status and shut down the system accordingly: Move motors to closed positions or exit the orientation program with an apology.

```
void lastchk()
```

```
{
```

    Define Variables

```
    char ribchk;
```

    Rib-in-orienter check variable

```
    char lcstep;
```

    Last check step number

```
    char lcstepl;
```

    Last check step present loop var

```
    char shutdwn;
```

    Shutdown option variable

```
    char shutdwnf;
```

    Shtdwn option valid input flag

    Begin User Interface

```
    printf("\nShutdown:\n");
```

        Repeat until correct input is entered

```
    for (ribchk = 'y'; ribchk != 'n';)
```

```
    {
```

        Check presence of a rib

```
        printf("Is there a rib in the orienter (y or n)? ");
```

```
        scanf("%c%c",&ribchk,&garbage);
```

        If incorrect input is given

```
        if (ribchk != 'y' && ribchk != 'n')
```

```
            printf("Incorrect entry...");
```

        If there is a rib in the orienter

```
        if (ribchk == 'y')
```

```
        {
```

```
            if (omode == STANDARD) printf("\n");
```

```

Repeat while there is a rib in the orienter
for (lcstepl = BEGIN; lcstepl != QUIT;)
{
  if (omode == STANDARD)
  {
    Present possible options
    printf("Choose from the following:\n");
    printf(" 1.) Open arms to release rib \n");
    printf(" 2.) Rib has been removed \n");
    printf(" 3.) Rib can't be removed \n");
    printf("Your choice: ");
    scanf("%c%c",&lcstep,&garbage);

    Option: Incorrect entry
    if (lcstep != '1' && lcstep != '2' && lcstep != '3')
      printf("Incorrect entry...");
  }

  Option: Open arms to release rib
  if (lcstep == '1' || omode == DEMO)
  {
    printf("\nOpening arms...");
    release();
    printf("\nPlease remove rib from orienter\n\n");
  }

  Options: Rib has been or cannot be removed
  if (lcstep == '2' || lcstep == '3' || omode == DEMO)
  {
    ribchk = 'n';
    lcstepl = QUIT;
  }
}
End: Presentation of rib in orienter options
}
End: Rib in orienter loop

If there isn't a rib in the orienter
if (ribchk == 'n')
{
  if (omode == STANDARD) printf("\n");
}

```

```

Repeat until valid input is entered
for (shutdwnf = INVALID; shutdwnf != VALID;)
{
    Check shutdown status
    printf("May a standard shutdown be performed at this point? ");
    scanf("%c%c",&shutdwn,&garbage);

    If a standard shutdown may be performed
    if (shutdwn == 'y')
    {
        shutdwnf = VALID;                Set input flag; valid
        printf("\nPerforming standard shutdown...\n");
        shtdwnmt();                      Move motors to closed pos.
        Provide system notes
        printf("\nSystem is shut down. \n");
        printf("If you intend to quit using the orienter, please ");
        printf("turn off the power:\n System (red power tree ");
        printf("switch) first, then the computer. Thanks.\n");
    }

    If a standard shutdown may not be performed
    else if (shutdwn == 'n')
    {
        shutdwnf = VALID;                Set input flag; valid
        if (omode == DEMO)
        {
            printf("See guide sheet for recommended procedures.");
        }
    }

    If input is incorrect
    else
    {
        shutdwnf = INVALID;              Set input flag; invalid
        printf("Incorrect entry...");
    }

} End: Shutdown status input loop
} End: Rib not in the orienter loop
} End: Rib presence input loop

} End: System check and shutdown procedure

```

---

## BCMDIF.H

Base Command Include File

by

Susie Ward

April 1994

This include file contains all my base level, functional procedures, (alias "stuff") which calls "their" (Omnitech Robotics) MC1000 and MC3000 variables and functions ("stuff").

Note: all /\* \*/ have been removed to facilitate file conversion and organization, but the notes have been changed to a different font to help separate them from the code.

---

\*\*\*\* Constant and variable definitions for my stuff \*\*\*\*

Constants: Flag characters for making the program more readable: standard and demonstration mode flags, loop continuation and quit flags, input needed, valid and invalid flags, rotation direction flags.

```
#define STANDARD 's'  
#define DEMO 'd'
```

```
#define BEGIN 'b'  
#define CONTINUE 'c'  
#define QUIT 'q'  
#define DONE 'd'
```

```
#define NEEDED 'n'  
#define VALID 'v'  
#define INVALID 'i'
```

```
#define CLOCKW 'c'  
#define CTCLOCKW 'r'
```

Constants: numerical values for pi, inch to quadraturecount conversion for linear actuation motors, degree to quadrature count conversion for rotary actuation motors degree to radian conversion.

```
#define PI 3.1415927  
#define INTOQD 2000.0  
#define DEGTOQD 44.7  
#define DEGTORAD 0.0174533
```

Motor variable arrays: These arrays were set up such that orientation procedures could generically define motors in relation to rib geometries. Then all motor values could also be stored and accessed or changed at one location according to simple array positions instead of irregular numerical values like 768, 993 etc, and the velocities of the motors could be changed through the timer values such that the necessary proportions could be maintained.

|   |                    |
|---|--------------------|
| <code>int bases[] = {768,992,993,994};</code> | Base addresses     |
| <code>int gains[] = {50,50,50,50};</code>     | Gains              |
| <code>int timers[] = {65,65,65,65};</code>    | Timers             |
| <code>int mvelos[] = {5,1,5,1};</code>        | Maximum velocities |
| <code>int accels[] = {2,1,2,1};</code>        | Accelerations      |

#### Generic globally accessed variables

|   |                                 |
|---|---------------------------------|
| <code>double bott, topp, langle, rangle;</code> | Rib dimensions                  |
| <code>char mainprog = BEGIN;</code>             | Main program flag               |
| <code>char omode;</code>                        | Orientation mode                |
| <code>char demonum;</code>                      | Demonstration number flag       |
| <br>  |                                 |
| <code>char garbage;</code>                      | Char for carriage return inputs |

#### \*\*\*\* Function and Procedure definitions for my stuff \*\*\*\*

Those described in this file

```
void initial(), setpzgtva(), gotopos(), waitcalc(), startmt(), tostart(), shtdwnmt(), release();
```

Those described and employed in cloopif.h

```
void startup(), calib(), get_geom(), orient(), wrapup(), lastchk();
double thesign(double);
int misrot(char);
```

#### \*\*\*\* Constant and variable definitions for their stuff \*\*\*\*

```
long int retval,hi,med,low;
long int ival1,pos;
int base;
```

#### \*\*\*\* Function and Procedure Definitions for their stuff \*\*\*\*

```
int regin(),get_gain(),get_pole(),get_zero(),get_accel();
void reset(),init(),trap_mode(),prop_mode(),int_mode(),pos_mode(),sel_mode();
```

```

void regout(),set_cmd_pos(),clr_act_pos(),set_final_pos();
void set_base(),set_pole(),set_zero(),set_gain(),set_timer(),set_accel();
void set_max_vel(),set_prop_vel(),set_int_vel(),set_status(),set_bipolar();
void set_unipolar(),open_loop_comm(),closed_loop_comm(),set_do(),set_x();
void set_y(),set_base(),set_offset(),set_max_adv(),set_vel_timer(),set_dac();
void set_pwm(),clr_emerg_flags(),set_sign_rev(),num_phases(),comm_count();
long get_act_pos(),get_cmd_pos(),get_final_pos();
int get_max_vel(),get_int_vel(),get_prop_vel(),get_act_vel(),get_status();
int get_di(),get_ring(),get_x(),get_y(),get_base(),get_offset();
int get_max_adv(),get_dac(),get_pwm();

```

---

My stuff...

Command series for initializing motors

```

void initial(int b)                                b from 0 to 3 for base addr 678, 992, 993, 994
{
  set_base(bases[b]);                             Set motor base address
  init();                                          Enter initiation mode
  clr_act_pos();                                  Clear motor position to 0
  set_cmd_pos(0);                                 Set motor command position
  pos_mode();                                     Enter position mode; hold until next command
}

```

End: Initialization procedure

Command series for setting poles, zeroes, gains, timers, velocities, acceleration.

```

void setpzgtva(int b)                              b from 0 to 3 for base addr 678, 992, 993, 994
{
  set_base(bases[b]);                             Set motor base address
  set_pole(0);                                    Set pole...
  set_zero(240);                                  zero...
  set_gain(gains[b]);                             gain...
  set_timer(timers[b]);                           timer...
  set_max_vel(mvelos[b]);                         maximum velocity...
  set_accel(accels[b]);                            acceleration...
}

```

to values designated in earlier variable definitions

End: Variable setting procedure

Command series for motion to designated position using a trapezoidal velocity profile.  
(All constants previously set during setpzgtva()).

```
void gotopos(int finpos)           Finpos defines final position
{
    sel_mode();                   Enter command selection mode
    set_final_pos(finpos);        Set final position
    trap_mode();                  Move to final pos in trapezoidal velocity
mode
}                                  End: Trapezoidal motion command procedure
```

Command series for answering the question, "Is the motor within a specified range of a certain position?"

```
char mposchk(int mbase, int mpos, int offrange)
{
    long poschk;                  mbase: motor base addr, mpos: check
    char inpos = 'n';             position, offrange: position error range

    set_base(mbase);              Position check variable
    poschk = get_act_pos();        Motor "in position" flag, default is "no."

    If motor is within the desired range of the designated motor position

    if (poschk < (mpos+offrange) && poschk > (mpos-offrange))
        inpos = 'y';              Set "in position" flag to "yes"

    return(inpos);                Return position status
}                                  End: Motor position check procedure
```

Command series for pausing between motion commands

```
void waitcalc(int wcbase, int wcpos)  wcbase: wait calculation motor base,
{                                     wcpos: motor pos at which the wait ends.
    While motor is not within range of the desired position, keep checking
    while (mposchk(wcbase, wcpos, 70) == 'n')
        mposchk(wcbase, wcpos, 70);
}                                  End: Wait generation procedure
```



"Startup meat" or command series for initializing motors and setting motor constants.

```
void startmt()
{
  int x;                                base counter variable

  printf("\nInitializing motors...");

  For each motor base address, 0 to 3 for addresses 768, 992, 993, and 994
  for (x = 0; x < 4; x++)
  {
    initial(x);                          Go through initialization procedure
  }

  printf("\nSetting motor constants...\n");

  For each motor base address, 0 to 3 for addresses 768, 992, 993, and 994
  for (x = 0; x < 4; x++)
  {
    setpzgtva(x);                         Go though constant setting procedure
  }

}                                          End: Motor reset procedure
```

Command series for moving motors to start positions

```
void tostart()
{
  set_base(993);                          Block base...
  gotopos(-32000);                         go to open position: -32000
  set_base(992);                          Right arm...
  gotopos(2800);                           go to open position: 2800
  set_base(994);                          Left arm...
  gotopos(2800);                           go to open position: 2800
  waitcalc(993,-32000);                   Wait until all motors are in
  waitcalc(994,2800);                     their open positions.

}                                          End: Startup position command procedure
```

Command series for opening arms to release a rib

```
void release()
{
  set_base(994);           Set base address; left arm
  gotopos(2800);          go to open position: 2800
  set_base(992);           Set base address; right arm
  gotopos(2800);          go to open position: 2800
  waitcalc(992,2800);      Wait until arms are open
}
                          End: Arm open position command procedure
```

"Shutdown meat" or, Command series for moving motors to closed positions.

```
void shtdwnmt()
{
  int x;                  Motor base counter variable
  set_base(994);          Initially, move the arms away
  gotopos(2800);          from each other and into
  set_base(992);          open positions to avoid
  gotopos(2800);          possible interference.
  set_base(993);
  gotopos(-10000);
  waitcalc(992,2800);      Wait until arms and block
  waitcalc(993,-10000);   have cleared each other
  delay(300);
}

                          For each motor or base address, from 0 to 3 for 768, 992, 993 and 994
for (x = 0; x < 4; x++)
{
  set_base(bases[x]);      Set base address
  set_timer(80);           Slow motors (timer: no chg in spd ratios)
  gotopos(0);             Move motor to closed posit.
}
waitcalc(993,0);          Wait until all motors are in
waitcalc(994,0);          their closed positions

                          For each motor or base address
for (x = 0; x < 4; x++)
{
  set_base(bases[x]);      Set base address
  set_timer(timers[x]);    Reset timer to previous val.
}
}
                          End: Shutdown procedure
```

Their stuff...

(Which came without any in-code documentation. See user manuals for command and register manipulation info.)

```
void reset()
{
    regout(5,0);
}
```

```
void init()
{
    regout(5,1);
}
```

```
void trap_mode()
{
    regout(0,8);
}
```

```
void prop_mode()
{
    regout(0,11);
}
```

```
void int_mode()
{
    regout(0,13);
}
```

```
void pos_mode()
{
    regout(0,0);
    regout(0,3);
    regout(0,5);
}
```

```
void sel_mode()
{
    regout(5,3);
}
```

```

void set_cmd_pos(ival1)
{
    low = (int)ival1 & 0x000000FF;
    med = ((int)ival1 >> 8) & 0x000000FF;
    hi = ((int)ival1 >> 16) & 0x000000FF;
    regout(12,hi);
    regout(13,med);
    regout(14,low);
}

```

```

void clr_act_pos()
{
    regout(19,0);
}

```

```

void set_final_pos(ival1)
long ival1;
{
    low = (int)ival1 & 0x000000FF;
    med = ((int)ival1 >> 8) & 0x000000FF;
    hi = ((int)ival1 >> 16) & 0x000000FF;
    regout(41,low);
    regout(42,med);
    regout(43,hi);
}

```

```

long get_cmd_pos()
{
    hi=regin(12);
    med=regin(13);
    low=regin(14);
    retval=hi*65536 + med*256 +low;
    return(retval);
}

```

```

long get_act_pos()
{
    low=regin(20);
    med=regin(19);
    hi=regin(18);
    retval=hi*65536 + med*256 +low;
    if (retval>8388607)
        retval -= 16777215;
    return(retval);
}

```

```
long get_final_pos()
{
    hi=regin(43);
    med=regin(42);
    low=regin(41);
    retval=hi*65536 + med*256 +low;
    return(retval);
}
```

```
void set_gain(ival1)
{
    regout(34,(int)ival1);
}
```

```
int get_gain()
{
    retval=regin(34);
    return(retval);
}
```

```
void set_pole(ival1)
{
    regout(33,ival1);
}
```

```
int get_pole()
{
    retval=regin(33);
    return(retval);
}
```

```
void set_zero(ival1)
{
    regout(32,(int)ival1);
}
```

```
int get_zero()
{
    retval=regin(32);
    return(retval);
}
```

```

void set_accel(ival1)
long ival1;
{
    regout(38,(int)ival1 & 0x000000FF);
    regout(39,(((int)ival1 & 0x0000FF00) >> 8));
}

int get_accel()
{
    low=regin(38);
    hi=regin(39);
    retval=hi*256 + low;
    return(retval);
}

void set_timer(ival1)
{
    regout(15,(int)ival1);
}

void set_max_vel(ival1)
{
    regout(40,(int)ival1);
}

int get_max_vel()
{
    retval=regin(40);
    return(retval);
}

void set_prop_vel(ival1)
long ival1;
{
    regout(35,(int)ival1 & 0x000000FF);
    regout(36,(((int)ival1 & 0x0000FF00) >> 8) );
}

int get_prop_vel()
{
    low=regin(35);
    hi=regin(36);
    retval=hi*256 + low;
    return(retval);
}

```

```

void set_int_vel(ival1)
{
    regout(60,(int)ival1);
}

int get_int_vel()
{
    retval=regin(60);
    return(retval);
}

int get_act_vel()
{
    hi=regin(53);
    low=regin(52);
    retval=hi*256 +low;
    return(retval);
}

void quit()
{
    exit(0);
}

void set_status(ival1)
{
    regout(7,(int)ival1);
}

int get_status()
{
    retval=regin(7);
    return(retval);
}

void set_bipolar()
{
    regout(0,2);
}

void set_unipolar()
{
    regout(0,10);
}

```

```

void open_loop_comm()
{
    regout(0,12);
}

void closed_loop_comm()
{
    regout(0,4);
}

void set_do(ival1)
{
    outp(base+3,(int)ival1);
}

int get_di()
{
    retval=(inp(base+4) & 0x0F);
    return(retval);
}

void set_ring(ival1)
{
    regout(24,(int)ival1);
}

int get_ring()
{
    retval=regin(24);
    return(retval);
}

void set_x(ival1)
{
    regout(26,(int)ival1);
}

int get_x()
{
    retval=regin(26);
    return(retval);
}

```



```
void set_y(ival1)
{
    regout(27,(int)ival1);
}
```

```
int get_y()
{
    retval=regin(27);
    return(retval);
}
```

```
void set_base(ival1)
{
    base=(int)ival1;
}
```

```
int get_base()
{
    retval=base;
    return(retval);
}
```

```
void set_offset(ival1)
{
    regout(28,(int)ival1);
}
```

```
int get_offset()
{
    retval=regin(28);
    return(retval);
}
```

```
void set_max_adv(ival1)
{
    regout(31,(int)ival1);
}
```

```
int get_max_adv()
{
    retval=regin(31);
    return(retval);
}
```

```

void set_vel_timer(ival1)
{
    regout(25,(int)ival1);
}

void set_dac(ival1)
{
    regout(8,(int)ival1);
}

int get_dac()
{
    retval=regin(8);
    return(retval);
}

void set_pwm(ival1)
{
    regout(9,(int)ival1);
}

int get_pwm()
{
    retval=regin(9);
    return(retval);
}

void clr_emerg_flags()
{
    regout(7,regin(7));
}

void set_sign_rev(ival1)
{
    if ((int)ival1 == 0) {
        regout(7,regin(7) & 0xfe);
    }
    else if ((int)ival1 == 1) {
        regout(7,regin(7) | 0x01);
    }
}

```

```

void num_phases(ival1)
{
  if ((int)ival1 == 3) {
    regout(7,regin(7) & 0xfd);
  }
  else if ((int)ival1 == 4) {
    regout(7,regin(7) | 0x02);
  }
}

```

```

void comm_count(ival1)
{
  if ((int)ival1 == 0) {
    regout(7,regin(7) & 0xfb); 0 for quadrature counts
  }
  else if ((int)ival1 == 1) {
    regout(7,regin(7) | 0x04); 1 for encoder counts
  }
}

```

## REGIN

```

int regin(reg)
int reg;
{
  delay(1);
  return(inp(base+reg*1024));
}

```

## REGOUT

```

void regout(reg,val)
int reg,val;
{
  delay(1);
  outp(base+reg*1024,val);
}

```



## OALGIF.C

Orientation Algorithm Include File

by  
Susie Ward  
April 1994

This file includes algorithms for all orientation types designated in the orient() procedure.

Note: All /\* \*/ have been removed for the purposes of organized file conversion and presentation, but all intermeshed comments have been converted to a smaller font such that they are more easily recognized as separate from the main code.

---

### Procedure common variable definitions

|                               |   |
|-------------------------------|---|
| int ang1qd, ang2qd;           | final arm angle position in quad counts   |
| int base1, arm1, base2, arm2; | motor const array designation variable    |
| long firstpos, lastpos;       | first and last base positions             |
| int intfpos;                  | interference base position                |
| long clearpos, tippos;        | rib clearance and tipping base positions  |
| double alar, arar;            | adjusted left and right angles in radians |
| double lowang;                | lowest arm angle in quad counts           |
| double ribl;                  | longest length across rib                 |
| float d = 4.25;               | 0 position dist between motor centers     |
| float ofar = 1.25;            | offset, motor center/arm rail             |
| float ofag = 2.25;            | offset, motor center/arm guard            |
| float h = 3.25;               | height, motor center/platform rail        |
| float lmcgt = 10.375;         | length, motor center/arm guard top        |
| float lmcgic = 2.375;         | length, mtr center/arm guard inner corner |

### Procedure definitions

|                   |   |
|-------------------|---|
| void bangasmt();  | base, angle variable assgnmt                |
| void type1a();    | orientation steps: first square rib         |
| void type1();     | orientation steps: nearly symmetric ribs    |
| void stdtype();   | standard orientation steps for types 2, 3   |
| void type2a();    | orientation steps: 1st rib with bott < topp |
| void type2();     | orientation steps: ribs with bott < topp    |
| void type3();     | orientation steps: ribs with bott > topp    |
| void tolastpos(); | orientation steps: motors to last pos       |
| void oerrmsg();   | orientation error messages                  |

## BANGASMT()

Universal base and arm position and angle assignment.

```
void bangasmt()
{
```

Variable definitions

```
double redge = 9.25/sin(rangle*DEGTORAD);           Right rib edge length
double ledge = 9.25/sin(langle*DEGTORAD);           Left rib edge length
double longest;                                       Longest bott or top edge length
```

Angle calculations

```
alar = (90 - langle)*DEGTORAD;                       adjusted left angle in radians
arar = (90 - rangle)*DEGTORAD;                       adjusted right angle in radians
lowang = 2800*(1.0/DEGTOQD)*DEGTORAD;                lowest arm angle: qd to rad
```

Address and final angle pos assignments and rib length calc, as determined by end angles.

```
if (langle < rangle)
{
  base1 = 0;
  arm1 = 3;
  base2 = 2;
  arm2 = 1;
  ang1qd = (90 - langle)*DEGTOQD;
  ang2qd = (90 - rangle)*DEGTOQD;
  ribl = sqrt(bott*bott + ledge*ledge
              - 2.0*bott*ledge*cos((180.0-langle)*DEGTORAD));
}
else
{
  base1 = 2;
  arm1 = 1;
  base2 = 0;
  arm2 = 3;
  ang1qd = (90 - rangle)*DEGTOQD;
  ang2qd = (90 - langle)*DEGTOQD;
  ribl = sqrt(bott*bott + redge*redge
              - 2.0*bott*redge*cos((180.0-rangle)*DEGTORAD));
}
if (ribl < bott) ribl = bott;
if (ribl < topp) ribl = topp;
```

Block position calculations: last, arm interference and rib clearing and tipping positions.

```
if (bott > topp) longest = bott;  
else longest = topp;
```

```
lastpos = -INTOQD*(bott - h*(tan(alar)+tan(arar))  
            + ofar*(1.0/cos(alar)+1.0/cos(arar)) - d);  
intfpos = -INTOQD*(0 - lmcgic*(sin(alar)+sin(arar))  
            + ofag*(cos(alar)+1.0/cos(arar)) - d);  
clearpos = -INTOQD*((ribl+1.0) - h*(tan(lowang)+tan(0))  
            + ofar*(1.0/cos(lowang)+1.0/cos(0)) - d);  
tippos = -INTOQD*((longest+1.0) - h*(tan(0)+tan(0))  
            + ofar*(1.0/cos(0)+1.0/cos(0)) - d);
```

Take out comment marks to verify calculation

```
printf("\nrangle = %5.2lf, langle = %5.2lf",rangle,langle);  
printf("\nledge = %5.2lf, redge = %5.2lf",ledge,redge);  
printf("\nribl = %5.2lf, clearpos = %d",ribl,clearpos);
```

```
} End: Universal variable assignment procedure
```

---

### TOLASTPOS()

Simple procedure to send arms and blocks to their final positions.

```
void tolastpos()  
{  
  set_base(993);  
  gotopos(lastpos);  
  waitcalc(993,lastpos);  
  set_base(bases[arm1]);  
  gotopos(ang1qd);  
  waitcalc(bases[arm1],ang1qd);  
  set_base(bases[arm2]);  
  gotopos(ang2qd);  
  waitcalc(bases[arm2],ang2qd);  
}
```

---

## TYPE1A()

The very first orientation algorithm generated, for small square ended ribs, as executed for first demonstration rib type. This algorithm was later modified the series of steps employed in type1(), which may also be used to orient this rib.

```
void type1a()
{
    Perform standard base and angle assignments
    bangasmt();

    set_base(992);           Move right arm...
    gotopos(0);             to the upright position
    waitcalc(992,0);        Wait until it arrives

    tolastpos();           Move all motors to last pos.
}                          End: Type 1A orientation
```

---



## TYPE10

Final orientation algorithm for all symmetric and nearly symmetric ribs.

```
void type10
{
    Perform standard base and angle assignments
    bangasmt();

    Calculate first block position
    firstpos = lastpos + 1000;

    Algorithm...

    If all block positions are within orienter range and don't cause interference
    if (lastpos < intfpos && lastpos > -36000 && clearpos > -36000 &&
        clearpos < 2000 && tippos > -36000 && tippos < 2800)
    {
        set_base(bases[arm2]);           Motion 1, described in section 4.5.2
        gotopos(-1800);
        waitcalc(bases[arm2],-1800);
        set_base(993);
        gotopos(firstpos);
        waitcalc(993,firstpos);

        set_base(bases[arm2]);           Motion 2 series
        gotopos(0);
        waitcalc(bases[arm2],0);
        delay(500);

        tolastpos();
    }

    else oerrmsg();
}
```

---

## TYPE2A()

The first orientation algorithm generated for ribs with a shorter bottom dimension, as executed in demonstration number two. This series of steps works if the rib has been misrotated by no more than 90 degrees in either direction. In order to accomodate 360 degrees of rotation, type2() was generated.

```
void type2a()
{
    Perform standard base and angle assignments
    bangasmt();

    Calculate first motor base position
    firstpos = -INTOQD*((rib1-3.0) - h*(tan(lowang)+tan(0))
                + ofar*(1.0/cos(lowang)+1/cos(0)) - d);

    If all designated motor positions are within orienter range and don't cause interference
    if (lastpos < intfpos && lastpos > -36000 && clearpos > -36000 &&
        clearpos < 2000 && tippos > -36000 && tippos < 2800)
    {
        set_base(bases[arm2]);           Motion 1, described in Appendix K.
        gotopos(0);
        waitcalc(bases[arm2],0);
        set_base(993);
        gotopos(firstpos);
        waitcalc(993,firstpos);

        set_base(bases[arm2]);
        gotopos(2800);
        set_base(bases[arm1]);           Motion 2 series.
        gotopos(-500);
        waitcalc(bases[arm1],-500);
        waitcalc(bases[arm2],2800);

        tolastpos();                     Move to last pos.
    }

    else oerrmsg();
}
    End: Type 2a rib orientation algorithm
```

---

## TYPE2()

Final orientation algorithm for ribs with a bottom dimension smaller than that of the top and misrotated up to 360 degrees.

```
void type2()
{
    Perform standard base and angle assignments
    bangasmt();

    If all designated motor positions are within orienter range and don't cause interference
    if (lastpos < intfpos && lastpos > -36000 && clearpos > -36000 &&
        clearpos < 2000 && tippos > -36000 && tippos < 2800)
    {
        stdtype();                Standard 360 degree mis-rotation steps
        stdtype();

        clearpos = clearpos + 2000;    Modify clear position to keep arm under rib

        set_base(bases[arm1]);        Perform std, slightly modified rotation step series
        gotopos(2800);                in reverse: move rib from longest to angled edge
        set_base(bases[arm2]);
        gotopos(0);
        waitcalc(bases[arm1],2800);
        waitcalc(bases[arm2],0);
        set_base(993);
        gotopos(2100);
        waitcalc(993,2100);

        set_base(bases[arm1]);
        set_timer(100);
        gotopos(-1000);
        set_timer(timers[arm1]);
        set_base(bases[arm2]);
        set_timer(100);
        gotopos(2300);
        set_timer(timers[arm2]);
        waitcalc(bases[arm1],-1000);
        waitcalc(bases[arm2],2300);
        delay(500);
    }
}
```

```

set_base(bases[arm1]);
gotopos(2800);
set_base(bases[arm2]);
gotopos(2800);
waitcalc(bases[arm1],2800);
delay(500);

if (bases[base2] == 768)
{
    set_base(993);
    gotopos(clearpos);
    waitcalc(993,clearpos/3);
    set_base(994);
    set_timer(100);
    gotopos(2400);
    waitcalc(994,2400);
    delay(500);
    gotopos(2800);
    waitcalc(994,2800);
    delay(500);
    gotopos(1200);
    waitcalc(994,1200);
    delay(500);
    gotopos(2800);
    waitcalc(994,2800);
    delay(500);
    gotopos(-1400);
    waitcalc(994,-1400);
    set_timer(timers[3]);
    delay(500);
    gotopos(0);
    waitcalc(994,0);
    delay(500);
}
else
{
    set_base(993);
    gotopos(clearpos);
    waitcalc(993,clearpos);
    delay(500);
}

```

```

set_base(bases[arm2]);
gotopos(0);
waitcalc(bases[arm2],0);
tolastpos();
}

else oerrmsg();

}

```

Move appropriate arm to avoid slip during motor rotation to final positions

---

### TYPE3()

Orientation steps for ribs with a bottomdimension greater than that of the top.

```

void type3()
{
    Perform standard base and angle assignments
    bangasmt();

    If all designated motor positions are within orienter range and don't cause interference
    if (lastpos < intfpos && lastpos > -36000 && clearpos > -36000 &&
        clearpos < 2000 && tippos > -36000 && tippos < 2800)
    {
        stdtype();
        stdtype();
        tolastpos();
    }

    else oerrmsg();

}

```

Perform std steps for square edge ribs twice to insure rib orientation  
Move motors to last positions

## STDTYPE()

Standard initial orientation steps for ribs with one square edge

```
void stdtype()
{
    If all designated motor positions are within orienter range and don't cause interference

    if (lastpos < intfpos && lastpos > -36000 && clearpos > -36000 &&
        clearpos < 2000 && tippos > -36000 && tippos < 2800)
    {
        set_base(bases[arm2]);           Motion 1 series as described in Section 4.5.2
        gotopos(2800);
        set_base(bases[arm1]);
        gotopos(0);
        waitcalc(bases[arm2],2800);
        waitcalc(bases[arm1],0);
        set_base(993);
        gotopos(2100);
        waitcalc(993,2100);

        set_base(bases[arm2]);
        set_timer(70);
        gotopos(-1000);
        set_timer(timers[arm2]);
        set_base(bases[arm1]);
        set_timer(70);
        gotopos(2300);
        set_timer(timers[arm1]);
        waitcalc(bases[arm2],-1000);
        waitcalc(bases[arm1],2300);
        delay(500);

        set_base(bases[arm2]);           Motion 2 series
        gotopos(2800);
        set_base(bases[arm1]);
        gotopos(2800);
        waitcalc(bases[arm1],2800);
        waitcalc(bases[arm2],2800);
        delay(500);
    }
}
```

```

if (bases[base1] == 768)
{
  set_base(993);
  gotopos(clearpos);
  waitcalc(993,clearpos/3);

```

Motion 3a

```

  set_base(994);
  set_timer(100);
  gotopos(2400);
  waitcalc(994,2400);
  delay(500);
  gotopos(2800);
  waitcalc(994,2800);
  delay(500);
  gotopos(1200);
  waitcalc(994,1200);
  delay(500);
  gotopos(2800);
  waitcalc(994,2800);
  delay(500);
  gotopos(-1400);
  waitcalc(994,-1400);
  delay(500);
  set_timer(timers[3]);
  gotopos(2800);
  delay(500);

```

Motion 4a series

```

if (abs(langle-rangle) > 19.0)
{
  set_base(992);
  gotopos(-700);
  waitcalc(992,-700);
  set_base(993);
  gotopos(tippos);
  waitcalc(993,tippos);
  delay(500);
  gotopos(clearpos);
  set_base(992);
  gotopos(2800);
  waitcalc(992,2800);
  delay(500);
}
}

```

Motion 5a series...

doesn't need to be performed  
for compound angled ribs





## OERRMSG()

Error messages corresponding to block pos problematic during orientation.

```
void oerrmsg()
{
    Notify of orientation problem
    printf("This rib cannot be oriented:");

    For each condition, make appropriate note
    if (lastpos >= intfpos)
        printf("\nRib geometry requires arm interference for orientation");
    if (lastpos < -36000)
        printf("\nThe final block position is out of range (too far) for this orienter");
    if (clearpos < -36000)
        printf("\nRib clearance position is out of range (too far) for this orienter");
    if (clearpos > 2000)
        printf("\nRib clearance position is out of range (too close) for this orienter");
    if (tippos < -36000)
        printf("\nRib tipping position is out of range (too far) for this orienter");
    if (tippos > 2800)
        printf("\nRib tipping position is out of range (too close) for this orienter");
    printf("\n");
}
```



## Appendix H: RIB ANGLE ASSIGNMENT

The format for prompting the user for the bottom edge dimension was straightforward, but it was more difficult to present a simple and easily understood method for designating how to measure and input rib end angles. The final method, as illustrated in figure H1 is described as follows: from the corresponding outside horizontals, enter the left angle (positive in the clockwise direction) and the right angle (positive in the counter-clockwise direction.)

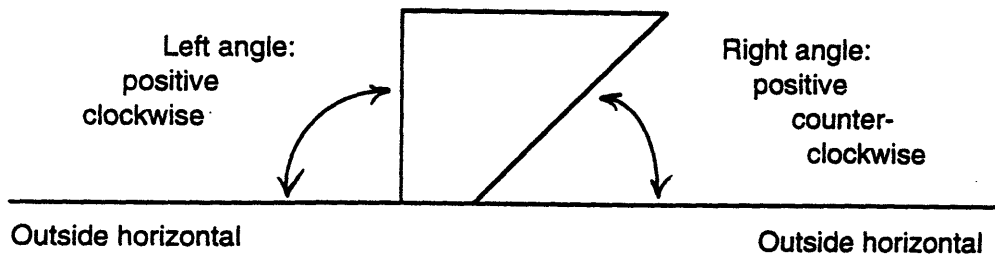


Figure H1: Illustration of Rib Angle Assignments



## Appendix I: CALCULATION OF REQUIRED ORIENTATION BLOCK POSITIONS

### GENERIC BLOCK POSITION:

The required separation between motor centers may be determined in two ways:

$$\begin{aligned}
 & \text{Initial distance between motors (in inches)} \\
 & \quad + \\
 & \text{Commanded motor position (negative and in quadrature counts)} \\
 & \quad \text{multiplied by -1 and the quadrature count/inch conversion factor} \\
 & \text{[separation = } d + \text{pos}(-1/\text{INTOQD})] \qquad \qquad \qquad \text{Eqn. 1}
 \end{aligned}$$

OR

$$\begin{aligned}
 & \text{Required space} \\
 & \quad + \\
 & \text{Linear displacement created by the varying arm angles and the height from} \\
 & \quad \text{the motor centers to the orienting platform rail.} \\
 & \quad - \\
 & \text{Linear displacement created by the varying arm angles and the offset from} \\
 & \quad \text{the line of the motor centers to the arm rails.} \\
 & \text{[separation = space + } x_h - x_f \text{]} \qquad \qquad \qquad \text{Eqn. 2}
 \end{aligned}$$

Combining equations 1 and 2, we get

$$d + \text{pos}(-1/\text{INTOQD}) = \text{space} + x_h - x_f \qquad \qquad \text{Eqn. 3}$$

And the form of the equation for the required commanded block position becomes

$$\text{pos} = -\text{INTOQD} * (\text{space} - d + x_h - x_f) \qquad \qquad \text{Eqn. 4}$$

Then from figure I1, the appropriate values may be substituted:

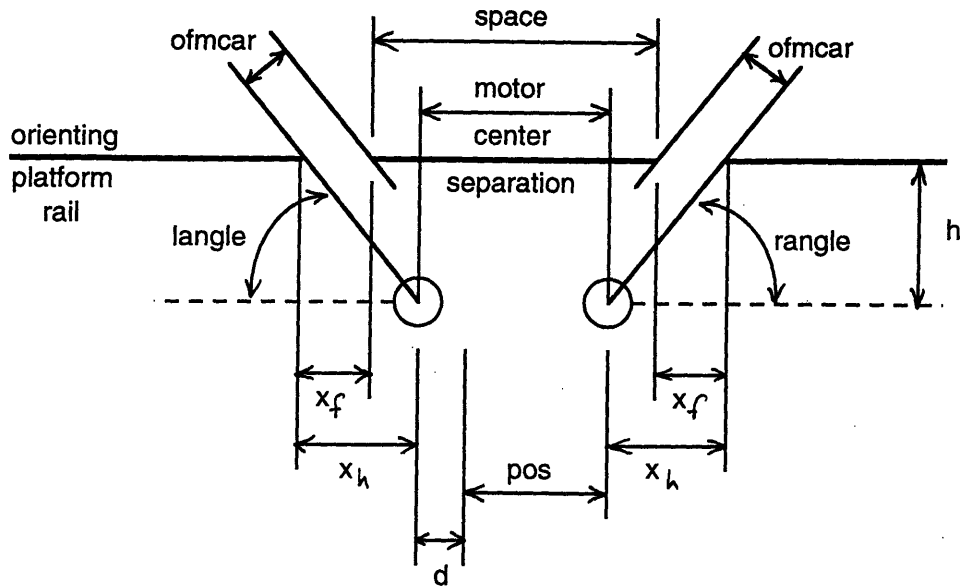


Figure 11: Illustration of Required Block Position Calculation Dimensions

According to figure 11,

$$\tan(90 - [l,r]angle) = x_h / h, \quad x_h = h * \tan(90 - [l,r]angle) \quad \text{Eqn. 5, 6}$$

$$\cos(90 - [l,r]angle) = ofmcar / x_f, \quad x_f = ofmcar / \cos(90 - [l,r]angle) \quad \text{Eqn. 6, 7}$$

Plugging these values into equation 4, the final value for a block position with a separation of (space), where the motor centers are originally (d) apart, and are (h) below the bottom rail,

$$\text{pos} = -\text{INTOQD} * ( \text{space} - d + h * ( \tan(90 - \text{langle}) + \tan(90 - \text{rangle}) ) - \text{ofmcar} * ( 1 / \cos(90 - \text{langle}) + 1 / \cos(90 - \text{rangle}) ) ) \quad \text{Eqn. 8}$$

#### **FINAL BLOCK POSITION:**

The final block position is calculated by simply substituting the bottom rib edge dimension into the generic block position equation as the required space.

#### **RIB TIPPING BLOCK POSITION:**

This position is necessary to tip over any square edged rib that might be standing on the smallest of its edges. If the rib is standing on its smallest edge, then the required distance between blocks is equal to the length of its opposite (longest) edge. Since this will always be either the top or the bottom dimension, the larger of the two can simply be substituted into the generic block position calculation as the required space.

#### **RIB CLEARANCE BLOCK POSITION:**

At certain points in the orientation algorithms, it is necessary to move the arms far enough apart from each other such that the rib can lie between them without being engaged. If the maximum possible length created by a rib is determined, then the appropriate block position can be calculated by substituting this length into the generic equation. Except in the case where the rib edges are compound angles (in which case the top or bottom dimension will be greatest), the largest dimension across a rib is the distance between two corners. As illustrated in figure I2, The appropriate corners depend on rib geometry: the upper corner corresponding to the smallest angle from the outside

horizontal will be the greatest distance from the opposite bottom corner. The calculation is as follows.

First, the lengths of the rib edges are calculated.

$$\text{right edge length (redge)} = \text{abs}(9.25 * \sin(\text{rangle})) \quad \text{Eqn. 9}$$

$$\text{left edge length (ledge)} = \text{abs}(9.25 * \sin(\text{langle})) \quad \text{Eqn. 10}$$

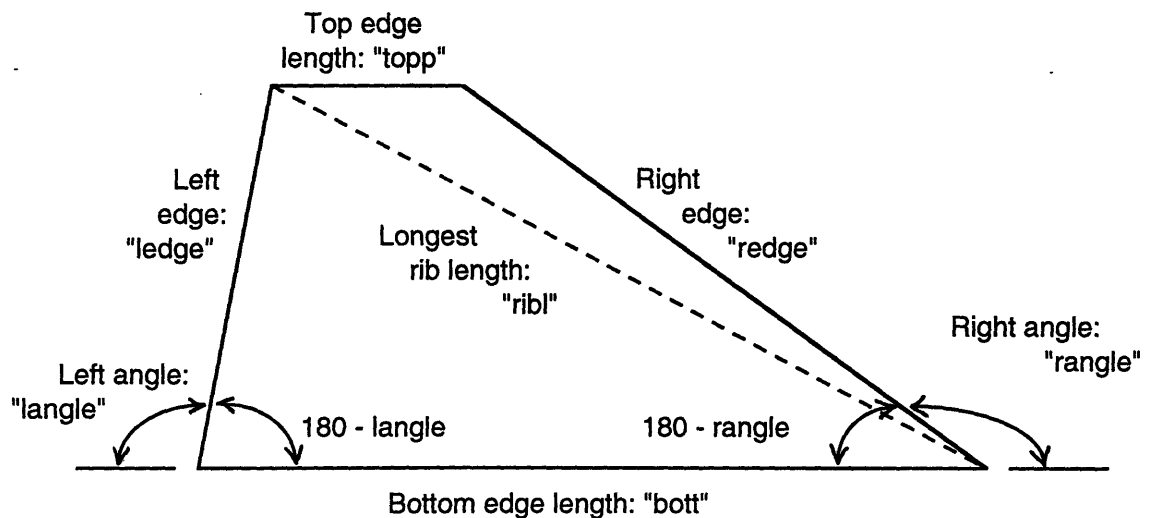
Then, the maximum rib length is calculated by the cosine law:

```

if (langle < rangle)
    ribl = sqrt(ledge2 + redge2 - 2*ledge*redge*cos(180 - langle))
otherwise
    ribl = sqrt(ledge2 + redge2 - 2*ledge*redge*cos(180 - rangle))

```

Once this distance is calculated it must be compared to the top and bottom rib dimensions to ensure that the appropriate value is used.



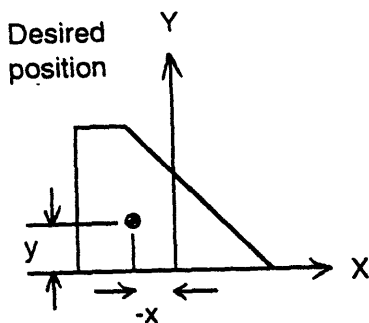
Since  $\text{langle} < \text{rangle}$ ,  
 According to the law of cosines,  $\text{ribl} = \text{bott}^2 + \text{ledge}^2 - 2 * \text{bott} * \text{ledge} * \cos(180 - \text{langle})$   
 However, if  $\text{bott} > \text{ribl}$  or  $\text{topp} > \text{ribl}$ , then  $\text{ribl}$  would be set as equal to the largest value.

Figure I2: Cosine Law and Longest Rib Length Dimensions



## Appendix J: CENTROID AND MIS-ROTATION CAPABILITY CALCULATIONS

In some cases the maximum angle to which a rib can be mis-rotated before entering the orienter must be determined. If the centroid location is calculated as shown in figure J1a, then, as shown in figure J1b, the maximum mis-rotation capability is the angle created by balancing the centroid over the edge corner corresponding to the clockwise or counter-clockwise rotation. The clockwise mis-rotation angle for geometries similar to that shown in figure J1c are negative, since they are unstable in their upright positions. However, the actual value may be conservatively reset to 90 degrees, since the corresponding algorithms can handle this mis-rotation. This appendix includes an example spreadsheet for calculating these values, and the actual equations are in Appendix G: the commented code that calculates and notifies the user of these capabilities.

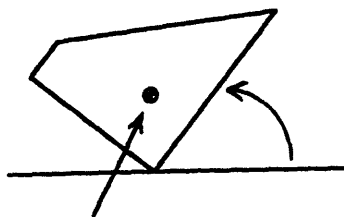


Centroid location:  
 Y: positive upward  
 from base  
 X: positive right  
 from base center

Figure J1a

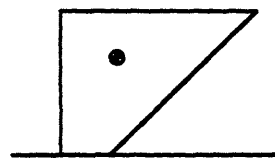
Figure J1b

Counter-  
 clockwise  
 misrotation



Centroid balanced over  
 edge corner which  
 corresponds to mis-  
 rotation direction

The calculated mis-rotation  
 value for this geometry will be  
 negative since the centroid is not  
 balanced in the upright position.



Therefore the counter-clockwise  
 mis-rotation value will remain  
 the same, and the clockwise  
 value will be estimated as 90 degrees.

Figure J1c

Figure J1: Illustration of Rib Centroid Position and Mis-Rotation Capability Calculations

| base  | top   | langle | rangle | height | degtors  | Rib Part | Area   | xbar  | ybar  | (xbar)*A | (ybar)*A |                        |
|-------|-------|--------|--------|--------|----------|----------|--------|-------|-------|----------|----------|------------------------|
| 24.05 | 2.00  | 140    | 140    | 9.25   | 0.017453 | center   | 222.48 | 0.00  | 4.63  | 0.00     | 1028.89  |                        |
| 24.05 | 2.00  | 140    | 140    | 9.25   | 0.017453 | left     | -50.98 | -8.35 | 6.17  | 425.74   | -314.41  |                        |
| 24.05 | 2.00  | 140    | 140    | 9.25   | 0.017453 | right    | -50.98 | 8.35  | 6.17  | -425.74  | -314.41  | --Critical Positions-- |
| 24.05 | 2.00  | 140    | 140    | 9.25   | 0.017453 | total    | 120.49 | 0.00  | 16.96 | 0.00     | 400.08   | Left alpha             |
| 24.05 | 2.00  | 140    | 140    | 9.25   | 0.017453 | centroid |        | 0.00  | 3.32  |          |          | 74.56                  |
| 12.88 | 2.00  | 120    | 120    | 9.25   | 0.017453 | center   | 117.29 | 0.00  | 4.63  | 0.00     | 542.47   |                        |
| 12.88 | 2.00  | 120    | 120    | 9.25   | 0.017453 | left     | -24.70 | -4.56 | 6.17  | 112.63   | -152.32  |                        |
| 12.88 | 2.00  | 120    | 120    | 9.25   | 0.017453 | right    | -24.70 | 4.56  | 6.17  | -112.63  | -152.32  | --Critical Positions-- |
| 12.88 | 2.00  | 120    | 120    | 9.25   | 0.017453 | total    | 67.89  | 0.00  | 16.96 | 0.00     | 237.84   | Left alpha             |
| 12.88 | 2.00  | 120    | 120    | 9.25   | 0.017453 | centroid |        | 0.00  | 3.50  |          |          | 61.08                  |
| 13.02 | 2.00  | 90     | 140    | 9.25   | 0.017453 | center   | 120.44 | 0.00  | 4.63  | 0.00     | 557.01   |                        |
| 13.02 | 2.00  | 90     | 140    | 9.25   | 0.017453 | left     | 0.00   | -6.51 | 3.08  | 0.00     | 0.00     |                        |
| 13.02 | 2.00  | 90     | 140    | 9.25   | 0.017453 | right    | -50.98 | 2.84  | 6.17  | -144.56  | -314.41  | --Critical Positions-- |
| 13.02 | 2.00  | 90     | 140    | 9.25   | 0.017453 | total    | 69.45  | -3.67 | 13.88 | -144.56  | 242.61   | Left alpha             |
| 13.02 | 2.00  | 90     | 140    | 9.25   | 0.017453 | centroid |        | -2.08 | 3.49  |          |          | 51.73                  |
| 7.34  | 2.00  | 90     | 120    | 9.25   | 0.017453 | center   | 67.90  | 0.00  | 4.63  | 0.00     | 314.01   |                        |
| 7.34  | 2.00  | 90     | 120    | 9.25   | 0.017453 | left     | 0.00   | -3.67 | 3.08  | 0.00     | 0.00     |                        |
| 7.34  | 2.00  | 90     | 120    | 9.25   | 0.017453 | right    | -24.70 | 1.89  | 6.17  | -46.68   | -152.32  | --Critical Positions-- |
| 7.34  | 2.00  | 90     | 120    | 9.25   | 0.017453 | total    | 43.20  | -1.78 | 13.88 | -46.68   | 161.70   | Left alpha             |
| 7.34  | 2.00  | 90     | 120    | 9.25   | 0.017453 | centroid |        | -1.08 | 3.74  |          |          | 34.67                  |
| 2.00  | 24.05 | 40     | 40     | 9.25   | 0.017453 | center   | 18.50  | 0.00  | 4.63  | 0.00     | 85.56    |                        |
| 2.00  | 24.05 | 40     | 40     | 9.25   | 0.017453 | left     | 50.98  | -4.67 | 6.17  | -238.33  | 314.41   |                        |
| 2.00  | 24.05 | 40     | 40     | 9.25   | 0.017453 | right    | 50.98  | 4.67  | 6.17  | 238.33   | 314.41   | --Critical Positions-- |
| 2.00  | 24.05 | 40     | 40     | 9.25   | 0.017453 | total    | 120.47 | 0.00  | 16.96 | 0.00     | 714.37   | Left alpha             |
| 2.00  | 24.05 | 40     | 40     | 9.25   | 0.017453 | centroid |        | 0.00  | 5.93  |          |          | 9.57                   |
| 2.00  | 13.02 | 90     | 40     | 9.25   | 0.017453 | center   | 18.50  | 0.00  | 4.63  | 0.00     | 85.56    |                        |
| 2.00  | 13.02 | 90     | 40     | 9.25   | 0.017453 | left     | 0.00   | -1.00 | 6.17  | 0.00     | 0.00     |                        |
| 2.00  | 13.02 | 90     | 40     | 9.25   | 0.017453 | right    | 50.98  | 4.67  | 6.17  | 238.33   | 314.41   | --Critical Positions-- |
| 2.00  | 13.02 | 90     | 40     | 9.25   | 0.017453 | total    | 69.48  | 3.67  | 16.96 | 238.33   | 399.97   | Left alpha             |
| 2.00  | 13.02 | 90     | 40     | 9.25   | 0.017453 | centroid |        | 3.43  | 5.76  |          |          | 37.58                  |
|       |       |        |        |        |          |          |        |       |       |          |          | -22.89                 |

Figure J2: Rib Centroid and Mis-Rotation Calculation Spreadsheet

| base  | top   | langle | rangle | height | degtorad | Rib Part | Area   | xbar  | ybar  | (xbar) <sup>2</sup> A | (ybar) <sup>2</sup> A |
|-------|-------|--------|--------|--------|----------|----------|--------|-------|-------|-----------------------|-----------------------|
| 12.00 | 23.02 | 90     | 40     | 9.25   | 0.017453 | center   | 111.00 | 0.00  | 4.63  | 0.00                  | 513.38                |
| 12.00 | 23.02 | 90     | 40     | 9.25   | 0.017453 | left     | 0.00   | -6.00 | 6.17  | 0.00                  | 0.00                  |
| 12.00 | 23.02 | 90     | 40     | 9.25   | 0.017453 | right    | 50.98  | 9.67  | 6.17  | 493.26                | 314.41                |
| 12.00 | 23.02 | 90     | 40     | 9.25   | 0.017453 | total    | 161.98 | 3.67  | 16.96 | 493.26                | 827.78                |
| 12.00 | 23.02 | 90     | 40     | 9.25   | 0.017453 | centroid |        | 3.05  | 5.11  |                       | 60.53                 |
| 10.00 | 10.00 | 40     | 140    | 9.25   | 0.017453 | center   | 92.50  | 0.00  | 4.63  | 0.00                  | 427.81                |
| 10.00 | 10.00 | 40     | 140    | 9.25   | 0.017453 | left     | 50.98  | -8.67 | 6.17  | -442.27               | 314.41                |
| 10.00 | 10.00 | 40     | 140    | 9.25   | 0.017453 | right    | 50.98  | 1.33  | 3.08  | -67.58                | -157.20               |
| 10.00 | 10.00 | 40     | 140    | 9.25   | 0.017453 | total    | 92.50  | -7.35 | 13.88 | -509.85               | 585.02                |
| 10.00 | 10.00 | 40     | 140    | 9.25   | 0.017453 | centroid |        | -5.51 | 6.32  |                       | -4.63                 |
| 4.00  | 4.00  | 60     | 120    | 9.25   | 0.017453 | center   | 37.00  | 0.00  | 4.63  | 0.00                  | 171.13                |
| 4.00  | 4.00  | 60     | 120    | 9.25   | 0.017453 | left     | 24.70  | -3.78 | 6.17  | -93.37                | 152.32                |
| 4.00  | 4.00  | 60     | 120    | 9.25   | 0.017453 | right    | 24.70  | 0.22  | 3.08  | -5.43                 | -76.16                |
| 4.00  | 4.00  | 60     | 120    | 9.25   | 0.017453 | total    | 37.00  | -3.56 | 13.88 | -98.80                | 247.28                |
| 4.00  | 4.00  | 60     | 120    | 9.25   | 0.017453 | centroid |        | -2.67 | 6.88  |                       | -5.73                 |
| 7.00  | 7.00  | 50     | 130    | 9.25   | 0.017453 | center   | 64.75  | 0.00  | 4.63  | 0.00                  | 299.47                |
| 7.00  | 7.00  | 50     | 130    | 9.25   | 0.017453 | left     | 35.90  | -6.09 | 4.63  | -218.52               | 166.03                |
| 7.00  | 7.00  | 50     | 130    | 9.25   | 0.017453 | right    | 35.90  | 0.91  | 4.63  | -32.77                | -166.03               |
| 7.00  | 7.00  | 50     | 130    | 9.25   | 0.017453 | total    | 64.75  | -5.17 | 13.88 | -251.28               | 299.47                |
| 7.00  | 7.00  | 50     | 130    | 9.25   | 0.017453 | centroid |        | -3.88 | 4.63  |                       | -4.71                 |
| 12.00 | 12.00 | 90     | 90     | 9.25   | 0.017453 | center   | 111.00 | 0.00  | 4.63  | 0.00                  | 513.38                |
| 12.00 | 12.00 | 90     | 90     | 9.25   | 0.017453 | left     | 0.00   | -6.00 | 6.17  | 0.00                  | 0.00                  |
| 12.00 | 12.00 | 90     | 90     | 9.25   | 0.017453 | right    | 0.00   | 6.00  | 6.17  | 0.00                  | 0.00                  |
| 12.00 | 12.00 | 90     | 90     | 9.25   | 0.017453 | total    | 111.00 | 0.00  | 16.96 | 0.00                  | 513.38                |
| 12.00 | 12.00 | 90     | 90     | 9.25   | 0.017453 | centroid |        | 0.00  | 4.63  |                       | 52.37                 |
| 2.00  | 5.26  | 40     | 130    | 9.25   | 0.017453 | center   | 18.50  | 0.00  | 4.63  | 0.00                  | 85.56                 |
| 2.00  | 5.26  | 40     | 130    | 9.25   | 0.017453 | left     | 50.98  | -4.67 | 6.17  | -238.33               | 314.41                |
| 2.00  | 5.26  | 40     | 130    | 9.25   | 0.017453 | right    | 35.90  | -1.59 | 3.08  | 56.98                 | -110.68               |
| 2.00  | 5.26  | 40     | 130    | 9.25   | 0.017453 | total    | 33.59  | -6.26 | 13.88 | -181.35               | 289.28                |
| 2.00  | 5.26  | 40     | 130    | 9.25   | 0.017453 | centroid |        | -5.40 | 8.61  |                       | -27.06                |
| 4.42  | 2.00  | 60     | 130    | 9.25   | 0.017453 | center   | 40.89  | 0.00  | 4.63  | 0.00                  | 189.09                |
| 4.42  | 2.00  | 60     | 130    | 9.25   | 0.017453 | left     | 24.70  | -3.99 | 3.08  | -98.56                | 76.16                 |
| 4.42  | 2.00  | 60     | 130    | 9.25   | 0.017453 | right    | 35.90  | -0.38 | 6.17  | 13.54                 | -221.37               |
| 4.42  | 2.00  | 60     | 130    | 9.25   | 0.017453 | total    | 29.69  | -4.37 | 13.88 | -95.01                | 43.88                 |
| 4.42  | 2.00  | 60     | 130    | 9.25   | 0.017453 | centroid |        | -2.86 | 1.48  |                       | -23.86                |
| 4.42  | 2.00  | 60     | 130    | 9.25   | 0.017453 | centroid |        |       |       |                       | 73.76                 |

Figure J3: Rib Centroid and Mis-Rotation Calculation Spreadsheet, cont.



## **Appendix K: EXPLANATION OF DEMONSTRATION ORIENTATION ALGORITHM NUMBER FOUR.**

One of the early attempts at orienting focused on ribs with a top dimension less than the bottom dimension as an individual rib type. This appendix explains the resulting algorithm, as it is not included in the standard code, except as demonstration orientation number four.

If a maximum mis-rotation of 90 degrees is assumed, this rib may present two contacts: it may be resting either on the angled end or the square end. Of these two possible configurations, the case where the rib is resting on its angled end is the easiest to orient, so it is the target contact. The necessary orienting steps are described below.

**Motion 0:** Preparatory motion as described in Chapter 4.

**Motion 1:** With arm A in its lowest position and arm B in its upright position the block moves the arms closer together.

This series of steps does not affect a rib which is lying on its square edge, but insures that the tip of a rib lying on its angled end will slide up arm A into the desired contact. Arm B needs to be in its upright position for this series of steps. If it is not, a rib in either of the two possible configurations might slide up the arm and become mis-rotated by the next series of steps.

**Motion 2: Both arms rotate toward arm B until arm B is in its lowest position and arm A is just past its upright position.**

**This series of steps either rotates a rib which is lying on its angled end into an upright position, or it slides the rib which is lying on its square edge up on to arm B.**

**Motion 3: The block moves to its final position followed by the arms.**

**The last set of motions creates the desired contact for both original rib configurations, so the motors can move to their final positions.**



Room 14-0551  
77 Massachusetts Avenue  
Cambridge, MA 02139  
Ph: 617.253.5668 Fax: 617.253.1690  
Email: docs@mit.edu  
<http://libraries.mit.edu/docs>

## **DISCLAIMER OF QUALITY**

Due to the condition of the original material, there are unavoidable flaws in this reproduction. We have made every effort possible to provide you with the best copy available. If you are dissatisfied with this product and find it unusable, please contact Document Services as soon as possible.

Thank you.

**Some pages in the original document contain color pictures or graphics that will not scan or reproduce well.**