A Control System for Laser Trimming
Thick Film Resistors and the Reliability Effects

by
Ryp R. Walters

Thesis submitted to the Faculty of
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

Approved:

__________________________
Dr. Aicha Elshabini-Riad, Chairperson

__________________________
Dr. F. W. Stephenson

__________________________
Dr. Joseph G. Tront
A Control System for Laser Trimming
Thick Film Resistors and the Reliability Effects

by

Ryp R. Walters

Electrical Engineering

Dr. Aicha Elshabini-Riad, Chairperson

Abstract

Since the development of thick film hybrid microelectronic processing, there has been a need for methods to adjust for tighter tolerances for electrical components through a trimming process. Components/elements, as produced, show a tolerance of the order of ±10% due to the variability of the screen printing process associated with film curing conditions. The methods that have arisen from this need encompass a variety of technologies and techniques. The usefulness of each method is based on its operation, flexibility, repeatability, and post-trim effects on the resistor's reliability.

The work in this thesis concerns the laser trimming of resistor components to a tight tolerance. It is the objective of this thesis to address the performance of an Nd:YAG laser operation and interface with a computer. The first task involves a computer hardware system to be interfaced to the laser control system, this task includes both design and implementation. The second task consists of a software operating environment to be flowcharted, written, and tested. The third task involves the computer interface driving the laser in the process of trimming resistor components using different types of cuts. The trimmed resistor performance is evaluated as part of the study.
Acknowledgements

I wish to express my most sincere gratitude and appreciation to Dr. Aicha Elshabini-Riad for her guidance and patience throughout the development and construction of this project. My thanks goes to Dr. F. W. Stephenson for keeping abreast of this project during all of its phases. I also thank Dr. Joseph Tront for his time as a member of my committee and for his feedback. Finally, I thank Monty Hayes for his help during the times of this project that were marked with confusion and chaos.

I could not have done this thesis without the enduring help and love that I received from my wife, Paige Walters, during the entire, gruesome adventure. It is because of these things and so many others that I dedicate this thesis to her. I wish that her presence in this project could be made more prevalent.
# Table of Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Abstract</strong></td>
<td>iii</td>
</tr>
<tr>
<td><strong>Acknowledgments</strong></td>
<td>iii</td>
</tr>
<tr>
<td><strong>Table of Contents</strong></td>
<td>iv</td>
</tr>
<tr>
<td><strong>List of Figures</strong></td>
<td>ix</td>
</tr>
<tr>
<td><strong>Chapter 1</strong> <strong>Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 <strong>Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.2 <strong>Objectives</strong></td>
<td>2</td>
</tr>
<tr>
<td>1.3 <strong>Proposed Process and Tests</strong></td>
<td>2</td>
</tr>
<tr>
<td>1.4 Thesis Structure</td>
<td>3</td>
</tr>
<tr>
<td><strong>Chapter 2</strong> <strong>Literature Review</strong></td>
<td>4</td>
</tr>
<tr>
<td>2.1 <strong>Methods of Resistor Trimming</strong></td>
<td>4</td>
</tr>
<tr>
<td>2.2 <strong>Techniques of Resistor Trimming</strong></td>
<td>5</td>
</tr>
<tr>
<td>2.2.1 The Single Straight Cut</td>
<td>5</td>
</tr>
<tr>
<td>2.2.2 The L Cut</td>
<td>7</td>
</tr>
<tr>
<td>2.2.3 The Multiple Straight Cut</td>
<td>8</td>
</tr>
<tr>
<td>2.3 <strong>Development of Applications</strong></td>
<td>8</td>
</tr>
<tr>
<td>2.4 <strong>Conclusion</strong></td>
<td>11</td>
</tr>
<tr>
<td><strong>Chapter 3</strong> <strong>Hardware System</strong></td>
<td>12</td>
</tr>
<tr>
<td>3.1 <strong>System Requirements</strong></td>
<td>12</td>
</tr>
<tr>
<td>3.2 <strong>Possible Interface Designs</strong></td>
<td>12</td>
</tr>
<tr>
<td>3.2.1 IBM XT with DOS and Available I/O Boards</td>
<td>12</td>
</tr>
<tr>
<td>3.2.2 IBM XT with DOS and a Constructed I/O Board</td>
<td>13</td>
</tr>
</tbody>
</table>
4.4.1.2.2 Analyze / Print a File 46
4.4.1.2.3 Edit a File 47
4.4.1.2.4 Run a file 48
4.4.1.3 Resistor Trimming 49
4.4.1.3.1 Resistance Value Acquisition 51
4.4.1.3.2 Single Straight Cut 51
4.4.1.3.3 Double Cut 51
4.4.1.3.4 L Cut 52
4.4.1.3.5 Multiple Straight Cuts 52
4.4.1.3.6 Measurement Test Routines 53
4.4.1.3.7 Trimming Defaults 53

4.4.2 FILEMAKE Program Operations 53

4.5 Conclusion 57

Chapter 5 Resistor Trimming and Aging Experimental Realization 58

5.1 Overview 58

5.1.1 Objectives 58
5.1.2 Testing Configuration 59
5.1.3 Thermal Aging Procedure 59
5.1.4 Testing Results 61

5.2 Analysis of Types of Cut, System Power, and Resistor Tolerances 61

5.2.1 The Single Straight Cut 61
5.2.2 The L Cut with the Elbow at 60% 64
5.2.3 The L Cut with the Elbow at 70% 64
5.2.4 The L Cut with the Elbow at 80% 73
5.2.5 The Double Cut 73
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>The Single Straight Cut</td>
<td>6</td>
</tr>
<tr>
<td>2.2</td>
<td>The L Cut</td>
<td>6</td>
</tr>
<tr>
<td>2.3</td>
<td>The Double Cut</td>
<td>9</td>
</tr>
<tr>
<td>2.4</td>
<td>The Double-Reverse Cut</td>
<td>9</td>
</tr>
<tr>
<td>2.5</td>
<td>The Serpentine Cut</td>
<td>10</td>
</tr>
<tr>
<td>3.1a</td>
<td>BMB A</td>
<td>18</td>
</tr>
<tr>
<td>3.1b</td>
<td>BMB C</td>
<td>18</td>
</tr>
<tr>
<td>3.2a</td>
<td>BAC A</td>
<td>19</td>
</tr>
<tr>
<td>3.2b</td>
<td>BAC B</td>
<td>19</td>
</tr>
<tr>
<td>3.2c</td>
<td>BAC C</td>
<td>19</td>
</tr>
<tr>
<td>3.3a</td>
<td>AC A</td>
<td>20</td>
</tr>
<tr>
<td>3.3b</td>
<td>AC B</td>
<td>20</td>
</tr>
<tr>
<td>3.3c</td>
<td>AC C</td>
<td>20</td>
</tr>
<tr>
<td>3.4</td>
<td>BAC Convertor Board</td>
<td>22</td>
</tr>
<tr>
<td>3.5</td>
<td>BMB Convertor Board</td>
<td>23</td>
</tr>
<tr>
<td>3.6</td>
<td>AC Convertor Board</td>
<td>24</td>
</tr>
<tr>
<td>4.1</td>
<td>Function Groups and IOP Break-up</td>
<td>28</td>
</tr>
<tr>
<td>4.2</td>
<td>PC Accumulator Usage</td>
<td>29</td>
</tr>
<tr>
<td>4.3</td>
<td>Laser Accumulator Usage</td>
<td>30</td>
</tr>
<tr>
<td>4.4</td>
<td>Comparator Flag/Laser Control</td>
<td>31</td>
</tr>
<tr>
<td>4.5</td>
<td>Bridge Range/Measuring System</td>
<td>31</td>
</tr>
<tr>
<td>4.6</td>
<td>Scanner Position and Channel</td>
<td>32</td>
</tr>
</tbody>
</table>
4.7 Display and Sorting
4.8 Overload Readout
4.9 Handler Information
4.10 The Main Menu
4.11 The Pattern Cut Menu
4.12 The Create a File Menu
4.13 The Resistor Trimming Menu
4.14 The Main Measurement Test Menu
4.15 The Measurement Test Value Selection Menu
4.16 The Trimming Defaults Menu
5.1 The Substrate
5.2 The Final Tolerances for the Straight Cut
5.3 The Breakup of Tolerances for the Straight Cut
5.4 The Tolerances of the First Cut of a L Cut
   with the Elbow at 60%
5.5 The Breakup of Tolerances of the First Cut of a L Cut
   with the Elbow at 60%
5.6 The Final Tolerances of a L Cut
   with the Elbow at 60%
5.7 The Breakup of the Final Tolerances of a L Cut
   with the Elbow at 60%
5.8 The Tolerances of the First Cut of a L Cut
   with the Elbow at 70%
5.9 The Breakup of Tolerances of the First Cut of a L Cut
   with the Elbow at 70%
5.10 The Final Tolerances of a L Cut
    with the Elbow at 70%  71
5.11 The Breakup of the Final Tolerances of a L Cut
    with the Elbow at 70%  72
5.12 The Tolerances of the First Cut of a L Cut
    with the Elbow at 80%  74
5.13 The Breakup of Tolerances of the First Cut of a L Cut
    with the Elbow at 80%  75
5.14 The Final Tolerances of a L Cut
    with the Elbow at 80%  76
5.15 The Breakup of the Final Tolerances of a L Cut
    with the Elbow at 80%  77
5.16 The Tolerances of the First Cut of a Double Cut  78
5.17 The Breakup of Tolerances of the First Cut of a Double Cut  79
5.18 The Final Tolerances of a Double Cut  80
5.19 The Breakup of the Final Tolerances of a Double Cut  81
5.20 The Tolerances of the First Cut of a Serpentine Cut  83
5.21 The Breakup of Tolerances of the First Cut of a Serpentine Cut  84
5.22 The Tolerances of the Second Cut of a Serpentine Cut  85
5.23 The Breakup of Tolerances of the Second Cut of a Serpentine Cut  86
5.24 The Tolerances of the Third Cut of a Serpentine Cut  87
5.25 The Breakup of Tolerances of the Third Cut of a Serpentine Cut  88
5.26 The Final Tolerances of a Serpentine Cut  89
5.27 The Breakup of the Final Tolerances of a Serpentine Cut  90
5.28 The Resistor Dimensions  92
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.29</td>
<td>The Numbering Format for the Resistors</td>
</tr>
<tr>
<td>5.30</td>
<td>The Single Straight Cut</td>
</tr>
<tr>
<td>5.31</td>
<td>The L Cut with the Elbow at 60%</td>
</tr>
<tr>
<td>5.32</td>
<td>The L Cut with the Elbow at 70%</td>
</tr>
<tr>
<td>5.33</td>
<td>The L Cut with the Elbow at 80%</td>
</tr>
<tr>
<td>5.34</td>
<td>The Double Cut</td>
</tr>
<tr>
<td>5.35</td>
<td>The Serpentine Cut</td>
</tr>
<tr>
<td>5.36</td>
<td>The Tolerances for the Control Substrate after Thermal Aging</td>
</tr>
<tr>
<td>5.37</td>
<td>The Breakup of Tolerances for the Control Substrate after Thermal Aging</td>
</tr>
<tr>
<td>5.38</td>
<td>The Tolerances for the Single Straight Cut after Thermal Aging</td>
</tr>
<tr>
<td>5.39</td>
<td>The Breakup of Tolerances for the Single Straight Cut after Thermal Aging</td>
</tr>
<tr>
<td>5.40</td>
<td>The Tolerances for the L Cut at 60% after Thermal Aging</td>
</tr>
<tr>
<td>5.41</td>
<td>The Breakup of Tolerances for the L Cut at 60% after Thermal Aging</td>
</tr>
<tr>
<td>5.42</td>
<td>The Tolerances for the L Cut at 70% after Thermal Aging</td>
</tr>
<tr>
<td>5.43</td>
<td>The Breakup of Tolerances for the L Cut at 70% after Thermal Aging</td>
</tr>
<tr>
<td>5.44</td>
<td>The Tolerances for the L Cut at 80% after Thermal Aging</td>
</tr>
</tbody>
</table>
5.45 The Breakup of Tolerances for the L Cut at 80% after Thermal Aging  

5.46 The Tolerances for the Double Cut after Thermal Aging  

5.47 The Breakup of Tolerances for the Double Cut after Thermal Aging  

5.48 The Tolerances for the Serpentine Cut after Thermal Aging  

5.49 The Breakup of Tolerances for the Serpentine Cut after Thermal Aging
Chapter 1

Introduction

1.1 Introduction

With its usefulness, diversity, and size, thick film processing has many applications. Along with this processing technique comes the inherent difficulties of obtaining and maintaining low tolerance components. This fact is particularly true for thick film resistors. Although their geometry can be quite accurate, the screen printing process associated with film curing conditions leads to limited control of the tolerance value. To alleviate this problem, various methods such as abrasive trimming [1,2], radio frequency trimming [3], and laser trimming [4-29] are used to adjust the resistance value to a tighter tolerance. Various standard techniques are applicable for all of the methods that can increase the resistance to a very fine tolerance. The actual tolerance range is a function of both the method and the technique. The trimming process requires that the resistor be designed for less than its final intended value.

The process of trimming a resistor is highly dependent on the capabilities and limitations of the method as well as the techniques used. Each method that can be used involves a control mechanism and a trimming mechanism; each is an essential part of operation. The techniques, such as the L cut and the serpentine cut, are standard for all methods but are better realized in some methods than in others. An abrasive trimmer, for instance, may not be able to make an accurate 90 degree turn that is required for the L cut. A laser system, on the other hand, can make sudden turns and keep the same travelling velocity.
1.2 Objectives

The main objective of this thesis is to design a new control interface and mechanism to connect a standard ESI Model 25 acousto-optic Q-switched Nd:YAG laser trimming system to an IBM XT. This entire system should be able to easily and accurately trim resistors using different techniques. The hardware interface is designed from the component level schematics of the ESI Model 25 laser system and from the requirements of the chosen Input/Output board for the IBM XT computer. When the hardware is completed, the software operating environment is designed for ease of use and for flexible operating options. With a completed control system, resistors are trimmed using different techniques. Both the trimmed as well as untrimmed resistors are then thermally aged for further component evaluation.

1.3 Proposed Process and Tests

The important aspect in the control system is that the computer must be able to control all of the parameters required in laser trimming. The hardware interface and software hand-shaking must be able to address all of the laser's control system functions whether all of the functions are required at one time or not. The software operating environment must provide the user with options to the available parameters as well as to routines for different techniques. The reliability of the resistors is tested by measuring the current leakage during and after thermal cycling has occurred.
1.4 Thesis Structure

This thesis consists of seven chapters. This chapter, Chapter 1, is a general introduction. Chapter 2 presents a literature survey of previously published work related to resistor trimming. In Chapter 3, an evaluation is made of possible hardware interface designs and a discussion is presented about requirements for the laser control system. The software is described in Chapter 4. This involves an understanding of the software system requirements, the discussion of constructed prototypes, and the presentation of the principles of operation of the completed software operating environment. Chapter 5 previews the resistor trimming configuration and the thermal aging procedure. Also in Chapter 5, the results are displayed and evaluated by power and frequency, by resistor geometry, and by post thermal aging stability. Chapter 6 concludes this thesis and provides recommendations for future research.
Chapter 2

Literature Review

2.1 Methods of Resistor Trimming

Due mainly to the limitations of technology and the price of the method, the original methods for resistor trimming were based on abrasive techniques that were similar to a fine streamed sand-blaster [1]. Through the advancement of technology, reliable and relatively inexpensive laser systems were constructed. The two laser systems that were put into production for use as laser trimmers were Carbon Dioxide (CO$_2$) lasers [2-4] and the acusto-optic, Q-switched Neodymium doped Yttrium Aluminum Garnet (Nd:YAG) lasers [3-4, 12-19, 22, 23-30]. The main difference between the laser systems is that the Nd:YAG laser has a ten times smaller line cutting width than the CO$_2$. Though the laser systems have far surpassed the accuracy and reliability of the abrasive systems, there is still some debate as to which system performs better under certain specific applications [5]. Most recently, other alternatives such as an electro-optic, Q-switched Nd:YAG systems [6], a radio frequency (RF) technique [7], and an ultrasonic trimming [8] have emerged. All of these methods are based on cutting into a resistor with as small of a straight line width as possible. Even though the obtainable accuracies are within the same range, there has always been discussion about the 'best' type of system [3-6] to obtain a stable performance over a period of time for a certain range of frequencies with minimum parasitics.
2.2 Techniques of Resistor Trimming

It is generally accepted [9-11] that there are three types of popular trimming techniques: the single straight cut, the L cut, and the multiple straight cut. These techniques are relevant for all of the trimming methods since all of the techniques involve straight line cuts into the resistor element.

2.2.1 The Single Straight Cut

The single straight cut (see Figure 2.1) is the simplest and fastest type of cut, but by far gives the least precision of the three techniques. The single straight cut starts at one edge of the resistor and cuts into the resistor perpendicular to the current flow. It is clear that as the cut proceeds further across the resistor, the change in the resistor value will increase more rapidly. The action makes it more difficult to stop the cut at the desired value because, during the time in between the measurement of the resistor's value, the resistance value will have greatly increased. By the time the control system has stopped the trimmer, the resistance will have already passed the desired value. Although this may hypothetically be bypassed by a very slow trim, other effects such as heating and width expansion play a major part in slow trims by adversely effecting the measurement of the resistance value as well as the future operating reliability.
Figure 2.1 The Single Straight Cut

Figure 2.2 The L Cut
2.2.2 The L Cut

The L cut (see Figure 2.2) is much more effective than the single straight cut and does not require much more time. The principle behind the L cut is that once the change in resistance value begins to rapidly increase as the trimmer is cutting perpendicular to the current flow, the trimmer changes direction and cuts serial to the current flow. This change in cutting direction significantly decreases the change in resistance per distance. This allows the trimming control system more time to shut the trimmer off while the resistor value is within tolerance and not have the trimmer cut the resistor out of tolerance while the control system is in the process of shutting the trimmer off. The decision about when the change in direction should be made can be based on a number of different criteria. First, the control system could be made to monitor the change in resistance as well as the resistance value. Once the change in resistance was, for example, three times what it was when the trim was first started, the direction would change. Another way would be to have the direction change when the trimmer was one third of the way through the resistor. A third way would be to do a straight cut to a percentage of the desired resistance value and then to change direction. The L cut offers more options than the single straight cut. The disadvantage is that the L cut can be ineffective on resistors that have a longer length (resistor distance across the conductive pad) than width (resistor distance on free sides between the pads). In these cases, the L cut may reach the end of the resistor and start cutting into the pad while still not being within the desired tolerance range.
2.2.3 The Multiple Straight Cut

The third type of cut is the multiple straight cut. The multiple straight cut is based on the single straight cut but uses multiple cuts to bypass the problem of cutting too deep on any one cut and making tolerance acquisition difficult. There are three standard types of multiple straight cuts. The first is the double straight cut (see Figure 2.3). This cut type uses two cuts, each originating on the same side of the resistor. The second is the double-reverse cut (see Figure 2.4). This is similar to the regular double cut except that the cuts begin from opposite sides of the resistor. The third type of cut is the serpentine cut (see Figure 2.5). This cut uses a number of cuts that alternate origination sides. Although the serpentine cut may seem to give the best results of all of the techniques, a capacitive effect arises due to the configuration of the cuts that can be damaging in high frequency operation. When using multiple cuts, care must be taken so that the spacing between the trims does not violate the rigid power capability of the resistor [10].

2.3 Development of Applications

Focusing on the acousto-optic, Q-switched, Nd:YAG laser in this thesis, there has been a lot of research dealing with specific applications. Aside from using the laser as a scribing tool [12], the Nd:YAG laser is quite a flexible tool. With regard to research dealing with the effects of resistor trimming of specific compositions of resistors, studies have been completed regarding the characteristics of Tantalum Nitride resistors on Silver [13] and the trimming effects on Birox and 1100 series thick film compositions [14]. These studies are accompanied by the laser trimming effects on polymer thick film
Figure 2.3 The Double Cut

Figure 2.4 The Double-Reverse Cut
Figure 2.5 The Serpentine Cut
resistors [15] and thermistors that require high accuracy [16-17]. The Nd:YAG provides the flexibility required for the trimming of different types of material. The documented research involved analyzing different power settings for their laser, not whether or not the Nd:YAG would actually cut their material. In measuring the accuracy and reliability of laser trimmed resistors, it is useful to measure against a relatively high standard such as the standard which is required for military applications [18]. The beam size of the Nd:YAG and the possible computer configurations allow for the precision and accuracy required for stringent standards.

2.4 Conclusion

The use of the Nd:YAG laser as a functional tool is well documented. Of the many possible methods to trim thick film resistors, the Nd:YAG laser has been a strong competitor with its precision and accuracy. A set of popular trimming techniques exist that allow for some flexibility in how a particular resistor may be trimmed. The straight cut is simple but not very accurate. The L cut is accurate but may be ineffective on certain types of resistor geometries. This geometry limitation also exists for the multiple straight cuts. The multiple straight cuts provide the tightest tolerance but may add unwanted capacitance at high frequency.
Chapter 3

Hardware System

3.1 System Requirements

The objective of this system is to be as precise and reliable in the full usage of the laser control system while allowing the user full capability to change all available system parameters. The laser control system is asynchronous; therefore, the interface must handle all of the required hand-shaking. The laser control system has three cables of 40 lines through which communication must be maintained. The interface should minimally use the data lines as well as the skip test line to control the laser system. An interrupt line is available but may not be required depending upon whether the interface is polled or interrupt driven.

3.2 Possible Interface Designs

To design the control system interface, all possible PC methodologies are examined. Each system is analyzed using a need, cost, and availability analysis. The following sections detail the different possible interface designs.

3.2.1 IBM XT with DOS and Available I/O Boards

The IBM XT is an 8 bit, 8 Megahertz, 8088 machine. Since a single laser operation requires a 24 bit command followed by an 8 bit input or output, it would take
the IBM XT three machine cycles to produce a single operation. Since DOS is a single
tasking operating system, a full hand-shaking scheme can be implemented with either
polled or interrupt driven I/O. This is because the computer can wait for the laser to
respond to the last task set to it. Since DOS can not perform any other tasks until the
laser responds, all of the response methods will perform the same. There are a variety of
I/O boards available for the XT that can reasonably perform the desired functions. To
fully interface with the laser, the XT requires three I/O Boards; although, not all of the
I/O lines will be used. Due to the asynchronous nature of the laser and the logic that it is
implemented in, some buffering capability will be required.

3.2.2 IBM XT with DOS and a Constructed I/O Board

Although very similar to the previous option, constructing new boards can
conserve a lot of space as well as streamline the hand-shaking both in hardware and in
software. The construction of a new board can eliminate ambiguities that occur in the
previous system, due to being able to choose how many I/O lines to construct. This new
construction can fit on one board. The need for buffering can also be put right on the I/O
board. This process requires about four months to design, simulate, wire-wrap, and test.

3.2.3 IBM AT with DOS

The IBM AT is a 16 bit, 12 Megahertz, 80286 machine. This can provide a much
faster processing speed because the AT has a faster clock, moves twice as many bits per
cycle, and has a much more efficient processor. Because the AT has a 16 bit data bus,
the range of available I/O boards fully supports the interface system's need for multiple
I/O lines and for full buffering. An I/O board for the AT will provide all necessary lines on a single board. This board would come at a much higher price. With DOS as the operating system, the interface can work as either polled or interrupt driven without any performance difference.

3.2.4 IBM AT with OS/2

This is equivalent to the previous option except for the major difference in operating conditions due to using OS/2 as an operating system. OS/2 has the capability to multi-task computer events. Using an interrupt driven interface, the user can run a laser program and then immediately afterwards be able to work on building a different laser program or even to switch to any other computer application. This can save the user time when a laser program needs to be run while other computer tasks also need immediate attention. The limitation is that to build the required software support for OS/2 will require an additional four months over DOS. The extra time is also due to the requirement that the system be interrupt driven.

3.2.5 Additional Line Conversion Box

In addition to all of the previous options, a line conversion box can be useful to change the 40 pin laser cable into whatever format the I/O board uses. In essence, the line conversion box will convert the pin-out of the laser control system to the pin-out of the interface system. For those options that do not have the buffering and pull-up capabilities, this will be an ideal place to build those capabilities into the system.
3.3 Evaluation of Possible Interface Designs

Comparing cost, time required, and added advantages, the options are quickly evaluated. Even though the IBM AT system with OS/2 seems almost ideal, the time required to develop the necessary software out-weighs the possible time advantages that OS/2 allows. Since the computer system will be a dedicated machine, the cost of using an AT can not be justified even though the AT would be much faster. The two remaining options involve the IBM XT. Although the construction of a new I/O board may use the computer ports and the number of available lines more effectively, there is no reason for these things to be a factor since the XT will be dedicated to the laser system. The time and cost involved in the construction of the new I/O board is much less efficient than purchasing available boards. Available boards will require the use of a constructed line conversion box for changing the pinouts of the laser to the interface and to add buffering and pull-up capabilities required for the system. Although there is a sacrifice in speed between the AT and the XT, the XT provides sufficient speed for a much better time and cost value.

3.4 Design Description

The hardware interface is made up of three 24-bit serial I/O boards, six cables, and one line conversion box. This interface goes together with the control software to send the right codes at the right time to the laser. There are also two sets of three cables. Each set contains the following cables: BAC, BMB, and AC. The first set of cables goes from the laser to the line conversion box. These cables are D-submarine 37 pin lines. A
second set of cables goes from the line conversion box to the laser. These are 40 pin straight connector lines.

To better understand the process of communicating with the laser, this description will assume that the coding of a function into a function number, an IOP number, and accumulator usage has already been done in the control software. The software translated code is sent to the I/O boards, which will be the first topic of discussion. From the I/O boards, the code goes through the first set of cables to the line conversion box. Here, the individual lines are rearranged into the form that the laser expects to see them. The line conversion box will be the second topic of discussion.

3.4.1 The I/O Boards

The key to the information exchange between the two systems is to organize the interface in a manner that both the PC and the laser can understand. There are a number of difficulties that arise due to the differences between the PC and the laser code usage. Since the laser was originally designed for use with a PDP/8e, the codes are made up of octal numbers (sets of three bits). With the BMB line, three of the binary digits are required in their complement form. The laser, on the other hand, does not independently produce complements of any line and communicates in hexadecimal (sets of four bits). This leads to a need to create a conversion standard that allows each system to operate in its regular environment. Note that the address and the IOP numbers for the laser are read from the charts as octal numbers (sets of three bits). The PC and I/O boards deal in hexadecimal code (sets of four bits).

The three 24-bit serial digital I/O interface boards are inside the PC and are given addresses 300H, 304H, and 308H for the BMB, AC, and BAC lines, respectively. The
I/O boards chosen are Omega Engineering, Inc.'s Model PI012, 24 Bit Parallel Digital I/O Interface Boards. Each of the I/O boards is broken up into 3 sets of 8-bit ports. Each port is given a letter from A to C. This requires that there be a correspondence between each laser cable and each bit coming into/out of the I/O boards. For better troubleshooting and understanding, each I/O board corresponds to one of the laser cables. Therefore, each board is labeled by the corresponding line. The BMB line carries the function address in six bits as well as three of the six address bits in their complement form. The BMB I/O board divides the bits up according to Figure 3.1a-b. The BAC line (Figure 3.2a-c) carries the PC accumulator of 12 bits, the IOP of 3 bits, and an initialization bit. The AC line (Figure 3.3a-c) carries the laser accumulator of 12 bits, an interrupt request bit, and a skip test bit. Listed with the laser line name is its internal, laser connector number.

The address is formed by taking the "10's" digit of the address, breaking it up into an octal number, and putting it into BMB03-BMB05 with BMB03 being the most significant bit(msb). After putting the complement of BMB03-BMB05 (i.e. BMB03' - BMB05') into its place, take the "1's" digit of the address, break it up into an octal number, and put it into BMB06-BMB08. To form a code, the address is loaded on the BMB line (and data into the PC accumulator on the BAC line if the particular function needs data). A '1' is loaded in the correct IOP number's bit and then a '0' placed into that bit. This has the effect of a pulse. If the function does not use the PC accumulator but does use the laser accumulator or the skip line, a '1' is loaded into the correct IOP line, and the data is received from the AC line. After the data has been received, the IOP line is dropped to '0'. The data on the AC line only remains valid while the correct IOP line is high.
<table>
<thead>
<tr>
<th>A7</th>
<th>A6</th>
<th>A5</th>
<th>A4</th>
<th>A3</th>
<th>A2</th>
<th>A1</th>
<th>A0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>BMB03 J1</td>
<td>BMB04 M1</td>
<td>BMB05 S1</td>
<td>BMB06 E2</td>
<td>BMB07 K2</td>
<td>BMB08 P2</td>
</tr>
</tbody>
</table>

Figure 3.1a BMB A

<table>
<thead>
<tr>
<th>C7</th>
<th>C6</th>
<th>C5</th>
<th>C4</th>
<th>C3</th>
<th>C2</th>
<th>C1</th>
<th>C0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>BMB03' H1</td>
<td>BMB04' L1</td>
</tr>
</tbody>
</table>

Figure 3.1b BMB C

Figure 3.1 The BMB Line
<table>
<thead>
<tr>
<th>A7</th>
<th>A6</th>
<th>A5</th>
<th>A4</th>
<th>A3</th>
<th>A2</th>
<th>A1</th>
<th>A0</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAC00</td>
<td>BAC01</td>
<td>BAC02</td>
<td>BAC03</td>
<td>BAC04</td>
<td>BAC05</td>
<td>BAC06</td>
<td>BAC07</td>
</tr>
<tr>
<td>B1</td>
<td>D1</td>
<td>E1</td>
<td>H1</td>
<td>J1</td>
<td>L1</td>
<td>M1</td>
<td>P1</td>
</tr>
</tbody>
</table>

Figure 3.2a  BAC A

<table>
<thead>
<tr>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>AC08</td>
<td>AC09</td>
<td>AC10</td>
<td>AC11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S1</td>
<td>D2</td>
<td>E2</td>
<td>H2</td>
</tr>
</tbody>
</table>

Figure 3.2b  BAC B

<table>
<thead>
<tr>
<th>C7</th>
<th>C6</th>
<th>C5</th>
<th>C4</th>
<th>C3</th>
<th>C2</th>
<th>C1</th>
<th>C0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>BINT</td>
<td>IOP4</td>
<td>IOP2</td>
<td>IOP1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V2</td>
<td>P2</td>
<td>M2</td>
<td>K2</td>
</tr>
</tbody>
</table>

Figure 3.2c  BAC C

Figure 3.2  The BAC Line
### Figure 3.3a AC A

<table>
<thead>
<tr>
<th>A7</th>
<th>A6</th>
<th>A5</th>
<th>A4</th>
<th>A3</th>
<th>A2</th>
<th>A1</th>
<th>A0</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC00</td>
<td>AC01</td>
<td>AC02</td>
<td>AC03</td>
<td>AC04</td>
<td>AC05</td>
<td>AC06</td>
<td>AC07</td>
</tr>
<tr>
<td>B1</td>
<td>D1</td>
<td>E1</td>
<td>E1</td>
<td>J1</td>
<td>L1</td>
<td>M1</td>
<td>P1</td>
</tr>
</tbody>
</table>

### Figure 3.3b AC B

<table>
<thead>
<tr>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>AC08</td>
<td>AC09</td>
<td>AC10</td>
<td>AC11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S1</td>
<td>D2</td>
<td>E2</td>
<td>H2</td>
</tr>
</tbody>
</table>

### Figure 3.3c AC C

<table>
<thead>
<tr>
<th>C7</th>
<th>C6</th>
<th>C5</th>
<th>C4</th>
<th>C3</th>
<th>C2</th>
<th>C1</th>
<th>C0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>SKIP</td>
<td>INTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>K2</td>
<td>M2</td>
</tr>
</tbody>
</table>

### Figure 3.3 The AC Line
3.4.2 The Line Conversion Box

The line conversion box basically converts the 37 pin cable to the 40 pin cable. Those lines that transfer information from the PC to the laser (i.e. BMB and BAC) are also buffered so that they can supply the laser's DTL transistors with enough current to operate. The AC lines have pull up resistors on them because the laser transistor logic goes tri-state when the lines are not in use.

Once the code is placed on the I/O board's ports, the signal needs to be transferred to the laser. As noted before, the individual line order that the laser and the PC use the are very different. The line conversion box is made of three perforated boards, one for each line. The BAC and BMB boards each have a 74LS244N buffer chip. The AC board has a pull-up resistor for each one of its data lines. Figure 3.4 displays the BAC board pin configuration with the line numbers from the PC cable, the board position of the individual lines from the PC, the pin and chip that line is connected to, the pin that line is buffered out to, the board position that the laser cable is connected to that takes that line, and the line number of each line on the laser cable. Figure 3.5 displays the BMB board pin configuration with the line numbers from the PC cable, the board position of the individual lines from the PC, the pin and chip that line is connected to, the pin that line is buffered out to, the board position that the laser cable is connected to that takes that line, and the line number of each line on the laser cable. Finally, Figure 3.6 shows the line number of each line from the laser's AC cable and the corresponding line number of each line on the PC's AC cable.
<table>
<thead>
<tr>
<th>PC Line Number</th>
<th>PC Board Position</th>
<th>Pin from PC</th>
<th>Chip</th>
<th>Pin to Laser</th>
<th>Laser Board Position</th>
<th>Laser Line Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAC00-22</td>
<td>12</td>
<td>2</td>
<td>U1</td>
<td>18</td>
<td>21</td>
<td>4</td>
</tr>
<tr>
<td>BAC01-24</td>
<td>13</td>
<td>4</td>
<td>U1</td>
<td>16</td>
<td>19</td>
<td>8</td>
</tr>
<tr>
<td>BAC02-26</td>
<td>14</td>
<td>6</td>
<td>U1</td>
<td>14</td>
<td>17</td>
<td>12</td>
</tr>
<tr>
<td>BAC03-28</td>
<td>15</td>
<td>8</td>
<td>U1</td>
<td>12</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>BAC04-30</td>
<td>16</td>
<td>11</td>
<td>U1</td>
<td>9</td>
<td>13</td>
<td>20</td>
</tr>
<tr>
<td>BAC05-32</td>
<td>17</td>
<td>13</td>
<td>U1</td>
<td>7</td>
<td>11</td>
<td>24</td>
</tr>
<tr>
<td>BAC06-34</td>
<td>18</td>
<td>15</td>
<td>U1</td>
<td>5</td>
<td>9</td>
<td>28</td>
</tr>
<tr>
<td>BAC07-36</td>
<td>19</td>
<td>17</td>
<td>U1</td>
<td>3</td>
<td>7</td>
<td>32</td>
</tr>
<tr>
<td>BAC08-13</td>
<td>1</td>
<td>2</td>
<td>U2</td>
<td>18</td>
<td>2</td>
<td>36</td>
</tr>
<tr>
<td>BAC09-15</td>
<td>2</td>
<td>4</td>
<td>U2</td>
<td>16</td>
<td>20</td>
<td>6</td>
</tr>
<tr>
<td>BAC10-21</td>
<td>3</td>
<td>6</td>
<td>U2</td>
<td>14</td>
<td>18</td>
<td>10</td>
</tr>
<tr>
<td>BAC11-19</td>
<td>4</td>
<td>8</td>
<td>U2</td>
<td>12</td>
<td>16</td>
<td>14</td>
</tr>
<tr>
<td>BIOP1-20</td>
<td>9</td>
<td>11</td>
<td>U2</td>
<td>9</td>
<td>14</td>
<td>18</td>
</tr>
<tr>
<td>BIOP2-18</td>
<td>8</td>
<td>13</td>
<td>U2</td>
<td>7</td>
<td>12</td>
<td>22</td>
</tr>
<tr>
<td>BIOP4-16</td>
<td>7</td>
<td>15</td>
<td>U2</td>
<td>5</td>
<td>10</td>
<td>26</td>
</tr>
<tr>
<td>BINT-14</td>
<td>6</td>
<td>17</td>
<td>U2</td>
<td>3</td>
<td>1</td>
<td>38</td>
</tr>
</tbody>
</table>

Figure 3.4 BAC Convertor Board
<table>
<thead>
<tr>
<th>PC Line Number</th>
<th>PC Board Position</th>
<th>Pin from PC</th>
<th>Chip</th>
<th>Pin to Laser</th>
<th>Laser Board Position</th>
<th>Laser Line Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMB03-26</td>
<td>14</td>
<td>2</td>
<td>U1</td>
<td>18</td>
<td>7</td>
<td>32</td>
</tr>
<tr>
<td>BMB04-28</td>
<td>15</td>
<td>4</td>
<td>U1</td>
<td>16</td>
<td>11</td>
<td>24</td>
</tr>
<tr>
<td>BMB05-28</td>
<td>16</td>
<td>6</td>
<td>U1</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>BMB06-28</td>
<td>17</td>
<td>8</td>
<td>U1</td>
<td>12</td>
<td>10</td>
<td>26</td>
</tr>
<tr>
<td>BMB07-28</td>
<td>18</td>
<td>11</td>
<td>U1</td>
<td>9</td>
<td>14</td>
<td>18</td>
</tr>
<tr>
<td>BMB08-28</td>
<td>19</td>
<td>13</td>
<td>U1</td>
<td>7</td>
<td>18</td>
<td>10</td>
</tr>
<tr>
<td>BMB03'-28</td>
<td>7</td>
<td>15</td>
<td>U1</td>
<td>5</td>
<td>5</td>
<td>36</td>
</tr>
<tr>
<td>BMB04'-28</td>
<td>8</td>
<td>17</td>
<td>U1</td>
<td>3</td>
<td>9</td>
<td>28</td>
</tr>
<tr>
<td>BMB05'-28</td>
<td>9</td>
<td>2</td>
<td>U2</td>
<td>18</td>
<td>13</td>
<td>20</td>
</tr>
</tbody>
</table>

Figure 3.5 BMB Convertor Board
<table>
<thead>
<tr>
<th>Laser Line Number</th>
<th>PC Line Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC00-4</td>
<td>36</td>
</tr>
<tr>
<td>AC01-8</td>
<td>34</td>
</tr>
<tr>
<td>AC02-12</td>
<td>32</td>
</tr>
<tr>
<td>AC03-16</td>
<td>30</td>
</tr>
<tr>
<td>AC04-20</td>
<td>28</td>
</tr>
<tr>
<td>AC05-24</td>
<td>26</td>
</tr>
<tr>
<td>AC06-28</td>
<td>24</td>
</tr>
<tr>
<td>AC07-32</td>
<td>22</td>
</tr>
<tr>
<td>AC08-36</td>
<td>19</td>
</tr>
<tr>
<td>AC09-6</td>
<td>17</td>
</tr>
<tr>
<td>AC10-10</td>
<td>15</td>
</tr>
<tr>
<td>AC11-14</td>
<td>13</td>
</tr>
<tr>
<td>INTR</td>
<td>20</td>
</tr>
<tr>
<td>SKIP</td>
<td>18</td>
</tr>
</tbody>
</table>

Figure 3.6 AC Convertor Board
3.5 Diagnostics

In order for the interface to be diagnosed for trouble-shooting purposes, VELTEST was written to test the data process from the PC to the laser. By monitoring either output of the linear motor's velocity digital-to-analog convertors on an oscilloscope, a diagnosis can be made about the accuracy of the BMB and the BAC lines. VELTEST simply ramps each of the motors velocities from a digital -125 to 125. This gives an analog output ramping from -0.67 Volts to +0.67 Volts. If the ramp function is not seen, there is a problem with the BMB line. If the ramp function is not continuous, there is a problem with the BAC line. To figure out which bit line is in error, LINETEST can be used to set and clear all of the bit lines on both the BAC and the BMB lines. The source code for VELTEST is found in Appendix X.

Once the input lines are secure, the AC line can be tested. This can be quickly seen from running the LASER program. When the laser control system is initializing, the laser head should be sent to its home position. If the returned "current position" is not (0,0), there is a problem with the AC line. To find the bit line in error, only the pull-up resistor values must be checked.
Chapter 4

Software / Programming

4.1 System Requirements

A completed interface will provide laser system register control, linear motor manipulation, and resistor trimming. Laser system register control involves sending values to specific registers and recognizing them. This will confirm that the register is being properly addressed and that the input lines to the laser system are operating. This will be witnessed by selecting the register that controls the laser's control panel lights. Linear motor manipulation requires the use of the previously established register selecting capability and an additional register output acquisition. This register output acquisition will be satisfied by the receipt and decoding of the linear motor position. The ability to trim resistors will be available once the linear motor manipulation is completed, but the resistor trimming requires high speed and high accuracy with motor movement and with data acquisition. The errors involved in resistor trimming are easily monitorable and upon system completion, will ensure a secure interface. All of the functions are included in one final operating environment that also includes an initialization sequence.

4.2 Component Description of Internal Laser Controls

For the Model 25 System, there are three lines through which communication takes place: BMB, BAC, and AC. Each line is unidirectional. The BMB and BAC lines
send information from the PC to the laser, while the AC line returns information from the laser to the PC.

All of the possible functions that the current Model 25 laser can accomplish are shown in Figure 4.1. Each function group has a unique address and consists of three functions. The individual functions are selected by pulsing the respective IOP line. The laser accumulator (AC lines) contains information that the laser has for the PC to read. The PC accumulator (BAC lines) has information that the PC has for the laser to read.

Knowing what laser function needs to be executed is only the beginning of the communication process. Figure 4.1 shows the function group in the left column and the tri-IOP function break-up in the right column. Figure 4.2 displays the PC accumulator usage, while Figure 4.3 shows the laser accumulator usage. Note that the meanings of some of the bits for certain accumulator usage are detailed in Figures 4.4 to 4.9.

The first step in accessing a function is to look it up in Figure 4.1 and record its address and its IOP number. If that function uses the PC accumulator, the required bits are needed from Figure 4.2 to do the function properly and must be obtained. If that function uses the laser accumulator, a note should be made about how the information is sent back to the PC through the laser accumulator so that it can be processed correctly. This information is available from Figure 4.3.

A complete operation to have the laser perform a particular function requires the following:

1. Load the address on the BMB line.
2. Load the PC accumulator on the BAC line.
3. Pulse the correct IOP line on the BAC line.

At this point, the laser knows what function is selected and acts on it.
<table>
<thead>
<tr>
<th>Function</th>
<th>Address</th>
<th>IOP1</th>
<th>IOP2</th>
<th>IOP4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programmable Time Delay</td>
<td>13</td>
<td>Skip on end of Delay</td>
<td>Load Delay, Start, Clear Flag</td>
<td>Send Probe Down</td>
</tr>
<tr>
<td>Comparator Flag/ Laser Control</td>
<td>15</td>
<td>Skip on &quot;0&quot; Crossing</td>
<td>Clear and Disable &quot;0&quot; Crossing Flag</td>
<td>Load Laser Control and Enable &quot;0&quot; Flag</td>
</tr>
<tr>
<td>Convotor Control/ Preamp Gain</td>
<td>16</td>
<td>Start Convortor &amp; Clr Overload Flag</td>
<td>Load Gain</td>
<td></td>
</tr>
<tr>
<td>Comparator Limit</td>
<td>16</td>
<td></td>
<td>Load Comparator</td>
<td></td>
</tr>
<tr>
<td>Convertor Readout</td>
<td>17</td>
<td>Skip on End Conversion</td>
<td>AC Clear</td>
<td>Read Convertor</td>
</tr>
<tr>
<td>Bridge Range</td>
<td>30</td>
<td></td>
<td>Load Range</td>
<td>Load Function</td>
</tr>
<tr>
<td>Generator Voltage</td>
<td>31</td>
<td></td>
<td>Load Generator Voltage</td>
<td></td>
</tr>
<tr>
<td>Detector Gain</td>
<td>31</td>
<td></td>
<td>Load Detector Gain</td>
<td></td>
</tr>
<tr>
<td>Conductance Standard</td>
<td>32</td>
<td>Skip on Strobe Done</td>
<td>Load Standard</td>
<td>Measurement System Strobe</td>
</tr>
<tr>
<td>Axis Select/Rest Test</td>
<td>40</td>
<td>Select the X-axis</td>
<td>Select the Y-axis</td>
<td>Skip on Rest</td>
</tr>
<tr>
<td>Table Position</td>
<td>41</td>
<td>Clear Counter</td>
<td>Read the MSB</td>
<td>Read the LSB</td>
</tr>
<tr>
<td>Distance to Go</td>
<td>42</td>
<td>Load the MSB</td>
<td>Load the LSB</td>
<td>Skip if No Count</td>
</tr>
<tr>
<td>Velocity &amp; Accel/ Home Latch</td>
<td>43</td>
<td>Load Velocity &amp; Acceleration</td>
<td>Clear the Home Latch</td>
<td>Skip on the Home Latch</td>
</tr>
<tr>
<td>Overtravel, Overshoot, Joystick</td>
<td>44</td>
<td>Skip on Interrupt</td>
<td>Clear Interrupt</td>
<td>Read Overtravel, Overshoot, Joystick</td>
</tr>
<tr>
<td>Overload Readout</td>
<td>52</td>
<td>Clear Overload Flag</td>
<td>Clear AC</td>
<td>Read Overload Status</td>
</tr>
<tr>
<td>Scanner, Position, and Channel</td>
<td>53</td>
<td>Skip if Ready</td>
<td>Load Position</td>
<td>Load Channel and Probe Number</td>
</tr>
<tr>
<td>Display and Sorting</td>
<td>56</td>
<td>Clear Manual Functions</td>
<td>Load Display and Sorting</td>
<td></td>
</tr>
<tr>
<td>Handler Information</td>
<td>56</td>
<td></td>
<td></td>
<td>Read the Manual Functions</td>
</tr>
</tbody>
</table>

Figure 4.1 Function Groups and IOP Break-up
<table>
<thead>
<tr>
<th>Function</th>
<th>Address</th>
<th>BAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programmable Time Delay</td>
<td>13</td>
<td>2048 1024 512 256 128 64 32 16 8 4 2 1</td>
</tr>
<tr>
<td>Comparator Flag/ Laser Control</td>
<td>15</td>
<td>X</td>
</tr>
<tr>
<td>Convertor Control/ Preamp Gain</td>
<td>16</td>
<td>X X X</td>
</tr>
<tr>
<td>Comparator Limit</td>
<td>16</td>
<td>MSB</td>
</tr>
<tr>
<td>Bridge Range/ Measurement System</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>Generator Voltage</td>
<td>31</td>
<td>8 4 2 1 0.5 0.25 0.125</td>
</tr>
<tr>
<td>Detector Gain</td>
<td>31</td>
<td>256 128 64 32 16 8 4 2 1 0.5 0.25 0.125</td>
</tr>
<tr>
<td>Conductance Standard</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>Distance to Go Upper</td>
<td>42</td>
<td>MSB</td>
</tr>
<tr>
<td>Lower</td>
<td>42</td>
<td></td>
</tr>
<tr>
<td>Velocity &amp; Accel/ Home Latch</td>
<td>43</td>
<td>64 32 16 8 4 2 1 0.5 0.25 16x</td>
</tr>
<tr>
<td>Scanner, Position, and Channel</td>
<td>53</td>
<td>MSB</td>
</tr>
<tr>
<td>Display/ Sorting</td>
<td>56</td>
<td>X X X X X X</td>
</tr>
</tbody>
</table>

Figure 4.2 PC Accumulator Usage
<table>
<thead>
<tr>
<th></th>
<th>17</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Convertor Readout</td>
<td></td>
<td>MSB</td>
<td></td>
<td></td>
<td></td>
<td>LSB</td>
</tr>
<tr>
<td>Table Position Upper</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
<td>MSB</td>
<td></td>
</tr>
<tr>
<td>Lower</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>LSB</td>
</tr>
<tr>
<td>Overload, Overshoot, and Joystick</td>
<td>44</td>
<td>+OT</td>
<td>-OT</td>
<td>OS</td>
<td>0 =</td>
<td>0 =</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+OT</td>
<td>-OT</td>
</tr>
<tr>
<td>Overload Readout</td>
<td>52</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Handler Information</td>
<td>56</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Figure 4.3 Laser Accumulator Usage
<table>
<thead>
<tr>
<th>AC Bit</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Limit Crossing Detector (0=normal (+to-), 1=reverse (-to+))</td>
</tr>
<tr>
<td>9</td>
<td>Laser On Unconditionally</td>
</tr>
<tr>
<td>10</td>
<td>Auxiliary Limit Crossing Selection (Else the Bridge Comparator is selected)</td>
</tr>
<tr>
<td>11</td>
<td>Laser On if Limit Crossing Flag is '1'</td>
</tr>
</tbody>
</table>

Figure 4.4 Comparator Flag / Laser Control

<table>
<thead>
<tr>
<th>Bridge Range</th>
<th>AC00</th>
<th>AC01</th>
<th>AC02</th>
<th>AC03</th>
<th>System Function</th>
<th>AC04</th>
<th>AC05</th>
<th>AC06</th>
<th>AC07</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Resistance</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2.</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Voltage</td>
<td>X</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3.</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.5 Bridge Range / Measuring System
<table>
<thead>
<tr>
<th>Channel</th>
<th>AC00</th>
<th>AC01</th>
<th>AC06 to AC11</th>
</tr>
</thead>
<tbody>
<tr>
<td>A (C1, P1)</td>
<td>0</td>
<td>1</td>
<td>Probe number (0-63)</td>
</tr>
<tr>
<td>B (C2, P2)</td>
<td>1</td>
<td>0</td>
<td>Probe number (0-63)</td>
</tr>
</tbody>
</table>

Figure 4.6 Scanner Position and Channel

<table>
<thead>
<tr>
<th>AC Bit</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Handler Advance</td>
</tr>
<tr>
<td>01</td>
<td>Initial Reject</td>
</tr>
<tr>
<td>02</td>
<td>Final Reject</td>
</tr>
<tr>
<td>10</td>
<td>Process Complete</td>
</tr>
<tr>
<td>11</td>
<td>In Process</td>
</tr>
</tbody>
</table>

Figure 4.7 Display and Sorting
<table>
<thead>
<tr>
<th>AC Bit</th>
<th>Overload Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Integrating Converter Overload (+)</td>
</tr>
<tr>
<td>01</td>
<td>Peak Overload (Any one or all bits 4-11)</td>
</tr>
<tr>
<td>04</td>
<td>Guard Overload (+)</td>
</tr>
<tr>
<td>05</td>
<td>Guard Overload (-)</td>
</tr>
<tr>
<td>06</td>
<td>Detector Preamp (+)</td>
</tr>
<tr>
<td>07</td>
<td>Detector Preamp (-)</td>
</tr>
<tr>
<td>08</td>
<td>Bridge (+)</td>
</tr>
<tr>
<td>09</td>
<td>Bridge (-)</td>
</tr>
<tr>
<td>10</td>
<td>Bridge Power (+)</td>
</tr>
<tr>
<td>11</td>
<td>Bridge Power (-)</td>
</tr>
</tbody>
</table>

Figure 4.8 Overload Readout

<table>
<thead>
<tr>
<th>AC Bit</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Emergency Stop</td>
</tr>
<tr>
<td>01</td>
<td>Single Step</td>
</tr>
<tr>
<td>02</td>
<td>Load Position</td>
</tr>
<tr>
<td>03</td>
<td>Index</td>
</tr>
<tr>
<td>04</td>
<td>Start Sequence</td>
</tr>
<tr>
<td>11</td>
<td>Resistance Only</td>
</tr>
</tbody>
</table>

Figure 4.9 Handler Information
A complete operation to read the status of a particular register requires the following:

1. Load the address for that register on the BMB line.
2. Pull the correct IOP line high on the BAC line.
3. Read in the laser accumulator data from the AC line.
4. Drop the IOP line to its ground state.

4.3 Prototype Description

Before the operating environment can be constructed, a series of test codes must be written to confirm the hardware construction, to verify laser control sequences, and to build a base for more complex function accessing. A detailed description follows on the accomplishment of each of the three benchmarks: laser system register control, linear motor manipulation, and resistor trimming.

4.3.1 Laser System Register Control

Controlling the laser system's registers is a basic step, but a step that requires that the hardware output functions are operating as well as an understanding of the laser function encoding process. The goal of this benchmark is to turn on and off the laser's control panel lights. In particular, the "In Process" and the "Process Complete" lights are used. From Figure 4.4, the address for loading data into the display-and-sorting is an octal 56 with a pulse on IOP 2. From Figure 4.7, the bit responsible for the "In Process" light is bit 11, and the "Process Complete" light is bit 10. A value of 1 in this register turns on the light, and a value of 0 turns off the light. Toggling between the two lights
was accomplished by using the ENDPRO and STARTPRO programs. ENDPRO turned on the "Process Complete" light and turned off the "In Process" light. STARTPRO turned on the "In Process" light and turned off the "Process Complete" light. The source code for ENDPRO and STARTPRO are located in Appendix I and II, respectively.

4.3.2 Linear Motor Manipulation with Joystick Control

Linear motor manipulation is accomplished through dialogue between the laser and the interface. Along with sending codes to the laser, the hardware and software must accept and decode data from the laser. The test program will involve using the joystick on the laser control panel to signal a direction and speed. The interface will recognize this request and send the appropriate function calls back to the laser. When the joystick is in its upright position, i.e. when no direction and speed are selected, the interface reads the current laser position. With this benchmark completed, the stability of the hardware and the process in which laser function encoding and decoding takes place are completely tested and verified. Information that the interface sends to the laser includes an axis select (octal address 40 and IOP 1 or 2), a velocity/acceleration setting (octal address 43 and IOP 1), the lower 12 bits of the distance to go (octal address 42 and IOP 2), and the upper 4 bits of the distance to go (octal address 42 and IOP 1). The data that the interface will receive from the laser includes a reading of the lowest 12 bits of the table position (octal address 41 and IOP 4), a reading of the highest 4 bits of the table position (octal address 41 and IOP 2), and a reading of the joystick position (octal address 44 and IOP 4).

The JOYMOVE program successfully completes the linear motor movement using the joystick. The program continuously polls both the x-axis and the y-axis
joystick registers to check for any movement. The laser treats the joystick as two separate joysticks, one for each direction. Therefore, there are two joystick registers. The joystick, itself, also has the capability to sense if it is at an angle greater than 45 degrees from its normal upright position. If it is at an angle greater than 45 degrees, it sets the appropriate flag in the appropriate joystick register where that the angle occurs. The steep angle of the joystick is used to reflect a faster speed of travel. When the joystick is in its upright position, the computer reads the current position of the laser. When the joystick is moved, the respective register(s) will reflect whether the joystick is angled in a positive or negative direction and whether that angle is greater than 45 degrees from its normal upright position. The computer recognizes the joystick movement on the next polling. The joystick registers are then decoded. The computer immediately encodes and sends a signal to the laser to move in the appropriate directions. On a slow move, the computer sends a velocity of 2 decimeters per second, while on a fast move it sends a velocity of 6 decimeters per second.

The table position that the laser sends back to the computer is incorrect until the linear motors are both moved across their origins. This is a laser initialization process, not an interface problem. At this point, this inconsistency is of no concern because the based on distances, not positions. Eventually, the initialization of the laser will be required for proper laser control system operation. In the final operating environment, an initialization sequence includes sending both motors to their origins.
4.3.3 Resistance Measurements

The final task only accounts for those functions that have not already been utilized. The remaining functions are those involved with resistor measurement. The only major difference between resistor measurement and resistor trimming is the continuous measurements with the laser on during a resistor trim. There are also other minor differences due to having a better first approximation of the nominal value of the unknown resistor with the resistor trim. This saves time during a trim and allows the latest measurement to better reflect the actual value of the resistor as it is being trimmed. It is sufficient to show for this task that the measurement process is properly accessed.

The laser uses a Wheatstone Bridge to calculate the resistance difference from a configurable comparison value. The measurement system in the laser returns the difference between the comparison value and the unknown resistance value. The first step in a measurement is to define a comparison value. In the measurement of an unknown resistance value, the comparison value will begin at the relatively low value of 500 ohms. The feedback that the computer receives from the laser measurement system after a measure includes a flag that represents if the comparison value is out of the measurable range. While the comparison value is out of range, the comparison value is multiplied by 2. This multiplication factor is small enough that it will not step the value over the measurable range of an unknown resistance value. When the value steps in range, it is stepped back out and then in again at a rate of 1.25 until the value is within the detectable difference range. The detectable difference range is only about half the size of the measurable range. Once the comparison value is within the detectable difference range, the computer decodes the difference and calculates the new comparison value. A new difference is recorded, and the comparison value is again recalculated. This
recalculation is required because it is possible to increase the gain when the comparison value is close to the actual value in order to achieve a much greater resolution. It only takes the computer two or three recalculation periods before the tightest resolution is achieved.

4.4 Principles of Operation

This section goes in-depth into the operating environment and the accessing of all available laser functions from the original code. For more information about the conceptualization of the environment, the illustrating flowcharts are provided in Appendix V and Appendix VIII. A by-product of this operating environment is a laser system user's manual that is included in Appendix X.

The original code was written and compiled in the Microsoft C 5.1 environment. There are two executable files: LASER and FILEMAKE. Some of the functions used in both LASER and FILEMAKE are from a separate library from the C TOOLS PLUS software package. In order to keep the terminology serial, a description of words to be used follows. A program is a C code file (with a .C extension) like LASER.C and SAVING.C. An executable program is a file with an .EXE extension that can be run. These are the LASER.EXE and FILEMAKE.EXE files. A function is a name inside a program that represents a set of commands and/or other functions. A module is the set of commands and/or functions that defines a function and is followed by (). In other words, a function is defined by a module that contains commands and/or other functions.

This code has been streamlined to reduce the size and confusion caused by hefty code. A difficulty that arises from this is that a lot of the functions are called by several other modules that will each do something different with the information provided.
Sometimes the reason that a module will perform a process that makes no sense with respect to a certain way that it is called makes perfect sense with respect to a different way. In the particular case of the 'set_up_a_loop()' module, the module is itself recursive. This means that under certain circumstances, the module will call itself. The header at the beginning of the module lists the users and modules called.

### 4.4.1 LASER Program Operation

The LASER program is the main file that processes the complete set of all laser functions. The LASER program is made up of five separate files: LASER.C, SAVING.C, NEWSTUFF.C, MEASINCD.C, and MOVEINCD.C. These are put together by the program list LASER.MAK. The program list has the proper library calls, including MT5_M5M.LIB from C TOOLS PLUS that defines certain external functions. For the LASER program, there is also a set of include files to allow for inter-program function calling. These include files are BWINDOW.H, GLOBLVAR.H, EXTRNVAR.H, MOVEINCD.H, MEASINCD.H, NEWSTUFF.H, LASER.H, SAVEINCD.H. The source code listings for LASER.C, SAVING.C, NEWSTUFF.C, MEASINCD.C, and MOVEINCD.C are included in Appendix VI, while the listing of the include files for LASER are in Appendix VII.

As the names of the four programs of the compiled LASER program suggest, each program contains the majority of a certain type of operations. The LASER.C program mainly coordinates the entire effort with the Main Menu as well as the single, individual operations such as the scribe, the circle cut, and the move option. The SAVING.C program contains all of the Pattern Cut Menu with pattern sequence options like creating, editing, and running patterns. The NEWSTUFF.C program incorporates all
of the resistor trimming menu systems and operations involved in the break-up of the a complete measurement system operation into sub-modules. The MOVEINCD.C program contains all of the code sequences that translates the intention to move, to turn laser on, etc. into number sequences that the laser understands. The MEASINCD.C program translates all of the resistor trimming intentions into correct number sequences for the laser. By subdividing the entire system and compiling it with a program list, any program or set of programs can be edited without having to re-compile the entire system. This saves time during testing, compiling, and future additions.

The executable program begins in the LASER program. The first function that LASER operates initializes all the laser systems registers as well as sends the laser to its home position. Once this initialization has been achieved, the laser system will not need to be initialized again until after the entire system has been shut down. At this point in the LASER program, the operation of the system is in the hands of the user. The main menu, Figure 4.10, is displayed, and the user is prompted to select an option. All of the screen display functions are from the C TOOL PLUS software package. Depending on the option chosen, the system will operate the particular sequence that may encompass all of the four programs. Listed below are the paths in which each option will go.

4.4.1.1 Single Operations

4.4.1.1.1 Move Distance

This option will perform a move of the users choice; with the joystick or with a distance entered by the user. The computer will then prompt for a further move. When all moving has been completed, the main menu is redisplayed.
Laser Operating System

What would you like to do?

1. Move distance
2. Scribe
3. Pattern cut
4. Cut a circle
5. Index
6. Resistor trimming
7. Hardware test
8. Test measurement system
9. Exit

Option:

System Status

Current Position: 0, 0

Figure 4.10 The Main Menu
4.4.1.2 Scribe

In a scribe, the user will define the endpoints by using either the joystick or distances, as well as define the cutting speed and the number of repetitions that the laser will make over the object. Once all of the scribes have been completed, the main menu is redisplayed.

4.4.1.3 Cut a Circle

This option takes advantage of a circle cutting module that is used for pattern cutting, but does not in itself cut the circle. It is responsible for getting the necessary information in order to run the other module. The two modules are not together because there are other modules that also need circles to be cut, but already have the necessary information that this main menu option does not have. As discussed before, the actual circle cutting module is one of modules that is called by many modules with different needs. The path to this cutting module requires that the user enter the necessary information as opposed to already having the information in a file. The necessary information consists of finding the origin of the circle, defining the radius, and determining whether or not the circle needs to be saved. After all of the information has been entered, the circle is actually cut.
4.4.1.4 Index

Indexing means the laser is sent back to its internal (0,0) position. This is accomplished by reading the internal position and moving the opposite directional distance.

4.4.1.5 Hardware Test

This is not a test of the laser hardware but of the interface hardware between the laser and the computer. This test is completed by simply indexing laser. If the laser will not index, something is wrong with the interface.

4.4.1.2 Pattern Cut Menu Options

This option has its own menu, the pattern cut menu. The pattern cut menu appears on the screen as it is shown in Figure 4.11. Depending on the users choice, one of the following will be done.

4.4.1.2.1 Create a File

After the user enters the name of the file to be created, the computer will check if that file already exists. If the file does already exist, the user is alerted, and the pattern cut menu is redisplayed. If the file does not already exist, the user is prompted for the velocity at which the laser will travel when the laser is ON. With this information stored, the create-a-file menu appears (see Figure 4.12). The user will enter the pattern
Laser Operating System

What would you like to do?

1. Create a file
2. Analyze / print a file
3. Edit a file
4. Run a file
5. Main menu

Option:

System Status

Current Position: 0, 0

Figure 4.11 The Pattern Cut Menu
Laser Operating System

What would you like to do?

1. Move to a position
2. Distance move
3. Laser On
4. Laser off
5. Circle cut
6. Start a loop
7. Repeat a loop
8. Exit

Option:

Systerm Status

Current Position: 0, 0

Figure 4.12 The Create a File Menu
sequence that he wants. The computer treats each single operation as an action with possible parameters. Each action is given a number. Depending on the user's next step, the computer stores that particular action number in the file and then prompts for the parameters of that step if any exist. For instance, the "distance move" will have two parameters, x and y distance, while the "laser on" will have no parameters. Since the parameters are obtained from the user as ASCII characters, the parameters must be transformed into numbers. The parameters are stored in the file right after the action number. Once the user has completed entering his sequence and selects the exit option, an "end of file" action number is saved in the file, and the file is closed. Control goes back to the pattern cut menu.

4.4.1.2.2 Analyze / Print a File

After the user enters the name of the file to be analyzed, the computer will check if that file already exists. If the file does not already exist, the user is alerted and the pattern cut menu is redisplayed. If the file does exist, the computer will ask the user if the file should be printed and analyzed. If the user wants the file printed, the user must turn on the printer. The computer will read the file line by line and keep track of each of the individual actions and their parameters. The circles, laser ons, laser offs, loops started, and loops completed are counted, and the final laser position would be are saved. If the file is to be printed, the computer reads each line and it sends a line to the printer that displays the action, its parameters, the instantaneous position of the laser, and the velocity of the laser. Nested loops are handled as a general case of looping with a single loop as the special case. The program keeps accurate tabs on the effects of loops by storing the number of times that each loop is to be completed along with the difference in
position between the start of the loop and the end of one repetition of that loop. This way, the number of times that each counted action is performed, as well as the final position, can be multiplied by the proper factor to represent all iterations of the nested loops. All of the final counts and the final position are displayed on the screen and sent to the printer if the user wanted the file sent there. Control is then passed back to the pattern cut menu.

4.4.1.2.3 Edit a File

After the user enters the name of the file to be edited, the computer will check if that file already exists. If the file does not already exist, the user is alerted, and the pattern cut menu is redisplayed. If the file does exist, the computer will create a back up of the file. After creating or overwriting a file with the same filename but with an '.old' extension, the computer copies each action, one at a time, with their parameters to the backup. When this task is completed, the backup is used as a source file and editing (inserting, deleting, or changing) is performed by confirming each action as the backup is copied back to the filename with the '.new' extension. Starting with the cutting velocity, each action is displayed one at a time on the screen, and the user is asked if that line is correct. If the user says that the line is correct, the next action will be displayed. If the user does not want the particular action, whether it is because it should not be there, something should be before it, or that the parameters are incorrect, the computer will prompt for the wanted changes. The first question asked is if other actions (commands) need to be inserted. If they do, the create-a-file menu (Figure 4.4.1.2.1) is displayed, and the user can enter as many other actions as needed that will precede the action that was in question. These other actions are stored in the '.new' file. Once this is done or if nothing
was to be inserted, the next question is to decide whether or not to delete the action in question. If the user responds to the deletion, the action is not transferred back to the '.new' extension, and the next action is displayed. If the user keeps the action and if the action has parameters, the user is asked if the parameter need to be changed. If so, the computer will prompt for the new parameters. Otherwise, the next action is displayed. Another option from the original prompt is 'q' for quit. This option will transfer the remaining actions from the backup file to the '.new' file.

4.4.1.2.4 Run a file

After the user enters the name of the file to be run, the computer will check if that file already exists. If the file does not already exist, the user is alerted and the pattern cut menu is redisplayed. If the file does exist, the computer processes the file by reading one action (with its parameters) and sending the appropriate command to the laser. If a loop is encountered, some fancy processing is required. This is accomplished by setting up an array that can hold all of the actions and their parameters that make up that loop. Nested loops are taken care of by calling a module recursively. This only adds another dimension to the array while allowing for multiple loops. This option processes the actions in the same way as editing is performed but it gets the actions and their parameters from the array as opposed to straight from the file. When the file has been completely processed, the final position as well as the number of circles, laser ons, laser offs, and loops are displayed. After this action is completed, the pattern cut menu is displayed.
4.4.1.3 Resistor Trimming

This option has its own menu, the resistor trimming menu. The resistor trimming menu appears on the screen as it is shown in Figure 4.13. All of the options under this menu except for the changing of the trimming defaults (Figure 4.16) require resistor probes to be in place on the laser and entered into the computer. Each option will request the probe numbers if no probes were previously entered; otherwise, a confirmation of the probes numbers will occur. Since there is no way that the computer can compare probe location with resistor location, the computer assumes that the user knows which resistor corresponds to the selected probes. The user is also responsible for making sure that the power supply is operating and that there are no open interlocks when a trim is to take place. If there is any problem during a laser trim or if the trim hit its final destination without trimming the resistor within tolerance, the user can strike the STOP button on the laser control to regain computer control and to turn off the laser.

The actual laser trim is a complicated process by which the computer carefully monitors the resistance value of a resistor as it is being cut. When the resistor is within the tolerance level, the laser is turned off and the trimming is completed. Laser_trim() is the module that coordinates this effort. When the user is ready, the laser is turned on and moves into the resistor. The first test is for a resistance that with large enough amplification is within the measuring range. When this is true, the computer compares the actual resistor value with the resistance value to be obtained. When the difference between the numbers is within the smallest tolerance, the trimming is completed.

Depending on the users choice, one of the functions described in sections 4.4.1.3.1 to 4.4.1.3.7 will be processed.
What would you like to do?

1. Get current resistance
2. Straight-cut trim
3. Double-cut trim
4. L-cut trim
5. Multiple-line-cut trim (serpentine)
6. Test routines for the measurement system
7. Imbedded resistor (not in use)
8. Change trimming defaults
9. Exit

Option:

Current Position: 0, 0

Figure 4.13 The Resistor Trimming Menu
4.4.1.3.1 Resistance Value Acquisition

After requesting or confirming which probes straddle the resistor, the computer calculates the resistance across the selected probes and displays the value in the bottom window.

4.4.1.3.2 Single Straight Cut

A straight cut is performed by acquiring a laser head position on each side of the resistor from the user. Once these points are entered by the user and the user confirms that everything is ready, the laser will start trimming from the first point and will move in the direction of the second point. When the resistance is within the accepted tolerance range, the laser will stop trimming and the user regains control.

4.4.1.3.3 Double Cut

For a double cut, three laser head positions are required. The first will be the starting position. The second position represents the direction in which the first cut will take place. The third position allows the computer to calculate the area of the resistor that is available for the starting and ending positions for the second cut. The laser trimming is accomplished by starting to trim from the first position towards the second position. When the resistance is within the tolerance of a percentage of the final resistance, the laser is turned off and the moves to the second starting position. This second starting position is based on a percentage of the available resistor area. The percentage of the final resistance and the percentage of available area that the computer
uses are parameters that can be changed from the change trimming defaults option (see Figure 4.16). The laser then starts to trim from this position in the same direction as the first cut until the resistance is within the tolerance of the final resistance value.

4.4.1.3.4 L Cut

As with a double cut, the L cut also requires three laser head positions. The first will be the starting position. The second position represents the direction in which the first cut will take place. The third position allows the computer to calculate the direction in which the laser will turn for the second cut. The laser trimming is accomplished by starting to trim from the first position toward the second position. When the resistance value is within the tolerance of a percentage of the final resistance value, the laser direction changes to that of the direction between the second position and the third position. The percentage of the final resistance that the computer uses is a parameter that can be changed from the change trimming defaults option (see Figure 4.16). The laser trims along this diagonal direction until the resistance is within the tolerance of the final resistance value.

4.4.1.3.5 Multiple Straight Cuts

A multiple straight cut follows the same principles as the double cut except that there can be more than two cuts and that the cuts alternate sides. The distance between cuts and the intermediate resistance values are calculated by the computer based on the number of cuts. The number of cuts is a trimming parameter that can be changed.
4.4.1.3.6 Measurement Test Routines

There are a lot of parts that make up the measurement system. For the most part, they are not completely independent of each other and can be integrated together. The possibility also exists that individual control might be wanted. All of these options are included in the measurement system test. The available options for the measurement system include setting the resistor value to test against, the generator voltage, the detector gain, and the pre-amp gain. Although the comparator value and a delay time are also listed, it is not taken into account in the program. Figure 4.14 displays the main measurement test menu with options to obtain feedback from the laser measurement system. The option to change the parameters is displayed in the measurement test value selection menu as seen in Figure 4.15.

4.4.1.3.7 Trimming Defaults

The trimming defaults menu, Figure 4.16, allows the user to change parameters that effect the different trims. These parameters include the percentages for L and double cut turns, the percentage of distance for the second double cut, and the number of multiple cuts.

4.4.2 FILEMAKE Program Operations

The FILEMAKE program is a subset of the LASER program that only does pattern sequence functions: creating, saving, editing, printing, and a test run. The FILEMAKE program is made up of only one program: FILEMAKE.C. There is a
1. Define test values
   - A Probe: 999
   - B Probe: 999
   - Test Resistance Value: 0
   - Range: 0
   - Standard: 0.000000000000
   - Generator Voltage: +0.000
   - Detectors Voltage: 0.000
   - Pre-amp gain (1 or 8): 0
   - Comparator value: 0
   - Delay: 0

2. Strobe the measurement system
3. Clear the overload flag
4. Find the unknown resistance
   9. Return to Main Menu

Option:

<table>
<thead>
<tr>
<th>Status</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convertor</td>
<td>0</td>
</tr>
<tr>
<td>End Delay</td>
<td>Ready</td>
</tr>
<tr>
<td>Overload</td>
<td>0</td>
</tr>
<tr>
<td>Ready Flag</td>
<td>Ready</td>
</tr>
<tr>
<td>End Conversion</td>
<td>Ready</td>
</tr>
<tr>
<td>Strobe Done</td>
<td>Ready</td>
</tr>
</tbody>
</table>

Figure 4.14 The Main Measurement Test Menu
Define test values
1. Change A Probe
2. Change B Probe
3. Change Test Resistance Value
   Range
   Standard
4. Change Generator Voltage
5. Change Detectors Voltage
6. Change Pre-amp gain (1 or 0)
7. Change Comparator value
8. Change Delay
   Strobe the measurement system
   Clear the overload flag
   Find the unknown resistance
9. Exit test value definitions

Option:

<table>
<thead>
<tr>
<th>System Status</th>
<th>Ready</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convertor 0</td>
<td>Ready</td>
</tr>
<tr>
<td>Overload 0</td>
<td>Ready</td>
</tr>
<tr>
<td>End Conversion: Ready</td>
<td>Ready</td>
</tr>
<tr>
<td>End Delay</td>
<td>Ready</td>
</tr>
<tr>
<td>Ready Flag</td>
<td>Ready</td>
</tr>
<tr>
<td>Strobe Done</td>
<td>Ready</td>
</tr>
</tbody>
</table>

Figure 4.15 The Measurement Test Value Selection Menu
**Laser Operating System**

1. Trim speed  
   \[ 0.125000 \]
2. Double-cut or I-cut turn percentage  
   \[ 80 \]
3. Second cut (Double-cut) percentage distance  
   \[ 20 \]
4. Perpendicular cuts - abort distance  
   \[ 50 \]
5. Adjacent (parallel) cuts - abort distance  
   \[ 50 \]
6. Number of serpentine cuts  
   \[ 5 \]
7. Restore defaults

9. Return to Laser Menu

   Option:

---

**System Status**

Current position: \( 0, 0 \)

---

Figure 4.16 The Trimming Defaults Menu
program list, FILEMAKE.MAK, that supplements this file in order to include the appropriate libraries, including MT5_M5M.LIB from C TOOLS PLUS. FILEMAKE requires the include files BWINDOW.H and GLOBLVAR.H. These include files for FILEMAKE are the same as for LASER and can, therefore, be found with the LASER include listings in Appendix VII. The source code listing for FILEMAKE.C is located in Appendix VIII.

The FILEMAKE executable program is made up of only one program, FILEMAKE.C. This program is an offshoot of SAVING.C, but has no laser oriented functions. FILEMAKE.C only contains information and abilities to create, edit, and screen run a pattern file. This program is almost an exact replica of SAVING.C, except that nothing is ever sent to or received from the laser. The operation of FILEMAKE.C is understood from the pattern cut menu options.

4.5 Conclusion

Once the prototypes were constructed and tested, the operating environment was constructed by type of operation. When the LASER program is run, the laser is initialized so that the laser control system will communicate properly with the PC. The main menu provides access to individual operations and to the pattern cut and resistor trimming menus. The pattern cut menu allows the user to create, edit, analyze, and run patterns. The resistor trimming menu provides the user with access to the various types of resistor trims as well as to a measurement testing menu. From the measurement testing menu, the laser's internal measurement system can be manipulated. A second program, FILEMAKE, allows users access to create, edit, and analyze patterns without having to be at the laser.
Chapter 5

Resistor Trimming and Aging Experimental Realization

5.1 Overview

This chapter is broken up into four sections. Section 5.1 discusses the testing objectives, the configuration and setup, and the preliminary testing results. By the end of section 5.1, an understanding of the process in which the resistors were trimmed and thermally aged will be known. The analysis of the results compared by the type of cut and the laser power applied is discussed in section 5.2. Section 5.3 evaluates the results by the type of cut and the resistor geometry. The results of the thermal aging are compiled in section 5.4.

5.1.1 Objectives

The objectives of the testing were to evaluate the types of cuts at two different levels. The first set of tests involved the best type of cut for the particular resistor geometry and the laser power applied. These same trimmed resistors were then tested for deterioration due to thermal aging. The resistor paste used was DuPont BIROX Series-1900 Paste number 1939.
5.1.2 Testing Configuration

The testing incorporated six sets of cuts and one set of control resistors. Four different system power configurations for a total of 24 groups of cuts and two groups of control resistors were used in the experimentation. Each group was made up of 9 different resistor geometries which consisted of the combinations of three different aspect ratios and three different surface areas. The six cuts used were a straight cut, an L cut with the elbow at 60%, an L cut with the elbow at 70%, an L cut with the elbow at 80%, a double cut with the shift at 70%, and a 4 cut serpentine cut. The four system power configurations included the combinations of a total output power of either 1.5 Watts or 2.0 Watts, and of a Q-switching frequency of either 3kHz or 6kHz. An attempt was made to use 10kHz instead of 6kHz, but the 10kHz beam could not completely burn through the resistor. The configuration of the substrate is shown in Figure 5.1. The resistors were trimmed to 1.4 times their printed resistance. This means that this evaluation is valid for resistors originally designed for approximately 70% of its intended value.

5.1.3 Thermal Aging Procedure

Each set of resistors underwent a thermal aging process that involved ramping up to 175 degrees C and then ramping back down to room temperature. A Fischer Programmable Ashing Furnace Model 497 was used. This furnace has the capability of ramping up to three temperatures. The furnace has a hold time at each temperature level before it continues to ramp to the next temperature. When the value set as the maximum temperature is reached, the temperature is held constant for a pre-programmed hold time. When the hold time is over, the furnace ramps back down to three temperatures.
Figure 5.1 The Substrate
For this thesis, all of the ramps were set at 10 degrees per minute and there were no intermediate hold times. In ramping up, the first temperature was 59 degrees C, the second temperature was 118 degrees C, and the third (maximum) temperature was 175 degrees C. This temperature was held for one hour. During the down ramping, the first temperature was 118 degrees C, the second temperature was 59 degrees C, and the last temperature was 32 degrees C (room temperature).

5.1.4 Testing Results

There were very little problems encountered during the trimming process. Of the 216 resistors that were trimmed, only four resistors were trimmed to an infinite resistance. Three of these were due to software data manipulation errors. The software was immediately fixed and the resistors were re-trimmed without any problems. The fourth resistor was due to a user error from the selection of a resistor other than the one that the probes were across. There were no data transfer errors. In general, all of the cuts were trimmed to a tolerance of less than 4%. The average tolerance for all of the trimmed resistors was about 0.9%.

5.2 Analysis of Types of Cut, System Power, and Resistor Tolerances

5.2.1 The Single Straight Cut

Since the straight cut involves only one cut, the results for this cut are based on one set of results. All other cuts can be reflected back to the straight cut since all of the cuts first trim into the resistor. Figure 5.2 shows the results of the straight cut. From a
Figure 5.2
The Final Tolerances for the Straight Cut
Frequency and Power

Figure 5.3
The Breakup of Final Tolerances for the Straight Cut
close inspection, both of the 3kHz trims caused less than optimal tolerances for almost half of the resistors. From Figure 5.3, this difference is noticed in both the average and the maximum tolerances. There is no other noticeable correlation between the tolerances and the applied power and frequency.

5.2.2 The L Cut with the Elbow at 60%

The L cut is made up of an initial straight cut with a 90 degree turn and a second, perpendicular cut. Figures 5.4 and 5.5 present the results of the first straight cut. There are no outstanding features about these cuts, although it is expected that the results would match those found from the single straight cut. The results from this first cut are in no way uniform or very accurate. The results from the second, perpendicular cut are much more uniform. The results from the first three resistors for all of the power and frequency combinations were not obtainable because this type of cut was ineffective. This will be explained later by the resistor geometry. Even though the obtained tolerances for the second cut were not all below 1%, the spread of the maximum and minimum values were close to the average tolerance. This is seen from Figures 5.6 and 5.7.

5.2.3 The L Cut with the Elbow at 70%

The first cut for this L cut produced relatively uniform distributions without any of the power and frequency combinations performing differently from the others. The distribution shown in Figure 5.8 shows this more than the general data in Figure 5.9. The average values for this first cut are quite close. The 3kHz trims yielded the noticeably
Figure 5.4
The Tolerances for the First Cut
of a L Cut with the Elbow at 60%
Figure 5.5
The Breakup of Tolerances for the First Cut of a L Cut with the Elbow at 60%
Figure 5.6
The Final Tolerances
of a L Cut with the Elbow at 60%
Frequency and Power

Figure 5.7
The Breakup of the Final Tolerances of a L Cut with the Elbow at 60%
Figure 5.8
The Tolerances for the First Cut of a L Cut with the Elbow at 70%
Frequency and Power

Figure 5.9
The Breakup of Tolerances for the First Cut of a L Cut with the Elbow at 70%
Figure 5.10
The Final Tolerances of a L Cut with the Elbow at 70%
Figure 5.11
The Breakup of the Final Tolerances of a L Cut with the Elbow at 70%
worse maximum tolerance values. The final results have a better tolerance than the first cut but, the uniformity was lost with resistors 8 and 9. Figure 5.10 shows how the power and frequency behaved completely differently. The average tolerance in Figure 5.11 remained relatively constant as was seen from the first cut. The worse maximum values and slightly higher averages were seen with the 2.0 Watt cuts as compared with the 3kHz cuts from before.

5.2.4 The L Cut with the Elbow at 80%

Aside from the close average tolerances, as seen in Figure 5.13, there was no continuity between the trimming performances. Figure 5.12 displays how the tolerance varies differently between the resistors. An interesting note is that the 3kHz cuts follow each other relatively closely across the resistors. This is also true for the two 5kHz groups of trims. The final tolerance followed closely with the results from the first cut. Once again, the tolerances showed no correlation other than that the 3kHz trims vary radically for the different resistors. Figures 5.14 and 5.15 show this in detail.

5.2.5 The Double Cut

The performances of the 3kHz trims for the first cut are seen in Figure 5.15 to be quite good. For the double cut, the 5kHz trims behave most nonuniformly. This is seen in both the higher average and the maximum values in Figure 5.17. This pattern, though, did not continue with the second straight cut. The results for the final tolerance have the best average tolerances. Even though the 3kHz cuts had higher maximum tolerances, all of the tolerances were below 1.6%, and all of the averages were below 0.8%.
Figure 5.12
The Tolerances for the First Cut of a L Cut with the Elbow at 80%
Frequency and Power

Figure 5.13
The Breakup of Tolerances for the First Cut of a L Cut with the Elbow at 80%
Figure 5.14
The Final Tolerances of a L Cut with the Elbow at 80%
Frequency and Power

Figure 5.15
The Breakup of the Final Tolerances
of a L Cut with the Elbow at 80%
Figure 5.16
The Tolerances for the First Cut of a Double Cut
Frequency and Power

Figure 5.17
The Breakup of Tolerances for the First Cut of a Double Cut
Figure 5.18
The Final Tolerances of a Double Cut
Figure 5.19
The Breakup of the Final Tolerances of a Double Cut
5.2.6 The Serpentine Cut

This serpentine cut was performed with four cuts. This is similar to doing two of the previous double cuts except that the cuts alternate sides of the resistor. The results of the first three cuts are displayed in Figures 5.20 to 5.25. The results are good for all of the cuts except for a few particular results that are inconsistent with the rest of the data. From Figures 5.26 and 5.27, it is evident that trims were quite accurate, but that some very odd inconsistencies occurred with resistors 5 and 7. Other than the outstanding data, the uniformity is evident.

5.2.7 Discussion

The most outstanding common factors among all of the cuts are with the average tolerance. All of the averages were less than 2% and they were all relatively even for each type of cut. The 3kHz cuts have the largest maximum value throughout the final tolerances. The uniformity of power and frequency combinations are seen strongly with the L Cuts. It is interesting to note, though, that the tightest tolerances were found particularly with the L Cuts with the elbow at 60%, the Double Cuts, and the Serpentine Cuts. This is discluding the invalid data for the Serpentine Cuts. The 5kHz at 1.5 Watt trims are the only trims that consistently had low average tolerances and low maximum tolerances. The lowest average tolerances were obtained from the Double Cuts and the Serpentine Cuts.

The most powerful of the cuts was the 3kHz at 2.0 Watt cuts. The least powerful was the 5kHz at 1.5 Watt cuts. It is surprising that the strongest of the cuts would not have prevailed more in the intricate cuts as it might have been expected to perform. A
Resistor Number

- 3kHz at 1.5 Watts
- 3kHz at 2.0 Watts
- 5kHz at 1.5 Watts
- 5kHz at 2.0 Watts

Figure 5.20
The Tolerances for the First Cut of a Serpentine Cut
Figure 5.21
The Breakup of Tolerances for the First Cut of a Serpentine Cut
Figure 5.22
The Tolerances for the Second Cut of a Serpentine Cut
Frequency and Power

-■- Average
-△- Maximum
○ Minimum

Figure 5.23
The Breakup of Tolerances for the Second Cut of a Serpentine Cut
Figure 5.24
The Tolerances for the Third Cut of a Serpentine Cut
Frequency and Power

Figure 5.25
The Breakup of Tolerances for the Third Cut
of a Serpentine Cut
Figure 5.26
The Final Tolerances of a Serpentine Cut
Frequency and Power

Figure 5.27
The Breakup of the Final Tolerances of a Serpentine Cut
possible explanation for this effect is that the stronger cuts effectively heated the resistor so that the measured resistance was not the normal operating resistance. When the trim was completed and the resistor cooled down, the resistance increased away from the original desired resistance. Another surprising result is that the best uniformity and average tolerances were obtained from the Double Cuts and the Serpentine Cuts. It was originally assumed that the L Cuts would have the best performance. Since the Double Cuts and the Serpentine Cuts can be evaluated as a series of straight cuts, it could be hypothesized that the quality would be the same as the Single Straight Cut. These multiple straight cuts performed better than the L Cuts because the percentage of resistance that was trimmed on the latter straight cuts was not as large as a Single Straight Cut. They also did not require the laser to trim as deep into the resistor. The L Cut, on the other hand, was already deep in the resistor when it turned to do its perpendicular cut and was trimming the resistor in an area that the resistance changed fast with respect to the measuring speed.

5.3 Analysis of Resistor Geometry, Types of Cuts, and Resistor Tolerances

The following evaluations are based on the aspect ratios of the trimmed resistors as seen in Figure 5.28. The correlation between the resistor’s number and the resistor is shown in Figure 5.29. Resistors 1 to 3 have a low aspect ratio of 1:3, resistors 4 to 6 are square with an aspect ratio of 1:1, and resistors 7 to 9 have a high aspect ratio of 5:1.
<table>
<thead>
<tr>
<th>Surface Area</th>
<th>Aspect Ratio</th>
<th>5:1</th>
<th>1:1</th>
<th>1:3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large</td>
<td>R7</td>
<td>R5</td>
<td>R3</td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>R9</td>
<td>R6</td>
<td>R2</td>
<td></td>
</tr>
<tr>
<td>Small</td>
<td>R8</td>
<td>R4</td>
<td>R1</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.28 The Resistor Dimensions
Figure 5.29 The Numbering Format for the Resistors
5.3.1 The Single Straight Cut

The Straight Cut worked most effectively with the low aspect ratio resistors (R1 to R3) at 5kHz which is shown in Figure 5.2. The square resistors (R4 to R6) took the Straight Cut well, but not with the same low tolerance as the low aspect ratio resistors. The high aspect ratio resistors (R7 to R9) turned out less than satisfactory.

5.3.2 The L Cut with the Elbow at 60%

The L Cuts with the elbow at 60% did not work on the low aspect ratio resistors because the second cut of the L Cut could not get the requested resistance without trimming into the resistor pad. The other two resistor aspect ratios had a very good response to this cut. The tolerances and uniformity are outstanding for the square and the high aspect ratio resistors. The tolerances on the first cut were not very good, but that is the basic advantage of the L Cut.

5.3.3 The L Cut with the Elbow at 70%

The L Cuts with the elbow at 70% were not as accurate as the elbow at 60%. In fact, this L Cut performed better on the low aspect ratio resistors than either the square or the high aspect ratio resistors. This was due to the laser trimming deeper into the resistor and trimming its second cut in an area that was still a high resistance change per resistance measurement. This did not happen to the low aspect ratio resistors because there was still a good amount of width remaining on the resistor when the second cut was begun. This data was extrapolated from Figures 5.8 to 5.10.
5.3.4 The L Cut with the Elbow at 80%

This type of cut followed very closely with the straight cut. The turn at 80% of the final value was very close to the stopping point of the final value of the straight cut. By comparing Figures 5.2 and 5.14, it is noticed that turning at 80% did not give any benefit and that there was no advantage between these cuts and the Single Straight Cuts.

5.3.5 The Double Cut

The Double Cut was programmed to turn at 70% of the final value. Surprisingly, the first cut of the Double Cuts and the first cut of the L Cuts with the elbow at 70% did not match very well by comparing Figure 5.8 and 5.16. The final values of the Double Cut had tolerances under 1% for the square and the high aspect ratio resistors. The low aspect ratio resistor did not perform well because the second cut was not far from the first cut.

5.3.6 The Serpentine Cut

The Serpentine Cut did very well, except for the two resistors with the longest length (resistors 5 and 7) as seen in Figure 5.26. Other than those two resistors, the tolerances and uniformity were excellent for the Serpentine Cuts. The square and the high aspect ratio resistors both performed well compared to the low aspect ratio resistors.
5.3.7 Discussion

It is commonly known that the Single Straight Cut is the least optimal type of cut. Although the Double Cuts and the Serpentine Cuts were clearly better than the Single Straight Cuts, the performance of the L Cuts were not so evident and were based strongly on the elbow percentage with respect to the resistor dimensions. The elbow at 60% worked well for the square and the high aspect ratio resistors, but the low aspect ratio resistors needed an elbow greater than that. The elbow at 70% and 80% worked well for the low aspect ratio resistors, but were not as effective on the square and the high aspect ratio resistors. Another interesting note is that the Serpentine Cuts worked well on the low aspect ratio resistors while the Double Cuts did not. This was due to the fact that the Serpentine Cut was accomplished by alternating cutting sides.

The results of the trims are most quickly seen by looking at the trimmed resistor. Figures 5.30 to 5.35 show images of the cuts on the square resistor 5. Figure 5.30 displays the Straight Cut as it was conceptualized in Figure 2.1. Figures 5.31 to 5.33 show the different L Cuts. It is quickly noticed how the elbow at 60% in Figure 5.31 has a longer second trim than the elbow at 70% and at 80%. The 70% and the 80% elbows differ only slightly in the length of the second cut due to the change in resistance per resistance measuring time. The Double Cut in Figure 5.34 and the Serpentine Cut in Figure 5.35 both show that by dividing up the final resistance among several cuts, no one cut went too deep into the resistor.
Figure 5.30 The Straight Cut
Figure 5.31  The L Cut with the Elbow at 60%
Figure 5.32  The L Cut with the Elbow at 70%
Figure 5.33 The L Cut with the Elbow at 80%
Figure 5.34 The Double Cut
Figure 5.35 The Serpentine Cut
5.4 Thermal Aging Results

The evaluation of the thermal aging results was based on resistor cracking along the trimmed line over time. The acquired results were quite diverse. The response of the control substrate that had no trims is shown in Figures 5.36 and 5.37. These results corresponded quite well to the specifications of the resistive paste that was used [31]. The Straight Cuts responded well, except for the 5kHz at 1.5 Watt trims (Figure 5.38). These went to an average of 5% from their original value (Figure 5.39). The other three groups averaged less than 1%. All of the low aspect ratio resistors that had L Cuts (Figures 5.40 to 5.45) held up well, also. The high aspect ratio resistors that had L Cuts were greatly affected by the thermal aging. The square resistors of the L Cuts with the elbow at 80% responded reasonably, but the high aspect ratio resistors reacted very poorly. The resistors trimmed with a L Cut with the elbow at 70% and at 3kHz with 1.5 Watts on power reacted five times worse than the other cuts on the average. The Double Cut (Figure 5.46 and 5.47) reacted the same way at 3kHz with 1.5 Watts of power. This was an interesting match because both of these cuts have had there turn/shift at 70%.

These were the only power, frequency, and type of cut combinations that showed any distinct pattern. Even with the spikes from the 3kHz cuts, the average tolerance for all of the Double Cuts was below 1%. The most tolerant and uniform distribution came from the Serpentine Cuts seen in Figures 5.48 and 5.49. The average tolerances were around 0.5% and all of the maximums and minimums varied by less than 1%. This type of cut was most related to the response of the control resistors.

In relating all of the types of cuts, the L Cuts seemed to be the most damaged from the aging process. This was probably due to the configuration of the L Cut. By having cuts in perpendicular directions, the L cut is susceptible to cracks in all directions.
Cracks that occurred off of the perpendicular cut will act like straight cuts that went directly into the current path. The response of the Serpentine Cuts was not surprising because no one cut penetrated deep into the resistor. Even though cracking may occur, the cracks would not disturb the current flow to any great extent.
Figure 5.36
The Tolerances for the Control Substrate after Thermal Aging
Figure 5.37
The Breakup of Tolerances for the Control Substrate after Thermal Aging
Figure 5.38
The Tolerances for the Single Straight Cut after Thermal Aging
Frequency and Power

Figure 5.39
The Breakup of Tolerances for the Single Straight Cut after Thermal Aging
Figure 5.40
The Tolerances for a L Cut
with the Elbow at 60% after Thermal Aging
Frequency and Power

Figure 5.41
The Breakup of Tolerances for a L Cut with the Elbow at 60% after Thermal Aging
Figure 5.42
The Tolerances for a L Cut
with the Elbow at 70% after Thermal Aging
Frequency and Power

-■- Average
-Δ- Maximum
○ Minimum

Figure 5.43
The Breakup of Tolerances for a L Cut with the Elbow at 70% after Thermal Aging
Figure 5.44
The Tolerances for a L Cut with the Elbow at 80% after Thermal Aging
Frequency and Power

Figure 5.45
The Breakup of Tolerances for a L Cut with the Elbow at 80% after Thermal Aging
Figure 5.46
The Tolerances for a Double Cut after Thermal Aging
Frequency and Power

- ■ Average
- ◇ Maximum
- ○ Minimum

Figure 5.47
The Breakup of Tolerances for a Double Cut after Thermal Aging
Figure 5.48
The Tolerances for a Serpentine Cut after Thermal Aging
Frequency and Power

- ■ Average
- △ Maximum
- ○ Minimum

Figure 5.49
The Breakup of Tolerances for a Serpentine Cut after Thermal Aging
Chapter 6

Conclusions and Recommendations

6.1 Conclusions

Laser trimming of thick film resistors is a very useful tool. The constructed laser control system interface effectively operates the laser systems functions through hardware and software. The hardware handles the line conversion, the correct buffering, and a means to communicate with the laser control system. The software takes responsibility for initializing the laser correctly, but allows the user the full realm of laser operations and parameters in an integrated operating environment. The operating environment includes a pattern storage structure that allows for sequences of patterns to be stored for later use, as well as being analyzed and printed.

The stability and effects of laser trimming are important to understand. Although the laser is a precise method of resistor trimming, the process in which the trimming is performed is important. Depending on the configuration of the resistor, a type of cut along with the power and frequency is selected. These are also a function of a necessary or suggested tolerance level and the long term stability requirements. The Serpentine Cut is the most effective type of cut for resistors without much length between the conductor pads. The Double Cut are shown to be stable for square and high aspect ratio resistors, but not for low aspect ratio resistors. The L Cuts are accurate, but require planning for the correct percentage of final resistance at which to turn. The stability of the L Cuts is
not very impressive. The Straight Cuts have relatively random results and are not suggested for use.

6.2 Recommendations

The current interface is limited for use by an XT. Future revisions may include upgrading the hardware to make use of faster machines and architectures. This will provide a more accurate resistor trimming function due to the polled nature of the interface. Additions to the current interface may also include adding diode banks to the communication lines for an easier diagnosis of line communication problems.

Further resistor trimming research may include cutting some Double-Reverse Cuts that will help in evaluating the difference between the Double Cut and the Serpentine Cut on low aspect ratio resistors. With the construction of a faster interface system, a second set of tests can evaluate the differences in actual performance.
References


30. Conversations with Kevin Woolbright from Directed Light, Inc.

APPENDIX I

ENDPRO SOURCE CODE

#include <conio.h>
#include <dos.h>

/* Initialize the addresses */
#define bmva  0x300
#define bmvc  0x302
#define bmvaom 0x303
#define bvaom  0x304
#define bvc  0x305
#define bvc  0x306
#define bccom  0x307
#define bccom  0x308
#define add56A  0x2E
#define add56C  0x02

/* Initialize the control bytes */
#define outct  0x80
/* Command config code for output all ports */
#define inct  0x9B
/* Command config code for input all ports */
#define lop2  0x02
#define clearlo  0x00

BEGIN /* Endpro.c */

PARAMETERS:

GLOBAL: NONE
LOCAL: NONE
CALLS: NONE

RETURNS: NONE

DATE: 10 February 1990
PROGRAMMER: Kyp Walters
LANGUAGE: Microsoft C Version 5.1
OS: MS-DOS Version 3.3

BEGIN

main()
{
    *** Initialize the I/O cards ***
}

END

*************************************************************************/
outp(bmbcom, outcbt);   /* INIT BMB FOR OUTPUT */
outp(baccom, outcbt);   /* INIT BAC FOR OUTPUT */
outp(accom, incbt);     /* INIT AC FOR INPUT */

/** Turn on the process complete light and turn off the in process light **/  
outp(bmba, addr56A);    
outp(bmbc, addr56C);    
outp(baca, 0x00);       
outp(bacb, 0x02);       
outp(bacc, lop2);       
outp(bacc, clearlop);   
}
APPENDIX II

STARTPRO SOURCE CODE

#include <conio.h>
#include <dos.h>

/* Initialize the addresses */
#define bmba 0x300
#define bmcb 0x302
#define bmbcom 0x303
#define baca 0x304
#define bacc 0x305
#define baccom 0x306
#define acccom 0x307
#define accom 0x308
#define addrs6a 0x31E
#define addrs6c 0x312

/* Initialize the control bytes */
#define outcbt 0x80 /* Command config code for output all ports */
#define incbt 0x9B /* Command config code for input all ports */
#define iop2 0x02
#define cleariop 0x00

*****************************************************************************
*****************************************************************************
*****************************************************************************

PROGRAM: Startpro.c
PURPOSE: To test the laser register accessing by turning on the In Process light
and turning off the Process Complete light.

PARAMETERS: GLOBL: NONE
LOCAL: NONE
CALLS: NONE
RETURNS: NONE

DATE: 10 February 1990
PROGRAMMER: Ryp Walters
LANGUAGE: Microsoft C Version 5.1
OS: MS-DOS Version 3.3

---------- INITIALS --------------- MODIFICATION

*****************************************************************************
*****************************************************************************
*****************************************************************************

main()
{
   /* Initialize the I/O cards */
outp(bmbcom, outcht);  /* INIT BM FOR OUTPUT */
outp(baccom, outcht);  /* INIT BAC FOR OUTPUT */
outp(accom, incht);    /* INIT AC FOR INPUT */

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

/*** Turn off the process complete light and turn on the in process light ***/
outp(bmba, addr56A);
outp(bmbc, addr56C);
outp(baca, 0x00);
outp(barb, 0x01);
outp(bact, iop2);
outp(bacc, cleariop);
}

128
APPENDIX III

JOYMOVE SOURCE CODE

```c
#include <defntion.h>
#include <conio.h>
#include <dos.h>
#include <moveincd.h>
#include <globval.h>
#include <moveinсид.c>

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

* PROGRAM: Joymove.c
* PURPOSE: Allow access to the full capabilities of the control panel's joystick
* PARAMETERS:
* GLOBAL: None
* LOCAL: x_velo - X axis velocity
*        y_velo - Y axis velocity
*        x_dist - X axis distance
*        y_dist - Y axis distance
*        xd - Temp X axis distance
*        yd - Temp Y axis distance
*        xv - Temp X axis velocity
*        yv - Temp Y axis velocity
*        finish - Counter
* CALLS: outp
*        rdjoy
*        move_norest
*
* DATE: 24 June 1990
* PROGRAMMER: Ryp Walters
* LANGUAGE: Microsoft C Version 5.1
* OS: MS-DOS Version 3.3
*
**************************************************************************/

main()
{
    int xd=0, yd=0, x_dist, y_dist, finish;
    float xv=0, yv=0, x_velo, y_velo;

    /****init I/O cards****/
    outp(bmbcom, outcbt);  /*INIT BMB FOR OUTPUT*/
    outp(bacom, outcbt);   /*INIT BAC FOR OUTPUT*/
    outp(acsom, incbts);  /*INIT AC FOR INPUT*/

    printf("\nHit 'e' for stop when you have finished playing with the joystick");

    finish = 1;
    while( finish == 1 )
    {
        rdjoy();

        if(xjoy == 1 )
        {
            move_norest(0, yd, 0, yv);
            xd = 0;
            xv = 0;
        }
        if(yjoy == 1 )
```
{  
    move_nolest(xd, 0, xv, 0);
    yd = 0;
    yv = 0;
}
if(xjoy := 1) /* test for motion in x */
{  
    xd = 32001;
    if(xjoy < 4)
    {
        xv = 2;
    }
    else
    {
        xv = 10;
    }
    if(xjoy == 2 || xjoy == 6)
    {
        xd = -1*xd;
    }
    move_nolest(xd, yd, xv, yv);
}
if(yjoy := 1)
{  
    yd = 32001;
    if(yjoy < 4)
    {
        yv = 2;
    }
    else
    {
        yv = 10;
    }
    if(yjoy == 2 || yjoy == 6)
    {
        yd = -1*yd;
    }
    move_nolest(xd, yd, xv, yv);
}
if(kbhlt() := 0)
{  
    finish = 0;
    move_nolest(0, 0, 0, 0);
}
}
finish = 1;
APPENDIX IV

VELTEST SOURCE CODE

#include <conio.h>
#include <dos.h>
#include <defintion.h>
#include <globval.h>
#include <moveircd.h>
#include <moveincd.c>

******************************************************************************

PROGRAM: Veltest.c

PURPOSE: To send a ramping velocity to the A boards on the linear motors

PARAMETERS:
  GLOBL: xpos     ypos
  LOCAL: x_velo   y_velo
            finish
            i, j     Counters

CALLS: outp    inp
        encode   send_vel

DATE: 24 June 1990
PROGRAMMER: Ryp Walters
LANGUAGE: Microsoft C Version 5.1
OS: MS-DOS Version 3.3

******************************************************************************

float x_velo, y_velo;
int i, j, finish;

/***init I/O cards***/

outp(bmbcom, outcbt);  /*INIT BMB FOR OUTPUT*/
outp(baccom, outcbt);  /*INIT BAC FOR OUTPUT*/
outp(acom, incbt);     /*INIT AC FOR INPUT*/

finish = 1;
while(finish == 1)
{
    x_velo = 0;
    y_velo = 0;

    /**< read in position ***/
    printf("\nstill going!");
    for(i=0; i<1000; i++)
    {
        printf("\n%i", i);
    }
for(j=127; j>0; j--)
{
    encode(-1, -1, j, j);
    send_vel();
}
for(j=0; j<127; j++)
{
    encode(1, 1, j, j);
    send_vel();
}

******************************************************************************
* MODULE: send_vel
* PURPOSE: To send the sequence to get the laser to move
* PARAMETERS: GLOBL: hlsbx    hlsby
              llsbx    llsby
              msbx    msby
              vhlsbx   vhlsby
              vlslbx   vlslby
* USERS: move_dist   move_morest
* CALLS: outp
* RETURNS: None
* DATE: 24 June 1990
* PROGRAMMER: Ryp Walters
* LANGUAGE: Microsoft C Version 5.1
* OS: MS-DOS Version 3.3
******************************************************************************

void send_vel()
{
    /*axis select x axis */
    outp(bma, addr40A); /*set addr 46 on a and c*/
    outp(bmbc, addr40C);
    outp(bacc, iopl1); /*pulse iop 1 for x axis select*/
    outp(bacc, cleariop);

    /* load velocity and acceleration */
    outp(bma, addr43A); /*set addr 43 on a and c*/
    outp(bmbc, addr43C);
    /*SET VELOCITY AT 1/4 MAX.*/
    outp(bacc, vhlsbx); /*send hls (ac 0 - 7)*/
    outp(bacc, vlslbx); /*send llsb (ac 8 - 11)*/
    outp(bacc, iopl1); /*pulse iop 1 for load velocity*/
    outp(bacc, cleariop);

    /* axis select y axis */
    outp(bma, addr40A); /*set addr 40 on a and c*/
    outp(bmbc, addr40C);
    outp(bacc, iopl2); /*pulse iop 2 for y axis select*/
    outp(bacc, cleariop);

132
/**** load velocity and acceleration ****/
outp(bmba, addr43A); /* set addr 43 on a and c */
outp(bmbc, addr43C);
outp(bca, vhisby);   /* send hlsb {ac 0 - 7} */
outp(bacb, vhisby);  /* send llsb {ac 8 - 11} */
curt(bacc, 'lop1');   /* pulse lop 1 for load velocity */
outp(bacc, cleariop);
APPENDIX V

LASER SOFTWARE FLOWCHART
LASER

MAIN()

Set up the window

Have the user turn on the laser

Initialize the laser interface and hardware

Option to exit?

Yes

Close all window functions

Exit

No

Display the main menu and prompt for option

Get the user's option

Move distance?

Yes

No

A

B

C
Option to move a distance?

Yes

Some_move()

Get user's option for joystick or distances

Not distances?

Yes

Display 'use the joystick'

Clear the manual functions register

No

Joinmove()

Load position?

Yes

Clear the manual functions register

No

Load the joystick and move accordingly

Get and transform (*) the distance and velocity

Move (**) the distance at the given velocity

Wait (***) for the move to be completed

Prompt the user for another move

Display the current position

* All numbers are entered as individual characters. They must be transformed into a number.

** All moves must be encoded in a way that the laser can understand.

*** Once a move is invoked, control returns to the caller. In order to be sure the move is finished, a check must be made.
L

Store the current position as the first position

Display the first position

Get and transform the distance

Calculate, store, and display the second position

Get and transform the velocity

Calculate the distance

Get the number of passes

M

M

Get the 'all ready' from the user

Turn on the laser beam

Passes left?

Yes

No

Move the distance

Reverse the direction

Turn off the laser beam

Display the current position

Prompt the user for another scribe

J
Pattern_cut()

Display the pattern cut menu

Prompt for the user's choice

Option between 1 and 5?

Yes

Get the name of the file

Alert the user that the file already exists

No

Option = 1
Create a file

Does the file exist?

Yes

Alert the user that the file already exists

No

Open the file with extension '.NEW'

Get and transform the cutting velocity from the user

A'

Switch on the option number

Option = 2
Analyze / Print a file

Does the file exist?

Yes

A'

No

Alert the user that the file does not exist

D'

C'
Switch on the option number

- Option = 3
  - Does the file exist? No: Alert the user that the file does not exist
  - Yes: Edit a file

- Switch on the option number
  - Option = 4
    - Does the file exist? No: Alert the user that the file does not exist
    - Yes: Do a file

- Switch on the option number
  - Option = 5
    - Quit: Exit
Build_file()

1. Display the build file menu and prompt for the user's choice

2. Move to a position?
   - Yes: Get and transform the new position
     - Display the position on the screen and save it in the file
   - No

3. Move a distance?
   - Yes: Get and transform the distance
     - Display the distance on the screen and save it in the file
   - No

4. Turn the laser on?
   - Yes: Display the laser on to the screen and save it in the file
   - No

5. Turn the laser off?
   - Yes: Display the laser off to the screen and save it in the file
   - No

6. Cut a circle?
   - Yes: Get the type of circle (saving or cutting the center)
     - Get the radius of the circle
     - Save the type and radius to the file
     - Display the loop on the screen and save it in the file
   - No: Display the type and radius to the screen

7. Start a loop?
   - Yes: Get and transform the number of repetitions
     - Display the end of the loop on the screen and save it in the file
   - No

8. End a loop?
   - Yes
   - End? Yes: Close the file and clear the screen
     - B'
   - No
   - No: Display the end of the loop or the screen and save it in the file
   - No
C'

Print the file?  

Yes

Set a register to flag for 'send file to printer'

No

Clear the register so that the file will not be printed

Have the user turn on the printer

Clear all counters and place holders

Valid pattern file?  

No

Alert the user of the error  

Yes

Get the cutting velocity and print it if requested

Action number = Exit?  

Yes

Screen display the final position and all counted items

No

Action number = Position move?  

Yes

Get the position and find the safest velocity at the traveled distance

No

Get the distance and find the safest velocity for that distance

Action number = Distance move?  

Yes

Print the position move, its parameters, and the status if requested

No

Print the distance move, its parameters, and the status if requested

Set the new position as the position moved to

Set the new position as the previous position plus the distance

H'

143
Append() 

- Open the backup files and clear the action number

- Is the original file valid? 
  - No → B'
  - Yes → Copy the cutting velocity from the original to the backup

- Action number = End of file?
  - Yes → I'
  - No → Get and copy the next action number

- Action number = Position move?
  - Yes → Copy the 'position to move to' to the backup
  - No

- Action number = Distance move?
  - Yes → Copy the 'distance to move' to the backup
  - No

- Action number = Circle?
  - Yes → Copy the radius and type of circle to the backup
  - No

- Action number = Start loop?
  - Yes → Copy the number of loops to the backup
  - No
Switch to reading from the backup and writing to the original

Get the cutting velocity

Correct cutting velocity? No

Yes

Get and transform the correct cutting velocity

Store the cutting velocity

Action number = End Yes

No

Read in the next action number

Close the backup file

Scroll the screen according to the current line

Action number = Position move? Yes

No

Action number = Distance move? Yes

No

J'
Prompt the user for correctness (yes, no, or quit)

Quit?

No?

Yes

Insert commands?

No

Delete this action?

No

Change the parameters?

Yes

Get and transform the new parameters

Is this action to be saved?

Yes

Run 'build_file()' to insert commands

Save the action and its parameters

No
L'

Prompt the user for correctness (yes, no, or quit)

Yes

Quit?

No

N'

No?

Yes

Insert commands?

No

Run 'build_file()' to insert commands

Yes

Delete this action?

No

M'

Yes

Is this action to be saved?

No

Save the action

No
Transfer_rest_of_file() →

- **Action number = End of File?**
  - Yes → B'
  - No → Get and save the next action number

- **Action number = Position move?**
  - Yes → Save the position to move
  - No

- **Action number = Distance move?**
  - Yes → Save the distance to move
  - No

- **Action number = Circle?**
  - Yes → Save the radius and the type of circle
  - No

- **Action number = Start loop?**
  - Yes → Save the number of repetitions
  - No
Do_file() 

Clear all counters and place holders 

Get the 'all ready' signal from the user 

Is the file a valid file? 

Yes 

Display the cutting velocity 

No 

Alert the user of the error 

B' 

Action number = End of file? 

Yes 

Display the final results and alert the user 

B' 

No 

Scroll the screen according to the current line 

Action number = Position move? 

Yes 

Get the position and calculate the max velocity 

Move to the position and display current status 

Set the position as the position just moved 

No 

Action number = Distance move? 

Yes 

Get distance and calculate the max velocity 

Move the distance and display current status 

Set the position as the old position plus the distance moved 

No 

O'
Action number = Laser on
  Yes → Increment the laser on counter → Turn the laser on and display current status
  No →

Action number = Laser off
  Yes → Increment the laser off counter → Turn the laser off and display current status
  No →

Action number = Circle
  Yes → Get the radius and the type of circle → Increment the circle counter
  No → Run 'circle_cut()' and display the current status

Action number = Start Loop?
  Yes → Get the repetitions and increment the loop counter → Set up the loop in an array (*)
  No → Do the array (**) → Increment the end of loop counter

Action number = End of loop?
  Yes → Display extra end of loop → Increment the end of loop counter
  No →

* All loops are put in an array format by set_up_a_loop()
** All arrays are processed by do_a_loop().
Set_up_a_loop()

Set loop conditions

Action number = End of loop?  
Yes → Return to caller  
No

Get the next action number and store it in the action array

Action number = Position move?  
Yes → Get the position and store it in the parameter array  
No

Action number = Distance move?  
Yes → Get the distance and store it in the parameter array  
No

Action number = Circle  
Yes → Get the radius and the type of circle and store it in the parameter array  
No

Action number = Start loop?  
Yes → Get the loop repetitions and store it in the parameter array  
No → Recursively call 'set_up_a_loop'
Do_a_loop()

Set up the loop number and the counter (# of loops completed)

Does the number of loops left = 0?

Yes

Return to the caller

No

Decrement the loop counter

Initialize counter

Action number = End of loop?

Yes

Get the next action number

No

Scroll the screen according to the current row

Action number = Position move?

Yes

Get the position and calculate the max velocity

No

Move to the position and display the current status

Set the position as the position just moved
```
Transform()

Is the first char. a number?
  Yes
    Clear the temporary and final number
    Is the first character a '-'?
      Yes
        Flag the negative register
      No
        Is the next char. a number?
          Yes
            Multiply the current value by 10 and add in the new digit
          No
            No
              Is the next character a '.'?
                Yes
                  Set the array pointer to the last character
                No
                  Add character into the current decimal value divided by 10
                    Is the character a number?
                      Yes
                        Decrement the array pointer
                      No
                        No
                          Are we back to the '.'?
                            Yes
                              Multiply the number by -1
                            No
                              Add the integer and the decimal parts together
                                Is the negative register flagged?
                                  Yes
                                    Multiply the number by -1
                                  No
                                    Return value to caller
```
Velocity_set()

Are both distances less than 49?

Yes: Return a velocity of 2 to the caller

No

Are both distances less than 199?

Yes: Return a velocity of 4 to the caller

No

Are both distances less than 699?

Yes: Return a velocity of 6 to the caller

No

Return a velocity of 8 to the caller
Laser_trim()

Get and transform the probe numbers

Set up the probe ring

Find_me()

Initialize the measurement system

Pick a resistance of 500kΩs

Test the resistor for in range

Valid?

Yes

Increase the resistor value

No

Increase the resistor value

Out of low range?

Yes

Get the feedback and calculate a better resistance

Get the feedback for the new resistance

Change in feedback > 1?

Yes

Get the new value for the resistor

Valid value?

Yes

No

Out of range?

Yes

No

Decrease the resistor value

O
Set up the measurement system

Have the user use the joystick to move to the first position

Store and display the first position

Have the user use the joystick to move to the second position

Store and display the second position

Are the positions correct?

Move back to the first position

Set all of the measurement system parameters

Display all of the values and feedback

Wait for an 'all clear' from the user

Turn on the laser and begin the move

Adjust the feedback gain and get the feedback

Out of range?

High gain?

R
R

Is the resistor within tolerance?

Yes

Turn off the laser and stop the move

Use 'find_me()' to get the current resistance

Display the final results

No

H

Index the laser

Did it work?

Yes

Display all okay

No

Display a hardware error

Display the current position

C

C
Measure_test()

Get and set up the probe ring and probes

Get the current feedback

Display the options and feedback

Get the users option

All processes complete?

Yes

Get the feedback

No

Get, transform, and set the new resistance

Yes

Get, transform, and set the new voltage

No

Get, transform, and set the new detector gain

Yes

No

S
S

Change the preamp gain?
Yes
Get, transform, and set the new preamp gain
No

Change the comparator?
Yes
Get, transform, and set the new comparator value
No

Strobe the system?
Yes
Strobe the measurement system
No

Clear an overload?
Yes
Clear the overload
No

Find the resistance?
Yes
Use 'findme()' to find the resistance
No

Disable the limit crossing?
Yes
Disable the limit crossing flag
No

Exit?
Yes
C
No

T
Distance Moving

Move_dist() and Move_norest()

Encode the distances and velocities

Move_norest()

Wait for the laser to stop

Send the x-axis velocity and lower 12 bits of distance

Send the y-axis velocity and lower 12 bits of the distance

Send the x-axis higher 4 bits of the distance

Send the y-axis higher 4 bits of the distance

Return to caller

Wait_for_rest()

Check the status of the x and y motors

Are both axes stopped?

No

Yes

Return to caller
APPENDIX VI

LASER SOURCE CODE

/******************************************************************************
 *
 LASER.C
 *
 ******************************************************************************/

#include <bwindow.h>
#include <moveinod.h>
#include <measincod.h>
#include <newstuff.h>
#include <stdio.h>
#include <glb1var.h>
#include <defntion.h>
#include <math.h>
#include <float.h>
#include <main.h>

/******************************************************************************
 *
 GLOBAL PARAMETERS:
 *
 DEFINN: acc        acb        - Port addresses for info transfer from
  aca       accom      laser to pc (ac port)
 ba          bacc      - Port addresses for info transfer from
 bac          baccom    laser to pc (bac port)
 bma          bmbc      - Port addresses for function call from
 bmcom       laser to pc (bm port)
 iop1       iop2        - Bmb value number for function load
 iop4        clearlop   - I/O board definition values for input
 outcbt      incbtt     vs. output
 addr13A    addr13C     - Address value sets for each function
 addr15A    addr15C
 addr16A    addr16C
 addr17A    addr17C
 addr30A    addr30C
 addr31A    addr31C
 addr32A    addr32C
 addr40A    addr40C
 addr41A    addr41C
 addr42A    addr42C
 addr43A    addr43C
 addr44A    addr44C
 addr52A    addr52C
 addr53A    addr53C
 addr54A    addr54C
 addr55A    addr55C
 addr56A    addr56C
 ON          OFF        - Status of laser beam on/off

 GLOBL:  BWINDOW - Window defined type cast for window pointer name
 sprobe    - The 'a' probe number
 bprobe    - The 'b' probe number
 circles    - The count of number of circles that are in the current file
 compare    - The comparator number sent to the laser in high speed trimming
 convertor  - The value returned from the laser in response to a resistor measure
 current    - The file that is currently being processed
 delay      - The delay value for the measurement bridge before it takes a reading
 end_conv   - The laser control skip bit for convertor done converting
 end_delay  - The laser control skip bit for the delay being over
 end_loops  - The count of the number of ended loops in the current file
 file_name  - The file name that the user types in - with no extension
 gain       - Gain for the measurement system before being sent to the pc
 gen_volt   - The generator voltage value for the measurement bridge

165
hcompare - The high byte of the sent comp. word - used for software checking
hdelay - The high byte of the sent delay word
hgain - The high byte of the sent gain word
hgen_volt - The high byte of the sent generator voltage word
hpreamp - The high byte of the sent preamplifier word
hrange - The high byte of the sent range word
hstandard - The high byte of the sent standard word
lcompare - The low byte of the sent comp. word - used for software checking
ldelay - The low byte of the sent delay word
lgain - The low byte of the sent gain word
lgen_volt - The low byte of the sent generator voltage word
lpreamp - The low byte of the sent preamplifier word
lrange - The low byte of the sent range word
lstandard - The low byte of the sent standard word
hlsby - The upper byte of the second x distance word after being encoded
hlsbx - The upper byte of the second y distance word after being encoded
hlsby - The lower byte of the second x distance word after being encoded
hlsbx - The lower byte of the second y distance word after being encoded
msbx - The only byte of the first x distance word after being encoded
msby - The only byte of the first y distance word after being encoded
laser_status - Current status of the laser beam on/off
loops - The number of loops in the current file
nested_loops - Array for all nested loops of the current file that is being run
new_file - The name of the user file with .new extension - the current file
num_loops - Counter as to which of the nested loops is currently being looped
off - Number of times that the current file turns the laser off
on - Number of times that the current file turns the laser on
old_file - The user file with .old extension - the backup of the last file
old - The file being backed up due to a revision in the current file
overload - The measurement system overload readout
pxlct - The name of the lower window - used for parameter viewing
pxpos - The x positions that are used for difference in distances
pypos - The y positions that are used for difference in distances
pre_amp - The value of the pre-amp which sets the bridge up for a measure
print - The file that is written to for future printing
prn_file - The user file with .prn extension - the printable function listing
probehigh - The high byte of the 'a' probe byte
probelow - The high byte of the 'b' probe byte
probelow - The low byte of the 'b' probe byte
pwin - The name of the main (upper) window
rl - The global version of the current resistance
radius - The radius of the circle to be cut
range - The value number of which range the current resistor is in
ready - The laser control skip bit for if the laser measurement is ready
resistor - The user inputted value for the resistor value to be trimmed to
reset - The laser control bit for if the linear motors (laser head) are moving
selection - The character string (2 char) that stores user responses
scan - The dummy variable that holds the end character of user input
speed - The type of a circle (with diagonal cut or with center unscathed)
standard - The ratio of the user inputted resistance and the range resistor
strobe_done - The skip bit for if the pre-measuring strobe is completed
units_of_step - The counter for which step number of the current nested loop
values - The parameters for nested loops during the running of a program
vel - The velocity that the laser is set to move at
vhsbx - The high bit of the x direction of the velocity after being encoded
vhsby - The high bit of the y direction of the velocity after being encoded
vlsbx - The low bit of the x direction of the velocity after being encoded
vlsby - The low bit of the y direction of the velocity after being encoded
x_axis - The x axis distance to move
y_axis - The y axis distance to move
x_displace - The difference distance between two points in the x direction
y_displace - The difference distance between two points in the y direction
x_position - The current x position for evaluation
y_position - The current y position for evaluation
x_travel - The x position or distance that is being read from/written to
y_travel - The y position or distance that is being read from/written to
xhome - The laser control bit for the x direction home position latch
yhome - The laser control bit for the y direction home position latch
xjoy - The laser control bit for the joystick movement in the x direction
yjoy - The laser control bit for the joystick movement in the y direction
xotos - The movement error value for the x linear motor
yotos - The movement error value for the y linear motor
 xpos - The current x position
 ypos - The current y position
MODULE: main

PURPOSE: Laser system operating system

PARAMETERS:

GLOBL: plist pwin
        scan radius
        xpos ypos

LOCAL: act_page - Active page number
        choice - Numeric value of user response to main menu
        col - Current column position of the cursor
        columns - Number of columns in the window
        high - Pointer for cursor high scan
        i - Counter variable
        low - Pointer for cursor low scan
        mode - Current mode of the window
        reps - Number of times to cut a circle
        row - Current row of the cursor
        yeah - Converted user input for loop control

CALLS: wncreate scmode
        scursor wndisplay
        wselect wncursor
        wncurmov wprintf
        wquery outp
        probe_select laser_off
        rdposltn wnscroll
        scribe laser_trim
        cut_a_pattern some_move
        index_pos wait_for_rest
        measure_test move_dist
        transform circle_cut
        wremove sccurset
        soplgcursor wndestroy

DATE: 24 June 1990
PROGRAMMER: Ryp Walters
LANGUAGE: Microsoft C Version 5.1
OS: MS-DOS Version 3.3

***************************************************************

void main()
{
    BORDER bord;
    WHERE location;
    int mode,columns,act_page;
    int row,col,high,low,choice;
    int reps, yeah, i;

    plist = wncreate(3,78,WHITE); /* Dimensions of window data area. */
    pwin = wncreate(18,78,CYAN);

    bord.type = BBRD_DDDD | BBRO_TCT; /* Box drawn with single */
    /* lines, having a top centered */
    /* title. */
    bord.attr = MAGENTA; /* Border will be magenta on black. */
    location.dev = scmode(&mode,&columns,&act_page);
    location.page = act_page;
    location.corner.col = 1;
    location.corner.row = 21;

    /* Other code goes here */
    return;

}
bord.tattr = REVERSE;  /* Title will be black on white. */
bord.pattr = ' System Status ';  /* Text of the title. */
if (NIL == wndsplay(plist,&location,&bord))
    return;  /* Quit if failure. */
location.corner.row = 1;
bord.pattr = ' Laser Operating System ';  /* Text of the title. */
if (NIL == wndsplay(pwin,&location,&bord))
    return;  /* Quit if failure. */
wmselect(pwin);  /* Display turn on laser in top window */
wncursor(pwin);
wncurmov(8,10);
wprintf("Turn on the Laser. (Press ENTER to continue)");
wnquery(selection, 1, & scan);  /* Wait for the user to respond */
wncurmov(8,10);
wprintf("Laser initialization sequence is in process.");

/****Init I/O cards****/
outp(bmbcom, outcbt);  /* INIT BMB FOR OUTPUT */
outp(baccom, outcbt);  /* INIT BAC FOR OUTPUT */
outp(accom, incbct);  /* INIT AC FOR INPUT */

outp(bacc,0x08);  /* Reset the system with a pulse on Bint line */
for(i=0;i<1000;i++)
    outp(bacc,0x00);
probe_select();  /* Set the multiplexer to acknowledge pc use */
laser_off();  /* Turn off the laser */

/****clear manual functions****/
outp(bmba, addr56a);  /* Set addr 56 on a and c */
outp(bmbc, addr56c);
outp(bacc, lop1);  /* Pulse lop 1 for clear manual functions */
outp(bacc, clearlop);

index_pos();  /* Send the laser to its internal home position */
wait_for_rest();
clnintr();  /* Clear all internal interrupts */
wmselect(plist);  /* Display the current position - bottom window */
wncurmov(0,2);
rdposln();
wprintf("Current Position: %6li, %6li", xpos,ypos);
choice = 0;
while(choice != 9)  /* While the choice is not an exit */
{
    wmselect(pwin);
    wmscroll(0,-1,-1,0);  /* Clear the upper screen */
    wncurmov(2,22);  /* Display the main menu */
    wncurmov(4,20);
    wncurmov(5,20);
    wncurmov(6,20);
    wncurmov(7,20);
    wncurmov(8,20);
    wncurmov(9,20);
    wncurmov(10,20);
    wncurmov(11,20);
wnprintf(" 8. Test measurement system ");
wncurmov(12,20);
wnprintf(" 9. Exit ");
wncurmov(14,20);
wnprintf(" Option:");

wnquery(selection, 2, &scan); /* Get the users selection */
if(*selection == 'm' || *selection == 'M' || *selection == 49)
    choice = 1;
if(*selection == 's' || *selection == 'S' || *selection == 50)
    choice = 2;
if(*selection == 'p' || *selection == 'P' || *selection == 51)
    choice = 3;
if(*selection == 'c' || *selection == 'C' || *selection == 52)
    choice = 4;
if(*selection == 'i' || *selection == 'I' || *selection == 53)
    choice = 5;
if(*selection == 'r' || *selection == 'R' || *selection == 54)
    choice = 6;
if(*selection == 'h' || *selection == 'H' || *selection == 55)
    choice = 7;
if(*selection == 't' || *selection == 'T' || *selection == 56)
    choice = 8;
if(*selection == 'e' || *selection == 'E' || *selection == 57)
    choice = 9;

switch(choice) /* Jump to the correct set of commands */
{
    case 1: /* Option to move a distance */
        while(choice == 1) /* While the user still wants a distance move */
        {
            wnscroll(0,-1,-1.0);
            /* Clear the screen */
            somew_move();
            /* Do a distance move */
            wnprintf("Is this the final point? (y/n)");
            /* See if the user wants another move */
            wnquery(selection,2,&scan);
            if(*selection == 'y' || *selection == 'Y')
                choice = 0;
            break; /* When all is done, go back to the main menu */
        }
    case 2: /* Option to scribe */
        while(choice == 2) /* While the user still wants to scribe */
        {
            wnscroll(0,-1,-1.0);
            /* Clear the screen */
            scribe();
            /* Do the scribing */
            wnselect(plist);
            /* Display the current position */
            wncurmov(0,2);
            rdposnt();
            wnprintf("Current position: %g1, %g1", xpos,ypos);
            wnselect(pwin);
            wncurmov(14,15);
            /* See if the user wants another scribe */
            wnprintf("Do you want to do another scribe? (y or n) ");
            /* wnquery(selection, 2, &scan); */
            if(*selection == 'y' && *selection == 'Y')
                choice = 0;
            break; /* When all is done, go back the main menu */
        }
    case 3: /* Option to cut a pattern */
        wnsroll(0,-1,-1.0);
        /* Clear the screen */
        cut_a_pattern();
        /* Go to the cut a pattern menu */
        break; /* When all is done, go back to the main menu */
}
case 4: /* Option to cut a circle */
    wnsrccoll(0,-1,-1,0); /* Clear the screen */
circle(); /* Do a circle cutting */
breatk; /* When all is done, go back to the main menu */

    case 5: /* Option to index the laser */
    wnsrccoll(0,-1,-1,0); /* Clear the screen */
    rdposltn(); /* Move to the 0,0 position */
    move_dist(-1*xpos,-1*ypos,10,10);
    wait_for_rest();

    for(i=0;i<1000;i++)/* Pause for a while */
        rdposltn();
    if(xpos != 0 & & ypos != 0)
        move_dist(-1*xpos,-1*ypos,1,1);
    wselect(plist);
    wncrmov(0,2);
    wait_for_rest();

    for(i=0;i<1000;i++)/* Pause just a tad bit more */
        rdposltn();
    wprintf("Current position: \$611, \$611 \`,xpos,ypos");
    wselect(pwin);
        break; /* When all is done, go back to the main menu */

    case 6: /* Option to do a laser resistor trim */
    wnsrccoll(0,-1,-1,0); /* Clear the screen */
    laser_menu(); /* Do a resistor trim */
    wselect(plist);
    wncrmov(0,2);
    wncrmov(0,2);
    rdposltn();
    wprintf("Current position: \$611, \$611 \`,xpos,ypos");

    wselect(pwin);
    break; /* When all is done, go back to the main menu */

    case 7: /* Option to check the hardware */
    wnsrccoll(0,-1,-1,0); /* Clear the screen */
    index_pos(); /* Index the laser position */
    wait_for_rest();

    for(i=0;i<1000;i++)/* Pause just a little while */
        rdposltn();
    wncrmov(5,5);
    rdposltn();
    if(xpos != 0 & & ypos != 0) /* If the index did not get us home, alert user */
        {wprintf("There is a problem with the hardware");
        wncrmov(6,5);
        wprintf("Current position: \$611, \$611 \`,xpos,ypos");
        }
    else /* If we made it home, all is okay */
        wprintf("Everything with the hardware is correct");
    wncrmov(8,5); /* Let the user absorb all the info */
    wprintf("(Press ENTER to continue) ");
    wquery(selection, i, &scan);
    wselect(plist); /* Display the current position */
    wncrmov(0,2);
    rdposltn();
    wprintf("Current position: \$611, \$611 \`,xpos,ypos");
    wselect(pwin);
        break; /* When all is done, go back to the main menu */

    case 8: /* Option to test out the measurement system */
    wnsrccoll(0,-1,-1,0); /* Clear the screen */
    measure_test(); /* Do a measurement system test */
    break; /* When all is done, go back to the main menu */
    } /* No other options!! */

} /* Go back to test for an exit or for main menu */
/ * Remove the window & restore the screen and cursor. */
wnremove(pwin);
wnremoveplist);
wndstroc(pwin); /* Clean up data structures. */
wndstrocplst); /* Clean up data structures. */
*/

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

/*
ODULE:  wait_for_rest

POSE: To wait for the laser to stop moving

PARAMETERS:   GLOBL: resting             laser_status

  USERS: main             scribe
        circle_cut     some_move
        move_dist     index_pos

  CALLS: skip_on_rest

  RETURNS: None

  DATE: 24 June 1990
  PROGRAMMER: Ryp Walters
  LANGUAGE: Microsoft C Version 5.1
  OS: MS-DOS Version 3.3
  *
  *  DATE   INITIALS    MODIFICATION
  *  --------   -----------  -----------------------------------

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

void wait_for_rest(void)
{
  resting = 0;
  while(resting == 0) /* Let's assume that we are moving */
    skip_on_rest();
  /* Keep testing until we are not moving */
}

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

/*
ODULE:  wait_for_rest2

POSE: To wait for the laser to stop moving, but also testing to see if the STOP
  button has been pressed

PARAMETERS:   GLOBL: resting             laser_status

  USERS: main             scribe
        circle_cut     some_move
        move_dist     index_dist

  CALLS: skip_on_rest     laser_off
        check_for_rest  move_dist

  RETURNS:   1 - If STOP
             0 - If okay to continue

  DATE: 24 June 1990
  PROGRAMMER: Ryp Walters
  LANGUAGE: Microsoft C Version 5.1
  OS: MS-DOS Version 3.3
  *
  *  DATE   INITIALS    MODIFICATION
  *  --------   -----------  -----------------------------------
  *  12/90     RRW       ------ check_for_stop to stop in middle of a move

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

171
int wait_for_rest2()
{
    resting = 0;
    while(resting == 0) /* Let's assume that we are moving */
    {
        if(check_for_stop() == 1) /* Check the stop button */
        {
            laser_off(); /* If STOP button, turn off the laser, stop */
            move_noreset(0,0,0,0); /* moving, reset laser status flag, and */
            laser_status = OFF; /* return with an error flag. */
            return(1); /* No STOP, keep checking for stopped moving */
        }
        skip_on_rest(); /* Stopped regularly, return with okay flag */
    }
    return(0); /* */
}

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

    MODULE: circle

    PURPOSE: To set up a circle cut from the main menu

    PARAMETERS: GLOBAL: resting       radius
                 scan                   selection

    LOCAL: numbers - Character buffer for numeric input (radius)
               reps - Number of times to do the circle
               yeah - Prompt variable for another circle
               type - Type of circle (cutting or saving the circle)

    USERS: main

    CALLS: wncurmov       wprintf
            wQUERY circle_cut
            some_move  transform
            wnscroll

    RETURNS: NONE

    DATE: 24 June 1990
    PROGRAMMER: Ryp Walters
    LANGUAGE: Microsoft C Version 5.1
    OS: MS-DOS Version 3.3

    ** DATE INITIALS MODIFICATION **
    ---------------------------------------------------------------

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

void circle()
{
    int reps, yeah, type;
    char numbers[5];

    yeah = 1; /* Assume that we are not at the origin */
    while(yeah == 1) /* While we are not sure if we are at the origin*/
    {
        wnscroll(0,-1,-1,0); /* Clear the screen */
        wncurmov(5,5); /* Clear the screen */
        wprintf("Is this the origin? [y/n]%!": /* Ask the user if we are there yet */
        wquery(\"selection\",2,&scan);
        if("selection == \"y\" || selection == \"Y\")
        yeah = 0; /* If we are there, then we can continue */
        else
        {
            wnscroll(0,-1,-1,0); /* Clear the screen */
            some_move();
        }
    } /* Check if we are sure about the origin */
wnscroll(0,-1,-1,0);  /* Clear the screen */
wncurmov(5,5);

wpinf("Do you want to keep the center? \(y/n\)?
wnquery(selection,2,&scan);
if(*selection == 'n' || *selection == 'N')  /* If not, flag a cut the circle */
type = 1;
else  /* If so, flag a save the circle */
type = 0;

yeah = 1;  /* Assume that some vaule will be wrong */
while(yeah == 1)  /* While we are not sure if the values are right*/

wncurmov(8,8);  /* Get the radius */
wnprint("What is the radius in integer values of .01mm? \(r\)?
wnquery(numbers,5,&scan);
radius = (int)transform(numbers);  /* Transform the radius from an ASCII string */
wncurmov(9,8);  /* Get the repetitions */
wnprint("How many repetitions? \(n\)?
wnquery(numbers,3,&scan);
reps = (int)transform(numbers);  /* Transform the repetitions from ASCII */
wncurmov(11,8);  /* Check if the user is happy yet */
wnprint("Is radius = \&d correct? \(y/n\)? \(r\)?
wnquery(selection,2,&scan);
if(*selection == 'y' || *selection == 'Y')  /* If so, all is well and we can continue */
yeah = 0;
}  /* Check to see if all is okay with the values */

wncurmov(13,8);  /* Let the user give an all clear */
wnprint("Press START when you are ready");
wait_for_start();

while(reps != 0)  /* While there are still reps to \&in */
{
    reps--;  /* Subtract one from the reps */
    circle_cut(type);  /* Do a circle cut of the wanted type */
}  /* Check if we are done yet */

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

**
** MODULE: scribe
**
** PURPOSE: Do a single operation scribe
**
** PARAMETERS:  
**  
**  GLOBAL:  
**  plist, posx
**  posy, pwin
**  vel
**  ypos, rt
**  selection, scan
**  x_displace, y_displace
**  x_position, y_position
**  vel
**  
**  LOCAL:  
**  passes - Number of times to do the scribe
**  
**  USERS: main
**  
**  CALLS:  
**  wncurmov, wnprint
**  wquery, joymove
**  wsavepos, wmove
**  wscroll, wselect
**  transform, laser_on
**  move_dist, laser_off
**  wait_for_reps?
**  move_norest
**  
**  RETURNS: NONE
**

173
void scribe() {
    char passes, numbers[7];

    wncurmov(5,5);          /* Figure out if it will be with joy or dist */
    wquery(*selection,2,&scan);

    if(*selection != 'd' & *selection != 'D')         /* If it is not with distances, yes joystick */
    {
        wncurmov(5,5);         /* Tell the user to use the joy to move to pos 1*/
        wprintf("Use the joystick to move to the first position and to align the base.");
        wncurmov(10,8);        /* Use the laser panel button to load the position.*/
        joymove();             /* Let the user use the joystick */
        rdpositn();            /* Store the current position as the 1st pos. */
        posx[0] = xpos;
        posy[0] = ypos;
        weselect(plist);       /* Display the current position */
        wncurmov(1,2);         /* Position #1: %6li, %6li, posx[0], posy[0]);
        wprintf("Use the joystick to move to the second position.");
        joymove();             /* Let the user use the joystick */
        rdpositn();            /* Store the current position as the 2nd pos. */
        posx[1] = xpos;
        posy[1] = ypos;
        weselect(plist);       /* Display the second position */
        wncurmov(2,2);         /* Position #2: %6li, %6li, posx[1], posy[1]);
        wprintf("Use the joystick to move to the third position.");
        joymove();             /* Let the user use the joystick */
        rdpositn();            /* Store the current position as the 3rd pos. */
        posx[2] = xpos;
        posy[2] = ypos;
        weselect(plist);       /* Display the third position */
    }

    else if(*selection != 'y' & *selection != 'Y')    /* If the user wants to use distances */
    {
        wncurmov(10,8);       /* Clear the screen */
        wprintf("Enter the x distance (less than +/-6000*)");
        wquery(numbers,6,&scan);
        x_displace = (int)transform(numbers);
        wncurmov(10,8);       /* Get the y distance */
        wprintf("Enter the y distance (less than +/-6000*)");
        wquery(numbers,6,&scan);
        y_displace = (int)transform(numbers);
        vel = velocity_set(x_displace, y_displace);    /* Calculate the best velocity to travel at */
        move_dist(x_displace, y_displace, vel, vel);
    }
}
wait_for_rest();            /* Move that distance */

wmscroll[0,-1,-1,0];       /* Clear the screen */
wmcurmov[1,5];             /* Ask the user if he is happy now */
wmprintf("Is the current position the first position? (y/n) ");
wmquery(selection,2,&scan);
wmprintf(" ");            /* Repeat if the user isn't happy */
readinput();              /* If all is okay, store current pos as 1st pos.*/
posx[1] = xpos;
posy[1] = ypos;

wmselect(plist);            /* Display the first position */
wmcurmov(1,2);
wmprintf("Position #1: %6d, %6d", posx[1], posy[1]);

wmselect(pwin);            /* Tell the user to enter the distance to cut */
wmscroll(0,-1,-1,0);       /* Clear the screen */
wmcurmov(5,5);
wprintf("Enter the distance to move with respect to the current position.");
wcurmov(8,8);              /* Get the x distance */
wprintf("Enter the x distance (less than +/-6000) ");
wquery(numbers,6,&scan);
x_position = (int)transform(numbers); /* Transform the x distance */
wcurmov(9,8);              /* Get the y distance */
wprintf("Enter the y distance (less than +/-6000) ");
wquery(numbers,6,&scan);
y_position = (int)transform(numbers); /* Transform the y distance */
posx[0] = posx[1] + x_position;  /* Calculate the second position */

posy[0] = posy[1] + y_position;

wmselect(plist);            /* Display the second position */
wcurmov(2,2);
wprintf("Position #2: %6d, %6d", posx[0], posy[0]);

wmselect(pwin);

wmscroll(0,-1,-1,0);       /* Clear the screen */
wcurmov(3,8);              /* Get the velocity to travel at */
wquery(numbers,7,&scan);
vel = transform(numbers);   /* Transform the velocity */
x_displace = (int)(posx[0] - posx[1]); /* Figure out the distance */
y_displace = (int)(posy[0] - posy[1]);

wmcurmov(3,5);             /* Get the number of passes */
wprintf("How many passes would you like to make? (max of 9) ");
wquery(numbers, 2, &scan);
passes = *numbers - 48;    /* Transform the passes */
if(passes <= 0 || passes > 9)  /* Don't let the user enter an invalid number */
    return;
wcurmov(8,9);             /* Wait for an all okay from the user */
wprintf("Press START when you are ready.");
wait_for_start();
laser_on();                /* Turn on the laser */

while((passes != 0) && wait_for_rest2() != 1)  /* While there are still passes to do and */
    {                                     /* no STOP button */
    passes--;                             /* Subtract one from the passes */
    move_noreset(x_displace, y_displace, vel, vel); /* Move the distance */
    x_displace = -1 * x_displace; /* Get the distance ready to move backwards */
y_displace = -1 * y_displace;
void circle_cut(int type)
{
    int origx, origy, negradius;
    int xax, yax;
    double angle, increment;

    laser_off(); /* Turn off the laser */
    rdposin(); /* Read and store the current pos as the center */
    x_travel = xpos;
    y_travel = ypos;
    wnselect(pplist);

    angle = 0; /* Initialize the angle */
    increment = 2*asin(.5/(double)radius); /* Calculate the angular increment step */
    wncurmov(0,40); /* Display the radius and increment */
    wprintf("The increment is %1f",increment);
    wncurmov(2,40);
    wprintf("The radius is %3d", radius);
    negradius = -1 * radius; /* Set the first comparison values */
origx = radius;
origy = 0;

vel = velocity_set(radius, radius); /* Set the velocity to travel the edge */
if(type == 1) /* If option to cut the circle, */
laser_on(); /* turn on the laser */
move_dist(negradius, 0, vel, 0); /* Move to the circumference */
if(wait_for_rest2() == 0) /* While there has not been a STOP button */
{
laser_on(); /* Turn on the laser regardless of option */
while((angle < 6.2831853) && (wait_for_rest2() != 1)) /* While no STOP and not a complete circle */
{
    angle = angle + increment; /* Find the new angle */
yax = (int)(radius*sin(angle)); /* Compute the components */
xax = (int)(radius*cos(angle));
    if(origx != xax || origy != yax) /* Check if movement is needed */
    {
        move_noret( origx - xax, yax - origy, 2, 2); /* If so, move it: */
        origx = xax; /* Change the location to the new location */
        origy = yax;
    }
/* Check if we are done */
wait_for_rest(); /* Make sure that the laser has finished moving */
laser_off(); /* Turn off the laser */
}
rdpositn(); /* Find out where we are */
move_dist(x_travel - xpos, y_travel - ypos, vel, vel); /* Move back to the center */
wncurmov(0, 40); /* Clear the lower screen of all circle info */
wprintf(" */
wncurmov(1, 40); /* */
wprintf(" */
wncurmov(2, 40); /* */
wprintf(" */
wnselct(pwin); /* */
);

*******************************************************************************
*
* MODULE: move_move
*
* PURPOSE: To move a distance either by the joystick or by distances
*
* PARAMETERS: GLOBL: plist pwin
* vel xpos
* ypos selection
* scan x_displace
* y_displace
*
* LOCAL: numbers - Character buffer for numeric input (distances)
*
* USERS: main do_file
*
* CALLS: wncurmov wnselct
* wquery joymove
* transform move_dist
* wait_for_rest2 wnselct
* rdpositn wnselcrll
*
* RETURNS: NONE
*
* DATE: 24 June 1990
* PROGRAMMER: Ryp Walters
* LANGUAGE: Microsoft C Version 5.1
* OS: MS-DOS Version 3.3
*
177
void some_move()
{
    char numbers[7];
    wncurmov(5,5);    /* Ask if user wants to use the joy or dist */
    wnprintf("Would you like to use the Joystick or Distances? (j/d)");
    wquery(selection,2,&scan);

    if ('selection' != 'd'&& 'selection' != 'D') /* If its not dist, its going to be joy*/
    {
        wncurmov(5,5);
        /* Tell the user to use the joy */
        wnprintf("Use the joystick to move and load position to finish");
        joymove();                             /* Let the user use the joystick */
    }
    else                                          /* If he wants to use distances */
    {
        wncurmov(8,8);
        /* Get the x distance */
        wnprintf("Enter the x distance (less than +/-6000)");
        wquery(numbers,6,&scan);
        x_displace = (int)transform(numbers);   /* Transform the x distance */
        wncurmov(9,8);
        /* Get the y distance */
        wnprintf("Enter the y distance (less than +/-6000)");
        wquery(numbers,6,&scan);
        y_displace = (int)transform(numbers);   /* Transform the y distance */
        wncurmov(10,8);
        /* Get the velocity */
        wnprintf("Enter the velocity (127.75 to .25)");
        wquery(numbers,7,&scan);
        vel = transform(numbers);               /* Transform the velocity */
        move_dist(x_displace, y_displace, vel, vel); /* Move the distance */
        wait_for_rest2();                        /* Wait for the laser to stop moving */
    }

    wselect(plist);                               /* Get and display the current position */
    wncurmov(0,2);
    rdposition();
    wnprintf("Current position: %f, %f ,xpos,ypos");
    wselect(pwin);
}

******************************************************************************

MODULE: transform

PURPOSE: To change a character buffer array into a numeric number

PARAMETERS: PARAM: buffer         - Character buffer to change

LOCAL: j                - Loop variable
    m                - Loop variable
    n                - Loop variable
    vi               - decimal part of the number
    value            - integer part of the number

USERS: build_file
    main
    scribe
    measure_test

RETURNS: Number in float form

DATE: 24 June 1990
PROGRAMMER: Ryp Walters
LANGUAGE: Microsoft C Version 5.1
OS: MS-DOS Version 3.3
float transform(char *buffer)
{
    float value, vl;
    int n,j,m;

    if((buffer[0] < 48 || buffer[0] > 57) && buffer[0] != 45 && buffer[0] != 46);
    else /* If the first character is not valid, quit */
    {
        /* If the first character is valid */
        n=0;
        /* Initialize the counters */
        value = 0;
        vl = 0;
        if(buffer[0] == 45) /* If the first character is negative, */
            n=1; /*Flag it */
        for(j=n; buffer[j] >= 48 && buffer[j] <= 57; j++) /* For each integer number */
            value = (value * 10) + buffer[j] - '0'; /* Add it in and shift */
        if(buffer[j] == 46) /* If, after the integers there is a period, */
        {
            for(m=6; m += j; --m) /* For each possible other character */
            {
                if(buffer[m] >= 48 && buffer[m] < 57) /* If the character is a number */
                    vl = (vl + buffer[m] - '0')/10; /* Add it in and shift */
            }
        }
        value = value + vl; /* Add the decimal and integer numbers */
        if(n==1) /* If the number is negative, */
            value = (-1)*value; /* Make the number negative */
    }
    return value; /* Return the number */
}

******************************************************************************

MODULE: vel_set

PURPOSE: Calculate the fastest velocity that the laser can safely move

PARAMETERS: PARAM: x_dist  - The x distance to travel
              y_dist  - The y distance to travel

USERS: cut_a_pattern  build_file
        main
        append
        scriber
        append
        measure_test

RETURNING: Number (velocity) in int form

DATE: 24 June 1990

PROGRAMMER: Ryp Walters

LANGUAGE: Microsoft C Version 5.1

OS: MS-DOS Version 3.3

******************************************************************************

int velocity_set(int x_dist, int y_dist)
{
    if(abs(x_dist) < 49 && abs(y_dist) < 49)
        return(2);
    if(abs(x_dist) < 19 && abs(y_dist) < 199)
        return(4);
if(abs(x_dist) < 699 && abs(y_dist) < 699)
    return(5);

return(8);
}
/*
 * NEUSTUFF.C
 */
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <stdlib.h>
#include <string.h>
#include <float.h>
#include <math.h>
#include <bwindow.h>
#include <measincd.h>
#include <newstuff.h>
#include <moveincd.h>
#include <extrnvar.h>
#include <defnition.h>
#include <main.h>

/*
 * MODULE: laser_menu
 * PURPOSE: To figure out what kind of trimming the user wants
 * PARAMETERS: GLOBAL: aprobe bprobe
 * convertor end_conv
 * gain gen_volt
 * overload plist
 * posx posy
 * pre_amp pwn
 * range resistor
 * scan standard
 * xpos ypos
 * rl selection
 * x_displace y_displace
 * LOCAL: okay - Prompt variable for another trim numbers - Character buffer for numeric input
 * USERS: main
 * CALLS: wncurmov wncurrent
 * wquery channel
 * channelb findme
 * wncollab scanf
 * arange strobe
 * clrol rdcnvert
 * rdoverload joymove
 * rpdsyn th select
 * move_dist pre_gain
 * agen_volt detect_gain
 * skend_conv disable_lim_cro
 * laser_on move_norest
 * laser_off
 * RETURNS: NONE
 * DATE: 24 June 1990
 * PROGRAMMER: Ryp Walters
 * LANGUAGE: Microsoft C Version 5.1
 * OS: MS-DOS Version 3.3
 */

void laser_menu(void)
{
    double r2;
    float resistor_value[9], initial_resistor;
    int serp_loops, choice;
    int double_cut_pos2dist_x, double_cut_pos2dist_y,
serp_cut_pos2dist_x, serp_cut_pos2dist_y;

wnselect(pwin);

aprobe = 999;
bprobe = 999;

choice = 0;

while(choice != 9) /* While the choice is not an exit */
{
    wnscreen(0,-1,-1,0); /* Clear the upper screen */
    wncurmov(2,22);
    wnprefix("What would you like to do?");
    wncurmov(4,20);
    wnprefix(" 1. Get current resistance ");
    wncurmov(5,20);
    wnprefix(" 2. Straight-cut trim ");
    wncurmov(6,20);
    wnprefix(" 3. Double-cut trim ");
    wncurmov(7,20);
    wnprefix(" 4. L-cut trim ");
    wncurmov(8,20);
    wnprefix(" 5. Multiple-line-cut trim (serpentine)");
    wncurmov(9,20);
    wnprefix(" 6. Test routines for the measurement system");
    wncurmov(10,20);
    wnprefix(" 7. Imbedded resistor (not in use)");
    wncurmov(11,20);
    wnprefix(" 8. Change trimming defaults");
    wncurmov(12,20);
    wnprefix(" 9. Exit");
    wncurmov(14,20);
    wnprefix("Option: *");

    wquery(selection, 2, &scan); /* Get the users selection */
    if(*selection == 'g'|| *selection == 'G' || *selection == 49)
        choice = 1;
    if(*selection == 's'|| *selection == 'S' || *selection == 50)
        choice = 2;
    if(*selection == 'd'|| *selection == 'D' || *selection == 51)
        choice = 3;
    if(*selection == 'l'|| *selection == 'L' || *selection == 52)
        choice = 4;
    if(*selection == 'm'|| *selection == 'M' || *selection == 53)
        choice = 5;
    if(*selection == 't'|| *selection == 'T' || *selection == 54)
        choice = 6;
    if(*selection == 'i'|| *selection == 'I' || *selection == 55)
        choice = 7;
    if(*selection == 'c'|| *selection == 'C' || *selection == 56)
        choice = 8;
    if(*selection == 'e'|| *selection == 'E' || *selection == 57)
        choice = 9;

    stop = 0;

    if(choice > 0 & & choice < 6)
    {
        if (aprobe == 999 || bprobe == 999)
            probe_request();
        else
            confirm_probes();

        switch(choice) /* Jump to the correct set of commands */
        {
            
        
    }
case 1: /* Option to find a resistance */
while(choice == 1)
{
    wmscroll(0,-1,-1,0); /* Clear the upper window */
    wnselect(pclist);
    wmscroll(0,-1,-1,0); /* Clear the lower window */
    current_resistor(); /* Get the current resistance */
    wncurmov(4,15); /* See if the user wants to find another resist */
    wprintf("Do you want to find another resistance? (y/n) ");

    wquery(selection, 2, &scan);

    if (*selection != 'n' && *selection != 'N')
    {
        probe_request();
        else
        {
            choice = 0;
        }
        break;
    }
    /* When all is done, go back to the main menu */
}

case 2: /* Option to laser trim a straight cut */
while(choice == 2)
{
    wmscroll(0,-1,-1,0); /* Clear the upper window */
    wnselect(pclist);
    wmscroll(0,-1,-1,0); /* Clear the lower window */
    current_resistor(); /* Get the current resistance */
    resistor_corners(); /* Find out where the resistor is */
    get_wanted_resistor(); /* Figure out what resistance the user wants */
    wnselect(pwin);
    wmscroll(0,-1,-1,0); /* Clear the upper window */
    wncurmov(10,15); /* Get the all ready from the user */
    wprintf("Press START when you are ready.");
    wnselect(pclist);
    wprintf("Press START when you are ready.");
    wait_for_start();

    laser_trim(); /* Do the scribing */
    wnselect(pclist); /* Display the current position */
    wselect(pwin);
    rdposn();
    wprintf("Current position: %6.1f, %6.1f", xpos, ypos);

    wselect(pwin);
    wncurmov(12,15);
    wprintf("The final resistance was %6.1f", resistor);
    wncurmov(14,15); /* See if the user wants another straight cut */
    wprintf("Do you want to do another straight-cut trim? (y/n) ");

    wquery(selection, 2, &scan);

    if (*selection != 'y' && *selection != 'Y')
    {
        choice = 0;
        else
        {
            probe_request();
        }
        break;
    }
    /* When all is done, go back the main menu */
}

while(choice == 3)
{
    wmscroll(0,-1,-1,0); /* Clear the upper window */
    wnselect(pclist);
    wmscroll(0,-1,-1,0); /* Clear the lower window */
    current_resistor(); /* Get the current resistance */
    resistor_corners(); /* Find out where the resistor is */
    get_wanted_resistor(); /* Figure out what resistance the user wants */
    r2 = r1;
    r1 = |r2 * (double)turn_after / 100| + ((double)initial_resistor * (100 - (double)turn_after)/100);
    resistor = (float)r1;
    wnselect(pwin);
    wmscroll(0,-1,-1,0); /* Clear the upper window */
    wncurmov(10,15); /* Get the all ready from the user */
    wprintf("Press START when you are ready.");
    wnselect(pclist);
    wprintf("Press START when you are ready.");
    wait_for_start();

    if (*selection != 'n' && *selection != 'N')
    {
        probe_request();
        else
        {
            choice = 0;
        }
        break;
    }
    /* When all is done, go back to the main menu */
}
laser_trim(); /* Do the scribing */
resistor_value[0] = resistor;
rl = r2;
resistor = (float)rl;
rposln();
double_cut_pos2dist_x = (int)((int)starting_corner_x - (int)xpos + (((int)opp_starting_corner_x - (int)adj_starting_corner_x) * double_cut_ratio/100));
double_cut_pos2dist_y = (int)((int)starting_corner_y - (int)ypos + (((int)opp_starting_corner_y - (int)adj_starting_corner_y) * double_cut_ratio/100));
wcurnovr(0,2);
mvlnorest(double_cut_pos2dist_x, double_cut_pos2dist_y, 4,4);
wait_for_rest();
wcurmovr(1,2);
if(stop != 1)
{
laser_trim();
resistor_value[1] = resistor;
}
wselect plist; /* Display the current position */
wcurmover(0,2);
rposln();
wprintf("Current position: %6li, %6li", xpos, ypos);
wselect pwn;
wcurnov(11,15);
wprintf("The first resistance was %12.0f \n", resistor_value[0]);
wcurmover(12,15);
wprintf("The final resistance was %12.0f \n", resistor_value[1]);
wcurmover(14,15); /* See if the user wants another scribe */
wquery(selection, 2, &scan);
if(*selection != 'y' & selection != 'Y')
choice = 0;
else
probe_request();
} break; /* When all is done, go back the main menu */
case 4: /* Option to cut a circle */
while(choice == 4) /* While the user still wants to scribe */
{
wscroll(0,-1,-1,0); /* Clear the upper window */
wselect plist;
wscroll(0,-1,-1,0); /* Clear the lower window */
current_resistor(); /* Get the current resistance */
initial_resistor = resistor;
resistor_corners(); /* Find out where the resistor is */
get_wanted_resistor(); /* Figure out what resistance the user wants */
r2 = rl;
rl = (r2 * ((double)turn_after / 100) + ((double)initial_resistor * (100 - (double)turn_after)/100);
resistor = (float)rl;
wselect pwn;
wscroll(0,-1,-1,0); /* Clear the upper window */
wcurmovr(10,15); /* Get the all ready from the user */
wprintf("Press START when you are ready.");
wselect plist;
wait_for_start();
laser_trim(); /* Do the scribing */
resistor_value[0] = resistor;
rl = r2;
resistor = (float)rl;
x_displace = (int)(opp_starting_corner_x - adj_starting_corner_x);
y_displace = (int)(opp_starting_corner_y - adj_starting_corner_y);
if(stop != 1)
{
laser_trim();
resistor_value[1] = resistor;
}
wselect plist; /* Display the current position */
wncurmov(0,2);
rdposln();
wprintf("Current position: 611, 611 ",xpos,ypos);

wselect(pwin);
wncurmov(11,15);
wprintf("The first resistance was 12.0f", resistor_value[0]);
wncurmov(12,15);
wprintf("The final resistance was 12.0f", resistor_value[1]);
wncurmov(14,15);
  /* See if the user wants another scribe */
wprintf("Do you want to do another L-cut trim? (y/n) ");

wnquery(selection, 2, &scan);

if(*selection == 'y' && *selection == 'y' )
  choice = 0;
else
  probe_request();
}
break; /* When all is done, go back the main menu */

case 5: /* Option to do a serpentine cut */
  while(choice == 5; /* While the user still wants to laser trim */
    
    wncurmov(0,-1,-1,0);
    wselectplist;
    wncurmov(0,-1,-1,0);
    /* Clear the lower window */
    current_resistor(); /* Get the current resistance */
    initial_resistor = resistor;
    resistor_corners(); /* Find out where the resistor is */
    get_wanted_resistor(); /* Figure out what resistance the user wants */
    r2 = r1;
    r1 = (double)((r2 - (double)initial_resistor) / (double)number_of_serpsc entrega) + (double)initial_resistor;

    resistor = (float)r1;
    wncurmov(0,-1,-1,0);
    wncurmov(10,15);
  /* Get the all ready from the user */
wprintf("Press START when you are ready.");
wselectplist;
  wait_for_start();

  laser_trim(); /* Do the scribing */

  resistor_value[0] = resistor;
wncurmov(0,2);
  rdposln();
  wprintf("Current position: 611, 611 ",xpos,ypos);
  for( serp_loops = 2; serp_loops <= number_of_serpsc entrega + 1; serp_loops++)
    
    r1 = ((r2 - (double)initial_resistor) * (double)serp_loops / (double)number_of_serpsc entrega) + (double)initial_resistor;
    rdposln();
    resistor = (float)r1;
    rdposln();
    if(pow(-1,(double)serp_loops) == 1)
      
      serp_cut_pos2dist_x = (int)((int)adj_starting_corner_x - (int)xpos +
      (int)((int)opp_starting_corner_x - adj_starting_corner_x) *
      serp_loops / number_of_serpsc entrega);

      serp_cut_pos2dist_y = (int)((int)opp_starting_corner_y - adj_starting_corner_y) *
      serp_loops / number_of_serpsc entrega;
    
    else
      
      serp_cut_pos2dist_x = (int)((int)starting_corner_x - (int)xpos +
      (int)((int)opp_starting_corner_x - adj_starting_corner_x) *
      serp_loops / number_of_serpsc entrega);

      serp_cut_pos2dist_y = (int)((int)opp_starting_corner_y - opp_starting_corner_y) *
      serp_loops / number_of_serpsc entrega;
    }
  move_norest(serp_cut_pos2dist_x, 0, 4,0);
  wait_for_rest();
  move_norest(10, serp_cut_pos2dist_y, 0,4);
  wait_for_rest();

185
x_displace = -1 * x_displace;
y_displace = -1 * y_displace;
if (stop := 1;
laser_trim();
resistor_value[serp_loops - 1] = resistor;
}
wnselect(pwln);
for (serp_loops = 0; serp_loops != number_of_serp_cuts; serp_loops++)
wncurmov(13 - number_of_serp_cuts + serp_loops,15);
wnprintf("The resistance at turn %d was %2.0f", 
        serp_loops, resistor_value[serp_loops]);
}
wncurmov(14,15); /* See if the user wants another laser trim */
wnprintf("Do you want to do another serpentine cut? (y/n) ");
wnquery(selection, 2, &scan);
if (*selection := 'y'&& *selection := 'Y')
    choice = 0;
else
    probe_request();
break; /* When all is done, go back the main menu */
case 6: /* Option to do test the measurement system */
    measure_test_screen();
    measure_test();
    break;
case 7: /* Option to do trim an imbedded resistor */
    break;
}
if (choice := 8)
    change_defaults();
}

******************************************************************************
* MODULE: change_defaults
* PURPOSE: To change the defaults for the different kinds of trims
* PARAMETERS: GLOBAL: plist
*             LOCAL: okay - Prompt variable for another trim
*             USERS: main
* CALLS: wncurmov wnprintf
* RETURNS: NONE
* DATE: 26 October 1990
* PROGRAMMER: Ryp Walters
* LANGUAGE: Microsoft C Version 5.1
* OS: MS-DOS Version 3.3
* DATE INITIALLS MODIFICATION
******************************************************************************

void change_defaults(void)
{
    int choice;
    char numbers[6];
    wnselect(pwln);
    wnscroll(0,-1,-1,0); /* Clear the upper screen */
    /* Display the main menu */
}
wncurmov(2,10);
wnprintf(" 1. Trim speed");
wncurmov(3,10);
wnprintf(" 2. Double-cut or L-cut turn percentage");
wncurmov(4,10);
wnprintf(" 3. Second cut (Double-cut) percentage distance");
wncurmov(5,10);
wnprintf(" 4. Perpendicular cuts - abort distance");
wncurmov(6,10);
wnprintf(" 5. Adjacent (parallel) cuts - abort distance");
wncurmov(7,10);
wnprintf(" 6. Number of serpentine cuts");
wncurmov(8,10);
wnprintf(" 7. Restore defaults");
wncurmov(10,10);
wnprintf(" 9. Return to Laser Menu");

choice = 0;
while(choice != 9)
{
    wncurmov(2,60);
    wnprintf("%4f", trim_speed);
    wncurmov(3,62);
    wnprintf("%3f", turn_after);
    wncurmov(4,62);
    wnprintf("%3f", double_cut_ratio);
    wncurmov(5,60);
    wnprintf("%3f", abort_dist_straight);
    wncurmov(6,60);
    wnprintf("%3f", abort_dist_lcut);
    wncurmov(7,63);
    wnprintf("%2f", number_of_serp_cuts);
    wncurmov(12,20);
    wnprintf("Option: ");
    wncurmov(32,28);
    wquery(selection, 2, &scan);  /* get the user's selection */
    /* Translate the possibilities into one number */
    if(*selection == 't' || *selection == 'T' || *selection == 49)
        choice = 1;
    if(*selection == 'd' || *selection == 'D' || *selection == 50)
        choice = 2;
    if(*selection == 's' || *selection == 'S' || *selection == 51)
        choice = 3;
    if(*selection == 'p' || *selection == 'P' || *selection == 52)
        choice = 4;
    if(*selection == 'a' || *selection == 'A' || *selection == 53)
        choice = 5;
    if(*selection == 'n' || *selection == 'N' || *selection == 54)
        choice = 6;
    if(*selection == 'r' || *selection == 'R' || *selection == 55)
        choice = 7;
    if(*selection == 'e' || *selection == 'E' || *selection == 57)
        choice = 9;
    switch(choice)
    { /* Jump to the correct set of commands */
        case 1:
            /* Option to move a distance */
            trim_speed = 100;
            while(trim_speed < 0 || trim_speed > 4)
            {
                wncurmov(15,10);
                /* Get the first probe number */
                wnprintf("Enter the trimming speed (between .125 and 4): ");
                wquery(numbers,5,&scan);
                trim_speed = transform_numbers;  /* Transform the first probe number */
            }
            wncurmov(2,60);
            wnprintf("%4f",trim_speed);
            wncurmov(15,10);
            /* Clear out the question area */
case 2: /* Option to move a distance */
    turn_after = 999;
    while( turn_after < 20 || turn_after > 100)
    {
        wncurmov(15,10);
        /* Get the first probe number */
        wpprintf("Enter the percentage of final value ");
        wncurmov(16,10);
        wpprintf("to turn at (between 20 and 100): ");
        wquery(numbers,4,&scan);
        turn_after = (int)(transform(numbers)); /* Transform the 1st probe number */
    }
    wncurmov(3,62);
    wpprintf("%31",turn_after);
    wncurmov(15,10);
    /* Clear out the question area */
    wpprintf(" ");
    wncurmov(16,10);
    /* Clear out the question area */
    wpprintf(" ");
    break;
    /* When all is done, go back to the main menu */

case 3: /* Option to move a distance */
    double_cut_ratio = 999;
    while( double_cut_ratio < 0 || double_cut_ratio > 100)
    {
        wncurmov(15,10);
        /* Get the first probe number */
        wpprintf("Enter the percentage of distance ");
        wncurmov(16,10);
        wpprintf("to turn for 2nd cut (between 0 and 100): ");
        wquery(numbers,4,&scan);
        double_cut_ratio = (int)(transform(numbers));
        /* Transform the first probe number */
    }
    wncurmov(4,62);
    wpprintf("%31",double_cut_ratio);
    wncurmov(15,10);
    /* Clear out the question area */
    wpprintf(" ");
    wncurmov(16,10);
    /* Clear out the question area */
    wpprintf(" ");
    break;
    /* When all is done, go back to the main menu */

case 4: /* Option to move a distance */
    abort_dist_straight = 0;
    while( abort_dist_straight < 1)
    {
        wncurmov(15,10);
        /* Get the first probe number */
        wpprintf("Enter the abort distance from approaching side: ");
        wncurmov(16,10);
        abort_dist_straight = (int)(transform(numbers)); /* Transform the first probe number */
    }
    wncurmov(5,60);
    wpprintf("%51",abort_dist_straight);
    wncurmov(15,10);
    /* Clear out the question area */
    wpprintf(" ");
    break;
    /* When all is done, go back to the main menu */

case 5: /* Option to move a distance */
    abort_dist_lcut = 0;
    while( abort_dist_lcut < 1)
    {
        wncurmov(15,10);
        /* Get the first probe number */
        wpprintf("Enter the abort distance from approaching resistor pad: ");
        wncurmov(16,10);
        abort_dist_lcut = (int)(transform(numbers)); /* Transform the first probe number */
    }
    wncurmov(6,60);
    wpprintf("%51",abort_dist_lcut);
    wncurmov(15,10);
    /* Clear out the question area */
    wpprintf(" ");
    break;
    /* When all is done, go back to the main menu */

case 6: /* Option to move a distance */
    number_of_serp_cuts = 999;
    while( number_of_serp_cuts < 1 || number_of_serp_cuts > 20)
```c
{
   wncurmov(15,10);    /* Get the first probe number */
   wnprintf("Enter the number of serpentine cuts (between 1 and 20): ");
   wnquery(numbers,3,&scan);
   number_of_serp_cuts = (int)(transform(numbers));
   /* Transform the first probe number */
}
wncurmov(7,63);
wnprintf("%21", number_of_serp_cuts);
wncurmov(15,10);    /* Clear out the question area */
wnprintf(" ");
break;               /* When all is done, go back to the main menu */

case 7:
   number_of_serp_cuts = 5;
   abort_dist_lcut = 50;
   abort_dist_straight = 50;
   double_cut_ratio = 20;
   turn_after = 80;
   trim_speed = 0.125;
   break;
}
}

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

MODULE: measure_test_screen
PURPOSE: To make the basic screen for testing with the measurement system
PARAMETERS:  GLOBL: plist
              LOCAL: okay - Prompt variable for another trim
              USERS: main
              CALLS: wncurmov  wnprintf

RETURNS:  NONE

DATE: 26 October 1990
PROGRAMMER: Ryp Walters
LANGUAGE: Microsoft C Version 5.1
OS: MS-DOS Version 3.3

------- INITIALS ------- MODIFICATION
--------------------------------------------------------------------------

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

void measure_test_screen(void)
{
   wnselect(pwin);
   wncroll(0,-1,-1,0);    /* Clear the upper screen */
   /* Display the main menu */
   wncurmov(1,10);
   wnprintf(" 1. Define test values");
   wncurmov(2,10);
   wnprintf(" 2. Probe");
   wncurmov(3,10);
   wnprintf(" A Probe");
   wncurmov(4,10);
   wnprintf(" B Probe");
   wncurmov(5,10);
   wnprintf(" Test Resistance Value");
   wncurmov(6,10);
   wnprintf(" Range");
   wncurmov(7,10);
   wnprintf(" Standard");
   wncurmov(8,10);
   wnprintf(" Generator Voltage");
   wncurmov(9,10);
   wnprintf(" Detectors Voltage");

   /* Display the test values */
   /* Perform the test operations */
   /* Display the test results */
   /* Save the test results */
   /* Display the confirmation message */
}
```
wnprintf('');
wnprintf('');
wnprintf('');
wnprintf('');
wnprintf('');
wnprintf('');
wnprintf('');
wnprintf('');
wnprintf('');
wnprintf('');
wnprintf('');
wnprintf('');

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

MODULE: measure_test
PURPOSE: To play with the measurement system
PARAMETERS: GLOBL: aprobe bprobe
            convertor end_conv
            gain gen_volt
            overload plst
            posx pcy
            pre_amp pwin
            range resister
            scan standard
            xpos ypos
            r1 selection
            x_displace y_displace
LOCAL: okay - Prompt variable for another trim
        numbers - Character buffer for numeric input
USERS: main
CALLS: wncurmov wprintf
        wnquery channels
        channelb findme
        wnscroll scanf
        arange strobe
        cirol rconver
        rdoverload jyomove
        rdposalin wnselct
        move_dist pre_gain
        aven_volt detect_gain
        skend_conv disable_lim_cro
        laser_on move_noreast
        laser_off

RETURNS: NONE
DATE: 24 June 1990
PROGRAMMER: Ryp Walters
LANGUAGE: Microsoft C Version 5.1
OS: MS-DOS Version 3.3
********************************************************************************
********************************************************************************

void measure_test(void)
{
    int okay, choice;
    char numbers[14];
choice = 0;
while(choice != 9)
    measure_test_screen();
display_feedback();
wnormap(17,20);
wnprintf("Option: ");
wnormap(17,28);
wnquery(selection, 2, &scan);   /* Get the users selection */
    /* Translate the possibilities into one number */
if(*selection == 'd' || *selection == 'D' || *selection == 49)
    choice = 1;
if(*selection == 's' || *selection == 'S' || *selection == 50)
    choice = 2;
if(*selection == 'c' || *selection == 'C' || *selection == 51)
    choice = 3;
if(*selection == 'f' || *selection == 'F' || *selection == 52)
    choice = 4;
if(*selection == 'e' || *selection == 'E' || *selection == 57)
    choice = 9;
if(choice > 0 && choice < 5)
    if(aprobe == 999 || bprobe == 999)
        probe_request();
    else
        confirm_probes();
switch(choice)  /* Jump to the correct set of commands */
{
    case 1:  /* Option to move a distance */
        measure_test_screen();
        change_measure_test_values();
        break;  /* When all is done, go back to the main menu */
    case 2:  /* Option to move a distance */
        strobe();
        break;
    case 3:  /* Option to move a distance */
        colrll();
        break;
    case 4:  /* Option to move a distance */
        findme();
        rl = resistor;
        render();
        break;
    }
wnselect(plist);
wnscroll(0,-1,-1,0); /* Clear the lower window */
wnselect(pwin);
/* ***********************************************************/

**
** MODULE: change_measure_test_values
**
** PURPOSE: To play with the measurement system
**
** PARAMETERS: ** GLOBAL:
** aprobe bprobe
** converter end_conv
** gain gen_volt
**
void change_measure_test_values(void)
{
    int okay, choice;
    char numbers[14];

    wncurmov(1,10);
    wnprompt(" ");
    wncurmov(2,10);
    wnprompt(" ");
    1. Change*);
    wncurmov(3,10);
    wnprompt(" ");
    2. Change*);
    wncurmov(4,10);
    wnprompt(" ");
    3. Change*);
    wncurmov(7,10);
    wnprompt(" ");
    4. Change*);
    wncurmov(8,10);
    wnprompt(" ");
    5. Change*);
    wncurmov(9,10);
    wnprompt(" ");
    6. Change*);
    wncurmov(10,10);
    wnprompt(" ");
    7. Change*);
    wncurmov(11,10);
    wnprompt(" ");
    8. Change*);
    wncurmov(12,10);
    wnprompt(" ");
    9. Exit test value definitions*);
    /* Translate the possibilities into one number */
    while(choice := 9)
    {
        wncurmov(17,20);
wnprintf("Option: ");
wncurmov(17,28);
wnquery(selection, 2, &scan);  /* Get the users selection */
    if(*selection == 'g'|| *selection == 'G' || *selection == 49)    
choice = 1;
    if(*selection == 's'|| *selection == 'S' || *selection == 50)    
choice = 2;
    if(*selection == 'd'|| *selection == 'D' || *selection == 51)    
choice = 3;
    if(*selection == 'l'|| *selection == 'L' || *selection == 52)    
choice = 4;
    if(*selection == 'm'|| *selection == 'M' || *selection == 53)    
choice = 5;
    if(*selection == 't'|| *selection == 'T' || *selection == 54)    
choice = 6;
    if(*selection == 'i'|| *selection == 'I' || *selection == 55)    
choice = 7;
    if(*selection == 'c'|| *selection == 'C' || *selection == 56)    
choice = 8;
    if(*selection == 'e'|| *selection == 'E' || *selection == 57)    
choice = 9;

if(choice > 0 && choice < 8) {
    switch(choice)      /* Jump to the correct set of commands */
    {      /* Option to move a distance */
        case 1:
            approve = 999;
            while(approve < 0 || approve > 40)
            {    /* Get the first probe number */
                wncurmov(17,10);
                wnprintf("Enter the A probe number (between 1 and 40): ");
                wnquery(numbers,3,&scan);
                approve = (int)(transform(numbers));
            }      /* Transform the first probe number */

            channela();

            wncurmov(3,60);
            wnprintf("%21s",approve);
            wncurmov(17,10);  /* Get the first probe number */
            wnprintf(" ");
            break;  /* When all is done, go back to the main menu */

        case 2:      /* Option to move a distance */
            bprobe = 999;
            while(bprobe < 0 || bprobe > 40)
            {    /* Get the first probe number */
                wncurmov(17,10);
                wnprintf("Enter the B probe number (between 1 and 40)");
                wnquery(numbers,3,&scan);
                bprobe = (int)(transform(numbers));
            }      /* Transform the first probe number */

            channelb();

            wncurmov(4,60);
            wnprintf("%21s",bprobe);
            wncurmov(17,10);  /* Get the first probe number */
            wnprintf(" ");
            break;  /* When all is done, go back to the main menu */

        case 3:      /* Option to move a distance */
            resistor = 0;
            while(resistor == 0)
            {    /* Get the first probe number */
                wncurmov(17,10);
                wnprintf("Enter the Resistor value to compare against");
                wnquery(numbers,14,&scan);
            }
resistor = transform(numbers); /* Transform the first probe number */
}

rl = resistor;

range;
wncurmov(5,59);
wnprint("%-12.0f",resistor);
wncurmov(9,61);
wnprint("%li",range);
wncurmov(7,54);
wnprint("%-13.12f",standard);
wncurmov(17,10);

break; /* When all is done, go back to the main menu */

case 4:

/* Option to move a distance */
gen_vol = 100;
while( gen_vol > 8 || gen_vol < -8 )

wncurmov(17,8);

/* Get the first probe number */
wnprint(Enter the Generator voltage {+/-.125,.25,.5,1,...,8} )
wnquery(numbers,7,&scan);
gen_vol = transform(numbers);
/* Transform the first probe number */
}
agen_vol();
wncurmov(8,57);
wnprint("%-7.3f",gen_vol);
wncurmov(17,10);
/* Get the first probe number */
break; /* When all is done, go back to the main menu */

case 5:
/* Option to move a distance */
gain = 513;
while( gain < 0 || gain > 512 )

wncurmov(17,10);
/* Get the first probe number */
wnprint(Enter the detector gain (0 to 511.875) )
wnquery(numbers,7,&scan);
gain = transform(numbers);
/* Transform the first probe number */
}
detect_gain();
wncurmov(9,57);
wnprint("%-7.3f",gain);
wncurmov(17,10);
/* Get the first probe number */
break; /* When all is done, go back to the main menu */

/* Option to move a distance */
case 6: pre_amp = 9;
while( pre_amp < 1 || pre_amp > 8 )

wncurmov(17,10);
/* Get the first probe number */
wnprint(Enter the Pre-amp gain [1 or 8] )
wnquery(numbers,2,&scan);
pre_amp = (int)(transform(numbers));
/* Transform the first probe number */
}
pre_gain();
wncurmov(10,51);
wnprint("%-11",pre_amp);
wncurmov(17,10);
/* Get the first probe number */
wnprint( /*
break; /* When all is done, go back to the main menu */

/* Option to move a distance */
case 7:
/* Get the first probe number */
compare = -1;
while( compare < 0 || compare > 4095 )

wncurmov(17,10);
/* Get the first probe number */
wnprint(Enter the Comparator value [0 to 4095] )
wnquery(numbers,5,&scan);
compare = (int)(transform(numbers));
/* Transform the first probe number */
}
acomparator();
wncurmov(11,58);
  wnprintf("%4u",compare);
wncurmov(17,10); /* Get the first probe number */
  wnprintf(" ");
break; /* When all is done, go back to the main menu */
case 8: /* Option to move a distance */
delay = -1;
while ( delay < 0 )
{
  wncurmov(17,10); /* Get the first probe number */
  wnprintf("Enter the measurement delay time");
  wquery(numbers,5,&scan);
  delay = (int)(transform(numbers)); /* Transform the first probe number */
}
wncurmov(12,58);
  wnprintf("%4u",delay);
wncurmov(17,10); /* Get the first probe number */
  wnprintf(" ");
break; /* When all is done, go back to the main menu */
}
display_feedback();
}

******************************************************************************
#
# MODULE: display_feedback
#
# PURPOSE: To display the measurement system feedback in the bottom window
#
# PARAMETERS: GLOBL: aprobe convertor gain overload posx pre_amp range scan xpos rl x_displace
#               bprobe end_conv gen_volt plist posy pwin resistor standard ypos selection y_displace
# LOCAL: okay numbers - Prompt variable for another trim
#        - Character buffer for numeric input
#
# USERS: main
#
# CALLS: wncurmov wnprintf
#         wquery channela findsw
#         wnscroll scanf
#         arange strobe
#         cirol rdconvert
#         rdoverload joymove
#         rdpositin wselect
#         move_dist pre_gain
#         a gen_volt detect_gain
#         skend_conv disable_lim_cro
#         laser_on move_norest
#         laser_off
#
# RETURNS: NONE
#
# DATE: 24 June 1990
# PROGRAMMER: Ryp Walters
# LANGUAGE: Microsoft C Version 5.1
# OS: MS-DOS Version 3.3
# _______________________________________________________________
void display_feedback(void)
{
    int okay;

    wncurmov(2,60);
    wpprintf("%21",aprobe);
    wncurmov(3,60);
    wpprintf("%21",bprobe);
    wncurmov(4,59);
    wpprintf("%12.0f",resistor);
    wncurmov(5,61);
    wpprintf("%11",range);
    wncurmov(6,54);
    wpprintf("%13.12t",standard);
    wncurmov(7,57);
    wpprintf("%7.3f",gen_volt);
    wncurmov(8,57);
    wpprintf("%7.3f",gain);
    wncurmov(9,61);
    wpprintf("%11",pre_amp);
    wncurmov(10,58);
    wpprintf("%4",compare);
    wncurmov(11,58);
    wpprintf("%4",delay);

    okay = 0;
    wnscroll(plist);
    wncurmov(0,-1,-1,0); /* Clear the lower window */
    while(okay == 0)
    {
        okay = 1;
        rdconvert();
        wncurmov(0,5);
        wpprintf("Converter %i",convertor);
        rdoverload();
        wncurmov(1,5);
        wpprintf("Overload %i",overload);
        skend_conv();
        wncurmov(2,5);
        wpprintf("End Conversion :");
        if(end_conv != 0)
        {
            wpprintf(" WAIT ");
            okay = 0;
        }
        else
            wpprintf(" Ready ");
        skend_delay();
        wncurmov(0,25);
        wpprintf("End Delay ");
        if(end_delay != 0)
        {
            wpprintf(" WAIT ");
            okay = 0;
        }
        else
            wpprintf(" Ready ");
        skready();
        wncurmov(1,35);
        wpprintf("Ready Flag ");
        if(ready != 0)
        {
            wpprintf(" WAIT ");
            okay = 0;
        }
        else
            wpprintf(" Ready ");
}
skstrobe();
wncurmov(2,35);
wnprintf("Strobe Done : ");
if(strobe_done != 0)
{
    wnprintf(" WAIT ");
    okay = 0;
}
else
    wnprintf(" Ready ");
}
wnselect(pwin);

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

MODULE: laser_trim

PURPOSE: To perform a resistor trim

PARAMETERS:    GLOBL: aprobe  bprobe
                convertor  end_conv
                gain  gen_volt
                overload  plist
                posx  posy
                pre_amp  pwin
                range  resistor
                scan  standard
                xpos  ypos
                rl  selection
                x_displace  y_displace

LOCAL: okay - Prompt variable for another trim
        numbers - Character buffer for numeric input

USERS: main

CALLS: wncurmov  wnprintf
        wquery  channela
        channelb  findme
        wncscroll  scanf
        arrange  strobe
        clrval  rcconvert
        rdoverload  joymove
        rdpositn  wnselect
        move_dist  pre_gain
        aget_volt  detect_gain
        sksend_conv  disable_lim_cro
        laser_on  move_norest
        laser_off

RETURNS: NONE

DATE: 24 June 1990
PROGRAMMER: Ryp Walters
LANGUAGE: Microsoft C Version 5.1
OS: MS-DOS Version 3.3

DATE    INITIALS    MODIFICATION
-------    --------    ---------------------

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

void laser_trim(void)
{
    int okay;
    char numbers[14];

    stop = 0;
    rl = resistor;  /* Store the value to trim to */
    arrange();  /* Send the value to the laser */
    strobe();  /* Strobe and clear the system for setup */
cirlo();
rdconvert();
rdoverload();
/* Get the measurement system feedback */

pre_amp = 1;
pre_gain();
/* Set and send the pre-amp gain value (1 or 8) */
gain = 1;
detect_gain();
/* Set and send the detector gain */
gen_vol = 8;
agen_vol = 1;
strobe();
cirlo();
/* Strobe and clear the system for setup */

skend_conv();
while(end_conv != 0 )
{
    skend_conv();
}
/* Wait for the measurement system to finish */

rdconvert();
rdoverload();
/* Get the measurement system feedback */

/* Display the current values and feedback */

/* wnselect{p1st};

/* wncurmov(1,25);
wnprintf("Resistor=\%10.0f Range=\%1 Stand=\%8f",rl,range,standard);
wncurmov(2,25);
wnprintf("Generator volt=\%10.0f Gain=\%10.0f",gen_vol,gain);
wncurmov(3,25);
wnprintf("Convert=\%1 Overload=\%4x",convertor,overload);
*/

gain = .5;

disable_lim_cro();
/* Disable the comparator laser interrupt */
laser_on();
/* Turn on the laser */
movemorest(x_displace, y_displace, trim_speed, trim_speed);
/* Start moving into the resistor */

while(gain != 4 && stop != 1)
{
    if(check_for_stop() == 1)
    {
        laser_off();
        movemorest(0,0,0,0);
        stop = 1;
    }
    gain = gain * .5;
detect_gain();
strobe();
cirlo();

    skend_conv();
    while(end_conv != 0 )
    {skend_conv();
    rdoverload();
    rdconvert();
    }
    /* Get the measurement system feedback */
    /* While we are out of range */

    cirlo();
    /* Wait for the measurement system to finish */

    if(check_for_stop() == 1)
    {
        laser_off();
        movemorest(0,0,0,0);
        stop = 1;
        overload = 0x80ff;
    }
    /* Check the stop button */
    /* If STOP button, turn off the laser, stop */
    /* moving, reset laser status flag, and */
    /* return with an error flag. */
    /* Adjust and send the gain */
    /* Strobe and clear the system for setup */
    /* Wait for the measurement system to finish */
    /* Get the measurement system feedback */
    /* While we are not in range with a high gain */

    see_if_in_range = 0;
    /* See if we are in range yet */
    /* See if we are at a high gain */

    cirlo();
    /* Clear the system overload */
#include <stdio.h>
#include <math.h>

#define PI 3.14159

float main(float x)
{
    float y = PI * x / 180;
    return y;
}

int main
{ }

if(stop == 0)
{ 
    skend_conv(); /* Wait for the measurement system to finish */
    while(end_conv != 0 )
    skend_conv(); /* Get the measurement system feedback */
    rdconvert();
}

else
{

    convertor = 0;

    while(convertor < -300)
    { 
        clr1(); /* While we are not within tolerance */
        skend_conv(); /* Clear the system overload */
        while(end_conv != 0)
        skend_conv(); /* Wait for the measurement system to finish */
        rdconvert(); /* Get the measurement system feedback */
        if(check_for_stop() == 1)
        { /* Check the stop button */
            stop = 1; /* If STOP button, */
            convertor = 0; /* return with an error flag. */
        }
        /* See if we are within tolerance */
        laser_off(); /* Turn off the laser */
        move_norest(0,0,0,0); /* STOP MOVING */

        findme(); /* Get the current resistance */
    }
}

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

* MODULE:   probe_request
* PURPOSE: To set the probe to the resistor is across
* PARAMETERS:   GLOBAL: aprobe bprobe
*               scan
* LOCAL: numbers - Character buffer for numeric input
* USERS:       laser_menu
* CALLS:       wncurmov wncurmov
*               wquery wquery
*               channela channela
*               wstorage wstorage
* RETURNS:     NONE
* DATE:        18 September 1991
* PROGRAMMER:  Ryp Walters
* LANGUAGE:    Microsoft C Version 5.1
* OS:          MS-DOS Version 4.1
* DATE INITIALLY MODIFICATION
* ******************************************************************************/

void probe_request(void)
{
    char numbers[14];
    wnscroll(0,-1,-1,0); /* Clear the upper screen */
    wncurmov(8,15); /* Get the first probe number */
    wprintf("Enter the A probe number: ");
    wquery(numbers,3,scant);
    aprobe = (int)(transform(numbers)); /* Transform the first probe number */
    wncurmov(10,15); /* Get the second probe number */

    wncurmov(6,15); /* Get the second probe number */
}
void confirm_probes(void)
{
    int okay;

    okay = 0;

    while(okay == 0)
    {
        wmscroll[0,0,-1,-1,0];   /* Clear the upper window */
        wncurmov[3,22];
        wprintf("The probes are currently set at:");
        wprintf("Probe A: %d",aprobe);
        wncurmov[9,35];
        wprintf("Probe B: %d",bprobe);
        wncurmov[13,15];
        wprintf("Are you sure that these are the right probes? (y/n) retal");
        wquery(selection, 2, &scan);  /* Get the users selection */
        if(*selection == 'y' && *selection == 'Y')
            probe_request();
        else
            okay = 1;
    }
}
void current_resistor(void)
{
    findme(); /* Get the resistors current value */
    r1 = resistor;
    wnselect(p1list);
    wncurrmov(2, 5); /* Display the current resistance */
    wprintf("The current resistance is %.0f", r1);
    wnselect(pwin);
}

MODULE: resistor_corners
PURPOSE: Check that the right probes are being tested
PARAMETERS: GLOBL: starting_corner(x,y) opp_starting_corner(x,y)
adj_starting_corner(x,y) selection
scan
LOCAL: okay - Prompt variable for another trim

USERS: laser_menu

CALLS: wncurrmov wprintf
wnselect wnscroll
joymove rdposition
abs wquery
move_dist

RETURNS: NONE

DATE: 18 September 1991
PROGRAMMER: Ryp Walters
LANGUAGE: Microsoft C Version 5.1
OS: MS-DOS Version 4.1

DATE INITIALS MODIFICATION
-------- -------- ---------------
void resistor_corners(void)
{
    int okay, scan;
    char selection[2];

    okay = 0;
    while(okay == 0)
    {
        wnsselect(pwin);
        wnscroll(0, -1, -1, 0);
        wncurmov(5,15);
        /* Tell the user to use the joy to go to pos 1 */
        wmprintf("Use the joystick to move to the corner");
        wncurmov(6,15);
        wmprintf(" you want the laser to first cut into.");
        wncurmov(8,15);
        wmprintf(" Use the laser panel button to load the position.");
        wncurmov(12,15);
        wmprintf(" You will have a chance to change wrong positions");
        wncurmov(13,15);
        wmprintf(" after all have been entered.");
        joymove();
        /* Let the user use the joystick */
    readposin();
    starting_corner_x = xpos;
    starting_corner_y = ypos;
        wnsselect(plist);
        /* Display the first position. */
        wncurmov(0,40);
        wmprintf(" Position #1: %6d, %6d", starting_corner_x, starting_corner_y);
        wnsselect(pwin);
        /* Tell the user to use the joy to go to pos 2. */
        wncurmov(5,15);
        wmprintf(" Use the joystick to move to the position");
        wncurmov(6,15);
        wmprintf(" directly across the resistor.");
        joymove();
        /* Let the user use the joystick */
    readposin();
    adj_starting_corner_x = xpos;
    adj_starting_corner_y = ypos;
        wnsselect(plist);
        /* Display the first position. */
        wncurmov(1,40);
        wmprintf(" Position #2: %6d, %6d", adj_starting_corner_x, adj_starting_corner_y);
        x_displace = (int)(adj_starting_corner_x - starting_corner_x);
        /* Check if the positions are okay */
        y_displace = (int)(adj_starting_corner_y - starting_corner_y);
        wnsselect(pwin);
        /* Tell the user to use the joy to go to pos 2. */
        if(abs(x_displace) > 10 && abs(y_displace) > 10)
        {
            wnscroll(0,-1,-1,0);
            /* Clear the upper window */
            wncurmov(8,15);
            wmprintf(" Adjust the base so that the cutting direction");
            wncurmov(9,15);
            wmprintf(" is either horizontal or vertical.");
            wncurmov(12,15);
            wmprintf(" Press ENTER when the base has been adjusted");
            wquery(&selection,2,&scan);
        }
        else
        {
            wncurmov(5,15);
            wmprintf(" Use the joystick to move to the position");
            wncurmov(6,15);
            wmprintf(" catty-corner to the first corner.");
            joymove();
            /* Let the user use the joystick */
        }
rdposition(); /* Store the current position as pos 2 */
opp_starting_corner_x = xpos;
opp_starting_corner_y = ypos;

wnselect(plist); /* Display the first position. */
wncurmov(2,40);
wprintf(“Position #3: %d, %d”,
    opp_starting_corner_x, opp_starting_corner_y);

wnselect(pwin); /* Ask the user if the positions are okay */
wscroll(0,-1,-1,0); /* Clear the upper window */
wncurmov(5,5);
wprintf(“Are these positions correct? (y/n)”);

wquery(‘selection’,2,&scan);

if(‘selection == ’y’|| ’selection == ’Y’) /* If user says okay, set the flag */
    okay = 1;
}

/* move back to the first position */
move_dist(-1*(int)(opp_starting_corner_x - starting_corner_x),
    -1*(int)(opp_starting_corner_y - starting_corner_y), 8, 8);

******************************************************************************

* MODULE: get_wanted_resistor
* PURPOSE: Get the value that the user wants the resistor trimmed to
* PARAMETERS: GLOBL: r1 resistor
* LOCAL: NONE
* USERS: laser_menu
* CALLS: wncurmov wprintf
* wselect findme
* RETURNS: NONE
* DATE: 18 September 1991
* PROGRAMMER: Ryp Walters
* LANGUAGE: Microsoft C Version 5.1
* OS: MS-DOS Version 4.1
******************************************************************************

void get_wanted_resistor(void)
{
    int okay;
    char numbers[14];

    findme(); /* Get the resistors current value */
    r1 = resistor;

    okay = 0; /* Assume that the inputted value is incorrect */
    while(okay == 0) /* While the value is incorrect */
    {
        wselect(pwin);
        wscroll(0,-1,-1,0); /* clear the upper window */
        wncurmov(5,5); /* Display the current resistance */
        wprintf(“The current resistance is %.0f”, r1);
        wncurmov(8,8);
        wprintf(“Tell the user to value to trim to”);
        wprintf(“What resistance would you like it to be?”);
    }
wnquery(numbers,14,&scan);
resistor = transform(numbers);    /* Transform the resistance to a number */
if( resistor < rl)                /* If the resistor is greater than the value */
    {                            /* that the user wants to trim to. */
        wncurmov(10,5);          /* Tell the user that it isn't possible */
        wnprintf("You can not get that resistance from this resistor!!!");
        wncurmov(11,6);
        wnprintf("Press ENTER to continue");
        wnquery(selection,2,&scan);
    }
else                                /* Otherwise all the values are okay */
    {                                /* Check to see if all the values are correct */
        okay = 1;
    }
rl = (double)resistor;
}
/* MEASINCD.C */

/*

MODULE: FINOME

PURPOSE: Main control system routines.

PARAMETERS: GLOBL: convotor end_conv

gain gen_volt

lastandard lstandard

overload pre_amp

rl resistor

standard

LOCAL: bottoml - Place holder for the current resistor

resistor_incr - Increment change for correct resistance

conv_old - Old convertor value

det_incr - Detector Increment

s_hex - New standard value

s_hex_old - Old standard value

USERS: laser_trim measure_test

CALLS: arange pre_gain

agen_volt detect_gain

strobe cirol

skend_conv rdoverload

rdconvert abs

RETURNS: NONE

DATE: 24 June 1990

PROGRAMMER: Ryp Walters

LANGUAGE: Microsoft C Version 5.1

OS: MS-DOS Version 3.3

DATE INITIALS MODIFICATION

===============================================================================

void findme()
{
    unsigned int s_hex, s_hex_old;
    float resistor_incr, det_lacr, bottoml;
    int conv_old;

    resistor = 500;
    /* Set and send an initial resistance

    range();
    pre_amp = 1;
    /* Set and send an initial preamp gain

    pre_gain();
    gain = 1;
    /* Set and send an initial detector

    detect_gain();

    return;
}

```
gen_volt = 8; /* Set and send an initial generator
voltage */
agen_volt();
strobe(); /* Strobe and clear the system for set-up */
crol();
skend_conv(); /* Wait for the measurement system to finish */
while(end_conv != 0)
    skend_conv();
rdoverload(); /* Get the system feedback */
rdconvert();
resistor_incr = 2; /* Set the initial resistor
incrementation */
while(overload != 0x8FF) /* While we are still out of range */
    
    resistor = (long int)(resistor*resistor_incr); /* Increment the resistance */
    arrange(); /* Send the new resistance */
    strobe(); /* Strobe and clear the system for set-up */
    crol();
    skend_conv(); /* Wait for the measurement system to finish */
    while(end_conv != 0)
        skend_conv();
    rdoverload(); /* Get the system feedback */
    rdconvert();
    if(overload & 0x800) /* If we passed it, */
        
        resistor_incr = .97; /* Reset the incrementation */
    
    /* See if we are in range yet */
    resistor_incr = 1.25; /* Reset the incrementation */
    conv_old = 0x47; /* Assume that we are at the far end of close */
while(conv_old == convertor) /* While we are still not within guessing range */
    resistor = (long int)(resistor*resistor_incr); /* Increment the resistance */
    arrange(); /* Send that resistance */
    strobe(); /* Strobe and clear the system for set-up */
    crol();
    skend_conv(); /* Wait for the measurement system to finish */
    while(end_conv != 0)
        skend_conv();
    rdconvert(); /* Get the system feedback */
    
    /* Check if we are close enough to guess */
    det_incr = 2; /* Set the detector gain increment */
    s_hex_old = 0; /* Set the initial difference in values */
s_hex = (hstandard*16) + lstandard;
gain = 1; /* Initialize the gain */
while(abs(s_hex - s_hex_old) > 0x001) /* While our last value put us closer to home */
    
    gain = (int)(gain*det_incr); /* Increment the gain */
    detect_gain(); /* Send the new gain */
    bottom = {convertor/(1024*gain*gen_volt)}; /* Make a new educated guess */
    resistor = 1 + (long int)(rl*(bottom + standard)); /* Strobe and clear the system for set-up */
    arrange(); /* Send the new guess */
    strobe(); /* Strobe and clear the system for set-up */
    crol();
    skend_conv(); /* Wait for the measurement system to finish */
    while(end_conv != 0)
        skend_conv();
    rdoverload(); /* Get the system feedback */
    rdconvert();
    s_hex_old = s_hex; /* Figure out the new feedback values */
    s_hex = (hstandard*16) + lstandard;
    
    /* Are we there yet */
```c
void astandard()
{
    int high = 0, low = 0;
    float s1;

    s1 = standard;                   /* Put the value into a place holder */
    standard = 0;
    high = high|0x80;                /* Figure out the right code to send */

    if(s1 > .5)
    {
        s1 = s1 - .5;
        standard = .5;
        high = 0x80;
    }

    if(s1 > .25)
    {
        s1 = s1 - .25;
        standard = standard + .25;
        high = high|0x40;
    }

    if(s1 > .125)
    {
        s1 = s1 - .125;
        standard = standard + .125;
        high = high|0x20;
    }

    if(s1 > .0625)
    {
        s1 = s1 - .0625;
        standard = standard + .0625;
        high = high|0x10;
    }

    if(s1 > .03125)
    {
        s1 = s1 - .03125;
        standard = standard + .03125;
        high = high|0x8;
    }

    if(s1 > .015625)
    {
        s1 = s1 - .015625;
    }
}```
standard = standard + .015625;
high = high|0x04;
}

if(s1 > .007#125)
{
 s1 = s1 - .0078125;
standard = standard + .0078125;
high = high|0x02;
}

if(s1 > .00390625)
{
 s1 = s1 - .00390625;
standard = standard + .00390625;
high = high|0x01;
}

if(s1 > .001953125)
{
 s1 = s1 - .001953125;
standard = standard + .001953125;
low  = low|0x08;
}

if(s1 > .0009765625)
{
 s1 = s1 - .0009765625;
standard = standard + .0009765625;
low  = low|0x04;
}

if(s1 > .00048828125)
{
 s1 = s1 - .00048828125;
standard = standard + .00048828125;
low  = low|0x02;
}

if(s1 > .000244140625)
{
 s1 = s1 - .000244140625;
standard = standard + .000244140625;
low  = low|0x01;
}

/**** Send out the standard ****/
out(bmba, addr32a); /* Set addr 32 on a and c */
out(bmbc, addr32c);
out(baca, high);  /* Set HSB AC(0-7) */
out(bacb, low);   /* Set LSB AC(8-11) */
out(bacc, iop2);  /* Pulse iop 2 for load */
out(bacc, cleariop);

hstandard = high;
lstandard = low;
}

******************************************************************************

// MODULE: range
// PURPOSE: Main control system routines.

// PARAMETERS:
GLOBL: range
   rl
   resistor
LOCAL: high - High byte of the range

// USERS:
findme
measure_test

/
void range()
{
    int high;
    range = 0;  // Initialize the place holders
    r1 = 0;
    while (r1 < resistor)  // Find out what range we are in
    {
        range *=
        r1 = pow(2, (3*(long)range));
    }
    standard = resistor/((float)r1);  // Compute the ratio of resistor to range
    astandard();  // Send that to the system
    if(range < 8)  // If the range is less than eight, adjust it
    {
        high = range;
        high <<= 5;
    }
    if(range == 8)  // If the range is eight, set its proper value
        high = 0x10;
    if(range == 9)  // If the range is nine, set its proper value
        high = 0x30;

    /** send out the range number ***/
    outp(bmba, addr30A);  // Set addr 30 on a and c
    outp(bmbc, addr30C);
    outp(baca, high);  // Set HSB AC(0-7)
    outp(bacb, 0x00);  // Set LSB AC(8-11)
    outp(bacc, lop2);  // Pulse lop 2 for load
    outp(bacc, clearlop);

    /** Send out the function ***/
    outp(bmba, addr30A);  // set addr 30 on a and c
    outp(bmbc, addr30C);
    outp(baca, 0x00);  // Set HSB AC(0-7)
    outp(bacb, 0x00);  // Set LSB AC(8-11)
    outp(bacc, lop4);  // Pulse lop 4 for load
    outp(bacc, clearlop);

    hrange = high;
    range = 0x00;
}

/*
 * MODULE:  agen_volt
*/
void gen_volt()
{
    int high = 0;
    float gl;

    gl = gen_volt; /* Figure out the right code to send */

    if (gl > 0)
        high = 0x10;
    else
        gl = gl * (-1);

    high = high & (int)gl;
    high <<= 1;
    if (gl >= (int)gl + .5) {
        high = high & 0x01;
        gl = gl - .5;
    }

    high <<= 1;
    if (gl >= (int)gl + .25) {
        high = high & 0x01;
        gl = gl - .25;
    }

    high <<= 1;
    if (gl > (int)gl) {
        high = high & 0x01;
    }

    /* Set addr 15 on a and c */
    outp(bmca, addr31A);
    /* Set HSB AC(0-7) */
    outp(bmca, addr31C);
    /* Set LSB AC(8-11) */
    outp(bmcb, high);
    outp(bmcb, 0x600);
    outp(bmcb, 0x802);
    outp(bmcb, cleario);

    lgen_volt = 0x06;
    hgen_volt = high;
}
MODULE:  detect_gain

PURPOSE:  detector gain

PARAMETERS:  GLOBL:  gain  hgain
              lgain

LOCAL:  gl  -  Temporary value of the gain
         high  -  High byte of the gain
         low  -  Low byte of the gain

USERS:  findme
        measure_test
        laser_trim

CALLS:  outp

RETURNS:  NONE

DATE:  24 June 1990
PROGRAMMER:  Ryp Walters
LANGUAGE:  Microsoft C Version 5.1
OS:  MS-DOS Version 3.3

---------------  INITIALS  ---------------------  MODIFICATION  
---------------  ---------  ---------------------  ------------------

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

void detect_gain()
{
  int high = 0, low = 0;
  float gl;

  if (gain > 511) /* Truncate the to the max gain */
    gain = 511;

  gl = gain; /* Figure out the right code to send */

  high = (int)gl;
  high >>= 1;

  if (gl != (int)gl + .5)
    {
      low = 0x01;
    }

  low <<= 1;
  if (gl >= (int)gl + .5)
    {
      low = low|0x01;
      gl = gl -.5;
    }

  low <<= 1;
  if (gl >= (int)gl + .25)
    {
      low = low|0x01;
      gl = gl -.25;
    }

  low <<= 1;
  if (gl > (int)gl)
    {
      low = low|0x01;
    }

  /* Send out the detector gl */

  outp(bmba, addr31a); /* Set addr 15 on a and c */
  outp(bmbc, addr31c);
  outp(baca, high); /* Set HSB AC[0-7] */
  outp(bacb, low); /* Set LSB AC[8-11] */

  211
outp(bacc, iop4);  /* Pulse iop 4 for load */
outp(bacc, cleariop);

bgain = high;
lgain = low;
}

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

** MODULE:    pre_gain
** PURPOSE:   Main control system routines.
** PARAMETERS: GLOBAL: hpre_amp  lpre_amp
**              pre_amp
**              LOCAL: high  - High byte of the pre_amp
**              USERS: findme  laser_trim
**              measure_test
** CALLS: outp
**
** RETURNS: NONE

** DATE: 24 June 1990
** PROGRAMMER: Ryp Walters
** LANGUAGE: Microsoft C Version 5.1
** OS: MS-DOS Version 3.3

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

void pre_gain()
{
    int high = 0;
    if (pre_amp == 8)   /* If the value is eight, set that code */
        high = 0x01;
    else                 /* Otherwise, set the code for one (1) */
        high = 0x00;

    /**** Send out the generator voltage ****/
    outp(bma, addr16A);  /* Set addr 16 on a and c */
    outp(bmbc, addr16C);
    outp(baca, 0x00);    /* Set HSB AC(0-7) */
    outp(bacb, high);    /* Set LSB AC(8-11) */
    outp(bacc, iop2);    /* Pulse iop 2 for load */
    outp(bacc, cleariop);

    hpre_amp = 0x00;
    lpre_amp = high;
}

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

** MODULE:    acomparator
** PURPOSE:   Main control system routines.
** PARAMETERS: GLOBAL: compare  hcompare
**              lcompare
**              LOCAL: high  - High byte of the comparator
**              high1  - Temporary byte holder
**              low   - Low byte of the comparator
*/

212
void acomparator()
{
    int high = 0, high1 = 0, low = 0;

    high = compare;
    /* Figure out the right code to send */
    high >>= 4;
    high1 = high;

    high1 <<= 4;
    low = compare - high1;

    /* *** Send out the comparator ***/
    outp(bmbs, addr\[6A]); /* Set addr 16 <n a and c */
    outp(bmbc, addr\[6C]);
    outp(baca, high); /* Set HSB AC(0-7) */
    outp(bacho, low); /* Set LSB AC(8-11) */
    outp(bacc, lop4); /* Pulse lop 4 for load */
    outp(bacc, clearlop);
    lcompare = low;
    hcompare = high;
}

void adelay()
{
    int high = 0, high1 = 0, low = 0;
    high = delay;
/* Figure out the right code to send */

high >>= 4;
high1 = high;

high1 <<= 4;
low = delay - high1;

/** Send out the delay **/

outp(bmba, addr13A);
outp(bmbc, addr13C);
outp(baca, high);
outp(bacc, low);
outp(bacc, lop2);
outp(bacc, clearlop);

delay = low;
hdelay = high;

}

/****************************************************************************
 * MODULE: cirol
 * PURPOSE: Clear overload and go.
 * PARAMETERS: None
 * USERS: findme laser_trim
 * measure_test
 * CALLS: outp
 * RETURNS: NONE
 * DATE: 24 June 1990
 * PROGRAMMER: Ryp Walters
 * LANGUAGE: Microsoft C Version 5.1
 * GS: MS-DOS Version 3.3
 * ***************************************************************************/

void cirol()
{

/** Send out the comparator **/

outp(bmba, addr16A);
outp(bmbc, addr16C);
outp(baca, 0x00);
outp(bacc, 0x00);
outp(bacc, lop1);
outp(bacc, clearlop);

}

/****************************************************************************
 * MODULE: skstrobe
 * PURPOSE: skip on a strobe done
 * PARAMETERS: GLOBL: strobe_done
 * ***************************************************************************/

214
void strobe()
{
	/** Send out the comparator ***/
	outp(bmba, addr32A); /* Set addr 16 on a and c */
	outp(bmbc, addr32C); /* Pulse iop 1 for load */
	outp(bacc, iopl1);
	strobe_done = inp(acc);
	strobe_done = strobe_done&0x02;
	outp(bacc, cleariop);
}

void skstrobe()
{

	MODULE: strobe

	PURPOSE: Main control system routines.

	PARAMETERS: GLOBAL: strobe_done

	LOCAL: i — Wait loop variable

	USERS: f'ndme laser_trim

	measure_test

	CALLS: outp strobe

	RETURNS: NONE

	DATE: 24 June 1990

	PROGRAMMER: Ryp Walters

	LANGUAGE: Microsoft C Version 5.1

	OS: MS-DOS Version 3.3

	----------------------------------

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

int i;

	/** Measure System Strobe ***/
	outp(bmba, addr32A); /* Set addr 32 on a and c */
	outp(bmbc, addr32C);
	outp(bacc, iopl4); /* Pulse iop 4 for load */
	for(i=0;i<1000;i++); /* Wait for a while */
outp(bacc, cleariop);

skstrobe();
while(strobe_done != 0) /* Wait for the system to finish */
   skstrobe();
}

******************************************************************************

  MODULE:  clracoon
  PURPOSE:  Clear the accumulator for the convertor.
  PARAMETERS:  None
  CALLS:  outp
  RETURNS:  NONE

  DATE:  24 June 1990
  PROGRAMMER:  Ryp Walters
  LANGUAGE:  Microsoft C Version 5.1
  OS:  MS-DOS Version 3.3

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

void clracoon()
{
   /* clear ac convertor */
   outp(bmbc, addr17A);   /* Set addr 17 on a and c */
   outp(bmbc, addr17C);
   outp(bacc, iop2);     /* Pulse iop 2 for CLEAR ac convertor */
   outp(bacc, cleariop);
}

******************************************************************************

  MODULE:  clracov
  PURPOSE:  Clear the accumulator for the overload.
  PARAMETERS:  None
  CALLS:  outp
  RETURNS:  NONE
  DATE:  24 June 1990
  PROGRAMMER:  Ryp Walters
  LANGUAGE:  Microsoft C Version 5.1
  OS:  MS-DOS Version 3.3

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

void clracov()
{
   /* clear ac overload */
/* Set addr 52 on a and c        */
outp(bmba, addr52A);
outp(bmbc, addr52C);
outp(bacc, iop21);          /* Pulse iop 2 for CLEAR ac overload */
outp(bacc, cleariop);

*******************************************************************************

* MODULE: skend_delay
* PURPOSE: skip on a strobe done
* PARAMETERS: GLOBL: end_delay
* CALLS: outp    inp
* RETURNS: NONE
* DATE: 24 June 1990
* PROGRAMMER: Ryp Walters
* LANGUAGE: Microsoft C Version 5.1
* OS: MS-DOS Version 3.3
*******************************************************************************

*******************************************************************************

void skend_delay()
{
    /* Send out the comparator */
    outp(bmba, addr13A);          /* Set addr 16 on a and c        */
    outp(bmbc, addr13C);
    outp(bacc, iop11);           /* Pulse iop 1 for load */
    end_delay = inp(acc);
    end_delay = end_delay&0X02;
    outp(bacc, cleariop);

*******************************************************************************

* MODULE: skready
* PURPOSE: Main control system routines.
* PARAMETERS: GLOBL: ready
*            USERS: measure_test channelb
*                   channela
*            CALLS: outp    inp
* RETURNS: NONE
* DATE: 24 June 1990
* PROGRAMMER: Ryp Walters
* LANGUAGE: Microsoft C Version 5.1
* OS: MS-DOS Version 3.3
*******************************************************************************
void skready()
{
    /** Send out the comparator **/  
    outp(bmba, addr53A);       /* Set addr 16 on a and c */
    outp(bmbc, addr53C);       /* */
    outp(bacc, iop1);          /* Pulse iop 1 for load */
    ready = inp(acc);
    ready = ready&0x02;
    outp(bacc, cleariop);
}

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

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

MODULE: skendconv

Purpose: Main control system routines.

Parameters:  GLOBL: end_conv

Users: findme  laser_trim

measure_test  rdconvert

Calls: outp  inp

Returns: NONE

Date: 24 June 1990

Programmer: Ryp Walters

Language: Microsoft C Version 5.1

OS: MS-DOS Version 3.3

Date  Initials  Modification

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

void skend_conv()
{
    /** Send out the comparator **/  
    outp(bmba, addr17A);       /* Set addr 16 on a and c */
    outp(bmbc, addr17C);       /* */
    outp(bacc, iop1);          /* Pulse iop 1 for load */
    end_conv = inp(acc);
    end_conv = end_conv&0x02;
    outp(bacc, cleariop);
}

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

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

MODULE: rdoverload

Purpose: Main control system routines.

Parameters:  GLOBL: overload

Local: high  - High byte of the overload
void rdoverload()
{
    int high=0, low=0;

    /* Measure System Strobe ***/
    outp(bmba, addr52a);        /* Set addr 52 on a and c */
    outp(bmbc, addr56c);        /* */
    outp(bacc, iop4);           /* Pulse iop 4 for load */
    high = inp(acd);            /* */
    low = inp(acb);             /* */
    low = low&0x0f;             /* */
    outp(bacc, cleariop);
    overload = high;            /* */
    overload <<= 4;             /* */
    overload = overload + low;  /* */
    overload = overload&0xff;   /* */
}

void rdconvert()
{ int high=0, low=0;

    /*** Measure System Strobe ***/
    outp(bmca, addr17A); /* Set addr 17 on a and c */
    outp(bmbc, addr17C); /* Pulse iop 4 for load */
    outp(bacc, iop4);  
    high = inp(acb);  
    low = low&0x0f;  
    outp(bacc, cleariop);
    convertor = high; /* Calculate the conv. in order to display */
    convertor <<= 4;
    convertor = convertor + low;
    if(convertor > 2047)
        convertor = convertor - 4096;

    end_conv = 2;
    while( end_conv != 0) /* Wait for the conversion to finish */
        skend_conv();

    strobe_done = 2;
    while( strobe_done != 0) /* Wait for the system to be ready */
        skstrobe();
}

//******************************************************************************
/*
 * MODULE: channelb
 * PURPOSE: Set CHANNEL B position.
 * PARAMETERS: GLOBL: hpbest
 *              probelow
 *              probehigh
 *              ready
 * LOCAL: high - High byte for channel B
 *        low  - Low byte for channel B
 * USERS: laser_trim
 *        measure_test
 * CALLS: outp
 *        skready
 *
 * RETURNS: NONE
 */
 /*
 * DATE: 24 June 1990
 * PROGRAMMER: Ryp Walters
 * LANGUAGE: Microsoft C Version 5.1
 * OS: MS-DOS Version 3.3
 */
 /*
 * DATENumber INITIALS MODIFICATION
 * --------------------------------
 */
*******************************************************************************/

void channelb()
{
    unsigned int high, low;

    low = (hpbest&0x0f);  /* Compute the right code */
    high = (hpbest&0xff0);  
    high >>= 4;
    high = high&0x80;

    /*** Send out the generator voltage ***/
    outp(bmca, addr15A);  /* Set addr 15 on a and c */
    outp(bmbc, addr15C); /* Set HSB AC(0-7) */
    outp(bacc, high);    /* Set LSB AC(8-11) */
    outp(bacb, low);  
}
void channels()
{
    unsigned int high, low;
    low = aprobes0x0f;
    high = aprobes0xff;
    high >>= 4;
    high = high>>0x40;
    /* Compute the right code to send */

    /* Set addr 15 on a and c */
    outp(bmba, addr53A);
    outp(bmbc, addr53C);
    outp(baca, high);
    outp(bacb, low);
    outp(bacc, 0x06);
    outp(bacc, clearlop);
    probehigh = high;
    probehigh = low;
    skready();
    while(ready != 0) /* Wait for the system to recover */
        skready();
}
void probe_select()
{
    outp(bmba, addr55A);   /* set addr 15 on a and c */
    outp(bmbc, addr55C);
    outp(bacca, 0x00);     /* Set HSB AC(0-7) */
    outp(bacb, 0x00);      /* Set LSB AC(8-11) */
    outp(bacc, 0x04);      /* Pulse lop 2 for load */
    outp(bacc, clearlop);
}
```c
#include <stdlib.h>
#include <conio.h>
#include <stdio.h>
#include <dos.h>
#include <window.h>
#include <moveincd.h>
#include <neaimax.h>
#include <newstuff.h>
#include <extrnvar.h>
#include <deftion.h>
#include <main.h>
#include <float.h>
#include <math.h>

/*
   MODULE: skip_on_rest
   PURPOSE: Skip test for if the laser is moving (resting) or not
   PARAMETERS: GLOBL: resting
               LOCAL: restx - control byte for x direction
                      resty - control byte for y direction
               USEFS: main: move_dist
                      scribe: circle_cut
                      some_move: do_file
                      do_a_loop
               CALLS: outp inp
   RETURNS: None
   DATE: 24 June 1990
   PROGRAMMER: Ryp Walters
   LANGUAGE: Microsoft C Versica 5.1
   OS: MS-DOS Version 3.3

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

void skip_on_rest()
{
    int restX, restY;

    /* skip on rest */
    /** x - axis select **/

cutp(bmba, addr40A); /* Set addr 40 on a and c */
cutp(bmbc, addr40C); /* Pulse iop1 for x-axis */
cutp(bacc, clearlop);

cutp(bmba, addr40A); /* Set addr 40 on a and c */
cutp(bmbc, addr40C); /* Pulse iop 4 for read masb */
cutp(bacc, iop4);

restx = inp(acc);
resty = restx & 0x02;

outp(bacc, clearlop);
```
/** y - axis select **/
outp(bmba, addr40A); /* Set addr 40 on a and c */
outp(bmbc, addr40C); /* Pulse iop2 for y-axis */
outp(bacc, cleariop);

/*** skip on rest /***/
outp(bmba, addr40A); /* Set addr 40 on a and c */
outp(bmbc, addr40C); /* Pulse iop 4 for read msb */
outp(bacc, iop4);
resty = inp(acc); /* Control bits; clear ac, interrupt, skip */
resty = resty & 0x02;
outp(bacc, cleariop);

/*** check if end of move /***/
if (restx == 0)
  if (resty == 0)
    resting = 1; /* set flag for end of last move */
}

******************************************************************************

MODULE: encode

PURPOSE: Transform the distance into octal code

PARAMETERS: GLOBL: hlsbx hlsby
             llsbx llsby
             msbx msby
             vhlstx vhlstby
             vllstx vllstby

PARAM: x_dist - X distance to transform
        x_velo - X velocity to transform
        y_dist - Y distance to transform
        y_velo - Y velocity to transform

LOCAL: temp - Temporary value holder

USES: move_norest move_dist

CALLS: abs

RETURNS:

DATE: PROGRAMMER: Ryp Walters
LANGUAGE: Microsoft C Version 5.1
OS: MS-DOS Version 3.3

DATE INITIALS MODIFICATION

******************************************************************************

void encode(int x_dist, int y_dist, float x_velo, float y_velo)
{
  int temp = 0x0000;

  if(x_dist < 0 )
    x_velo = 128 - x_velo;

  if(y_dist < 0)
    y_velo = 128 - y_velo;

  224
/*** encode numbers ***/

/*** velocity ***/

/*** x-axis ***/

vhlsbx = 0x00;
vlhsbx = 0x00;

vhlsbx = abs((int)x_velo);

if(x_dist < 0 ){
    vhlsbx = 0x80|vhlsbx;
    x_dist = x_dist * (-1);
}

if(x_velo > (int)x_velo + .25){
    vlhsbx = 0x08;
    x_velo = x_velo-.5;
}

if(x_velo > (int)x_velo){
    vlhsbx = vlhsbx|0x04;
}

/*** y-axis ***/

vhlsby = 0x00;
vlhsby = 0x00;

vhlsby = abs((int)y_velo);

if(y_dist < 0 ){
    vhlsby = 0x80|vhlsby;
    y_dist = y_dist * (-1);
}

if(y_velo > (int)y_velo + .25){
    vlhsby = 0x08;
    y_velo = y_velo-.5;
}

if(y_velo > (int)y_velo){
    vlhsby = vlhsby|0x04;
}

/*** position ***/

/*** x-axis ***/

msbx = 0x00;
hsbx = 0x00;
llsbx = 0x00;

temp = x_dist & 0x0000;
temp >>= 12;                     /* Shift right 12 to right align */
msbx = temp;

temp = x_dist & 0x00FF0;
temp >>= 4;
hsbx = temp;
llsbx = x_dist & 0x00000;

/*** y axis ***/

msby = 0x00;
hsby = 0x00;
llsb = 0x00;

temp = y_dist & 0x0000;
temp >>= 12;                     /* Shift right 12 to right align */
msby = temp;

temp = y_dist & 0x00FF0;
temp >>= 4;
hsby = temp;
llsby = y_dist & 0x000F;
}

 /*******************************************************************************/
 /* MODULE: move_no_rest */
 /* PURPOSE: Distance move with no initial wait for laser to not be moving */
 /* PARAMETERS: PARAM: x_dist - X distance to move */
 /* x_velo - Y velocity to move */
 /* y_dist - Y distance to move */
 /* y_velo - Y velocity to move */
 /* USERS: laser_trim index_pos */
 /* joymove */
 /* CALLS: encode move_it */
 /* RETURNS: None */
 /* DATE: 24 June 1990 */
 /* PROGRAMMER: Ryp Walters */
 /* LANGUAGE: Microsoft C Version 5.1 */
 /* OS: MS-DOS Version 3.3 */
 /*******************************************************************************/

void move_norest(int x_dist, int y_dist, float x_velo, float y_velo)
{
    double angle;
    if(x_velo == 0 || y_velo == 0 || x_dist == 0 || y_dist == 0);
    else
    if(x_velo == y_velo)
    {
        angle = atan((double)(abs(y_dist)/abs(x_dist)));
        x_velo = x_velo * (float)(cos(angle));
        y_velo = y_velo * (float)(sin(angle));
    }
    encode(x_dist, y_dist, x_velo, y_velo);
    move_it();
}

 /*******************************************************************************/
 /* MODULE: move_dist */
 /* PURPOSE: To move a distance after the laser has stopped moving */
 /* PARAMETERS: PARAM: x_dist - X distance to move */
 /* x_velo - Y velocity to move */
 /* y_dist - Y distance to move */
 /* y_velo - Y velocity to move */
 /* USERS: main scribe */
 /* laser_trim circle_cut */
 /* some_move index_pos */
 /* do_file do_a_loop */
 /* CALLS: encode wait_for_rest */
 /* RETURNS: None */
 /* DATE: 24 June 1990 */
 /* PROGRAMMER: Ryp Walters */
 /* LANGUAGE: Microsoft C Version 5.1 */
 /* OS: MS-DOS Version 3.3 */
 /******************************************************************************/
void move_dist(int x_dist, int y_dist, float x_velo, float y_velo)
{
    double angle;
    if(x_velo == 0 || y_velo == 0 || x_dist == 0 || y_dist == 0)
    else
    if(x_velo == y_velo)
    {
        angle = atan((double)(abs(y_dist)/abs(x_dist)));
        x_velo = x_velo * (float)(cos(angle));
        y_velo = y_velo * (float)(sin(angle));
    }
    encode(x_dist, y_dist, x_velo, y_velo);
    wait_for_rest();
    move_lt();
}

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

MODULE:  move_lt

PURPOSE: To send the sequence to get the laser to move

PARAMETERS: GLOBL:  hlsbx  hlsby
             llsbx  llisby
             mlsbx  mlsby
             vhlsbx  vhlsby
             vllsbx  vllsby

USERS: move_dist  move_norest

CALLS:  outp

RETURNS: None

DATE:  24 June 1990

PROGRAMMER: Ryp Walters

LANGUAGE:  Microsoft C Version 5.1

OS:  MS-DOS Version 3.3

DATE  INITIALS  MODIFICATION

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

void move_lt()
{
    /* axis select x axis */
    output(bmba, addr40A);
    /* Set addr 40 or a and c */
    output(bmbc, addr40C);
    output(bcc, iopl1);
    /* Pulse iop 1 for x axis select */
    output(bacc, clearicp);

    /* load location as distance to go */
```plaintext
/** load velocity and acceleration /**
outp(bmca, addr3A); /* Set addr 43 on a and c */
outp(bmcc, addr3C); /* SET VELOCITY AT 1/4 MAX */
outp(baca, vlsbx); /* Send llsb (ac 0 - 7) */
outp(bacb, vlsbx); /* Send llsb (ac 8 - 11) */
outp(bacc, iop1); /* Pulse iop 1 for load velocity */
outp(bacc, cleariop);

/** distance to go - send lsb /**
outp(bmca, addr42A); /* Set addr 42 on a and c */
outp(bmcc, addr42C); /* Send llsb (ac 0 - 7) */
outp(bacb, llsbx); /* Send llsb (ac 8 - 11) */
outp(bacc, iop2); /* Pulse iop 2 for load lsb */
outp(bacc, cleariop);

/** axis select y axis /**
outp(bmca, addr40A); /* Set addr 40 on a and c */
outp(bmcc, addr40C); /* Pulse iop 2 for y axis select */
outp(bacc, cleariop);

/** load location as distance to go /**
/** load velocity and acceleration /**
outp(bmca, addr3A); /* Set addr 43 on a and c */
outp(bmcc, addr3C); /* SET VELOCITY AT 1/4 MAX */
outp(baca, vlsby); /* Send llsb (ac 0 - 7) */
outp(bacb, vlsby); /* Send llsb (ac 8 - 11) */
outp(bacc, iop1); /* Pulse iop 1 for load velocity */
outp(bacc, cleariop);

/** distance to go - send lsb /**
outp(bmca, addr42A); /* Set addr 42 on a and c */
outp(bmcc, addr42C); /* Send llsb (ac 0 - 7) */
outp(bacb, llsby); /* Send llsb (ac 8 - 11) */
outp(bacc, iop2); /* Pulse iop 2 for load lsb */
outp(bacc, cleariop);

/** axis select x axis /**
outp(bmca, addr40A); /* Set addr 40 on a and c */
outp(bmcc, addr40C); /* Pulse iop 1 for x axis select */
outp(bacc, cleariop);

/** distance to go - send msb /**
outp(bmca, addr42A); /* Set addr 42 on a and c */
outp(bmcc, addr42C); /* Put msb on ac */
outp(bacc, iop1); /* Pulse iop 1 for load msb */
outp(bacc, cleariop);

/** axis select y axis /**
outp(bmca, addr40A); /* Set addr 40 on a and c */
outp(bmcc, addr40C); /* Pulse iop 2 for y axis select */
outp(bacc, cleariop);

/** distance to go - send msb /**
outp(bmca, addr42A); /* Set addr 42 on a and c */
outp(bmcc, addr42C); /* */
```
void clrhome()
{

    outp(bmba, addr40A);      /* Set addr 40 on a and c */
    outp(bmbc, addr40C);
    outp(bacc, iop1);         /* Pulse iop1 for the x-axis */
    outp(bacc, cleariop);

    outp(bmba, addr43A);      /* Set addr 43 on a and c */
    outp(bmbc, addr43C);
    outp(bacc, iop2);         /* Pulse iop2 for CLEAR HOME LATCH */
    outp(bacc, cleariop);

    outp(bmba, addr40A);      /* Set addr 40 on a and c */
    outp(bmbc, addr40C);
    outp(bacc, iop2);         /* Pulse iop2 for the y-axis */
    outp(bacc, cleariop);

    outp(bmba, addr43A);      /* Set addr 43 on a and c */
    outp(bmbc, addr43C);
    outp(bacc, iop2);         /* Pulse iop 4 for CLEAR HOME LATCH */
    outp(bacc, cleariop);
}

MODULE: rdjoy
PURPOSE: Read the joystick values
PARAMETERS: GLobl: xjoy  yjoy
void rdjoy()
{
    /* x-axis select */
    outp(bmbc, addr40A); /* Set addr 40 on a and c */
    outp(bmbc, addr40C);
    outp(bacc, iop1); /* Pulse iop1 for the x-axis */
    outp(bacc, cleariop);

    /* read overtravel and overshoot */
    outp(bmbc, addr44A); /* Set addr 44 on a and c */
    outp(bmbc, addr44C);
    outp(bacc, iop4); /* Pulse iop 4 for read otosjoy */
    xjoy = inp(acb);
    xjoy = xjoy & 0x07;
    outp(bacc, cleariop);

    /* y-axis select */
    outp(bmbc, addr40A); /* Set addr 40 on a and c */
    outp(bmbc, addr40C);
    outp(bacc, iop2); /* Pulse iop2 for the y-axis */
    outp(bacc, cleariop);

    /* read overtravel and overshoot */
    outp(bmbc, addr44A); /* Set addr 44 on a and c */
    outp(bmbc, addr44C);
    outp(bacc, iop4); /* Pulse iop 4 for read otosjoy */
    yjoy = inp(acb);
    yjoy = yjoy & 0x07;
    outp(bacc, cleariop);
}

MODULE: laser_on
PURPOSE: Turn the laser on
PARAMETERS: None

CALLS: outp
void laser_on()
{

    /** Turn on the laser **/

    outp(bmba, addr15A);  /* Set addr 15 on a and c */
    outp(bmbc, addr15C);
    outp(baca, 0x00);     /* Set MSB AC(0-7) */
    outp(bacb, 0x04);     /* Set LSB AC(8-11) */
    outp(bacc, lop);      /* Pulse lop 4 for load */
    outp(bacc, clearlop);

    /** Disable the comptator **/

    outp(bmba, addr15A);  /* Set addr 15 on a and c */
    outp(bmbc, addr15C);
    outp(bacc, lop2);     /* Pulse lop2 for load */
    outp(bacc, clearlop);
}


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

MODULE:  disable_lim_cro
*
PURPOSE: Disable the limit crossing flag
*
PARAMETERS: None
*
USERS: laser_trim measure_test
*
CALLS: outp
*
RETURNS: None
*
DATE:  24 June 1990
*
PROGRAMMER: Ryp Walters
*
LANGUAGE: Microsoft C Version 5.1
*
OS: MS-DOS Version 3.3
*
*******************************************************************************/

void disable_lim_cro()
{

    /** Disable the comptator **/

    outp(bmba, addr15A);  /* Set addr 15 on a and c */
    outp(bmbc, addr15C);
    outp(bacc, lop2);     /* Pulse lop2 for load */
    outp(bacc, clearlop);
}

*******************************************************************************/
void laser_off()
{
    /* Turn off the laser */
    outp(bmbs, addr15A);        /* Set addr 15 on a and c */
    outp(bmbs, addr15C);        /* Set HSB AC(0-7) */
    outp(bach, 0x00);           /* Set LSB AC(8-11) */
    outp(bacc, lop4);           /* Pulse lop 4 for load */
    outp(bacc, clearlop);       /* */
}

void rdposltn()
{
    long int total;
}
```c
int msb, hlsb, llsb;

    /*** axis select x axis /***/
    outp(bmba, addr40A);  /* Set addr 40 on a and c */
    outp(bmbc, addr40C);  /* */
    outp(bacc, iop1);    /* Pulse iop 1 for x axis select */
    outp(bacc, cleariop); /* */

    /*** read location /***/
    /*** table pos - read msb /***/
    outp(bmba, addr41A);  /* Set addr 41 on a and c */
    outp(bmbc, addr41C);  /* */
    outp(bacc, iop2);    /* Pulse iop 2 for read msb */
    msb = inp(acb);      /* MSB (ac 8-11) */
    msb = msb & 0x0F;    /* Clean out the input */
    outp(bacc, cleariop); /* */

    /*** table pos - read lsb /***/
    outp(bmba, addr41A);  /* Set addr 41 on a and c */
    outp(bmbc, addr41C);  /* */
    outp(bacc, iop4);    /* Pulse iop 4 for read lsb */
    hlsb = inp(acb);      /* HLSB (ac 0-7) */
    llsb = inp(acb);      /* LLSB (ac 8-11) */
    llsb = llsb & 0x0F;   /* Clean out the input */
    outp(bacc, cleariop); /* */

    /*** put together number /***/
    total = 0x00000000;  /* Clear out the number */
    total <<= 8;        /* */
    total &= 0xFFFFFFF0; /* Make room for HLSB */
    total |= hlsb;      /* clear the space just in case */
    total <<= 4;        /* OR it in */
    total &= 0xFFFFFFF0; /* Make room for LLSB */
    total ^= 0xFFFFFFF0; /* clear the space just in case */
    total |= llsb;      /* */
    total = 32767 - total;
    xpos = total;

    /*** axis select y axis /***/
    outp(bmba, addr40A);  /* Set addr 40 on a and c */
    outp(bmbc, addr40C);  /* */
    outp(bacc, iop2);    /* Pulse iop 2 for y axis select */
    outp(bacc, cleariop); /* */

    /*** read location /***/
    /*** table pos - read msb /***/
    outp(bmba, addr41A);  /* Set addr 41 on a and c */
    outp(bmbc, addr41C);  /* */
    outp(bacc, iop2);    /* Pulse iop 2 for read msb */
    msb = inp(acb);      /* MSB (ac 9-11) */
    msb = msb & 0x0F;    /* Clean up the input */
    outp(bacc, cleariop); /* */

    /*** table pos - read lsb /***/
    outp(bmba, addr41A);  /* Set addr 41 on a and c */
    outp(bmbc, addr41C);  /* */
```

outp(bacc, iop4);  /**< Pulse iop 4 for read lsb */

hlsb = inp(ac);  /**< HLSB (ac 0 - 7) */
llsb = inp(acb);  /**< LLSB (ac 8 - 11) */
llsb = llsb & 0x0F;  /**< clean up the input */
outp(bacc, cleariop);

    /*** put together number ***/

    total = 0x00000000;
    /*** Clear out number ***/
    total = msb;
    total <<= 8;
    /*** Make room for hlsb ***/
    total &= 0xFFFFF000;
    /*** clear the space just in case ***/
    total |= hlsb;
    /*** OR it in ***/
    total <<= 4;
    /*** Make room for llsb ***/
    total &= 0xFFFFF000;
    /*** clear the space just in case ***/
    total |= llsb;
    total = 32767 - total;
    ypos = total;
}

 /*******************************************************************************/
 * MODULE:  power
 * PURPOSE: Change the auxiliary power settings
 * PARAMETERS:  PARAM: pwr        - New power setting (0 - 8)
 *               rep        - New rep rate (0 - 8)
 * CALLS:  outp
 * RETURNS:  None
 * DATE:  24 June 1990
 * PROGRAMMER:  Ryp Walters
 * LANGUAGE:  Microsoft C Version 5.1
 * OS:  MS-DOS Version 3.3
 * DATE  MODIFICATION
 * -------  ---------------
 *          24 June 1990  
 /*******************************************************************************/

void power(int pwr, int rep)
{
    /**< Set the power level and rep. ***/

    outp(bma, addr54A);  /**< Set addr 54 on a and c */
    outp(bmc, addr54C);
    outp(bca, pwr);  /**< Set power */
    outp(bcb, rep);  /**< Set reps */
    outp(bacc, iop1);  /**< Pulse iop1 to load power */
    outp(bacc, cleariop);
}

 /*******************************************************************************/
 * MODULE:  rdotos
 * PURPOSE:  To read the overtravel and overshoot latches
 * PARAMETERS:  GLOBE: xotos      yotos
 * USERS:  index_pos

234
void rdotos()
{
    /** x-axis select **/
    outp(bmba, addr40A);  /* Set addr 40 on a and c */
    outp(bmbc, addr40C);  /* Pulse iop1 for the x-axis */
    outp(bacc, cleariop);

    /** read overtravel and overshoot ***/
    outp(bmba, addr44A);  /* Set addr 44 on a and c */
    outp(bmbc, addr44C);  /* Pulse iop 4 for read msb */
    outp(bacc, iop4);

    xotos = inp(aca);
    outp(bacc, cleariop);

    /** y-axis select **/
    outp(bmba, addr40A);  /* Set addr 40 on a and c */
    outp(bmbc, addr40C);  /* Pulse iop2 for the y-axis */
    outp(bacc, cleariop);

    /** read overtravel and overshoot ***/
    outp(bmba, addr44A);  /* Set addr 44 on a and c */
    outp(bmbc, addr44C);  /* Pulse iop 4 for read msb */
    outp(bacc, iop4);

    yotos = inp(aca);
    outp(bacc, cleariop);
}

MODULE: skhome
PURPOSE: Sklp test for the home latches
PARAMETERS: GLOBL, xhome yhome
USRS: index_pos
CALLS: outp inp
RETURNS: None
DATE: 24 June 1990
PROGRAMMER: Ryp Walters
LANGUAGE: Microsoft C Version 5.1
OS: MS-DOS Version 3.3
void skhome()
{
    /** x - axis select **/
    outp(bmcb, addr40A); /* Set addr 40 on a and c */
    outp(bmcb, addr40C);
    outp(bacc, lop1);   /* Pulse lop1 for the x-axis */
    outp(bacc, clearlop);

    /** skip on home latch ***/
    outp(bmcb, addr43A); /* Set addr 43 on a and c */
    outp(bmcb, addr43C);
    outp(bacc, lop4);   /* Pulse lop 4 for read msb */
    xhome = inp(acc);   /* Control bits; clear ac, interrupt, skip */
    xhome = xhome & 0x02;
    outp(bacc, clearlop);

    /** y - axis select **/
    outp(bmcb, addr40A); /* Set addr 40 on a and c */
    outp(bmcb, addr40C);
    outp(bacc, lop2);   /* Pulse lop2 for the y-axis */
    outp(bacc, clearlop);

    /** skip on home latch ***/
    outp(bmcb, addr43A); /* Set addr 43 on a and c */
    outp(bmcb, addr43C);
    outp(bacc, lop4);   /* Pulse lop 4 for read msb */
    yhome = inp(acc);   /* Control bits; clear ac, interrupt, skip */
    yhome = yhome & 0x02;
    outp(bacc, clearlop);
}

******************************************************************************

MODULE: clrintr

PURPOSE: To clear the interrupts

PARAMETERS: None

USERS: index_pos

CALLS: outp

RETURNS: None

DATE: 24 June 1990
PROGRAMMER: Ryp Walters
LANGUAGE: Microsoft C Version 5.1
OS: MS-DOS Version 3.3

******************************************************************************

void clrintr()
/** x - axis select **/
outp(bmba, addr40A);  /* Set addr 40 on a and c */
outp(bmbc, addr40C);
outp(bacc, iop1);    /* Pulse iop1 for x-axis */
outp(bacc, cleariop);

/** Clear an interrupt **/
outp(bmba, addr44A);  /* Set addr 44 on a and c */
outp(bmbc, addr44C);
outp(bacc, iop2);    /* Pulse iop2 for load */
outp(bacc, cleariop);

/** y - axis select **/
outp(bmba, addr40A);  /* Set addr 40 on a and c */
outp(bmbc, addr40C);
outp(bacc, iop2);    /* Pulse iop2 for y-axis */
outp(bacc, cleariop);

/** Clear an interrupt **/
outp(bmba, addr44A);  /* Set addr 44 on a and c */
outp(bmbc, addr44C);
outp(bacc, iop2);    /* Pulse iop2 for load */
outp(bacc, cleariop);


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

*  MODULE:  index_pos
*  PURPOSE: To index the laser position [ home position ]
*  PARAMETERS:  GLOBL:  xhome  xotos
*               xpos  yhome  yotos
*               ypos
*  LOCAL:  x_dtemp  - X distance to travel to cause a latch
*          y_dtemp  - Y distance to travel to cause a latch
*  USERS:  main
*  CALLS:  clirintr  rdotos
*          move_norest  clrhhome
*          skhome  rdpositin
*          move_dist  wait_for_rest
*  RETURNS:  None
*  DATE:  24 June 1990
*  PROGRAMMER:  Ryp Walters
*  LANGUAGE:  Microsoft C Version 5.1
*  OS:  MS-DOS Version 3.3
*  DATE  INITIALS  ----------------------------------------------------------
*  -----------------------------------------------
*******************************************************************************/

void index_pos()
{
  int  x_dtemp, y_dtemp;
  clihome();    /* Clear the home latch and the interrupt reg. */
  clirintr();
}
x_dtemp = 6767; /* Set some very large distance to travel */
y_dtemp = 6767;

move_norest(x_dtemp, y_dtemp, 7, 7); /* Start to move that distance */
while (x_dtemp != 0 || y_dtemp != 0) /* While our axial flags are not set */
{
    rdotos(); /* See if we have any overtravel */
    skhome(); /* See if we have passed home */
    if (xhome == 0 || (xotoes&0x80) == 0) && x_dtemp != 0) /* If the x axis passed home or hit the edge */
    { /* but we haven't already done something */
        x_dtemp = 0;
        move_norest(0, y_dtemp, 0, 7); /* Flag the x axis flag */
        move_norest(0, y_dtemp, 0, 0); /* Stop x direction but keep current y */
    }
    if (yhome == 0 || (yotoes&0x80) == 0) && y_dtemp != 0) /* If the y axis passed home or hit the edge */
    { /* but we haven't already done something */
        y_dtemp = 0;
        move_norest(x_dtemp, 0, y_dtemp, 0, 0); /* Flag the y axis flag */
        move_norest(x_dtemp, 0, y_dtemp, 0, 7); /* Stop y direction but keep current x */
    }
}
clrintr(); /* Check if both flags have been hit */

if(xhome == 0) /* If the x axis wasn't a home flag */
x_dtemp = -6767; /* Set a backwards direction */

if(yhome == 0) /* If the y axis wasn't a home flag */
y_dtemp = -6767; /* Set a backwards direction */

move_norest(x_dtemp, y_dtemp, 7, 7); /* Move according to the home flags */
while (x_dtemp != 0 || y_dtemp != 0) /* While both flags aren't home flagged */
{
    skhome(); /* Check the home status */
    if(xhome == 0) && x_dtemp != 0) /* If X home and haven't already flagged it */
    { /* Set the flag */
        x_dtemp = 0;
        move_norest(0, y_dtemp, 0, 7); /* Stop X movement but keep Y */
    }
    if(yhome == 0) && y_dtemp != 0) /* If y home and haven't already flagged it */
    { /* Set the flag */
        y_dtemp = 0;
        move_norest(x_dtemp, 0, 0, y_dtemp, 0); /* Stop Y movement but keep X */
    }
}
rdposint(); /* Check the flags */
movcdist(-xpos, -ypos, 3, 3); /* Find out where we are */
wait_for_rest(); /* Move to pos 0,0 */
wait_for_rest(); /* Wait for laser to stop */

******************************************************************************

* MODULE: joymove
* PURPOSE: Joystick movement
* PARAMETERS: $GLOBAL: xjoy yjoy
* LOCAL: finish - Loop variable for load position button
*     msb - Most significant byte
*     xd - X distance to travel
*     xv - X velocity to travel
*     yd - Y distance to travel
*     yv - Y velocity to travel
*     USERS: scribe laser_trim

238
void joymove()
{
    int msb, xd=0, yd=0, finish;
    float xv=0, yv=0;

    /**************************************************************************/
    /** clear manual functions **/**
    /**************************************************************************/
    outp(bmba, addr56A);
    outp(bmbc, addr56C);
    outp(bacc, lop1);
    outp(bacc, cleariop);

    finish = 1;
    while( finish == 1 )
    {
        rdjoy();
        /* Read the joystick */
        if(xjoy == 1 )
        {
            move_norest(0,xd,0,yv);
            /* If the x axis is not angled */
            xd = 0;
            /* Stop all movement in the x direction */
            xv = 0;
            /* Reset the x distance and velocity */
        }
        if(yjoy == 1 )
        {
            move_norest(xd,0,xv,0);
            /* If the y axis is not angled */
            yd = 0;
            /* Stop all movement in the y direction */
            yv = 0;
            /* Reset the y distance and velocity */
        }
        if(xjoy == 1 )
        {
            xd = 32001;
            if(xjoy < 4 )
            {
                xv = 2;
            }
            else
            {
                xv = 10;
            }
            if(xjoy == 2 || xjoy == 6 )
            {
                xd = -1*xd;
            }
            move_norest(xd, yd, xv, yv);
            /* Move!! */
        }
        if( yjoy == 1 )
        {
            yd = 32001;
            if(yjoy < 4 )
            {
                yv = 2;
            }
            else
            {
                yv = 18;
            }
            /**************************************************************************/
            /* Set some big distance */
            /**************************************************************************/
            /* If the angle is small, set a slow vel. */
    */
if(yjoy == 2 || yjoy == 6)   /* If angle is down, set distance as neg. */
    yd = -1*yd;
move_norest(xd, yd, xv, yv);   /* Move!! */
}

/*** read manual functions ***/
outp(bmba, addr56A);          /* Set addr 56 on a and c */
outp(bmbc, addr56C);          /* Pulse iop 1 for clear manual functions */
    msb = inp(aca)&0xf8;
outp(bacc, cleariop);
    if((msb&0x20) == 0)          /* If the load position button is pressed */
    {
        move_norest(0,0,0,0);    /* Stop all moving */
        finish = 0;              /* Flag a get out of here */
    }
    /* Check for a get out of here */

/*** clear manual functions ***/
outp(bmba, addr56A);          /* Set addr 56 on a and c */
outp(bmbc, addr56C);          /* Pulse iop 1 for clear manual functions */
outp(bacc, lop1);            /* */
outp(bacc, cleariop);

/******************************************************************************
 * MODULE:    wait_for_start
 * PURPOSE:   To wait until the start button has been pressed
 * PARAMETERS: LOCAL: msb       - Most significant byte
 *             USERS: scribe   do_file
 *             CALLS: outp        inp
 * RETURNS:   None
 * DATE:      24 June 1990
 * PROGRAMMER: Ryp Walters
 * LANGUAGE:  Microsoft C Version 5.1
 * OS:        MS-DOS Version 3.3
 * DATE       INITIALS        MODIFICATION
 * -----------              ---------------
 ******************************************************************************/
void wait_for_start() {
    int msb;

    msb = 0xff;

    /*** read manual functions ***/
    outp(bmba, addr56A);          /* Set addr 56 on a and c */
    outp(bmbc, addr56C);
    while((msb&0x08) != 0)        /* While no START button */
    { 

240
outp(bacc, iop4);  /* Pulse iop 4 for read manual functions */
msb = inp(aca)&0x80;  /* Check that START button!! */
outp(bacc, cleariop);
}  /* Did we get it?? */

/** clear manual functions **/
outp(bmba, addr56A);  /* Set addr 56 on a and c */
outp(bmbc, addr56C);
outp(bacc, iop1);  /* Pulse iop 1 for clear manual functions */
outp(bacc, cleariop);

/*********************************************************/
/* MODULE:  check_for_stop */
/* PURPOSE:  To check if the stop button has been pressed */
/* PARAMETERS:  LOCAL:  msb - Most significant byte */
/* USERS:  scribe do_file do_a_loop */
/* CALLS:  outp inp */
/* RETURNS:  0 - Keep going (no stop) */
/* 1 - STOP!! */
/* DATE:  24 June 1990 */
/* PROGRAMMER:  Ryp Walters */
/* LANGUAGE:  Microsoft C Version 5.1 */
/* OS:  MS-DOS Version 3.3 */
/* DATE | INITIALS | MODIFICATION */
/*********************************************************/

int check_for_stop()
{
  int msb;

  msb = 0xff;

  /** read manual functions **/
  outp(bmba, addr56A);  /* Set addr 56 on a and c */
  outp(bmbc, addr56C);
  outp(bacc, iop1);  /* Pulse iop 1 for clear manual functions */
  msb = inp(aca)&0x80;
  outp(bacc, cleariop);

  /** clear manual functions **/
  outp(bmba, addr56A);  /* Set addr 56 on a and c */
  outp(bmbc, addr56C);
  outp(bacc, iop1);
  outp(bacc, cleariop);

  if((msb&0x80) == 0)
    return(1);  /* We've been snagged by a stop */
  return(0);  /* Return a STOP for the rest of the system */
  /* Otherwise, send a no stop */
}
void cut_a_pattern()
{
    int true, action, option;
    char numbers[10];

    true = 1; /* Defining variables. */
    while(true = 1)
    {
        wmscroll(0,-1,-1,0);
        wncurmov(3,26);
        wnprintf("what would you like to do?"); /* Print */
        wncurmov(5,22);
        wnprintf("1. Create a file"); /* Print */
        /* Other code... */
    }
}
wncurmov(6,22);

wprintf(" 2. Analyze / print a file "); /* Print
wncurmov(7,22);

wprintf(" 3. Edit a file "); /* Print
wncurmov(8,22);

wprintf(" 4. Run a file ");
wncurmov(9,22);

wprintf(" 5. Main menu");
wncurmov(12,26);

wprintf("Option: ");

wnquery(selection, 2, &scan);

if(*selection == 'c'|| *selection == 'C' || *selection == '1')
  option = 1;
if(*selection == 'a'|| *selection == 'A' || *selection == '2')
  option = 2;
if(*selection == 'e'|| *selection == 'E' || *selection == '3')
  option = 3;
if(*selection == 'r'|| *selection == 'R' || *selection == '4')
  option = 4;
if(*selection == 'm'|| *selection == 'M' || *selection == '5')
  option = 5;
if(option > 0 && option < 5) /* If it is one of the first four options. */
{
  if(file_name == "") /* option == 1 */
  {
    wncurmov(14,20); /* Move cursor.
    wprintf("Enter the name of the file: "); /* Print
    wnquery(file_name, 9, &scan); /* Find next selection.
    strncpy(old_file,file_name,9); /* Store the backup file.
    strncpy(new_file,file_name,9);
    strcat(old_file, "*.old");
    strcat(new_file, "*.new");
  }
  else
  {
    wncurmov(14,20); /* Move cursor.
    wprintf("Enter the name of the file: "); /* Print
    if(new_file[0] = kbgetkey(&scan)) != 'z')
    {
      wncurmov(14,48); /* Move cursor.
      wprintf(" "); /* Print
      wncurmov(14,48);
      kbplace(1,new_file[0],scan);
      wnquery(file_name, 9, &scan); /* Get next request.
      strncpy(new_file,file_name,9);
      strncpy(old_file,file_name,9); /* Store the backup.
      strcat(old_file, "*.old");
      strcat(new_file, "*.new");
    }
    else
    {
      strncpy(old_file,file_name,9);
      strncpy(new_file,file_name,9);
      strcat(old_file, "*.old"); /* Store the backup with .old extension.
      strcat(new_file, "*.new");
    }
  }
}

switch(option)

  case 1:
    if ((current = fopen(new_file, "r")) != NULL) /* If the new file exists. */
    {
      wnscroll(0,-1,-1,0); /* Scroll window
      wncurmov(5,5);
      wprintf("This file already exists. "); /* Print
      wncurmov(7,5); /* Move cursor. */
    }
wnprintf("{Press ENTER to continue}*");
wnquery(selection, 1, &scan);     /* Get the next selection. */
}
else
{
close(current);     /* Close the current file. */
current = fopen(new_file.*, "w");
action = 8;        /* If action is a resistor trim. */
wnprintf(current, ",\n", action);   /* Print */
wnscroll(0,-1,-1,0);     /* Scroll window */
wncurmov(5,5);
wnprintf("What cutting velocity would you like to set? (0 = 20.625) *"); /* Print */
wnquery(numbers, sizeof(numbers), &scan); /* Get the next selection. */
vel_cut = transform(numbers);
wnprintf(current, "%4.3f\n", vel_cut);    /* Print */
buid_file();    /* Do module. */
wnprintf(current, ",n9\n");    /* Close file. */
close(current);
}
break;
}
case 2:
if ((current = fopen(new_file.*, "r")) == NULL)    /* If the file doesn't exist. */
{
wnscroll(0,-1,-1,0);
wncurmov(5,5);             /* Move cursor. */
wnprintf("This file does not exist. *");   /* Print */
wncurmov(7,5);
wnprintf("(Press ENTER to continue) *");
wnquery(selection, sizeof(selection), &scan);
close(current);
break;
}
wnscroll(0,-1,-1,0);     /* Move cursor. */
wncurmov(6,10);
wnprintf("Do you want the file sent to the printer? (Y/n)"); /* Print */
wnquery(selection,2,&scan);
if (*selection == 'Y' || *selection == 'Y')    /* Selection is a yes. */
{
wncurmov(8,10);
wnprintf("Turn on the printer and press ENTER.*");  /* Print */
wncurmov(0,0);
wnquery(selection,1,&scan);    /* Find next selection. */
pattern_check(1);
}
else
pattern_check(0);    /* Do module. */
wnscroll(0,-1,-1,0);     /* Scroll window. */
wncurmov(5,0);
wnprintf("The final position is %6.1f, %6.1f, x_position, y_position");
wnprintf("\nThere were %d loops started,loops\n");
wnprintf("The laser was turned on at %d times,\n");
wnprintf("The laser was turned off at %d times,\n");
wnprintf("There were %d circles,circles\n");
wnprintf("\n{Press ENTER to continue}");
wncurmov(1,0,0,1);
wnquery(selection,1,&scan); /* Get next selection. */
close(current);     /* Close the file. */
break;
}
case 3:
if ((current = fopen(new_file.*, "r")) == NULL)
{
wnscroll(0,-1,-1,0);     /* Scroll window. */
wncurmov(5,5);
wnprintf("This file does not exist. *");    /* Print */
wncurmov(7,5);
wnprintf("(Press ENTER to continue) *");
wnquery(selection, 1, &scan);     /* Get next selection. */
close(current);
break;
}
wnscroll(0,-1,-1,0);     /* Move cursor */
append();
fclose(current);  /* Close file. */
break;

case 4:
if (current = fopen(new_file,"r")) == NULL)  /* If new file doesn't exist */
{
    wncrnmov(5,5);  /* Scroll window. */
    wncrnmov(7,5);  /* Print */
    wncrnmov(new_file);  /* Print */
    do_file();  /* Do module. */
    fclose(current);
    break;
}
wncrnmov(0,-1,-1,0);  /* Scroll window. */
do_file();  /* Do module. */
fclose(current);
break;

case 5:
return;
}

******************************************************************************

MODULE:  build_file

PURPOSE: Create menu system - create a file from scratch

PARAMETERS:
    GLOBL: current      scan
            selection    x_displace
            y_displace

    LOCAL: numbers - Character buffer for numeric input
         times - Number of times to process a loop
         rad - Radius of a circle
         types - Type of circle (cutting/saving center)

    USERS: cut_a_pattern append

    CALLS: wncrnmov
            wncrnmov
            wncrnmov
            wncrnmov
            wpnrmov
            wquery
            wprintf
            transform

RETURNS: NONE

DATE: 24 June 1990
PROGRAMMER: Ryp Walters
LANGUAGE: Microsoft C Version 5.1
OS: MS-DOS Version 3.3

******************************************************************************
wncurmov(1,26); /* Move cursor */
wnprintf("What would you like to do?\n"); /* Print */
wncurmov(3,28); /* Move cursor. */
wnprintf("1. Move to a position\n");
wncurmov(4,28);
wnprintf("2. Distance move \n");
wncurmov(5,28);
wnprintf("3. Laser On \n");
wncurmov(6,28);
wnprintf("4. Laser off\n");
wncurmov(7,28);
wnprintf("5. Circle cut \n");
wncurmov(8,28);
wnprintf("6. Start a loop\n");
wncurmov(9,28);
wnprintf("7. Repeat a loop\n");
wncurmov(11,28);
wnprintf("9. Exit\n");
wncurmov(13,34);
wnprintf("Option: \n");

wnquery(selection, 2, &scan);

if(*selection == 'n' || *selection == 'm' || *selection == '1')
{
  fprintf(current, "\n\n"); /* Print */
  wncurmov(15,20);
  wnprintf("To what position would you like to move?\n"); /* Print. */
  wncurmov(15,20);
  wnprintf("X direction: \n");
  wnquery(numbers, 10, &scan);
  x_displace = (int)transform(numbers); /* Calculate x displacement. */
  wncurmov(16,20);
  /* Move cursor */
  wnprintf("Y direction: \n");
  wnquery(numbers, 10, &scan);
  y_displace = (int)transform(numbers);
  fprintf(current, "\n\n", x_displace);
  fprintf(current, "\n\n", y_displace);
  /* Print. */
  wnscroll(1,-1,-1,0);
  /* Scroll one line. */
  wncurmov(2,40);
  wnprintf("Position \n51, %51, x_displace, y_displace\n");
}

if(*selection == 'd' || *selection == 'D' || *selection == '2')
/* If it is the second selection. */
{
  fprintf(current, "\n\n"); /* Print. */
  wncurmov(15,20);
  /* Print */
  wnprintf("What distance would you like to move?\n");
  wncurmov(15,20);
  wnprintf("X direction: \n"); /* Print */
  wnquery(numbers, 10, &scan);
  x_displace = (int)transform(numbers); /* New x distance. */
  wncurmov(16,20);
  wnprintf("Y direction: \n");
  wnquery(numbers, 10, &scan); /* Get next selection. */
  y_displace = (int)transform(numbers);
  fprintf(current, "\n\n", x_displace);
  fprintf(current, "\n\n", y_displace);
  /* Print */
  wnscroll(1,-1,-1,0);
  /* Lower window. */
  wncurmov(2,40);
  /* Select lower window. */
  wnprintf("Distance \n51, %51, x_displace, y_displace\n"); /* Print */
}

if(*selection == 'o' || *selection == 'O' || *selection == '3')
{
  fprintf(current, "\n\n"); /* Print */
  wnscroll(1,-1,-1,0);
  /* Scroll one line. */
  wncurmov(2,40);
  wnprintf("Laser on\n"); /* Print */
}
if('selection == 'f' || *selection == 'F' || *selection == '4')
{
  printf(current,"\n4\n") ;  /* Print */
  wnsselect(plist);  /* Select lower window. */
  wnscroll(1,-1,-1,0);
  wncursmov(2,40);
  wpnprintf(" Laser off ");  /* Print */
}

if('selection == 'c' || *selection == 'C' || *selection == '5')
{
  printf(current,"\n5\n");  /* Print */
  wncursmov(13,12);  /* Move cursor. */
  wnquery(numbers, 2, &scan);
  if('numbers == 'y' || *numbers == 'Y' )
    type = 0;
  else
    type = 1;  /* if it is a circle with a center. */
  wncursmov(15,12);
  wpnprintf("What radius would you like the circle? (in .01mm )");  /* Print */
  wnquery(numbers, 10, &scan);  /* Get the next selection. */
  rad = (int)transform(numbers);
  printf(current, "\n\", rad);
  printf(current, "\n\", type);  /* Print */
  wnsselect(plist);
  wnscroll(1,-1,-1,0);
  wncursmov(2,40);
  if(type == 1)
    wpnprintf(" Circle w/o cntr \%4i", rad);  /* Print */
  else
    wpnprintf(" Circle w/ center\%4i", rad);
}

if('selection == 's' || *selection == 'S' || *selection == '6')
{
  printf(current,"\n6\n") ;  /* Print */
  wncursmov(13,20);
  wpnprintf(" HOW many times would you like to loop? ");  /* Print */
  wnquery(numbers, 10, &scan);
  times = (int)transform(numbers);  /* Calculate the number of loops. */
  printf(current, "\n\", times);  /* Print */
  wnsselect(plist);
  wnscroll(1,-1,-1,0);  /* Scroll one line. */
  wncursmov(2,40);
  wpnprintf(" Start loop \%2i times", times);  /* Print */
}

if('selection == 'r' || *selection == 'R' || *selection == '7')
{
  printf(current,"\n7\n") ;  /* Print */
  wnsselect(plist);  /* Select lower window. */
  wnscroll(1,-1,-1,0);
  wncursmov(2,40);
  wpnprintf(" Repeat a loop ");  /* Print */
}

wnselect(plist);
wpnprintf("Current Position: \%6d, \%6d ",xpos,ypos);  /* Print */
wselect(pwin);  

'selection = ' 
wnselect(plist);
wnscroll(3,-1,-1,0);  /* Scroll three lines. */
wncursmov(0,2);
wpnprintf("Current Position: \%6d, \%6d ",xpos,ypos);  /* Print */
wselect(pwin);  /* Select upper window */

/*************************************************************/
void append()
{
    int row, action, times;
    char choice[2], choice1[2], numbers[10];

    old = fopen(old_file, "w");        /* Open extension file. */
    action = 0;
    fscanf(current, "%li", &action);    /* Scan file for selection. */
    fprintf(old, "%li\n", action);    /* Print */
    if(action == 8)
    {
        wncrmmov(0, -1, -1, 0);
        wncrmmov(10, 10, 0);
        fprintf("There is an error in this program!!!"); /* Print */
        wquery(choice, 2, &choice);
        fclose(old); /* Close extension file */
        return;
    }
    fscanf(current, "%s", &vel_cut);
    fprintf(old, "%3f\n", vel_cut);   /* Print */

    while(action != 9)                  /* As long as it is not an exit. */
    {
        fscanf(current, "%li", &action);
        fprintf(old, "%li\n", action); /* Print */
        if(action == 1)                  /* If it is a move position. */
        {
            fscanf(current, "%li", &x_travel);
            fscanf(current, "%li", &y_travel); /* Scan file for the y travel distance. */
            fprintf(old, "%li\n", x_travel);  /* Print */
            fprintf(old, "%li\n", y_travel);
        }
        if(action == 2)                  /* If it is a move distance. */
{  
    fscanf(current, "%i", &x_travel);  
    fscanf(current, "%i", &y_travel);  
    /* Scan file for the y travel distance. */  
    fprintf(old, "%i\n", x_travel);  
    /* Print */  
    fprintf(old, "%i\n", y_travel);  
    /* Print */  
}  

if(action == 5)  
    /* If it is a circle set */  
    {  
        fscanf(current, "%i", &radius);  
        /* Scan file for the radius. */  
        fscanf(current, "%i", &speed);  
        /* Scan file for the speed. */  
        fprintf(old, "%i\n", radius);  
        /* Print */  
        fprintf(old, "%i\n", speed);  
        /* Print */  
    }  

if(action == 6)  
    /* If it is a loop set */  
    {  
        fscanf(current, "%i", &times);  
        /* Scan file for the times. */  
        fprintf(old, "%i\n", times);  
        /* Print */  
    }  

fclose(old);  
fclose(current);  
/* Close file. */  

old = fopen(old_file,"r");  
/* Opening back up file. */  

row = 3;  
 fscanf(old, "%i", &action);  
    /* Scan file for the action. */  
    fscanf(old, "%f", &vel_cut);  
    /* Scan file for the cutting velocity. */  
    wnscroll(0,-1,-1,0);  
    /* Scroll window. */  
    wnprintf(" Cutting Velocity \%.3f\n", vel_cut);  
    /* Print */  
    wnprintf("
 Is this correct? (y/n)\n");  
    /* Find the next selection. */  
    wnqurey(choice,2,&scan);  
    /* Find the next selection. */  
    wncurmov(row,40);  
    /* Print */  
    wnprintf("\n");  
    /* Print */  
    if(choice == 'n' || choice == 'N')  
    {  
        wncurmov(+row,1);  
        wnprintf(" What cutting velocity do you want? (0 - 20.625)\n");  
        /* Print */  
        wnqurey(numbers,10,&scan);  
        /* Find the cutting velocity. */  
        vel_cut = transform(numbers);  
        /* Find the cutting velocity. */  
        fprintf(current, "\n\n\n",action);  
        /* Print. */  
        fprintf(current, "\%.3f\n", vel_cut);  
    }  

while(action != 0)  
    /* As long as it is not an exit. */  
    {  
        fscanf(old, "%i", &action);  
        if(action > 13)  
        {  
            wnscroll(5,-1,-1,0);  
            /* Scroll 5 lines. */  
            row = row - 5;  
        }  
        wncurmov(row,1);  
    }  

if(action == 1)  
    /* If it is a move */  
    {  
        "choice1 = 'n';  
        fscanf(old, "%i", &x_travel);  
        fscanf(old, "%i", &y_travel);  
        /* Scan backup for the y travel distance. */  
        wnprintf(" Position  \$i1, \$i2, \$i3, \$i4, \$i5, \$i6, \$i7, \$i8, \$i9\n", x_travel, y_travel);  
        /* Print */  
        wnprintf(" Is this correct? (y/n/q)\n");  
        /* Find the next selection. */  
        wnqurey(choice,2,&scan);  
        /* Find the next selection. */  
        wncurmov(row,40);  
        /* Print */  
        wnprintf("\n");  
        /* Print */  
        if(choice == 'q' || choice == 'Q')  
        {  
            fprintf(current, "\n\n\n",action);  
            /* Print */  
            fprintf(current, "%i\n", x_travel);  
            /* Print */  
            fprintf(current, "%i\n", y_travel);  
            /* Print */  
            transfer_rest_of_file();  
            /* Do module. */  
            return;  
        }  
    }  
}
{
wncurrmov(row,35);
wnprintf("Do you want to insert commands? (y/n)"); /* Print */
wnquery(choice1,2,&scan);
wncurrmov(row,35);
wnprintf(" "); /* Print */
if(*(choice1 == 'y' || *choice1 == 'Y')
{
    build_file(); /* Do module. */
    wnscroll(0,-1,-1,0); /* Scroll window. */
    row = 1;
    wncurrmov(+row,1); /* Move cursor. */
    wnprintf(" Position %s1, %s1", x_travel, y_travel); /* Print */
}
wncurrmov(row,35);
wnprintf("Delete this position move? (y/n)"); /* Print */
wnquery(choice1,2,&scan); /* Get the next selection. */
wncurrmov(row,35);
wnprintf(" "); /* Print */
if(*(choice1 == 'y' && *choice1 == 'Y') /* If choice is a yes. */
{
    wncurrmov(row,35);
    wnprintf("Keep the position the same? (y/n)"); /* Print */
    wnquery(choice1,2,&scan); /* Get next selection. */
    wncurrmov(row,35);
    wnprintf(" "); /* If choice == 'n', | 'choice == 'N')
{
    wncurrmov(+row,1);
    wnprintf(" What position would you like to move to? ' ");
    wncurrmov(+row,1);
    wnprintf(" X direction:"); /* Print */
    wnprintf(" "); /* Print */
    wnquery(numbers,10,&scan);
    x_travel = (int)transform(numbers);
    wncurrmov(+row,1);
    wnprintf(" Y direction:"); /* Print */
    wnquery(numbers,10,&scan);
    y_travel = (int)transform(numbers);
}
}
if(*(choice1 == 'y' && *choice1 == 'Y') /* If choice is a yes. */
{
    fprintf(current,"\n\n\n",action);
    fprintf(current,"\n\n", x_travel);
    fprintf(current,"\n\n", y_travel); /* Print */
}
}
if(action == 2) /* If it is a move distance. */
{
    'choice1 = 'n';
    fscanf(ol, "%c", &x_travel)); /* Scan backup file for the x travel distance. */
    fscanf(ol, "%c", &y_travel);
    wnpnprintf(" Distance %s1, %s1", x_travel, y_travel); /* Print */
    wnprintf(" Is this correct? (y/n/q) ");
    wnquery(choice1,2,&scan); /* Find the next selection. */
    wncurrmov(row,40);
    wnprintf(" "); /* Print */
    if(*(choice1 == 'q' || *choice1 == 'Q')
    {
        fprintf(current,"\n\n\n",action);
        fprintf(current,"\n\n", x_travel);
        fprintf(current,"\n\n", y_travel);
        transfer_rest_of_file();
        return;
    }
    if(*(choice1 == 'n' || *choice1 == 'N') /* Choice is a no. */
    {
        wncurrmov(row,35); /* Move cursor */
        wnprintf("Do you want to insert commands? (y/n)"); /* Print */
        wnquery(choice1,2,&scan);
        wncurrmov(row,35);
        wnprintf(" "); /* Print */
    }
}
250
{ build_file(); /* Do module. */
 wnscrollo(0,-1,-1,0); /* Define the row. */
 wncurmov(+row,1); wnprintf(‘ Distance \$51, \$51*, x_travel, y_travel\); /* Print */
}
 wncurmov(row,35); /* Move cursor.
 wnprintf(‘Delete this distance move? (y/n)\'); /* Print
 wnquery(choice1,2,&scan); wncurmov(row,35);
 wnprintf(‘* ‘); /* Print
 if(‘choice1 := ‘y’ & choice1 != ‘Y’)
 { wncurmov(row,35);
  wnprintf(‘ Keep the distance the same? (y/n)\'); /* Print
  wnquery(choice,2,&scan); /* Get the next selection */
  wncurmov(row,35);
  wnprintf(‘* ‘); /* Print
  if(‘choice := ‘n’ || ‘choice == ‘N’)
  { wncurmov(+row,1); /* Move cursor.
    wnprintf(‘ What distance would you like to move? \*\); wncurmov(+row,1);
    wnprintf(‘ X direction:*\'); /* Print
    wnquery(numbers,10,&scan);
    x_travel = (int)transform(numbers); /* Fine the x travel distance. */
    wncurmov(+row,1);
    wnprintf(‘ Y direction:*\); /* Print
    wnquery(numbers,10,&scan); /* Get the next selection. */
    y_travel = (int)transform(numbers);
  }
 }
 if(‘choice1 := ‘y’ & choice1 != ‘Y’) /* If the choice is a yes. */
 { fprintf(current,"\n@i\n",action);
  fprintf(current, ‘\%\n’, x_travel); /* Print
  fprintf(current, ‘\%\n’, y_travel);
 }
}

if(action == 3) /* If it is a velocity set */
{ ‘choice1 = ‘n’;
  wnprintf(‘ Laser on Is this correct? (y/n/q)\); /* Print
  wnquery(choice,2,&scan); /* Get the next selection. */
  wncurmov(row,40);
  wnprintf(‘* ‘); /* Print
  if(‘choice == ‘q’ || ‘choice == ‘Q’) /* Choice is a quit. */
  { fprintf(current,"\n@i\n",action);
   transfer_rest_of_file();
   return;
  }
  if(‘choice == ‘n’ || ‘choice == ‘N’)
  { wncurmov(row,35);
    wnprintf(‘Do you want to insert commands? (y/n)\'); /* Print
    wnquery(choice1,2,&scan); /* Get the next selection. */
    wncurmov(row,35);
    wnprintf(‘ ‘); /* Print
    if(‘choice1 := ‘y’ || ‘choice1 == ‘Y’)
    { build_file(); /* Do module. */
      wnscrollo(0,-1,-1,0);
      row = 1; /* Define row. */
      wncurmov(0,row,1); /* Laser on */
    }
  wncurmov(row,35);
  wnprintf(‘Delete this laser cn? (y/n)\); /* Print
  wnquery(choice1,2,&scan); /* Find the next selection. */
  wncurmov(row,35);
  wnprintf(‘* ‘); /* Print

251
if ('choice1' == 'y' && 'choice1' != 'Y')
    fprintf(current, "\n\n\n", action); /* Print */
}

if (action == 4) /* If it is a laser off. */
{
    'choice1' = 'n';
    wmprintf(" Laser off \n\n\n\n", action); /* Print
    wmpquery(choice2, &scap); /* Get the next selection.
    wncurmov(row, 40);
    wmprintf(" "); /* Print
    if ('choice == 'q' || 'choice' == 'Q') /* If the choice is a quit
    {fprintf(current, "\n\n\n", action); /* Print
        transfer_rest_of_file(); /* Do module.
        return;
    }
    if ('choice' == 'n' || 'choice' == 'N')
    {
        wncurmov(row, 35);
        wmprintf("Do you want to insert commands? (y/n/\n\n\n", action); /* Print
        wmpquery(choice1, 2, &scap);
        wncurmov(row, 35); /* Move cursor.
        wmprintf(" "); /* Print
        if ('choice' == 'y' || 'choice1' == 'Y')
        { DO module.
            wmscroll(0, -1, -1, 0);
            row = 1;
            wncurmov(\srow\, 1); /* Do module.
            wmprintf(" Laser off "); /* Print
        }
        wncurmov(row, 35);
        wmprintf("Delete this laser off? (y/n)\n\n\n", action); /* Print
        wmpquery(choice1, 2, &scap);
        wncurmov(row, 35); /* Get the next selection.
        wncurmov(row, 35);
        wmprintf(" "); /* Print
        if ('choice' == 'y' || 'choice1' == 'Y') /* Choice is no.
            fprintf(current, "\n\n\n", action); /* Print
        }
    }

if (action == 5) /* If it is a circle set */
{
    'choice1' = 'n';
    fscanf(old, "\n\n\n", radius);
    if (speed == 1)
        wmprintf(" Circle w/o cntr %4i", radius); /* Print
    else
        wmprintf(" Circle w/ center%4i", radius);
        wmprintf(" Is this correct? (y/n/q)\n\n\n", action); /* Print
        wmpquery(choice2, &scap);
        wncurmov(row, 40);
        wmprintf(" "); /* Print
        if ('choice' == 'q' || 'choice' == 'Q') /* Choice is a quit.
        {fprintf(current, "\n\n\n", action); /* Print
            fprintf(current, "\n\n\n", radius);
            fprintf(current, "\n\n\n", speed);
            transfer_rest_of_file(); /* Do the module.
            return;
        }
    }
}

if ('choice' == 'n' || 'choice' == 'N')
{
    wncurmov(row, 35); /* Do the module.
    wmprintf("Do you want to insert commands? (y/n)\n\n\n", action); /* Print
    wmpquery(choice1, 2, &scap); /* Get the next selection.
    wncurmov(row, 35);
    wmprintf(" "); /* Print
    if ('choice1' == 'y' || 'choice1' == 'Y') /* Choice is a yes.
    {build_file();
        wmscroll(0, -1, -1, 0); /* Scroll window.
    }
row = 1;
wncurmov(++row,1);
if(speed == 1)
   wprintf(" Circle w/o cntr %4i", radius);
else
   wprintf(" Circle w/ center%4i", radius); /* Print */
}
wncurmov(row,35);
wprintf("Delete this circle? (y/n)\n"); /* Print */
wquery(choice1,2,&scan); /* Get the next selection. */
wncurmov(row,35);
wprintf(" "); /* Print */
if(choice1 == 'y' && choice1 != 'Y')
   wncurmov(row,35);
wprintf(" Keep the same circle and radius? (y/n)\n", radius); /* Print */
wquery(choice1,2,&scan); /* Get the next selection. */
wncurmov(row,35);
wprintf(" "); /* Print */
if(choice == 'n' || choice == 'N')
   wncurmov(++row,1);
wprintf(" What radius do you want? (in .01mm)\n"); /* Print */
wquery(numbers,10,&scan);
radius = (int)transform(numbers); /* Calculate the radius. */
wncurmov(++row,1);
wprintf(" "); /* Print */
wquery(numbers,2,&scan);
if(numbers == 'n' || numbers == 'N')
   speed = 1;
else
   speed = 0; /* If it is a circle with a center. */
}

if(choice1 != 'y' && choice1 != 'Y') /* Choice is a yes. */
{
   fprintf(current, \n\n\naction);
   fprintf(current, *$n\n", radius); /* Print */
   fprintf(current, *$n\n", speed);
}

if(action == 6) /* If it is a loop */
{
   *choice1 = 'n';
   fscanf(old, "$i", &times);
   wprintf(" Start loop %2i times", times); /* Print */
   wprintf(" Is this correct? (y/n/q)\n"); /* Print */
   wncurmov(row,40);
   wprintf(" "); /* Print */
   if(choice == 'q' || choice == 'Q') /* If it is a quit. */
   {
      fprintf(current, \n\n\naction);
      /* Print */
      fprintf(current, "$n\n", times);
      transfer_rest_of_file(); /* Do the module. */
      return;
   }
   if(choice == 'n' || choice == 'N') /* Choice is a no. */
   {
      wncurmov(row,35);
wprintf("Do you want to insert commands? (y/n)*\n"); /* Print */
      wquery(choice1,2,&scan); /* Get the next selection. */
wncurmov(row,35);
   wprintf(" "); /* Print */
   if(choice1 == 'y' || choice1 == 'Y') /* Choice is a yes. */
   {
      build_file();
wscroll(0,-1,-1,0);
      row = 1;
wncurmov(++row,1);
wprintf(" Start loop %2i times", times); /* Print */
   }
   wncurmov(row,35);
wprintf("Delete this start of a loop? (y/n)*\n"); /* Print */
wnquery(choice),2,&scan); /* Get the next selection. */
wncurmov(row,35);  
wnprintf(" "); /* Print */
if(*choice1 == 'y' && *choice1 != 'Y')  
{  
   wncurmov(row,35);  
   wnprintf("Keep the %i loop repetitions? (y/n)*", times); /* Print */  
   wnquery(choice,2,&scan);  
   wncurmov(row,35); /* Move cursor. */  
   wnprintf(" ");  
   if(*choice == 'n' || *choice == 'N')  
   {  
      wncurmov(+row,1);  
      wnprintf("How many times do you want? "); /* Print */  
      wnquery(numbers,10,&scan); /* get the next selection. */  
      times = (int)transform(numbers); /* Calculate the number of loops. */  
   }  
}  

if(*choice1 == 'y' && *choice1 != 'Y')  
{  
   printf(current,"\n%i\n",action); /* Print */  
   printf(current, ";\n", times);  
}  

if(action == 7) /* If it is a repeat loop */  
{  
   *choice1 = 'n';  
   wnprintf("End loop Is this correct? (y/n/q)*");  
   wnquery(choice,2,&scan); /* Get the next selection. */  
   wncurmov(row,40);  
   wnprintf(" "); /* Print */  
   if(*choice == 'q' || *choice == 'Q') /* Choice is a quit. */  
   {  
      printf(current,"\n\%i\n",action); /* Print */  
      transfer_rest_of_file(); /* Do module. */  
      return;  
   }  
   if(*choice == 'n' || *choice == 'N') /* Choice is a no. */  
   {  
      wncurmov(row,35);  
      wnprintf("Do you want to insert commands? (y/n)*"); /* Print */  
      wnquery(choice1,2,&scan);  
      wncurmov(row,35);  
      wnprintf(" "); /* Print */  
      if(*choice1 == 'y' || *choice1 == 'Y')  
      {  
         build_file(); /* Do module. */  
         wnscroll(0,-1,-1,0); /* Scroll window. */  
         row = 1;  
         wncurmov(+row,1);  
         wnprintf(" End loop "); /* Print */  
      }  
      wncurmov(row,35); /* Move cursor. */  
      wnprintf("Delete this loop repetition? (y/n)*"); /* Print */  
      wnquery(choice1,2,&scan);  
      wncurmov(row,35);  
      wnprintf(" "); /* Print */  
      if(*choice1 == 'y' && *choice1 != 'Y') /* If choice is a no. */  
      {  
         printf(current,"\n\%i\n",action); /* Print */  
      }  
}  

if(action == 9) /* If it is a exit */  
{  
   *choice = 'n';  
   wnprintf("End of file Is this correct? (y/n)*");  
   wnquery(choice,2,&scan); /* Find the next selection. */  
   wncurmov(row,40);  
   wnprintf(" "); /* Print */  
   if(*choice == 'n' || *choice == 'N')  
   {  
      wncurmov(row,35); /* Move the cursor. */  
      wnprintf("Do you want to insert commands? (y/n)*"); /* Print */  
      wnquery(choice1,2,&scan);  
}
void transfer_rest_of_file()
{
    int action, times;

    while(action != 9) /* As long as it isn't an exit. */
    {
        fscanf(old, "%i", &action);
        fprintf(current, "\n\n\n\n", action); /* Print */
        if(action == 1) /* If it is a move */
            {
                fscanf(old, "%i", &x_travel);
                fscanf(old, "%i", &y_travel);
                fprintf(current, "\n\n\n\n", x_travel);
                fprintf(current, "\n\n\n\n", y_travel);
            }
        if(action == 2) /* If it is a move */
            {
                fscanf(old, "%i", &x_travel);
                fscanf(old, "%i", &y_travel);
                fprintf(current, "\n\n\n\n", x_travel);
                fprintf(current, "\n\n\n\n", y_travel);
            }
        if(action == 5) /* If it is a circle set */
            {
                fscanf(old, "%i", &radius);
                fprintf(current, "\n\n\n\n", radius);
            }
    }
}
fscanf(old, "%i", &speed);
fprintf(current, "%i\n", radius);  /* Print */
fprintf(current, "%i\n", speed);  /* */
}

if(action == 6) /* If it is a loop */
{
    fscanf(old, "%i", &times);  /* Scan file for the number of loops. */
    fprintf(current, "%i\n", times);  /* Print */
}

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

MODULE:  pattern_check

PURPOSE: Analyze and optionally print the file

PARAMETERS:  GLOBL:  current       print
               prn_file     radius
               scan         selection
               x_position   y_position
               x_travel     y_travel
               speed        laser_status
               vel          vel_cut

PARAM:  print_it            - Option to send the file to the printer

LOCAL:  action              - Laser action variable
        count             - Array counter
        i                - Loop counter
        k                - Loop counter
        loop_times      - Array for the # of times a loop is done
        looping          - Array counter for loop to keep track of
        nested_x        - Array for the x posit. during the loops
        nested_y        - Array for the y posit. during the loops
        times           - Number of times to process a loop
        row             - Row number of the screen

USERS:  cut_a_pattern

CALLS:  fopen              fprintf
         fscanf             wmscroll
         wnquery            wmprintf
         velocity_set

RETURNS:  NONE

DATE:  24 June 1990
PROGRAMMER:  Ryp Walters
LANGUAGE:  Microsoft C Version 5.1
OS:  MS-DOS Version 3.3

-------------  INITIALS  -------------------------------
             -------------------------------
***************************************************************************/

void pattern_check(int print_it)
{
    int i, k, row, count;
    int action, times[5];
    int nested_x[5], nested_y[5], loop_times[5], looping;

    laser_status = OFF;            /* Defining initial values. */
    count = 0;
    for(k=0; k<5; k++)
    {
        times[k] = 0;
        nested_x[k] = 0;
    }
nested_y[k] = 0;
loop_times[k] = 0;

x_position = 0;
y_position = 0;
circles = 0;
loops = 0;
end_loops = 0;
on = 0;
off = 0;
looping = 0;
row = 1;

if(print_it == 1)
  fprintf(stderr, " Action Parameters Position Velocity\n";
  /* Print */
action = 0;

  fscanf(current, "%li", &action); /* Scan file for the action. */
  if(action == 0)
    /* If it isn't a resistor trim */
  {
    wncursmov(10,10); /* Scroll window. */
    wnprompt("There is an error in this program!!!"); /* Print */
    wquery(selection,2,&scan);
    fclose(current);
    return; /* Return */
  }

  fscanf(current, "%f", &vel_cut);
if(print_it == 1)
  fprintf(stderr, " Cutting Vel. %7.3f
action, vel_cut;

while(action != 9) /* While action isn't exit. */
{
  fscanf(current, "%li", &action); /* Scan file for the action. */

  if(action == 1) /* If it is a move */
  {
    fscanf(current, "%li", &x_travel); /* Scan file for the x travel distance */
    fscanf(current, "%li", &y_travel);
    if(laser_status == ON)
      /* If laser is on */
      vel = vel_cut;
    else
      vel = velocity_set(x_travel-x_position, y_travel-y_position); /* Find the safe veloctiy */
  }

  if(print_it == 1)
    fprintf(stderr, " Position %5i,%5i %5i,%5i %7.3f\n",
            action, x_travel, y_travel, x_position, y_position, vel);

  x_position = x_travel; /* Calculate the x position. */
  y_position = y_travel;
}

if(action == 2) /* If it is a move */
{
  fscanf(current, "%li", &x_travel); /* Scan file for the x travel distance. */
  fscanf(current, "%li", &y_travel);
  if(laser_status == ON)
    /* Calculate the cutting velocity */
    vel = velocity_set(x_travel, y_travel);
  else
    vel = velocity_set(x_travel, y_travel);

  if(print_it == 1)
    fprintf(stderr, " Distance %5i,%5i %5i,%5i %7.3f\n",
            action, x_travel, y_travel, x_position, y_position, vel);

  x_position = x_travel + x_position; /* Calculate the x position. */
  y_position = y_travel + y_position;
}

if(action == 3) /* If it is a laser on */
{}

k = 1;
if(looping != 0)
{
  for(i=1; i <= looping + 1; i++) /* Increment the number of loops. */
  {

257
k = k * loop_times[i]; /* keep track of the number of loops. */
}

on = on + k; /* Keep track of the number of times the laser has been on. */
if(print_it == 1)
  fprintf(stdout, " Laser on %5i,%5i %7.3f\n", action, x_position, y_position, vel);
/* Print. */
laser_status = ON;
}

if(action == 4) /* If it is a laser off */
{
  k = 1; /* Number of times laser has been on. */
  if(looping != 0)
    for(i=1; i != looping + 1; i++) /* Keeping track of the number of loops. */
    {
      k = k * loop_times[i];
    }
  off = off + k; /* Keeping track of the number of times the laser has been off. */
  if(print_it == 1)
    fprintf(stdout, " Laser off %5i,%5i %7.3f\n", action, x_position, y_position, vel);
  laser_status = OFF; /* Turning laser off. */
}

if(action == 5) /* If it is a circle set */
{
  fscanf(current, "%1", &radius); /* Scan file for the radius */
  fscanf(current, "%1", &speed);
  k = 1; /* Number of circles */
  if(looping != 0)
    for(i=1; i != looping + 1; i++) /* Keeping track of the number of times */
    {
      k = k * loop_times[i];
    }
  circles = circles + k; /* Number of circles. */
  if(print_it == 1)
    {
      if(speed == 1) /* If it is a circle with a center. */
        fprintf(stdout, " Circle w/o cntr%5i %5i,%5i 2.000\n", action, radius, x_position, y_position, vel);
      else
        fprintf(stdout, " Circle w/center%5i %5i,%5i 1.000\n", action, radius, x_position, y_position, vel);
    }
}

if(action == 6) /* If it is a start loop */
{
  fscanf(current, "%1", &times[count]); /* Scan file for the number of loops. */
  nested_x[count] = x_position;
  nested_y[count] = y_position;
  looping++; /* Increment */
  loop_times[looping] = times[count]; /* Calculate the number of loops. */
  loops++;
  if(print_it == 1)
    fprintf(stdout, " Start Loop %3i times %5i,%5i %7.3f\n", action, times[count], x_position, y_position, vel);
  count++;
/* Increment */
}

if(action == 7) /* If it is an end loop */
{
  if(count != 0) /* If is isn't the first loop */
    num_times[count - 1] = times[count]; /* Calculate the number of loops. */
  }

258
{ count--; 
  x_position = nested_x[count] + times[count]*{x_position - nested_x[count]}; /* Calculate the x position. */ 
  y_position = nested_y[count] + times[count]*{y_position - nested_y[count]}; 
  end_loops++; 
  if(print_it == 1) /* If it is to be sent to the printer. */ 
  { 
    fprintf(stderr, " \r- Loop repetition \n"); 
    fprintf(stderr, " \r\%li End loop \n\%li,\%li \n\%7.3f\n", 
      action, x_position, y_position, vel); 
  } 
  looping--; /* Decrement the number of loops */ 
  else 
  { 
    end_loops++; /* Keep track of the number of loops completed */ 
    if(print_it == 1) 
    { 
      fprintf(stderr, " \r\%li EXTRA END LOOP \n\%li,\%li \n\%7.3f\n", 
        action, x_position, y_position, vel); 
    } 
  } 
  
  if(print_it == 1) 
  fprintf(stderr, " \r\%li End of File \n\%li,\%li \n\%7.3f\n", 
    action, x_position, y_position, vel); /* Print */ 
  if(print_it == 1) /* If it is to be sent to the printer. */ 
  { 
    fprintf(stderr, " \r\n\nThe final position is \%li, \%li, \%li, \n\n\%7.3f\n", 
      x_position, y_position); /* Send the following messages to the printer. */ 
    fprintf(stderr, " \r\n\nThere were %li loops started.*,loops); 
    fprintf(stderr, " \r\n\nThere were %li loops ended.*,end_loops); 
    fprintf(stderr, " \r\n\nThe laser was turned on %li times.*,on); 
    fprintf(stderr, " \r\n\nThe laser was turned off %li times.*,off); 
    fprintf(stderr, " \r\n\nThere were %li circles.*,circles); 
    fprintf(stderr, "); 
    fprintf(stderr, "); 
    fprintf(stderr, "); 
    fprintf(stderr, "); 
    fprintf(stderr, "); 
    fprintf(stderr, "); 
    fprintf(stderr, "); 
    fprintf(stderr, "); 
    fprintf(stderr, "); 
    fprintf(stderr, "); 
  } 
} 

******************************************************************************

* MODULE:  do_file
* PURPOSE:  Base loop execution (internal loops are processed separately)
* PARAMETERS:  GLOBAL:  circles  current
*              end_loops  loops
*              num_loops  off
*              on  plist
*              pmin  resting

259
void do_file()
{
    int i, j, bits, times, row, col, action;

    pattern_check(0);
    if((loops != end_loops) || (on != off) || (x_position != 0) || (y_position != 0))
    {
        wncursorm(7,10);
        wperror("This file is not operationally correct!!");
        wperror("Do you want to run the program anyway? {y/n}");
        wperror("Selection = ");
        if(*selection == 'n' || *selection == 'N') /* If selection is no. */
            return;
        fclose(current);
        current = fopen(new_file, 'r'); /* Open new file. */
        laser_off(); /* Laser off. */
        laser_status = OFF; /* Defining variables. */
        loops = 0;
        end_loops = 0;
        x_position = 0;
        y_position = 0;
        circles = 0;
        on = 0;
        off = 0;
wnscroll(0,-1,-1,0);  /* Scroll window */
wncurvomov(7,10);
wpprint("Is the current position the origin? (y/n)\n");  /* Print */
wquery(selection,2,\&scan);
if(*selection == 'n' || *selection == 'N')  /* Selection is no. */
   some_move();  /* Module. */
rdposn();
x_displace = xpos;  /* New X position. */
y_displace = ypos;
wnscroll(0,-1,-1,0);  /* Move cursor. */
wncurvomov(7,10);
wprint("Press \"START\" when you are ready. ");  /* Print */
wait_for_start();  /* Module */
action = 0;

wnscroll(0,-1,-1,0);  /* Scroll window */
wcurvomov(1,1);
wpprintf("%s Action Parameters Position Velocity\n");  /* Print */
fsconf(current, \"%11s, \&action\n");
if(action == 8)
{
   wncscroll(0,-1,-1,0);  /* Scroll window */
wcurvomov(10,10);
wprint("There is an error in this program!!!\n");  /* Print */
wquery(selection,2,\&scan);
forclose(current);
return;
}
fsconf(current, \"%f\n\", \&vel_cut\n");
row = 2;
wncurrow(row,1);
wpprintf("%11s Cutting Vel. \$7.3f\n\n\0, 0 0.000\n\", action, vel_cut\n");

while(action <= 9)  /* If it is an exit */
{
if(wait_for_resett2[] == 1)
{
   move_dist(0,0,0,0);  /* Don't move */
laser_off();  /* Laser off */
laser_status = OFF;
rdposn();  /* Module */
vel = velocity_set( -xpos + x_displace, -ypos + y_displace\n");
move_dist( -xpos + x_displace, -ypos + y_displace, vel, vel\n");  /* Move distance. */
break;
}
fsconf(current, \"%11s, \&action\n");
if(++row > 17)
{
   wnsroll(5,-1,-1,0);  /* Scroll five lines. */
   row = row - 5;
}
wncurrow(row,1);

if(action == 1)  /* If it is a move */
{
   fsconf(current, \"%1s, \&x_travel\n");  /* Scan file. */
   fsconf(current, \"%1s, \&y_travel\n");
   rdposn();
   if(laser_status == ON)
      vel = vel_cut;
   else
      vel = velocity_set(x_travel - x_displace, y_travel - y_displace\n");
move_dist(x_travel - x_displace, y_travel - y_displace, vel, vel\n");  /* Move distance. */
   wpprintf("%11s Position \$51,\$51 %51,\$51 \$7.3f\n\n\", action, x_travel, y_travel, x_position, y_position, vel\n");
x_position = x_travel;
y_position = y_travel;
}

if(action == 2)  /* If it is a move */

{  
  fscanf(current, "%f", &x_travel);
  fscanf(current, "%f", &y_travel);  
  if(laser_status == ON)  
    vel = vel_cux;  
    /* Set cutting velocity. */  
  else  
    vel = velocity_set(x_travel, y_travel);  
  move_dist(x_travel, y_travel, vel, vel);  
    /* Move distance. */  
  wprintf("%11.3f\n",  
    action, x_travel, y_travel, x_position, y_position, vel);  
  x_position = x_travel + x_position;  
    /* New x position. */  
  y_position = y_travel + y_position;
}  

if(action == 3)  
  /* If it is a laser on */  
{  
  on++;  
  wprintf("%11.3f\n",  
    action, x_position, y_position, vel);  
    /* Print */  
  wnesday(plist);  
  wncm(0,2);  
    /* Move cursor */  
  rdposn();  
    /* Do module */  
  wprn(11, 61i, 61i, x_pos, ypos);  
    /* wnesday(pwin); */  
  laser_on();  
    /* Do module */  
  laser_status = ON;  
}  

if(action == 4)  
  /* If it is a laser off */  
{  
  off++;  
  wprintf("%11.3f\n",  
    action, x_position, y_position, vel);  
    /* Print */  
  laser_off();  
  wnesday(plist);  
    /* Lower window. */  
  wncm(0,2);  
  rdposn();  
    /* wnesday(pwin); */  
  laser_status = OFF;  
}  

if(action == 5)  
  /* If it is a circle set */  
{  
  fscanf(current, "%f", &radius);  
    /* Scan file. */  
  circles++;  
  laser_off();  
  fscanf(current, "%f", &speed);  
    /* Scan file. */  
  if(speed == 1)  
  {  
    wprintf("%11.3f\n",  
      action, radius, x_position, y_position);  
    /* Scan file. */  
    circle_cut(1);  
  }  
  else  
    wprintf("%11.3f\n",  
      action, radius, x_position, y_position);  
    /* Scan file. */  
    circle_cut(0);  
  }  
    /* Lower window. */  
  wnesday(plist);  
  wncm(6,2);  
  rdposn();  
    /* wnesday(pwin); */  
  wait_for_res();  
  if(laser_status == ON)  
    laser_on();  
}
if(action == 6) {
    /* If it is a start loop */
    fscanf(current, "%i", &times);
    wprintf("%li Start Loop 0 %li times %li,%li
        action, times, x_position, y_position, vel");
    loops++;
    num_loops = 0;
    set_up_a_loop();
    num_loops = 0;
    if(do_a_loop(times) == 1)
    {
        move_dist(0,0,0,0); /* Don't move. */
        laser_off(); /* Do module. */
        laser_status = OFF; /* Laser is off. */
        vel = velocity_set(-xpos + x_displace, -ypos + y_displace); /* New velocity. */
        move_dist(-xpos + x_displace, -ypos + y_displace, vel, vel);
        break;
    }
    wait_for_rest();
    wprintf("%li End Loop 0 %li,%li
        x_position, y_position, vel");
    /* Print */
    end_loops = 1;
    wncursor(&row,&col);
    /* Move cursor. */
}
if(action == 7) {
    /* If it is an repeat loop */
    wprintf("- Extra end of repetition ");
    /* Print */
    wprintf("%li End loop %li,%li
        action, x_position, y_position, vel");
    /* Increment number of endloops. */
    end_loops++;
    return;
}

wncurbodrow();

wprintf("End of File %li,%li
        y_position, vel");
/* Print */

wmscroll(0,-1,-1,0); /* Scroll 8 lines.

wncurunov(9,1);
/* The final position is %li, %li, \n        x_position, y_position);*/

wprintf("\n There were %li loops started.*\,loops\); /* Print */

wprintf("\n There were %li loops ended.*\,end\_loops\); /* Print */

wprintf("\n The laser was turned on %li times\,*\,on\); /* Print */

wprintf("\n There were %li circles\,*\,circles\); /* Print */

wselect(plist); /* Select lower window.

wcurmov(0,2);
/* Do module.

wprintf("Current Position: %li, %li
        \,xpos, ypos\); /* Print */

laser_off(); /* Do module. */

wselect(plist); /* Lower window.

wprintf("\n (PRESS ENTER to continue)\); /* Print */

wquery(selection,2,&scan);
/* Close file. */

/****************** M U L T I P L E D X E M S 1 0 /***********/
/*
 * MODULE: set_up_a_loop
 * PURPOSE: To store in memory the (nested) loops of the file
 * PARAMETERS: GLOBL: current nested_loops
 * */

263
void set_up_a_loop()
{
    int action, times, loop_number, units_of_step;
    float velo_set;

    action = 0;            /* Set initial value. */
    loop_number = num_loops;
    num_loops++;           /* As long as it is not a 7 */
    units_of_step = 0;
    while(action != 7)
    {
        fscanf(current, "%li", &action);   /* Scan file. */
        nested_loops[loop_number][units_of_step] = action;    /* Calculate nested loops. */

        if(action == 1)       /* If it is a move */
        {
            fscanf(current, "%li", &x_travel);   /* Scan file for x travel distance. */
            fscanf(current, "%li", &y_travel);   /* Scan file for y travel distance. */
            values[loop_number][units_of_step][0] = x_travel;
            values[loop_number][units_of_step][1] = y_travel;  /* Find location for pointer. */
        }

        if(action == 2)       /* If it is a move */
        {
            fscanf(current, "%li", &x_travel);   /* Scan file for x travel distance. */
            fscanf(current, "%li", &y_travel);   /* Scan file for y travel distance. */
            values[loop_number][units_of_step][0] = x_travel;
            values[loop_number][units_of_step][1] = y_travel;  /* Find location for pointer. */
        }

        if(action == 5)       /* If it is a circle set */
        {
            fscanf(current, "%li", &radius);    /* Find radius. */
            fscanf(current, "%li", &speed);     /* Find type. */
            values[loop_number][units_of_step][0] = radius;
            values[loop_number][units_of_step][1] = speed;
        }

        if(action == 6)       /* If it is a start loop */
        {
            fscanf(current, "%li", &times);    /* Find number of times to do loop. */
            values[loop_number][units_of_step][0] = times;
            set_up_a_loop();                    /* Do module. */
        }

        units_of_step += 1;
    }
}
int do_a_loop(int count)
{
    int row, col, times, counter, loop_number, action, units_of_step;

    loop_number = num_loops;
    wncurpos(&row,&col);

    for (counter = 0; counter < count; counter++)
    {
        num_loops = loop_number;
        units_of_step = 0;
        action = 0;
        while (action != 7) /* As long as it isn't a resistor trim. */
            
            if (wait_for_rest2() == 1)
                return(1);
            
            action = nested_loops[loop_number][units_of_step];
    
            if (++row > 13) /* If the number of rows is more than 13 */
{  
  wnscroll(5,-1,-1,0);  /* Scroll window five lines. */  
  row = row - 5;  /* Calculate new row for window. */  
}  
wncurrmov(row,1);  /* Move cursor. */  

if(action == 1)  /* If it is a move */  
{  
  x_travel = values[loop_number][units_of_step][0];  /* Calculate new x position. */  
  y_travel = values[loop_number][units_of_step][1];  
  rdpsonth();  /* Do module. */  
  if(laser_status == ON)  
    vel = vel_cut;  /* Find new velocity. */  
  else  
    vel = velocity_set(x_travel - xpos + x_displace,  
      y_travel - ypos + y_displace);  
  move_dist(x_travel - x_pos + x_displace,  
      y_travel - y_pos + y_displace, vel, vel);  
  / * Find the distance to move. */  
  wnprintf("%5i  Position  %5i,%5i  %7.3f\n",  
      action, x_travel, y_travel, x_position, y_position, vel);  
      /* Print. */  
  x_position = x_travel;  /* Find new y position. */  
  y_position = y_travel;  
}  

if(action == 2)  /* If it is a move */  
{  
  x_travel = values[loop_number][units_of_step][0];  /* Find new x travel distance. */  
  y_travel = values[loop_number][units_of_step][1];  
  if(laser_status == ON)  
    vel = vel_cut;  /* Find new velocity. */  
  else  
    vel = velocity_set(x_travel, y_travel);  /* Find new velocity */  
  move_dist(x_travel, y_travel, vel, vel);  
  wnprintf("%5i  Distance  %5i,%5i  %7.3f\n",  
      action, x_travel, y_travel, x_position, y_position, vel);  
      /* Print */  
  x_position = x_travel + x_position;  /* Find new x position. */  
  y_position = y_travel + y_position;  
}  

if(action == 3)  /* If it is a laser on */  
{  
  on++;  
  wnprintf("%5i  Laser on  %5i,%5i  %7.3f\n",  
      action, x_position, y_position, vel);  
      /* Print */  
  wnselect(plist);  /* Select lower window. */  
  wncurrmov(0,2);  
  rdpsonth();  
  wnprintf("Current Position: %6i, %6i  *",xpos, ypos);  
      /* Print */  
  wnselect(pwin);  /* Select upper window. */  
  laser_on();  
  laser_status = ON;  /* Change laser status to on. */  
}  

if(action == 4)  /* If it is a laser off */  
{  
  off++;  
  wnprintf("%5i  Laser off  %5i,%5i  %7.3f\n",  
      action, x_position, y_position, vel);  
      /* Print */  
  laser_off();  
  wnselect(plist);  
  wncurrmov(0,2);  
  rdpsonth();  /* Do module. */  
  wnprintf("Current Position: %6i, %6i  *",xpos, ypos);  
  wnselect(pwin);  /* Select lower window. */  
  laser_status = OFF;  
}  

if(action == 5)  /* If it is a circle set */  
{  
  /* Code for circle set */  
}  

266
{ radius = values(loop_number)[units_of_step][0]; speed = values(loop_number)[units_of_step][1]; /* Find type of circle. */
circles++; /* Turn the laser off. */
if(speed == 1)
    { wprintf("%li  Circle w/o cnt%li  %li,%li  2.000\n", action, radius, x_position, y_position);
        /* Print. */
circle_cut(1); /* Do module. */
}
else
    { wprintf("%li  Circle w/center%li  %li,%li  2.000\n", action, radius, x_position, y_position);
        /* Print. */
circle_cut(0); /* Do module. */
}
wnselect(p_list); /* Select lower window */
wncurrmov(0,2); /* Do module. */
wprintf("Current Position: %li,%li  \%.xpos, ypos\n", x_pos, y_pos);
/* Print. */
wnselect(p_win); /* Select lower window. */
wait_for_rest();
if(laser_status == ON) /* If laser is on. */
    { laser_on();
    }
if(action == 6) /* If it is a start loop */
    { times = values(loop_number)[units_of_step][0]; /* Calculate number of times. */
um loops++;
wprintf("%li  Start Loop %i  %li times  %li,%li  \%.7.3f\n", action, loop_number + 1, times, x_position, y_position, vel);
        /* Print. */
loops++; /* If it is the first loop. */
if(do_a_loop(times) == 1)
    { return(1);
wprintf("%li  End Loop %i  %li,%li  \%.7.3f\n", loop_number + 1, x_position, y_position, vel);
        /* Print */
end_loops += 1; /* Increment the number of endloops. */
wncurpos(&row, &col); /* Move cursor. */
}
}
units_of_step++;
}
return(0);
APPENDIX VII

LASER/FILEMAKE INCLUDE FILES

/* main.h */
void main(void);
void scribe(void);
void some_move(void);
void circle_cut(int);
float transform(char *);
void wait_for_rest(void);
int wait_for_rest2(void);
void circle(void);
int velocity_set(int, int);
/* */

/* moveincd.h */
void skip_on_rest(void);
void encode(int, int, float, float);
void move_norest(int, int, float, float);
void move_dist(int, int, float, float);
void move_t(void);
void ciirhome(void);
void rdjoy(void);
void laser_on(void);
void disable_lim_cro(void);
void laser_off(void);
void rdposln(void);
void power(int, int);
void rdotos(void);
void skhome(void);
void ciirinr(void);
void index_pos(void);
void j祢ove(void);
void wait_for_start(void);
int check_for_stop(void);
/* */

/* savincd.h */
void cdecl cut_a_pattern(void);
void cdecl build_file(void);
void cdecl append(void);
void cdecl pattern_check(int);
void cdecl do_file(void);
int cdecl do_a_loop(int);
void cdecl set_up_a_loop(void);
void cdecl transfer_rest_of_file(void);
/* */
/* measincl.h */

void astandard(void);
void arange(void);
void asgen_volt(void);
void detect_gain(void);
void pre_gain(void);
void accomparator(void);
void delay(void);
void cirol(void);
void skstrobe(void);
void strobe(void);
void clraccon(void);
void clracov(void);
void skend_delay(void);
void skready(void);
void skend_conv(void);
void rdconvert(void);
void channelb(void);
void channela(void);
void probe_select(void);
void findme(void);

/* */

/* newstuff.h */

void laser_menu(void);
void change_defaults(void);
void measure_test_screen(void);
void measure_test(void);
void change_measure_test_values(void);
void display_feedback(void);
void laser_trim(void);
void probe_request(void);
void confirm_probes(void);
void current_resistor(void);
void resistor_corners(void);
void get_wanted_resistor(void);

/* */

/* globlvar.h */

unsigned int resting, xjoy, yjoy, xtos, ytos, xhome, yhome;
int x_axis, y_axis, maxx, llbx, llsbx, whisbx, vllsbx;
int msby, hlsby, lisby, whlsby, vllsb;
long int xpos, ypos;
float vel;

unsigned int strobe_done, ready, end_conv, end_delay, hprobe, aprobe, range;
unsigned int pre_amp, compare, delay, overload;
unsigned int probehigh, probealow, probehigh, probealow, hdelay, ldelay;
unsigned int hrange, lrange, hstandard, lstandard, hgen_volt, lgen_volt;
unsigned int hgain, lgain, hpre_amp, lpre_amp, hcompare, lcompare;

int convertor;
float resistor, standard, gen_volt, gain;

BWINDOW *pwin, *plist;

int radius, scan, x_travel, y_travel;
char selection[3];
char file_name[9];
char old_file[12];
char new_file[12];
char prn_file[12];

FILE *old, *current, *print;
/* measincd.h */

void astandard(void);
void arange(void);
void aleg_volt(void);
void aleg_gain(void);
void aleg_spec(void);
void scanout(void);
void strobe(void);
void circon(void);
void circonv(void);
void skeno_delay(void);
void skeno_adv(void);
void skeno_conv(void);
void rd overrun(void);
void rdconvert(void);
void channelb(void);
void channela(void);
void probe_select(void);
void findme(void);

/* newstuff.h */

void laser_menu(void);
void change_defaults(void);
void measure_test_screen(void);
void measure_test(void);
void change_measure_test(void);
void display_feedback(void);
void laser_trim(void);
void probe_request(void);
void confirm_probes(void);
void confirm_resistor(void);
void resistor_corners(void);
void get_resistance(void);

/* globvar.h */

unsigned int xreset, yreset, xtos, ytos, xhome, yhome;
int xaxis, yaxis, msbx, hsbx, llsbx, vlsbx, vllsbx;
int msby, hisby, llsby, vlsby, vllsb;
long int xpos, ypos;
float vel;

unsigned int strobe_done, ready, end_conv, end_delay, bprobe, aprobe, range;
unsigned int pre_amp, compare, delay, overload;
unsigned int probehigh, probehigh, probelow, probelow, hdelay, idelay;
unsigned int hrange, lrange, hstandard, lstandard, hgen_volt, lgen_volt;
unsigned int hgain, lgain, hpre_amp, lpre_amp, hcompare, lcompare;
int convertor;
float resistor, standard, gen_volt, gain;

BWINDOW *pwin, *plist;

int radius, scan, x_travel, y_travel;
char selection[3];

char file_name[9];
char old_file[12];
char new_file[12];
char prn_file[12];

FILE *old, *current, *print;
float vel, vel_cut;
int loops, end_loops, on, off, x_position, y_position, circles;
int nested_loops[30][200];
int values[30][200][2];
int x_displace, y_displace, num_loops, units_of_step;

double rl;
int stop, laser_status, speed;

long int starting_corner_x, starting_corner_y,
    adj_starting_corner_x, adj_starting_corner_y,
    opp_starting_corner_x, opp_starting_corner_y;

int number_of_serp_cuts = 5, abort_dist_lcut = 50, abort_dist_straight = 50;
int double_cut_ratio = 20, turn_after = 80;
float trim_speed = 0.125;

/** */

/* extrnvar.h */

extern unsigned int xresting, xjoe, yjoy, xctos, yotos, xhome, yhome;
extern int x_axis, y_axis, mxbx, hlsbx, llsbx, vhlsbx, vlshx;
extern int mshby, hlsby, llsby, vhlsby, vlshby;
extern long int xpos, ypos;
extern float vel;

extern unsigned int strobe_done, ready, end_conv, end_delay, bprobe, aprobe, range;
extern unsigned int pre_amp, compare, delay, overload;
extern unsigned int probe-high, probe-low, probe-high, probe-low, hdelay, ldelay;
extern unsigned int hrange, lrange, hstandard, lstandard, hgen_volt, lgen_volt;
extern unsigned int hgain, lgain, hpre_amp, lpre_amp, hcompare, lcompare;
extern int convertor;
extern float resistor, standard, gen_volt, gain;

extern long int posx[2], posy[2];
extern BWINDOW *pwin, *plist;

extern int radius, scan, x_travel, y_travel;
extern char selection[3];

extern char file_name[9];
extern char old_file[12];
extern char new_file[12];
extern char prn_file[12];

extern FILE *old, *current, *print;

extern float vel, vel_cut;
extern int loops, end_loops, on, off, x_position, y_position, circles;
extern int nested_loops[30][200];
extern int values[30][200][2];
extern int x_displace, y_displace, num_loops, units_of_step;

extern double rl;
extern int stop, laser_status, speed;

extern long int starting_corner_x, starting_corner_y,
        adj_starting_corner_x, adj_starting_corner_y,
        opp_starting_corner_x, opp_starting_corner_y;

extern int number_of_serp_cuts, abort_dist_lcut, abort_dist_straight;
extern int double_cut_ratio, turn_after;
extern float trim_speed;

/** */

270
/* defntion.h */
#define bmba 0x300
#define bmbc 0x302
#define bmcbcom 0x303
#define baca 0x304
#define bacb 0x305
#define baccc 0x306
#define bacccom 0x307
#define acca 0x308
#define accb 0x309
#define accc 0x30a
#define acccom 0x30b
#define outcbr 0x80 /*command config code for output all ports*/
#define inckt 0x9b /*command config code for input all ports*/
#define addr13a 0x0b
#define addr13c 0x06
#define addr15a 0x0d
#define addr15c 0x06
#define addr16a 0x0e
#define addr16c 0x06
#define addr17a 0x0f
#define addr17c 0x06
#define addr30a 0x18
#define addr30c 0x04
#define addr31a 0x19
#define addr31c 0x04
#define addr32a 0x1a
#define addr32c 0x04
#define addr40a 0x20
#define addr40c 0x03
#define addr41a 0x21
#define addr41c 0x03
#define addr42a 0x22
#define addr42c 0x03
#define addr43a 0x23
#define addr43c 0x03
#define addr44a 0x24
#define addr44c 0x03
#define addr52a 0x2a
#define addr52c 0x02
#define addr53a 0x2b
#define addr53c 0x02
#define addr54a 0x2c
#define addr54c 0x02
#define addr55a 0x2d
#define addr55c 0x02
#define addr56a 0x2e
#define addr56c 0x02

#define iop1 0x01
#define iop2 0x02
#define iop4 0x04
#define cleariop 0x00

#define ON 1
#define OFF 0

/* */
APPENDIX VIII

FILEMAKE SOFTWARE FLOWCHART
Switch on the option number

Option = 3
Does the file exist?

Yes
Edit a file

No
Alert the user that the file does not exist

E

Option = 4
Does the file exist?

Yes
Do a file

No
Alert the user that the file does not exist

F

Option = 5
Quit
Exit

B
Build_file()

- Display the build file menu and prompt for the user's choice

- Move to a position?
  - Yes: Get and transform the new position; Display the position on the screen and save it in the file
  - No

- Move a distance?
  - Yes: Get and transform the distance; Display the distance on the screen and save it in the file
  - No

- Turn the laser on?
  - Yes: Display the laser on to the screen and save it in the file
  - No

- Turn the laser off?
  - Yes: Display the laser off to the screen and save it in the file
  - No

- Cut a circle?
  - Yes: Get the type of circle (saving or cutting the center); Get the radius of the circle
    - Display the type and radius to the screen; Save the type and radius to the file
  - No

- Start a loop?
  - Yes: Get and transform the number of repetitions; Display the loop on the screen and save it in the file
  - No

- End a loop?
  - Yes: Display the end of the loop or the screen and save it in the file
  - No

- End?
  - Yes: Close the file and clear the screen
  - No
Print the file?

Yes → Set a register to flag for 'send file to printer'

No → Clear the register so that the file will not be printed

→ Have the user turn on the printer

→ Clear all counters and place holders

Valid pattern file?

No → Alert the user of the error

Yes → Get the cutting velocity and print it if requested

Action number = Exit?

Yes → Screen display the final position and all counted items

No → Action number = Position move?

Yes → Get the position and find the safest velocity at the traveled distance

No → Print the position move, its parameters, and the status if requested

Action number = Distance move?

Yes → Get the distance and find the safest velocity for that distance

No → Set the new position as the previous position plus the distance

Print, if requested, the final position and all counted items
Action number = Laser on?
  Yes → Increment the laser on counter with respect to loops → Print the laser on and status if requested
  No

Action number = Laser off?
  Yes → Increment the laser off counter with respect to loops → Print the laser off and the status if requested
  No

Action number = Circle?
  Yes → Get the radius and type of circle, increment the circle counter → Print the circle radius, type of circle, and the status if requested
  No

Action number = Start loop?
  Yes → Get and store the loop repetitions, store current position, and increment counter → Print the start of loop and the status if requested
  No

Action number = End of loop?
  Yes → Calculate the final position from the starting and ending position, increment the counter → Print the end of loop and the status if requested
  No
Append()

Open the backup files and clear the action number

Is the original file valid?
No → B
Yes

Copy the cutting velocity from the original to the backup

Action number = End of file?
Yes → I
No

Get and copy the next action number

Action number = Position move?
Yes → Copy the 'position to move to' to the backup
No

Action number = Distance move?
Yes → Copy the 'distance to move' to the backup
No

Action number = Circle?
Yes → Copy the radius and type of circle to the backup
No

Action number = Start loop?
Yes → Copy the number of loops to the backup
No
Switch to reading from the backup and writing to the original

Get the cutting velocity

Correct cutting velocity? No

Get and transform the correct cutting velocity

Yes

Store the cutting velocity

Action number = of file? End Yes

Close the backup file

No

Read in the next action number

Scroll the screen according to the current line

Action number = Position move? Yes

K

No

Action number = Distance move? Yes

K

No

J
Prompt the user for correctness (yes, no, or quit)

Quit?

Yes ➔ N
No

No?

Yes ➔ Run 'build_file()' to insert commands
No

Insert commands?

Yes

Delete this action?

Yes ➔ Get and transform the new parameters
No

Change the parameters?

Yes ➔ Is this action to be saved?
No

Get and transform the new parameters

Yes ➔ Save the action and its parameters
No

Is this action to be saved?

Yes

M
L

Prompt the user for correctness (yes, no, or quit)

Quit?

Yes

N

No

No?

Yes

Insert commands?

Run 'build_file()' to insert commands

No

Delete this action?

Yes

No

Is this action to be saved?

Yes

Save the action

No

M
Transfer_rest_of_file()

Action number = End of File?
  Yes -> B
  No

Get and save the next action number

Action number = Position move?
  Yes -> Save the position to move
  No

Action number = Distance move?
  Yes -> Save the distance to move
  No

Action number = Circle?
  Yes -> Save the radius and the type of circle
  No

Action number = Start loop?
  Yes -> Save the number of repetitions
  No
Do_file()

Clear all counters and place holders

Get the 'all ready' signal from the user

Is the file a valid file?
  Yes
  No Alert the user of the error → B

Display the cutting velocity

Action number = End of file?
  Yes
  No

Scroll the screen according to the current line

Action number = Position move?
  Yes
  No

Action number = Distance move?
  Yes
  No

Get the position and calculate the max velocity

Get distance and calculate the max velocity

Display the move to the position and the current status

Display the move a distance and the current status

Set the position as the position just moved

Set the position as the old position plus the distance moved

B

O

P
Action number = Laser on

Yes: Increment the laser on counter
     Display the action and the current status

No:

Action number = Laser off

Yes: Increment the laser off counter
     Display the action and the current status

No:

Action number = Circle

Yes: Get the radius and the type of circle
     Increment the circle counter
     Display the action and the current status

No:

Action number = Start Loop?

Yes: Get the repetitions and increment the loop counter
     Set up the loop in an array (*)

No:

Action number = End of loop?

Yes: Display extra end of loop
     Increment the end of loop counter

No:

* All loops are put in an array format by set_up_a_loop()

** All arrays are processed by do_a_loop().
Set_up_a_loop()

Set loop conditions

Action number = End of loop?
  Yes: Return to caller
  No:

Get the next action number and store it in the action array

Action number = Position move?
  Yes: Get the position and store it in the parameter array
  No:

Action number = Distance move?
  Yes: Get the distance and store it in the parameter array
  No:

Action number = Circle
  Yes: Get the radius and the type of circle and store it in the parameter array
  No:

Action number = Start loop?
  Yes: Get the loop repetitions and store it in the parameter array
  No: Recursively call 'set_up_a_loop'
Do_a_loop()

Set up the loop number and the counter (# of loops completed)

Does the number of loops left = 0?

Yes → Return to the caller

No → Initialize counter

Decrement the loop counter

Yes → Action number = End of loop?

No → Get the next action number

Scroll the screen according to the current row

Action number = Position move?

Yes → Get the position and calculate the max velocity

No → Move to the position and display the current status

Set the position as the position just moved

Q
Velocity_set()

Are both distances less than 49?
  Yes → Return a velocity of 2 to the caller
  No

Are both distances less than 199?
  Yes → Return a velocity of 4 to the caller
  No

Are both distances less than 699?
  Yes → Return a velocity of 6 to the caller
  No → Return a velocity of 8 to the caller
APPENDIX IX

FILEMAKE SOURCE CODE

/**************************************************************************
 FILEMAKE.C
**************************************************************************/

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <float.h>
#include <windows.h>
#include <globivar.h>
#include <defntion.h>

void cdecl pattern_cut(void);
void cdecl build_file(void);
void cdecl append(void);
void cdecl pattern_check(int);
void cdecl do_file(void);
int cdecl do_a_loop(int);
void cdecl set_up_a_loop(void);
void cdecl transfer_rest_of_file(void);
float transform(char *);
int velocity_set(int, int);

**************************************************************************

GLOBAL PARAMETERS:

* DEFIN: aca acom - Port addresses for info transfer from
  ac  acm - Port addresses for info transfer from
  baca bacb - Port addresses for info transfer from
  bacc baccom - Port addresses for function call from
  bmbca bmbcom - Port addresses for function call from
  iop1 iop2 - Bmb value number for function load
  iop4 clearchp - I/O board definition values for input
  outcbt incbt - Address value sets for each function
  addr13a addr13c - Status of laser beam on/off
  addr15a addr15c
  addr17a addr17c
  addr30a addr38c
  addr31a addr31c
  addr32a addr32c
  addr40a addr40c
  addr41a addr41c
  addr43a addr43c
  addr44a addr44c
  addr52a addr52c
  addr53a addr53c
  addr54a addr54c
  addr55a addr55c
  addr56a addr56c

GLOBL: BWINDOW - Window defined type cast for window pointer name
aprobe - The "a" probe number
bprobe - The "b" probe number

291
void main()
{
    BORDER bord;
    WHERE location;
    int mode, columns, act_page;
    int row, col, high, low, choice;
    int reps, yeah, i;

    plist = wncreate(3, 78, WHITE); // Dimensions of window data area.
    pwin = wncreate(10, 78, CYAN);
    bord.type = BBRD_DDDDD | BRRD_TCT; // Box drawn with single lines, having a top
                                        // centered title.
bord.attr = HAGENTA; /* Border will be magenta on black. */
location.dev = scmode(&mode,&columns,&act_page);
location.page = act_page;
location.corner.col = 1;
location.corner.row = 21;
bord.ttattr = REVERSE; /* Title will be black on white. */
bord.pptitle = " System Status "; /* Text of the title. */
if (NIL == wndsplay(plist,&location,&bord))
return; /* Quit if failure. */
location.corner.row = 1;
bord.pptitle = " Laser Operating System "; /* Text of the title. */
if (NIL == wndsplay(pwin,&location,&bord))
return; /* Quit if failure. */
pattern_cut();
/* Remove the window & restore the screen and cursor. */
wnremove(pwin);
wnremove(plist);
wndstvoy(pwin); /* Clean up data structures. */
wndstvoy(plist); /* Clean up data structures. */

/**************************************************************************
* MODULE: pattern_cut
* PURPOSE: Main menu system for building, analyzing, editing, and test running.
* PARAMETERS:
*  PUBLIC: GLOBL: NULL current
*  PUBLIC: file_name new_file
*  PUBLIC: old_file vel_cut
*  PUBLIC: scan selection
*  PUBLIC: x_travel y_travel
*  PUBLIC: LOCAL: action Laser action variable
*  PUBLIC: numbers Character Buffer for numeric input
*  PUBLIC: true Loop conditional (loop until 5.Exit)
*  PUBLIC: option Menu variable (create, analyze,...)
*  PRIVATE: USERS: main
*  PRIVATE: CALLS: wnscroll wncurmov
*  PRIVATE: wprintf wquery
*  PRIVATE: strncpy strcat
*  PRIVATE: fopen append
*  PRIVATE: fclose fprintf
*  PRIVATE: transform build_file
*  PRIVATE: pattern_check do_file
*  PRIVATE: kbplace kbgetkey
*  PRIVATE: RETURNS: NONE
*  PRIVATE: DATE: 24 June 1990
*  PRIVATE: PROGRAMMER: Ryp Walters
*  PRIVATE: LANGUAGE: Microsoft C Version 5.1
*  PRIVATE: OS: MS-DOS Version 3.3
*  PRIVATE: DATE INITIjUALS MODIFICATION
* 24/90 Rcw Add previous used file as the next file to use
**************************************************************************/

void pattern_cut()
{
int true, action, option;
char numbers[10];

true = 1; /* Defining variables. */
while(true = 1)
{
   wnscrollo(0.1, -1.0, 1.0);
   wncurvov(3, 26);
   wmprintf("What would you like to do?\n"); /* Print
   wncurvov(5, 22);
   wmprintf(" 1. Create a file\n"); /* Print
   wncurvov(6, 22);
   wmprintf(" 2. Analyze / print a file \n"); /* Print
   wncurvov(7, 22);
   wmprintf(" 3. Edit a file\n"); /* Print
   wncurvov(8, 22);
   wmprintf(" 4. Run a file (test the sequence)\n");
   wncurvov(9, 22);
   wmprintf(" 5. Exit to DOS\n");
   wncurvov(12, 26);
   wmprintf("Option: \n");
   wquery\(\text{selection}, 2, &scan\);
   if\(\text{\textbf{\texttt{selection == 'c' || selection == 'C' || selection == '1')}}\)
      option = 1;
   if\(\text{\textbf{\texttt{selection == 'a' || selection == 'A' || selection == '2')}}\)
      option = 2;
   if\(\text{\textbf{\texttt{selection == 'e' || selection == 'E' || selection == '3')}}\)
      option = 3;
   if\(\text{\textbf{\texttt{selection == 'r' || selection == 'R' || selection == '4')}}\)
      option = 4;
   if\(\text{\textbf{\texttt{selection == 'm' || selection == 'M' || selection == '5')}}\)
      option = 5;
   if(option > 0 & & option < 5) /* If it is one of the first four options. */
      if\(\text{\textbf{\texttt{file\_name == '' || option == 1}}\)
      wncurvov(14, 20); /* Move cursor.
      wmprintf("Enter the name of the file: \n"); /* Print
      wquery\(\text{file\_name, 9, &scan}\); /* Find next selection.
      strncpy\(\text{old\_file, file\_name, 9}\); /* Store the backup file.
      strncpy\(\text{new\_file, file\_name, 9}\);
      strcat\(\text{old\_file, ".old"}\);
      strcat\(\text{new\_file, ".new"}\);
   else
      wncurvov(14, 20); /* Move cursor.
      wmprintf("Enter the name of the file: \n"); /* Print
      if\(\text{\textbf{\texttt{new\_file[0] = KBgetkey(&scan)}} \quad \text{\textbf{\texttt{= 13}}}\)
      {
         wncurvov(14, 48); /* Move cursor.
         wmprintf("\n"); /* Print
         wncurvov(14, 48); /* Move cursor
         kbplace\(1,\text{new\_file[0], scan}\); /* Get next request.
         wquery\(\text{file\_name, 9, &scan}\); /* Get next request.
         strncpy\(\text{new\_file, file\_name, 9}\);
         strncpy\(\text{old\_file, file\_name, 9}\);
         strcat\(\text{old\_file, ".old"}\);
         strcat\(\text{new\_file, ".new"}\);
      } else
      {
         strncpy\(\text{old\_file, file\_name, 9}\);
         strncpy\(\text{new\_file, file\_name, 9}\);
         strcat\(\text{old\_file, ".old"}\); /* Store the backup with .old extension.
         strcat\(\text{new\_file, ".new"}\);
      }
}
switch(option) {
  case 1:
    if (!current = fopen(new_file, "r")) { /* If the new file exists. */
        wncursor(0, -1, -1, 0); /* Scroll window */
        wnprint("This file already exists."); /* Print */
        wncursor(7, 5); /* Move cursor. */
        wnprint("(Press ENTER to continue)");
        wquery(selection, 1, &scan); /* Get the next selection. */
      }
    else
      {close(current); /* Close the current file. */
       current = fopen(new_file, "w");
       action = 8; /* If action is a resistor trim. */
       fprintf(current, 
 stubborn
         , action); /* Print */
       wncursor(0, -1, -1, 0); /* Scroll window */
       wncursor(5, 5);
       wnprint("What cutting velocity would you like to set? (0 - 20.625)"); /* Print */
       wquery(numbers, sizeof(numbers), &scan); /* Get the next selection. */
       vel_cut = transform(numbers);
       fprintf(current, "%.3f\n", vel_cut); /* Print */
       build_file(); /* Do module. */
       fprintf(current, "\n\n\n");
       fclose(current); /* Close file. */
      }
    break;
  case 2:
    if (!current = fopen(new_file, "r")) { /* If the file doesn’t exist. */
        wncursor(0, -1, -1, 0);
        wncursor(5, 5);
        wnprint("This file does not exist."); /* Print */
        wncursor(7, 5);
        wnprint("(Press ENTER to continue)");
        wquery(selection, sizeof(selection), &scan);
        fclose(current);
        break;
      }
    wncursor(0, -1, -1, 0); /* Move cursor. */
    wncursor(6, 10);
    wnprint("Do you want the file sent to the printer? [y/n]"); /* Print */
    wquery(selection, 2, &scan);
    if ("selection == 'Y' || selection == 'Y'"); /* Selection is a yes. */
      {wncursor(8, 10);
       wnprint("Turn on the printer and press ENTER."); /* Print */
       wquery(selection, 1, &scan); /* Find next selection. */
       pattern_check(1);
      }
    else
      pattern_check(0); /* Do module. */
    wncursor(0, -1, -1, 0); /* Scroll window. */
    wncursor(5, 5);
    wnprint(" The final position is %6i, %6i, ", x_position, y_position);
    wnprint("n There were %i loops started., loops");
    wnprint("n There were %i loops ended., end_loops");
    wnprint("n The laser was turned on %i times,,on");
    wnprint("n The laser was turned off %i times,,off");
    wnprint("n There were %i circles,,circles");
    wnprint("n\n Press ENTER to continue!*");
    wquery(selection, 1, &scan); /* Get next selection. */
    fclose(current); /* Close the file. */
    break;
  case 3:
    if (!current = fopen(new_file, "r")) { /* If the new file doesn’t exist. */
      wncursor(0, -1, -1, 0);
      wncursor(5, 5);
      wnprint(" The final position is %6i, %6i, ", x_position, y_position);
      wnprint("n There were %i loops started., loops");
      wnprint("n There were %i loops ended., end_loops");
      wnprint("n The laser was turned on %i times.,on");
      wnprint("n The laser was turned off %i times.,off");
      wnprint("n There were %i circles, circles");
      wnprint("n\n (Press ENTER to continue)!");
      wquery(selection, 1, &scan); /* Get next selection. */
      fclose(current); /* Close the file. */
      break;
    }
  }
void build_file() { /* Scroll window. */ wncurmmov(5,5); /* Print */ wncurmmov(7,5); /* Press ENTER to continue */ wnquery(selection, 1, &scan); /* Get next selection. */ fclose(current); break;

wnscroll(0,-1,-1,0); /* Move cursor */ append(); /* Close file. */ fclose(current); break;

case 4:
if ((current = fopen(new_file,"r")) == NULL) /* If new file doesn't exist */ {
  wncurmmov(0,-1,-1,0); /* Scroll window. */ wncurmmov(5,5); /* Print */ wncurmmov(7,5); /* Press ENTER to continue */ wnquery(selection, sizeof(selection), &scan); fclose(current); break;
}

wnscroll(0,-1,-1,0); /* Scroll window. */ do_file(); /* Do module. */ fclose(current); break;

case 5:
return;
}
}
int times, rad, type; /* Define variables. */
char numbers[10];

'selection = 'a';

while (*selection != 'e' && *selection != 'E' && *selection != '9')
    /* While selection is not an exit. */
{
    wnscroll(0,-1,-1,0);
    wnprintf("\n"); /* Move cursor */
    wnprintf("What would you like to do?"); /* Print */
    wnprintf("1. Move to a position");
    wnprintf("2. Distance move *");
    wnprintf("3. Laser On");
    wnprintf("4. Laser Off");
    wnprintf("5. Circle cut");
    wnprintf("6. Start a loop");
    wnprintf("7. Repeat a loop");
    wnprintf("9. Exit");
    wnprintf("Option: ");
}

wnquery(*selection, 2, &scan);

if (*selection == 'm' || *selection == 'M' || *selection == 'l')
    /* If it is the second selection. */
{
    fprintf(current, "\n1\n"); /* Print */
    wcumov(13,20);
    wnprintf("To what position would you like to move?"); /* Print */
    wcumov(15,20);
    wnprintf("X direction:");
    wnquery(numbers, 10, &scan);
    x_displace = (int)transform(numbers); /* Calculate X displacement. */
    wcumov(16,20);
    wnprintf("Y direction:");
    wnquery(numbers, 10, &scan);
    y_displace = (int)transform(numbers);
    fprintf(current, "\n", x_displace);
    fprintf(current, "\n", y_displace); /* Print */
    wselect(plist);
    wnscroll(1,-1,-1,0); /* Scroll one line. */
    wcumov(2,40);
    wnprintf("Position %5i, %5i", x_displace, y_displace);
}

if (*selection == 'd' || *selection == 'D' || *selection == '2')
    /* If it is the second selection. */
{
    fprintf(current, "\n2\n"); /* Print */
    wcumov(13,20);
    /* Print */
    wnprintf("What distance would you like to move?");
    wcumov(15,20);
    wnprintf("X direction:"); /* Print */
    wnquery(numbers, 10, &scan);
    x_displace = (int)transform(numbers); /* New X distance. */
    wcumov(16,20);
    wnprintf("Y direction:");
    wnquery(numbers, 10, &scan); /* Get next selection. */
    y_displace = (int)transform(numbers);
    fprintf(current, "\n", x_displace); /* Print */
    fprintf(current, "\n", y_displace);
    wselect(plist); /* Lower window. */
    wnscroll(1,-1,-1,0);
    wcumov(2,40); /* Select lower window. */
    wnprintf("Distance %5i, %5i", x_displace, y_displace); /* Print */
}
if(*selection == 'O' || *selection == '0' || *selection == '3')
{
    fprintf(current,"\n3\n"); /* Print */
    wnselect(plist); /* Scroll one line. */
    wncurmov(1,1,1,0); /* Scroll one line. */
    wncurmov(2,0); /* Laser on */
    wprintf(" "); /* Print */
}

if(*selection == 'F' || *selection == 'F' || *selection == '4')
{
    fprintf(current,"\n4\n"); /* Print */
    wnselect(plist); /* Select lower window. */
    wncurmov(1,-1,1,0); /* Move cursor. */
    wncurmov(2,0); /* Laser off */
    wprintf(" "); /* Print */
}

if(*selection == 'C' || *selection == 'C' || *selection == '5')
{
    fprintf(current,"\n5\n"); /* Move cursor. */
    wncurmov(13,12); /* Move cursor. */
    wprintf("\n?\n") /* Do you want to save the center? (y/n) */; /* Print */
    wquery(numbers, 10, &scan); /* Get the next selection. */
    rad = (int)transform(numbers); /* Transform the radius. */
    fprintf(current, "%i\n", rad); /* Print */
    wncurmov(13,12); /* Move cursor. */
    wncurmov(15,12); /* Move cursor. */
    wprintf("\n?\n") /* What radius would you like the circle? (in .01mm) */; /* Print */
    wquery(numbers, 10, &scan); /* Get the next selection. */
    rad = (int)transform(numbers); /* Transform the radius. */
    fprintf(current, "%i\n", rad); /* Print */
    wncurmov(1,1,1,0); /* Move cursor. */
    wncurmov(2,40); /* Move cursor. */
    printf("Circle w/o cntlr C:\r\n", rad); /* Print */
    else
    wprintf("Circle w/ center C:\r\n", rad); /* Print */
}

if(*selection == 'S' || *selection == 'S' || *selection == '6')
{
    fprintf(current,"\n6\n"); /* Print */
    wncurmov(13,20); /* Move cursor. */
    wncurmov(13,20); /* Move cursor. */
    wprintf("How many times would you like to loop? "); /* Print */
    wquery(numbers, 10, &scan); /* Get the next selection. */
    times = (int)transform(numbers); /* Calculate the number of loops. */
    fprintf(current, "%i\n", times); /* Print */
    wncurmov(1,1,1,0); /* Scroll one line. */
    wncurmov(2,40); /* Move cursor. */
    wprintf("Start loop %i times "); /* Print */
}

if(*selection == 'R' || *selection == 'R' || *selection == '7')
{
    fprintf(current,"\n7\n"); /* Print */
    wncurmov(1,1,1,0); /* Move cursor. */
    wncurmov(1,1,1,0); /* Move cursor. */
    wncurmov(2,40); /* Repeat a loop */
    wprintf(" "); /* Print */
}

wnselect(plist); /* Select lower window. */
wnselect(plist); /* Select lower window. */
wncurmov(0,2); /* Move cursor. */
wprintf("Current Position: %i, %i ",xpos,ypos); /* Print */
wnselect(pwin); /* Select lower window. */

*selection = ' '; /* Print */
wnselect(plist);
wmncroll(3,-1,-1,0);  /* Scroll three lines. */
wncurmov(0,2);
wnprintf("Current Position: \%6ld, \%6ld ", xpos,ypos);  /* Print */
wnselect(pwin);  /* Select upper window */
}

******************************************************************************

* MODULE:  append
* PURPOSE: Edit the file
* PARAMETERS:  GLOBAL:
new_file
old_file
old
radius
X_travel
y_travel
vel_cut
speed
choice
- Laser action variable
choice
- User choice character variable
choice
- User choice character variable
times
- Character buffer for numeric input
row
- Number of times to process a loop
row
- Row number of the screen

USERS:
cut_a_pattern

CALLS:
ofopen
fopen

fscanf

fprintf
wnprintf
wncurmov
wnquery
fclose
transform
build_file

RETURNS: NONE

DATE: 24 June 1990
PROGRAMMER: Ryp Walters
LANGUAGE: Microsoft C Version 5.1
OS: MS-DOS Version 3.3

******************************************************************************

void append()
{
int row, action, times;
char choice[2], choice[2], numbers[10];

old = fopen(old_file, "w");  /* Open extension file. */
action = 0;

fscanf(current, "%li", &action);  /* Scan file for selection. */
fprintf(old, "\n\n\n", action);  /* Print */

if(action != 0)
{
wmncroll(0,-1,-1,0);  /* Move cursor. */
wncurmov(10,10);  /* Move cursor. */
wnquery("There is an error in this program!!!");  /* Print */
cfclose(old);
return;
}

fscanf(current, "%f", &vel_cut);
fprintf(old, "\f\n\n", vel_cut);  /* Print */

while(action != 9)  /* As long as it is not an exit. */
{
fprintf(old, "\n\n\n", action);  /* Print */
}
if(action == 1) /* If it is a move position. */
{
    fscanf(current, "%i", &x_travel);
    fscanf(current, "%i", &y_travel);
    /* Scan file for the x travel distance. */
    printf(old, "%i", x_travel); /* Print */
    printf(old, "%i", y_travel); /* Print */
}

if(action == 2) /* If it is a move distance. */
{
    fscanf(current, "%i", &x_travel);
    fscanf(current, "%i", &y_travel);
    /* Scan file for the y travel distance. */
    printf(old, "%i", x_travel); /* Print */
    printf(old, "%i", y_travel); /* Print */
}

if(action == 5) /* If it is a circle set */
{
    fscanf(current, "%i", &radius); /* Scan file for the radius. */
    fscanf(current, "%i", &speed);
    printf(old, "%i", radius); /* Print */
    printf(old, "%i", speed);
}

if(action == 6) /* If it is a loop set */
{
    fscanf(current, "%i", &times);
    printf(old, "%i", times); /* Print */
}

fclose(old); /* Close file. */
fclose(current); /* Opening back up file. */

row = 3;
scanf(old, "%i", &action); /* Scan file for the action. */
scanf(old, "%f", &vel_cut); /* Scroll window. */
wcscat(row, l); /* Is this correct? (y/n)! */
wmprintf("Cutting Velocity %f", vel_cut); /* Print */
wmprintf("Is this correct? (y/n)!"; /* Find the next selection. */
wmcurmov(row, 40); /* Print */

if(('choice' == 'n') || ('choice' == 'N'))
{
    wcscat(row, l); /* What cutting velocity do you want? (0 - 20.625)! */
    wmprintf("What cutting velocity do you want? (0 - 20.625)!"; /* Print */
    wquery(numbers, 10, &scan); /* Find the cutting velocity */
    wcsxform(numbers); /* Find the cutting velocity */
}
printf(current, "%s
action); /* Print. */
printf(current, "%s
vel_cut); /* Print. */

while(action != 9) /* As long as it is not an exit. */
{
    fscanf(old, "%i", &action);
    if (++row > 13) /* If the number of rows is more than 13 */
    {
        wcsnscroll(5, -1, -1, 0); /* Scroll 5 lines. */
        row = row - 5;
    }
    wcscat(row, l);
}

if(action == 1) /* If it is a move */
{
    'choice' = 'n';
    fscanf(old, "%i", &x_travel);
    fscanf(old, "%i", &y_travel); /* Scan backup for the y travel distance. */
    wcscat(row, l); /* Position */
    wcsxform(row, l); /* Print */
    wcscat(row, l); /* Is this correct? (y/n/q)! */
    wquery(choice, 2, &scan); /* Find the next selection. */
    wcscat(row, l); /* Find the next selection. */
wnprintf("\n\f\n");
if(*choice == 'q' || *choice == 'Q') /* If choice is a quit. */
{
    fprintf(current,"\n\f\n",action); /* Print */
    fprintf(current, "%d", x_travel);
    fprintf(current, "%d", y_travel);
    transfer_rest_of_file(); /* Do module. */
    return;
}
if(*choice == 'n' || *choice == 'N')
{
    wncurmov(row,35);
    wnprintf("Do you want to insert commands? (y/n)\n"); /* Print */
    wquery(choice1,2,&scan);
    wncurmov(row,35);
    wnprintf("\n"); /* Print */
    if(choice1 == 'y' || *choice1 == 'Y')
    {
        build_file(); /* Do module. */
        wmscroll(0,-1,-1,0); /* Scroll window. */
        row = 1;
        wncurmov(+row,1); /* Move cursor. */
        wnprintf("Position \%5s, \%5s, x\_travel, y\_travel\n"); /* Print */
    }
    wncurmov(row,35);
    wnprintf("Delete this position move? (y/n)\n"); /* Print */
    wquery(choice1,2,&scan); /* Get the next selection. */
    wncurmov(row,35);
    wnprintf("\n"); /* Print */
    if(choice1 != 'y' && *choice1 != 'Y') /* If choice is a yes. */
    {
        wncurmov(row,35);
        wnprintf("Keep the position the same? (y/n)\n"); /* Print */
        wquery(choice1,2,&scan); /* Get next selection. */
        wncurmov(row,35);
        wnprintf("\n"); /* Print */
    }
    if(*choice == 'n' || *choice == 'N')
    {
        wncurmov(+row,1);
        wnprintf("What position would you like to move to? \n");
        wncurmov(+row,1);
        wnprintf("\nX direction\n"); /* Print */
        wquery(numbers10,&scan);
        x_travel = (int)transform(numbers);
        wncurmov(+row,1);
        wnprintf("\nY direction\n"); /* Print */
        wquery(numbers10,&scan);
        y_travel = (int)transform(numbers);
    }
}
if(choice1 == 'y' && *choice1 == 'Y') /* If choice is a yes. */
{
    fprintf(current,"\n\f\n",action);
    fprintf(current, "%d", x_travel);
    fprintf(current, "%d", y_travel); /* Print */
}

if(action == 2) /* If it is a move distance. */
{ 
    *choice = 'n';
    fscan(oid, "%*d", &x_travel); /* Scan backup file for the x travel distance. */
    fscan(oid, "%*d", &y_travel);
    wnprintf("Distance \%5s, \%5s, x\_travel, y\_travel\n"); /* Print */
    wnprintf("Is this correct? (y/n/q)\n"); /* Print */
    wquery(choice2,&scan); /* Find the next selection. */
    wncurmov(row,40); /* Print */
    wnprintf("\n"); /* Print */
    if(*choice == 'q' || *choice == 'Q')
    {
        fprintf(current,"\n\f\n",action);
        fprintf(current, "%d", x_travel); /* Print */
        fprintf(current, "%d", y_travel);
        transfer_rest_of_file();
    }
}
return;
}
if(*choice == 'n' || *choice == 'N')  /* Choice is a no. */
{
    wncurmov(row,35);  /* Move cursor */
    wnpprintf("Do you want to insert commands? (y/n)"); /* Print */
    wquery(choice1,2,&scan);
    wncurmov(row,35);
    wnpprintf(' '); /* Print */
    if(*choice1 == 'y' || *choice1 == 'Y')
    {
        build_file();  /* Do module. */
        wncscroll(0,-1,-1,0);
        row = 1;  /* Define the row. */
        wncurmov(;++row,1);
        wnpprintf(" Distance %5i, %5i, x_travel, y_travel\n"); /* Print */
        wncurmov(row,35);  /* Move cursor. */
        wnpprintf("Delete this distance move? (y/n)\n"); /* Print */
        wquery(choice1,2,&scan);
        wncurmov(row,35);
        wnpprintf(' '); /* Print */
    }
    if(*choice == 'n' || *choice == 'N')
    {
        wncurmov(;++row,1);  /* Move cursor. */
        wnpprintf(" What distance would you like to move? \n");
        wncurmov(;++row,1);
        wnpprintf(" X direction:\n"); /* Print */
        wquery(numbers,10,&scan);
        x_travel = (int)transform(numbers);  /* Find the x travel distance. */
        wncurmov(;++row,1);
        wnpprintf(" Y direction:\n"); /* Print */
        wquery(numbers,10,&scan);  /* Get the next selection. */
        y_travel = (int)transform(numbers);
    }
    if(*choice1 == 'y' && *choice1 == 'Y')  /* If the choice is a yes. */
    {
        fprintf(current,\"%s\n\n",action);
        fprintf(current,\"%s\n", x_travel); /* Print */
        fprintf(current,\"%s\n", y_travel);
    }
}
if(action == 3)  /* If it is a velocity set */
{
    *choice1 = 'n';  /* Laser on Is this correct? (y/n/q)? */
    wnpprintf(" Laser on\n");/* Print */
    wquery(choice,2,&scan);  /* Get the next selection. */
    wncurmov(row,40);
    wnpprintf(' '); /* Print */
    if(*choice == 'q' || *choice == 'Q')  /* Choice is a quit. */
    {
        fprintf(current,\"%s\n\n",action);
        transfer_rest_of_file();
        return;
    }
    if(*choice == 'n' || *choice == 'N')
    {
        wncurmov(row,35);
        wnpprintf(" Do you want to insert commands? (y/n)\n"); /* Print */
        wquery(choice1,2,&scan);  /* Get the next selection. */
        wncurmov(row,35);
        wnpprintf(' '); /* Print */
        if(*choice1 == 'y' || *choice1 == 'Y')
        {
            build_file();  /* Do module. */
```c
wncscroll(0,-1,-1,0);
row = 1;

wncurmov( +row, 1); /* Define row.

wprintf(" Laser on "); /* Print */

wncurmov(row, 35);
wprintf("Delete this laser on? (y/n)"); /* Print */
wquery(choice1,2,&scan); /* Find the next selection.

wncurmov(row,35);
wprintf(" "); /* Print */
if(choice1 != 'Y' || choice1 != 'Y')
  fprintf(current,"\n%li\n",action); /* Print */
}

if(action == 4) /* If it is a laser off. */
{
  choice = 'n';
  wprintf(" Laser off Is this correct? (y/n)"); /* Print */
  wncurmov(row,40);
wprintf(" "); /* Print */
  if(choice == 'q' || choice == 'Q') /* If the choice is a quit */
  {
      fprintf(current,"\n%li\n",action); /* Print */
      transfer_rest_of_file(); /* Do module. */
      return;
  }
  if(choice == 'n' || choice == 'N')
  {
      wncurmov(row, 35);
wprintf("Do you want to insert commands? (y/n)"); /* Print */
      wncurmov(choice1,2,&scan); /* Move cursor.
      wncurmov(row,35);
wprintf(" "); /* Print */
      if(choice1 != 'y' || choice1 != 'Y')
      {
    build_file(); /* Do module. */
    wncscroll(0,-1,-1,0);
    row = 1;
    wncurmov( +row,1); /* Do module. */
    wprintf(" Laser off "); /* Print */
    wncurmov(row,35);
wprintf("Delete this laser off? (y/n)"); /* Print */
      wncurmov(choice1,2,&scan); /* Get the next selection.
      wncurmov(row,35);
wprintf(" "); /* Print */
      if(choice1 != 'y' && choice1 != 'Y') /* Choice is no. */
      fprintf(current,"\n%li\n",action); /* Print */
    }
  }

if(action == 5) /* If it is a circle set */
{
  choice1 = 'n';
  fscanf(old, "%d", &radius);
  fscanf(old, "%d", &speed);
  if(speed == 1)
    wprintf(" Circle w/o cntr %d", radius); /* Print */
  else
    wprintf(" Circle w/ center%4d", radius);
  wprintf(" Is this correct? (y/n)"); /* Print */
  wncurmov(row,40);
wprintf(" "); /* Print */
  if(choice == 'q' || choice == 'Q') /* Choice is a quit. */
  {
      fprintf(current,"\n%li\n",action); /* Print */
      fprintf(current, "%d", radius);
      fprintf(current, "%d", speed); /* Do the module. */
      return;
    }
  if(choice == 'n' || choice == 'N')
```
{ wncurmov(row,35); /* Do the module. */
  wnpref("Do you want to insert commands? (y/n)"); /* Print */
  wquery(choice1,2,&scan); /* Get the next selection. */
  wncurmov(row,35);
  wnpref(" "); /* Print */
  if(*choice1 == 'y' || *choice1 == 'Y') /* Choice is a yes. */
  {
    build_file();
    wnscroll(0,-1,-1,0); /* Scroll window. */
    row = 1;
    wncurmov(row,1);
    if(speed == 1)
      wnpref(" Circle w/o cntr %41", radius);
    else
      wnpref(" Circle w/ center%41", radius); /* Print */
  }
  wncurmov(row,35);
  wnpref(" Delete this circle? (y/n)"); /* Print */
  wquery(choice1,2,&scan); /* Get the next selection. */
  wncurmov(row,35);
  wnpref(""); /* Print */
  if(*choice1 == 'y' && *choice1 == 'Y')
  {
    wncurmov(row,35);
    wnpref(" Keep the same circle and radius? (y/n)r, radius"); /* Print */
    wquery(numbers,10,&scan);
    radius = (int)rttransform(numbers); /* Calculate the radius. */
    wncurmov(row,1);
    wnpref(" Do you want to save the center? (y/n)"); /* Print */
    wquery(numbers,2,&scan);
    if(*numbers == 'n' || *numbers == 'N')
      speed = 1;
    else
      speed = 0; /* If it is a circle with a center. */
  }
  if(*choice == 'n' || *choice == 'N') /* Choice is a no. */
  {
    fprintf(current,"\%11n",action);
    fprintf(current,"\%11n",radius); /* Print */
    fprintf(current,"\%11n",speed);
  }
  else
  {
    if(action == 6) /* If it is a loop */
    {
      *choice1 = 'n';
      fscanf(old, "%1", &times);
      wnpref(" Start loop \%21 times", times); /* Print */
      wnpref(" Is this correct? (y/n/q)"); /* Print */
      wquery(choice1,2,&scan); /* Get the next selection. */
      wncurmov(row,40);
      wnpref(" "); /* Print */
      if(*choice == 'q' || *choice == 'Q') /* If it is a quit. */
      {
        fprintf(current,"\%11n",action);
        fprintf(current,"\%11n",times); /* Print */
        transfer_rest_of_file(); /* Do the module. */
        return;
      }
      if(*choice == 'n' || *choice == 'N') /* Choice is a no. */
      {
        wncurmov(row,35);
        wnpref(" Do you want to insert commands? (y/n)"); /* Print */
        wquery(choice1,2,&scan); /* Get the next selection. */
        wncurmov(row,35);
        wnpref(" "); /* Print */
    }}}}
if(*choice == 'y' || *choice == 'Y')  /* Choice is a yes. */
{
    build_file();
    wncursor(0,-1,-1,0);
    row = 1;
    wncursor(+row,1);
    wprintf(" Start loop  %2d times", times);  /* Print */
}

wncursor(row,35);
wprintf("Delete this start of a loop? (y/n)*");  /* Print */
wquery(choice,2,&scan);  /* Get the next selection. */

wncursor(row,35);
wprintf(" ");  /* Print */

if(*choice != 'y' && *choice != 'Y')
{
    wncursor(row,35);
wprintf(" Keep the %2d loop repetitions? (y/n)*, times);  /* Print */
wquery(choice,2,&scan);
    wncursor(row,35);  /* Move cursor. */
wprintf(
    
    if(*choice == 'n' || *choice == 'N')
    {
        wncursor(+row,1);
        wprintf(" How many times do you want? ");  /* Print */
wquery(numbers,10,&scan);  /* Get the next selection. */
        times = (int)transform(numbers);  /* Calculate the number of loops. */
    }
}

if(*choice != 'y' && *choice != 'Y')
{
    fprintf(current,"\n\n!=n",action);  /* Print */
    fprintf(current,"\n\n!=n",times);
}

if(action == 7)  /* If it is a repeat loop */
{
    *choice = 'n';
    wncursor(choice,2,&scan);  /* Get the next selection. */
    wncursor(row,40);
wprintf(" ");  /* Print */

if(*choice == 'q' || *choice == 'Q')  /* Choice is a quit. */
{
    fprintf(current,"\n\n!=n",action);  /* Print */
    transfer_rest_of_file();  /* Do module. */
    return;
}

if(*choice == 'n' || *choice == 'N')  /* Choice is a no. */
{
    wncursor(row,35);
wprintf("Do you want to insert commands? (y/n) *");  /* Print */
wquery(choice,2,&scan);
    wncursor(row,35);
wprintf(" ");  /* Print */

if(*choice == 'y' || *choice == 'Y')
    {
        build_file();  /* Do module. */
        wncursor(0,-1,-1,0);  /* Scroll window. */
        row = 1;
        wncursor(+row,1);
        wprintf(" End loop ");  /* Print */
    }

wncursor(row,35);  /* Move cursor. */
wprintf(" Delete this loop repetition? (y/n)*");  /* Print */
wquery(choice,2,&scan);
    wncursor(row,35);
wprintf(" ");  /* Print */

if(*choice != 'y' && *choice != 'Y')  /* If choice is a no. */
    {
        fprintf(current,"\n\n!=n",action);  /* Print */
    }
}

if(action == 9)  /* If it is an exit */
{
void transfer_rest_of_file()
{
    int action, times;

    while(action != 9) /* As long as it isn't an exit. */
    {
        scanf(old, "%d", &action); /* Print */
        fprintf(current, "\n\n\n\n", action);
        if(action == 1) /* If it is a move */
        {
            scanf(old, "%d", &x_travel); /* Scan backup for the x travel distance. */
            scanf(old, "%d", &y_travel);
            fprintf(current, "%d\n", x_travel);
            fprintf(current, "%d\n", y_travel);
        }
        if(action == 2) /* If it is a move */
    }
}
{  
  fscanf(old, "%i", &x_travel);  /* Scan file for the x travel distance. */  
  fscanf(old, "%i", &y_travel);  
  fprintf(current, "\n\n", x_travel);  /* Print */  
  fprintf(current, "\n\n", y_travel);  
}

if (action == 5)  /* If it is a circle set */  
{  
  fscanf(old, "%i", &radius);  /* Scan file for the radius. */  
  fscanf(old, "%i", &speed);  
  fprintf(current, "\n\n", radius);  /* Print */  
  fprintf(current, "\n\n", speed);  
}

if (action == 6)  /* If it is a loop */  
{  
  fscanf(old, "%i", &times);  /* Scan file for the number of loops. */  
  fprintf(current, "\n\n", times);  /* Print */  
}

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

*  MODULE: pattern_check
*  PURPOSE: Analyze and optionally print the file
*  PARAMETERS:  
*    GLOBL: current print  
*            prn_file radius  
*            scan selection  
*            x_position y_position  
*            x_travel y_travel  
*            speed laser_status  
*            vel vel_cut  
*    PARAM: print_it - Option to send the file to the printer
*    LOCAL: action - Laser action variable
*            count - Array counter
*            i - Loop counter
*            k - Loop counter
*            loop_times - Array for the # of times a loop is done
*            looping - Array counter for loop to keep track of
*            nested_x - Array for the x posit. during the loops
*            times - Number of times to process a loop
*            row - Row number of the screen
*    USERS: cut_a_pattern
*    CALLS: fopen printf
*            fscanf wncscroll
*            wncurmov wnpprintf
*            wnquery fclose
*  RETURNS: NONE
*  DATE: 24 June 1990
*  PROGRAMMER: Ryp Walters
*  LANGUAGE: Microsoft C Version 5.1
*  OS: MS-DOS Version 3.3
*  --------------------------------------------------------------------------------
***************************************************************************/

void pattern_check(int print_it)
{
  int i, k, row, count;
int action, times[5];
int nested_x[5], nested_y[5], loop_times[5], looping;

laser_status = OFF;   /* Defining initial values. */
count = 0;
for(k=0;k<5;k++)
{
    times[k] = 0;
    nested_x[k] = 0;
    nested_y[k] = 0;
    loop_times[k] = 0;
}

x_position = 0;
y_position = 0;
circles = 0;
loops = 0;
end_loops = 0;
on = 0;
off = 0;
looping = 0;
row = 1;

if(print_it == 1)
{
    fprintf(stderr,* "\n\n\nAction Parameters Position Velocity\n\n\n*/

action = 0;

fscanf(current, "%ld", &action);   /* Scan file for the action. */
if(action != 8)   /* If it isn't a resistor trim */
{
    wncscroll(0,-1,-1,0);   /* Scroll window. */
    wncursor(10,10);

    wprintf("There is an error in this program!!!");   /* Print */
    wquery(selection,2,2&scan);
    fclose(current);
    return;
}   /* Return */

fscanf(current, "%s", &vel_cut);
if(print_it == 1)
    fprintf(stderr,* "\n\nCutting Vel. %7.3f 0, 0 0.000\n\nAction, vel_cut\n\n\n*/

while(action != 9)   /* While action isn't exit. */
{
    fscanf(current, "%ld", &action);   /* Scan file for the action. */

    if(action == 1)   /* If it is a move */
    {
        fscanf(current, "%ld", &x_travel);   /* Scan file for the x travel distance */
        fscanf(current, "%ld", &y_travel);
        if(laser_status == ON)   /* If laser is on */
            vel = vel_cut;
        else
            vel = vel = velocity_set(x_travel-x_position, y_travel-y_position);

    }   /* Find the safe velocity */

    if(print_it == 1)
        fprintf(stderr,* "\n\nPosition %5d,%5d  %5d,%5d  %7.3f\n\n\nAction, x_travel, y_travel, x_position, y_position, vel;\n\n\n\n*/

    x_position = x_travel;
    /* Calculate the x position. */
    y_position = y_travel;
}

if(action == 2)   /* If it is a move */
{
    fscanf(current, "%ld", &x_travel);   /* Scan file for the x travel distance. */
    fscanf(current, "%ld", &y_travel);
    if(laser_status == ON)   /* If laser is on */
        vel = vel_cut;
    else
        vel = vel = velocity_set(x_travel, y_travel);

    if(print_it == 1)
        fprintf(stderr,* "\n\nDistance %5d,%5d  %5d,%5d  %7.3f\n\n\nAction, x_travel, y_travel, x_position, y_position, vel;\n\n\n\n*/

    x_position = x_travel + x_position;
    /* Calculate the x position. */
}
y_position = y_travel + y_position;
}

if(action == 3) /* If it is a laser on */
{
    k = 1;
    if(looping == 0)
    {
        for(i=1; i <= looping + 1; i++) /* Increment the number of loops. */
        {
            k = k * loop_times[i]; /* keep track of the number of loops. */
        }
    }
    on = on + k; /* Keep track of the number of times the laser has been on. */
    if(print_it == 1)
        fprintf(stderr, "\r%i Laser on \n", action, x_position, y_position, vel); /* Print.
    laser_status = ON;
}

if(action == 4) /* If it is a laser off */
{
    k = 1; /* Number of times laser has been on. */
    if(looping == 0)
    {
        for(i=1; i <= looping + 1; i++) /* Keeping track of the number of loops. */
        {
            k = k * loop_times[i];
        }
    }
    off = off + k; /* Keeping track of the number of times the laser has been off. */
    if(print_it == 1)
        fprintf(stderr, "\r%i Laser off \n", action, x_position, y_position, vel); /* Turning laser off. */
    laser_status = OFF;
}

if(action == 5) /* If it is a circle set */
{
    fscanf(current, "\%i", &radius); /* Scan file for the radius */
    fscanf(current, "\%i", &speed);
    k = 1; /* Number of circles */
    if(looping == 0)
    {
        for(i=1; i <= looping + 1; i++) /* Keeping track of the number of times */
        {
            circle has been done. */
            k = k * loop_times[i];
        }
    }
}

circles = circles + k; /* Number of circles. */
if(print_it == 1)
{
    if(speed == 1) /* If it is a circle with a center. */
        fprintf(stderr, "\r%i Circle w/o cntr\n", action, radius, x_position, y_position, vel);
    else
        fprintf(stderr, "\r%i Circle w/center\n", action, radius, x_position, y_position, vel);
}

if(action == 6) /* If it is a start loop */
{
    fscanf(current, "\%i", &times[count]); /* Scan file for the number of loops. */
    nested_x[count] = x_position;
    nested_y[count] = y_position;
    looping++; /* Increment */
    loop_times[looping] = times[count]; /* Calculate the number of loops. */
}
loops++;
if(print_it == 1)
    fprintf(stderr, "\r\n\l1 Start Loop \%3i times \%5i,\%5i \$7.3f\n", action, times[count], x_position, y_position, vel);
    count++;
    /* Increment */
}
if(action == 0) /* If it is an end loop */
    if(count == 0) /* If is isn't the first loop */
    count--;  k n
    x_position = nested_x[count] + times[count] * (x_position - nested_x[count]);
    /* Calculate the x position. */
    y_position = nested_y[count] + times[count] * (y_position - nested_y[count]);
    end_loops++;
    if(print_it == 1) /* If it is to be sent to the printer. */
    {
        fprintf(stderr, "\r- Loop repetition \n");
        fprintf(stderr, "\r\n\l1 End loop \%5i,\%5i \$7.3f\n", action, x_position, y_position, vel);
    }
    looping--; /* Decrement the number of loops */
else
    {  k n
        end_loops++;
        /* Keep track of the number of loops completed */
        if(print_it == 1)
            fprintf(stderr, "\r\n\l1 EXTRA END LOOP \%5i,\%5i \$7.3f\n", action, x_position, y_position, vel);
    }
}
if(print_it == 1)
    fprintf(stderr, "\r\n\l1 End of File \%5i,\%5i \$7.3f\n", action, x_position, y_position, vel);
    /* Print */
if(print_it == 1) /* If it is to be sent to the printer. */
    {
        fprintf(stderr, "\r\n\nThe final position is \%6i, \%6i, X_position, Y_position);";
        fprintf(stderr, "\r\nThere were \%i loops started.*,loops);";
        fprintf(stderr, "\r\nThere were \%i loops ended.*,end_loops);";
        fprintf(stderr, "\r\nThe laser was turned on \%i times.*,on);";
        fprintf(stderr, "\r\nThe laser was turned off \%i times.*,off);";
        fprintf(stderr, "\r\nThere were \%i circles.*,circles);";
    };
    fprintf(stderr, ");";
    fprintf(stderr, ");";
    fprintf(stderr, ");";
    fprintf(stderr, ");";
    fprintf(stderr, ");";
    fprintf(stderr, ");";
    fprintf(stderr, ");";
}

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

311
void do_file()
{
int i, j, bits, times, row, col, action;

pattern_check(0);

if((loops := end_loops) || (on := off) || (x_position != 0) || (y_position != 0))
{
    wnscroll(0,-1,-1,0);  /* Scroll window */
    wncurmov(7,10);
    wprintf("This file is not operationally correct!\n");  /* Print */
    wncurmov(9,10);
    wprintf("Do you want to run the program anyway? (y/n)\n");  /* Print */
    wquery(selection,2, &scan);
}
if(*selection == 'n' || *selection == 'N')  /* If selection is no. */
    return;

fclose(current);

current = fopen(new_file,"r");  /* Open new file. */

laser_status = OFF;  /* Defining variables. */
loops = 0;
end_loops = 0;
x_position = 0;
y_position = 0;
circles = 0;
on = 0;
off = 0;

wnscroll(0,-1,-1,0);
wncurrmov(7,10);  /* Move cursor.  */
wprintf("Press ENTER when you are ready.  ");  /* Print */
wnquery(selection,2,&scan);
action = 0;

wnscroll(0,-1,-1,0);  /* Scroll window  */
wncurrmov(1,1);
wprintf("#  Action  Parameters  Position  Velocity");  /* Print*/
 fscanf(current, "%li", &action);
if(action != 8)  
{
  wnscroll(0,-1,-1,0);  /* Scroll window  */
  wncurrmov(10,10);
  wprintf("There is an error in this program!!!");  /* Print */
  wnquery(selection,2,&scan);
  fclose(current);
  return;
}

fscanf(current, "%f", &vel_cut);
row = 2;
wncurrmov(++row,1);
wprintf("%li  Cutting Vel.  $7.3f  0, 0  0.000\n",action, vel_cut);

while(action != 9)  /* If it is an exit */
{
  fscanf(current, "%li", &action);
  if(*++row > 17)
  {
    wnscroll(5,-1,-1,0);  /* Scroll five lines. */
    row = row - 5;
  }
  wncurrmov(row,1);

  if(action == 1)  /* If it is a move */
  {
    fscanf(current,"%li", &x_travel);  /* Scan file. 
    fscanf(current,"%li", &y_travel);
    if(laser_status == ON)
      vel = vel_cut;
    else
      vel = velocity_set(x_travel - xpos + x_displace,
                         y_travel - ypos + y_displace);
    wprintf("%li  Position  %5i,%5i  %5i,%5i  $7.3f\n", 
            action, x_travel, y_travel, x_position, y_position, vel);
    x_position = x_travel;
    y_position = y_travel;
    wnquery(selection,2,&scan);
  }

  if(action == 2)  /* If it is a move */
  {
    fscanf(current,"%li", &x_travel);
    fscanf(current,"%li", &y_travel);
    if(laser_status == ON)
      vel = vel_cut;
    else
      vel = velocity_set(x_travel, y_travel);
    wprintf("%li  Distance  %5i,%5i  $7.3f\n", 
            action, x_travel, y_travel, x_position, y_position, vel);
    x_position = x_travel + x_position;  /* New x position. */
    y_position = y_travel + y_position;
    wnquery(selection,2,&scan);
  }

  if(action == 3)  /* If it is a laser on */
  {
    on++;
    wprintf("%li  Laser on  %5i,%5i  $7.3f\n", 

action, x_position, y_position, vel);
laser_status = ON;
wnquery(selection,2,&scan);
}

if(action == 4) /* If it is a laser off */
{
  off++;
  wprintf("%i Laser off %i,%i %7.3f\n", action, x_position, y_position, vel);
laser_status = OFF;
wnquery(selection,2,&scan);
}

if(action == 5) /* If it is a circle set */
{
  fscanf(current, "%i", &radius); /* Scan file. */
circles++;
  fscanf(current, "%i", &speed); /* Scan file. */
  if(speed == 1)
    wprintf("%i Circle w/o cntr%5i %5i,%5i 2.000\n", action, radius, x_position, y_position);
    /* Scan file. */
  else
    wprintf("%i Circle w/center%5i %5i,%5i 2.000\n", action, radius, x_position, y_position);
    /* Scan file. */
wnquery(selection,2,&scan);
}

if(action == 6) /* If it is a start loop */
{
  fscanf(current, "%i", &times);
  wprintf("%i Start Loop 0 %i times %5i,%5i %7.3f\n", action, times, x_position, y_position, vel);
  loops++;
  num_loops = 0;
  set_up_a_loop();
  num_loops = 0;
  if(do_a_loop(times) == 1)
    { /* Laser is off. */
laser_status = OFF;
  break;
    }
  wprintf("%s End Loop 0 %5i,%5i %7.3f\n", action, x_position, y_position, vel);
  /* Print */
  end_loops ++;
  wncurpos(row, &col);
  /* Move cursor. */
  wnquery(selection,2,&scan);
}

if(action == 7) /* If it is a repeat loop */
{
  wprintf("%s Extra end of repetition "); /* Print */
  wprintf("%i End loop %5i,%5i %7.3f\n", action, x_position, y_position, vel);
  end_loops++;
  /* Increment number of endloops. */
  return;
}
}

wncurmov(++row,1);
wprintf("%s End of File %5i,%5i %7.3f\n", action, x_position, y_position, vel);
wncscroll(8,-1,-1,0); /* Scroll 8 lines. */
wncurmov(9,1);
wprintf("The final position is %i,%i.*, x_position, y_position);
wprintf("%s There were %i loops started.*, loops); /* Print */
wprintf("%n There were %i loops ended.*, end_loops);
wprintf("%n The laser was turned on %i times.*, on);
wprintf("%n There were %i circles.*, circles);
wprintf("%n (Press ENTER to continue))*; /* Print */
wmquery(selection,2,&scan);
fclose(current); /* Close file. */

/******************************************************************************
 *
 * MODULE: set_up_a_loop
 *
 * PURPOSE: To store in memory the (nested) loops of the file
 *
 * PARAMETERS: GLOBL: current nested_loops
 * num_loops speed
 * values radius
 * x_travel y_travel
 *
 * LOCAL: action - Laser action variable
 * loop_number - Current loop that is being processed
 * velo_cut - Cutting velocity of the laser
 * times - Number of times to process a loop
 * units_of_step - Single operation pointer in a loop
 *
 * USERS: do_file set_up_a_loop
 *
 * CALLS: fscanf set_up_a_loop
 *
 * RETURNS: NONE
 *
 * DATE: 24 June 1990
 * PROGRAMMER: Ryp Walters
 * LANGUAGE: Microsoft C Version 5.1
 * OS: MS-DOS Version 3.3
 *
 */
*******************************************************************************/

void set_up_a_loop()
{
    int action, times, loop_number, units_of_step;
    float velo_set;

    action = 0; /* Set initial value. */
    loop_number = num_loops;
    num_loops++;
    units_of_step = 0;

    while(action != 7) /* As long as it is not a 7 */
    {
        fscanf(current, "%d", &action); /* Scan file. */
        nested_loops[loop_number][units_of_step] = action; /* Calculate nested loops. */

        if(action == 1) /* If it is a move */
        {
            fscanf(current, "%f", &x_travel); /* Scan file for x travel distance. */
            fscanf(current, "%f", &y_travel);
            values[loop_number][units_of_step][0] = x_travel;
            values[loop_number][units_of_step][1] = y_travel; /* Find location for pointer. */
        }

        if(action == 2) /* If it is a move */
        {
            fscanf(current, "%f", &x_travel); /* Scan file for x travel distance. */
            fscanf(current, "%f", &y_travel);
            values[loop_number][units_of_step][0] = x_travel;
            values[loop_number][units_of_step][1] = y_travel;
        }

        if(action == 5) /* If it is a circle set */
        {
            fscanf(current, "%f", &radius); /* Scan file for radius. */
            values[loop_number][units_of_step][0] = radius;
            values[loop_number][units_of_step][1] = radius;
        }
    }
int do_a_loop(int count)
{
    int row, col, times, counter, loop_number, action, units_of_step;

    loop_number = num_loops;
    wncurpos(&row, &col);

    return;
}
for(counter = 0; counter < count; counter++)
{
    num_loops = loop_number;
    units_of_step = 0;
    action = 0;
    while(action := 7) /* As long as it isn't a resistor trim. */
    {
        action = nested_loops[loop_number][units_of_step];
        if (++row > 13) /* If the number of rows is more than 13 */
        {
            wnscrol(5,-1,-1,0); // Scroll window five lines.
            row = row - 5;
            // Calculate new row for window.
        }
        wncursmov(row,1); // Move cursor.
    }
    if(action == 1) /* If it is a move */
    {
        x_travel = values[loop_number][units_of_step][0]; /* Calculate new x position. */
        y_travel = values[loop_number][units_of_step][1];
        if(laser_status == ON)
            vel = vel_cut; /* Find new velocity. */
        else
            vel = velocity_set(x_travel - xpos + x_displace,
                                y_travel - ypos + y_displace);
        wnpri("%d Position %d,%%d %d,%%d %%%d,%%d \n", action, x_travel, y_travel, x_position, y_position, vel);
        /* Print.*/
        x_position = x_travel;
        y_position = y_travel;
        wnpri("%d Position %d,%%d %d,%%d %%%d,%%d \n", action, x_travel, y_travel, x_position, y_position, vel);
        /* Find new y position. */
    }
    if(action == 2) /* If it is a move */
    {
        x_travel = values[loop_number][units_of_step][0]; /* Find new x travel distance. */
        y_travel = values[loop_number][units_of_step][1];
        if(laser_status == ON)
            vel = vel_cut; /* Find new velocity. */
        else
            vel = velocity_set(x_travel, y_travel); /* Find new velocity */
        wnpri("%d Distance %d,%%d %d,%%d %%%d,%%d \n", action, x_travel, y_travel, x_position, y_position, vel);
        /* Print.*/
        x_position = x_travel + x_position; /* Find new x position. */
        y_position = y_travel + y_position;
        wnpri("%d Position %d,%%d %d,%%d %%%d,%%d \n", action, x_travel, y_travel, x_position, y_position, vel);
        /* Find new y position. */
    }
    if(action == 3) /* If it is a laser on */
    {
        on++; /* Print */
        wnpri("%d Laser on %d,%%d %d,%%d \n", action, x_position, y_position, vel);
        wnpri("%d Laser_status on %d,%%d \n", action, x_position, y_position, vel);
        /* Change laser status to on. */
    }
    if(action == 4) /* If it is a laser off */
    {
        off++; /* Print.*/
        wnpri("%d Laser off %d,%%d %d,%%d \n", action, x_position, y_position, vel);
        wnpri("%d Laser_status on %d,%%d \n", action, x_position, y_position, vel);
        /* Change laser status to off. */
    }
    if(action == 5) /* If it is a circle set */
    {
        radius = values[loop_number][units_of_step][0];
        speed = values[loop_number][units_of_step][1]; /* Find type of circle. */
        circle++; /* Print.*/
        if(speed == 1)
            wnpri("%d Circle w/o cnt %d,%%d %d,%%d %d,%%d \n", action, radius, x_position, y_position);
    }
}
else
    wprintf("%li Circle w/center%li %li,%li 2.000\n", action, radius, x_position, y_position);
    wquery(selection,2,&scan);
}

if(action == 6) /* If it is a start loop */
{
    times = values[loop_number][units_of_step][0]; /* Calculate number of times. */
    num_loops++;
    wprintf("%li Start Loop %li %li times %li,%li 7.3f\n", action, loop_number + 1, times, x_position, y_position, vel); /* Print. */
    loops++;
    if(do_a_loop(times) == 1) /* If it is the first loop. */
        return(1);
    wprintf("%li End Loop %li, %li,%li 7.3f\n", loop_number + 1, x_position, y_position, vel); /* Print. */
    end_loops += 1; /* Increment the number of endloops. */
    wcursor(dx, &col); /* Move cursor. */
}

units_of_step++;
}

return(0);

/******************************************************************************
 * MODULE:  transform
 * PURPOSE: To change a character buffer array into a numeric number
 * PARAMETERS:  PARAM: buffer - Character buffer to change
 * LOCAL:  j - Loop variable
 *         m - Loop variable
 *         n - Loop variable
 *         v1 - decimal part of the number
 *         value - integer part of the number
 * USERS:  cut_a_pattern build_file
 * RETURNS: Number in float form
 * DATE:  24 June 1993
 * PROGRAMMER:  Ryp Walters
 * LANGUAGE:  Microsoft C Version 5.1
 * OS:  MS-DOS Version 3.3
 * DATE INITIALS MODIFICATION
 * ---------------------------------------------------------------
 * ******************************************************************************/

float transform(char *buffer)
{
    float value, v1;
    int n, j, m;

    if((buffer[0] < 48 || buffer[0] > 57) && buffer[0] == 45) /* If the first character is not valid, quit */
        else /* If the first character is valid */
        {
        n=0; /* Initialize the counters */
        value = 0;
        v1 = 0;
        if(buffer[0] == 45) /* If the first character is negative, */
            n=1; /*Flag it */
            return(0);
            return(1);
        return(0);
    }
}
for(j=n; buffer[j] >= 48 && buffer[j] <= 57; j++) /* For each integer number */
    value = (value * 10) + buffer[j] - '0'; /* Add it in and shift */
if(buffer[j] == 46) /* If, after the integers there is a period */
    { /* For each possible other character */
        for(m=j; m != j; --m)
            if(buffer[m] >= 48 && buffer[m] <= 57) /* If the character is a number */
                v1 = (v1 + buffer[m] - '0')/10; /* Add it in and shift */
        } /* Do for all other characters */
    value = value + v1; /* Add the decimal and integer numbers */
if(n==i) /* If the number is negative */
    value = (-1)*value; /* Make the number negative */
return value; /* Return the number */

MODULE: vel_set

PURPOSE: Calculate the fastest velocity that the laser can safely move

PARAMETERS: PARAM: x_dist - The x distance to travel
            y_dist - The y distance to travel

USERS: cut_a_pattern build_file

RETURNS: Number (velocity) in int form

DATE: 24 June 1990
PROGRAMMER: Ryp Walters
LANGUAGE: Microsoft C Version 5.1
OS: MS-DOS Version 3.3

------------------------------------- MODIFICATION
-------------------------------------

int velocity_set(int x_dist, int y_dist)
{
    if(abs(x_dist) < 49 && abs(y_dist) < 49)
        return(2);
    if(abs(x_dist) < 199 && abs(y_dist) < 199)
        return(4);
    if(abs(x_dist) < 699 && abs(y_dist) < 699)
        return(6);
    return(8);
}
APPENDIX X

LASER SYSTEM USER'S MANUAL

10/16/90
Revised 2/25/91

Prepared by
Ryp Walters
Introduction

This system is designed so that anyone can operate the ESI Model 25 Laser System through the controlling IBM XT. Although the system is simple to use, the laser operator needs to be trained in the hardware of the entire system prior to operation. Correct operation of the system hardware, in particular with the laser power supply, is critical for equipment performance as well as proper care of the equipment. It is necessary to understand both the hardware and software characteristics in operating the laser, as covered in the user's manual.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>321</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>322</td>
</tr>
<tr>
<td>Table of Figures</td>
<td>324</td>
</tr>
<tr>
<td>Chapter 1. Laser Software</td>
<td>325</td>
</tr>
<tr>
<td>1.1 The Main Menu</td>
<td>325</td>
</tr>
<tr>
<td>1.1.1 Move Distance</td>
<td>325</td>
</tr>
<tr>
<td>1.1.2 Scribe</td>
<td>327</td>
</tr>
<tr>
<td>1.1.3 Pattern Cut</td>
<td>328</td>
</tr>
<tr>
<td>1.1.4 Cut a Circle</td>
<td>328</td>
</tr>
<tr>
<td>1.1.5 Index</td>
<td>329</td>
</tr>
<tr>
<td>1.1.6 Resistor Trimming</td>
<td>329</td>
</tr>
<tr>
<td>1.1.7-8 Hardware and Measurement System Test</td>
<td>330</td>
</tr>
<tr>
<td>1.1.9 Exit</td>
<td>330</td>
</tr>
<tr>
<td>1.2 Pattern Cut Menu</td>
<td>330</td>
</tr>
<tr>
<td>1.2.1 Create a File</td>
<td>330</td>
</tr>
<tr>
<td>1.2.1.1 Position Movement</td>
<td>333</td>
</tr>
<tr>
<td>1.2.1.2 Distance Movement</td>
<td>333</td>
</tr>
<tr>
<td>1.2.1.3 Laser On</td>
<td>333</td>
</tr>
<tr>
<td>1.2.1.4 Laser Off</td>
<td>333</td>
</tr>
<tr>
<td>1.2.1.5 Circle</td>
<td>334</td>
</tr>
<tr>
<td>1.2.1.6 Start A Loop</td>
<td>334</td>
</tr>
<tr>
<td>1.2.1.7 Repeat Loop</td>
<td>334</td>
</tr>
</tbody>
</table>
1.2.1.8 Exit

1.2.2 Edit

1.2.3 Analyze / Print a File

1.2.4 Run A File

1.2.5. Main Menu

Chapter 2. Filemake

Chapter 3. Hardware
## Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.</td>
<td>Main Menu</td>
<td>326</td>
</tr>
<tr>
<td>Figure 2.</td>
<td>Pattern Cut Menu</td>
<td>331</td>
</tr>
<tr>
<td>Figure 3.</td>
<td>Create a File Menu</td>
<td>332</td>
</tr>
<tr>
<td>Figure 4.</td>
<td>The Hardware Diagram</td>
<td>340</td>
</tr>
</tbody>
</table>
Chapter 1. Laser Software

1.1 The Main Menu

This software is completely menu driven. All single-action operations are accessed through the Main Menu, while all pattern cutting operations (repeated operations) are contained in a sub-menu. The program is started by calling the executable file LASER from DOS. Upon entering the program, a prompt will appear to turn the laser system ON. This only refers to the laser machinery, not to the laser power supply. After the laser has been turned on and ENTER has been pressed, the laser initialization sequence will occur. When this is completed, the Main Menu will appear as shown in Figure 1.

The screen is divided into two windows. The upper window is the active user window which displays the user's current options. The lower window is the parameter window that displays the current laser position as well as other information dealing with the user's previous choices.

A description of the Main Menu options is as follows:

1.1.1 Move Distance

Move Distance allows the user to actively change the current position of the laser system. This can be done by either using the joystick on the laser keypad or by entering the distance to move into the PC. Both techniques are very useful. The joystick method is most applicable when an exact position is not necessary. The move is completed by pressing the load position button on the laser keypad. Examples of the joystick's use include retrieving a magnification of the material by viewing it on the monitor or finding certain
Laser Operating System

What would you like to do?

1. Move distance
2. Scribe
3. Pattern cut
4. Cut a circle
5. Index
6. Resistor trimming
7. Hardware test
8. Test measurement system
9. Exit

Option:

System Status

Current Position: 0, 0

Figure 1. The Main Menu
characteristics of the material such as corners, sides, or figures. The numeric distance method is best suited for precise distance requirements, like the half-way point between two known objects. If either the distance or velocity is 0, no change will result.

A combination of the two methods might also be required. This is seen when a 2" X 2" substrate needs to be scribed into two 1" X 2" pieces. The joystick can be used to place the position at a particular corner. A distance of 2540 motor steps (1") in the appropriate axial direction will then yield the correct initial position for the scribe. One motor step equals .01 mm (2.54 mil).

It is important to remember with any type of move that the laser system has position limits. It is very easy to forget about the location of the current position. This may result in trying to move to a position that is out of the laser's range. The joystick method will recover from a limit bump, but a circuit breaker in the upper left corner of the left front cabinet may be tripped. This is solved by moving the laser arm by hand to its center and resetting the circuit breaker. The distance method will not allow movements that would move out of range. Upon entering an out of range distance, the computer will alert the user of the error and will request a valid distance.

1.1.2 Scribe

Scribe is the single-scribe routine. The computer will prompt for the way in which the scribing end points will be entered: by joystick or by distances. Most applications of this routine will use the joystick, which prompts the user for both the initial and the final points. The points are entered by pressing the LOAD POSITION button on the laser keypad. Some applications may require the use of specific distances. The program currently assumes that the user is already at the initial point for a distance scribe. The computer only needs the distance to the final point. After loading the positions with either the joystick or the distance
to travel, the computer will ask for the number of times that the complete distance should be scribed. A reply of zero will cancel the routine. The user should note that after scribing, the final position may not be at the same initial position. After the scribe has been completed, the computer will ask if more scribing needs to be done. A response of NO will return control to the Main Menu.

1.1.3 Pattern Cut

Pattern Cut is a separate menu system that allows for sequences of actions to be created, printed/analyzed, edited, and run. Discussion of this option is left until 1.2.

1.1.4 Cut a Circle

Cut a Circle is a routine that cuts a complete circle of a given radius. Valid radii are from 1 to 3000 motor steps. It starts by asking if the current position is the origin of the circle. A response of NO will allow movement to that position using the techniques described in the move distance routine. When the origin is set, the user is asked what kind of circle he wants: saving-the-circle or cutting-the-circle.

Both types of circles will travel at the same speed without any pause as they circumscribe the circle. The only difference is that the saving-the-circle option turns the laser on after it has moved to the circumference but before it starts to move along it. The cutting-the-circle option turns the laser on at the center of the circle and cuts a radial line as the laser moves form the center to the circumference. It then proceeds around the circumference leaving the circle radially scribed. The saving-the-circle option is good for most applications, but it occasionally has an unwanted effect on small radius circles. This is due to the fact that when the laser is turned on, it pulses with a greater intensity than when it normally operates. This produces a hole of about 2 motor steps (1 mil) in diameter. With
circles that have radii greater than about twenty motor steps, this fringing is negligible due to the curvature of the circle. For circles with relatively large radii, the saving-the-circle option is reliable to use without complications. Circles with radii less than about twenty motor steps are most efficiently created using the cut-the-circle option, assuming that the inner portion of the circle does not need to be saved. With the cut-the-circle option, the continuity of the circumference is preserved. After the circle has been completed, the computer will ask if another circle needs to be cut. A response of YES will allow for a change in the origin and/or a change in the radius. A response of NO will return control to the Main Menu.

1.1.5 Index

Index is a quick way to move the laser to its home position (0,0).

1.1.6 Resistor Trimming

Resistor Trimming is the routine that performs a resistor trim. After entering the probe positions, the current value of the resistor that is connected across those probes is displayed. The computer will ask for the (higher) resistance to which the resistor is to be trimmed. After using the joystick to enter positions on both sides of the resistor, the start button must be pressed to begin the trim. Note that the computer has no way of checking the relation between the position that was loaded and where the resistor really is located. Incorrect positions can lead to computer lock-up. After the trim has been completed, the computer will ask if another resistor trim needs to be done. Only one resistor can be trimmed at a time. A response of NO will return control to the Main Menu.
1.1.7-8 Hardware and Measurement System Test

Hardware and Measurement System Test are test routines for the purpose of maintenance and trouble-shooting and should only be used by the laser system supervisor.

1.1.9 Exit

Exit will return the system to DOS.

1.2 Pattern Cut Menu

The third choice of the Main Menu is for a pattern cut. The pattern cut menu (Figure 2) includes all file oriented functions. These functions are to create a file, analyze/print a file, edit a file, and run a file. No other choices from the Main Menu include any file oriented functions. When a file needs to be constructed and/or tested, the program "Filemake" should be run instead of "Laser". The "Filemake" program does not require the use of the laser and can be run on any system. (Remember that the "Laser" program requires that the laser machinery be running.) Further information on filemake is in the next section. A synopsis of the pattern cut menu functions is as follows:

1.2.1 Create a File

Create a File is a function that builds a new file. The origin is assumed that wherever the laser is located when the file is to be run. Therefore, the file assumes a starting position (0,0). The computer will then prompt for an eight character filename without any extension. Once the file has been opened, the cutting velocity of the pattern
What would you like to do?

1. Create a file
2. Analyze / print a file
3. Edit a file
4. Run a file
5. Main menu

Option:

Current Position: 0, 0

Figure 2. The Pattern Cut Menu
Laser Operating System

What would you like to do?

1. Move to a position
2. Distance move
3. Laser On
4. Laser off
5. Circle cut
6. Start a loop
7. Repeat a loop

9. Exit

Option:

System Status

Current Position: 0, 0

Figure 3. The Create a File Menu
will be requested. This is not the travelling velocity, but the cutting velocity that the laser will have whenever the laser is on. See Chapter 4 for good velocities for the material that is to be cut. The laser operation list (Figure 3) will then appear.

An explanation of these operations is as follows:

1.2.1.1 Position Movement

Position Movement will prompt for the desired position. The program assumes that it will start at position (0,0). This position is not the internal laser: position, but an arbitrary starting point. This is acceptable because when the file is running, the user is prompted to move to the starting point. This starting point will then be defined as (0,0) and the file will run with respect to it.

1.2.1.2 Distance Movement

Distance Movement will prompt for the distance to move from the current position.

1.2.1.3 Laser On

Laser On will turn the laser on at that point in the file. If the laser was previously turned on, it will remain on.

1.2.1.4 Laser Off

Laser Off will turn the laser off at that point in the file. If the laser was previously turned off, it will remain off.
1.2.1.5 Circle

Circle is a routine that cuts a circle of any radius. Regardless of whether the laser is ON or OFF, the laser will be ON when the circle is cut. The circle velocities are preset and unchangeable. There are two types of cuts, cutting-the-circle and saving-the-circle. The difference between the two methods was described in the Cut a circle operation from the Main Menu.

1.2.1.6 Start A Loop

Start a Loop is the open marker for a loop sequence. The computer will ask for the number of times the loop is to be processed.

1.2.1.7 Repeat Loop

Repeat a Loop is the close marker for a loop sequence. All loops that have been started MUST have a repeat or else the file will not be processed correctly and the computer will lock up.

1.2.1.8 Exit

Exit will close the file and return to the pattern cut menu.

Once a file has been created, it is a very good idea to analyze it so that it can be checked for errors. It is also good practice to do a distance move as the last step on all files. This allows for greatest gain in the use of an analysis.

On an exit, the file is automatically saved. There is no need to "save" the file. It is advisable to keep backups of your files by copying the filename with a ".new" extension onto a backup disk. In order to use a file, it must be in the same directory as the Laser Program.
No change of directory or drive is allowed. Once a file has been created, the user MUST fill out a file information sheet so that other people will know that the file exists, what it does, and who uses it. This sheet will also be good for keeping track of changes and for writing comments for next time the file will be used. A printout of the file should also be kept along with this information sheet.

1.2.2 Edit

Edit is a function that steps the user through a file line by line. The cutting velocity will be displayed first and the prompt will say "Is this correct? (y/n)". Entering 'n' will allow a new cutting velocity to be entered. A 'y' will continue to the next operation. The prompt will now say "Is this correct? (y/n/q)". Entering 'q' will quit the editor and save all previous changes. If the operation is correct, press 'y' and ENTER. Simply pressing ENTER will assume a 'y' reply. With this, the next operation will appear with the "Is this correct? (y/n/q)" prompt.

If the operation is not correct, press 'n' and ENTER. An incorrect operation means one or more of the following:

- commands need to be inserted before this operation,
- the operation should be deleted (skipped),
- the parameters are incorrect (wrong distance, radius,...).

After pressing 'n' and ENTER, the prompt "Do you want to insert commands? (y/n)" will appear. If commands do need to be inserted, enter 'y'. This will take the user to the create a file menu discussed above. Select option "9. Exit" when all required commands are inserted. This will return the user to the edit.
The next prompt is to delete (skip) the operation. A 'yes' response for the delete skips to the next operation and deletes the previous operation from the file. A 'no' response for the delete leads to a prompt for if the current parameters are correct. Those operations that do not have parameters (laser on, off, end loop) will skip this prompt and go on to the next operation. For those operations that do have parameters, this is the time to change them if they are incorrect.

To change the parameters, enter 'n'. The editor will then prompt for the correct parameters. After changing the parameters or selecting not to, the operation is stored in the file and the next operation is listed.

This process can be continued until the file has been completely stepped through or until 'q' is pressed. This quit will save all changes made to the file during the edit and will then quit the editor.

1.2.3 Analyze / Print a File

Analyze / Print a File will keep track of the position, as well as the number of laser ON times, laser OFF times, loops started, loops repeated, and circles. This allows for a checking sequence between what the file is expected to do and what it will really do when it is run. As described previously with creating a file, it is a good practice to do a distance move (not a position move) back to the origin as the final operation before exiting the creation of a file. This way, when the final analysis is shown, if the final position is not (0,0), it will be obvious that an error exists. Doing a position move to (0,0) as the final operation will not give this insight because the computer will tell the laser to move whatever distance is required to get back to (0,0), whether it was the intended distance or some other distance. This function also prompts for whether the file and analysis should be sent to the printer.
The printed file contains the operation name, all of the operation's parameters, the position the laser would be at BEFORE the operation is processed, and the velocity that the operation would be processed at. This gives the position of the laser as it will be when it is running which can be checked with the original diagram. The final analysis is also printed. This is ideal for debugging as well as including with the file information sheet to show the exact values and parameters. It is also very important to keep written reports of the files.

1.2.4 Run A File

Run A File is the actual function that sends the file to the laser for processing. It starts out by asking if the current position of the laser is the position to be stored as the origin for the file. If the current position is not the origin, the joystick and/or distances may be used to get to the origin as described in the Move distance option (1.1.1) of the Main Menu. The final prompt is to hit the START button on the laser keypad when everything is ready. Once this button is pressed, the file will be acted upon, so be sure that the power is set, that the status is ON, and that the interlocks are closed. To break the file before completion, the STOP button on the keypad may be pressed. This will break the file, turn the laser off, and return to the origin of the file. The file may be run again without any problems.

1.2.5 Main Menu

The option returns to the main menu.
Chapter 2. Filemake

This is a stand alone version of the Pattern Cut Menu found in the laser program. The main use of this program is to allow the creation and analysis of files without having to be in the laser environment. The laser system is required to be ON when the "LASER" program is used. The "FILEMAKE" program can be used on any computer. This is very important during times in which the laser is needed for many different projects. With this program, the laser operator for each project can have his own place to make and analyze files other than on the computer dedicated to the laser. This should cut down on the amount of actual system time required by any one project.

This program is exactly like the Pattern Cut Menu except that the Main Menu option is an exit to DOS option and the run option will only run the file on the screen. With the run option, pressing ENTER after each operation will cause the next operation to appear. This option can be useful in the debugging process. Specific information on the other options of this program can be found in the Pattern Cut Menu description.

The created file is saved in whichever drive and directory that this program is running. To use the file on the laser, use the DOS copy command to copy the filename with the ".new" extension to a floppy and then copy it onto the laser system computer's hard drive. If the filename was "Test", the DOS command to copy the file from a floppy to the hard drive would be "Copy a:\Test.new d:\Laser".
Chapter 3. Hardware

The laser system is made up of three different units; the IBM computer, the laser machinery, and the laser power supply. To fully operate the system, all three units need to be operating. The computer has its own power switches on its side and on the monitor. As seen in Figure 4, the laser is turned ON by switching the middle toggle switch (B) on the power box down to local mode. Both off and remote modes are OFF. It is a very good practice to keep the line power circuit breaker (A) OFF when the system is not in use. This must be turned ON before the middle switch will turn on the laser machinery. The laser system and laser power circuit breakers can always be left on.

The laser power supply will only turn ON if the laser machinery is ON. At this point, it is mandatory to turn ON the water by completely opening the water valve located near the wall behind the power supply. The air should also be turned ON at this point. With the machinery ON, the power supply can be turned ON by inserting the key (a), turning it a quarter turn clockwise, and pushing up the switch (b). The power supply panel should have its switches set according to Figure 4 with the start button(c) lit. After the start button is brightly lit, press it. This should cause the ammeter (d) to go from 0 to approximately 13 amps. If the ammeter returns to 0 (and the start button lights up), wait until the light is completely lit and press the start button(c) again.

The sequence of remaining set-up events is very important. The switches must be set as in figure 4 before proceeding.
* First, adjust the ammeter with the knob (e) to the required number of amps for the material that is to be cut. This can be found on the materials chart in Chapter 4.

* Second, turn the mode selector(f) to GATE ON. This will allow the viewing of frequency setting on the wattmeter/frequency gage (g).

* Turn the gage selector (h) to FREQ and vary the frequency using the internal FREQ adjust screw (i) and the x1/x10 switch(j). The frequency is read by the lower scale of the gage (g) in Khz. When this is set (according to the material to be cut and as listed in Chapter 4), switch the mode (f) back to EXTERN GATED.

* With one of the laser interlocks open (usually the stage cover), switch the laser status switch(k) from ST BY (stand-by) to ON. At this point, the power supply is ready for action.

To change the frequency or number of amps after the supply has been configured, open a laser interlock (stage cover), switch the laser status to ST BY(k), do the adjustments(e,f,g,h,i,j), switch the laser status to ON(k), and continue the processing.

The laser machinery does not have any configuration, but it is very important to realize that the x and y motors that move the laser optics do have upper, lower, right, and left limits. It is VERY important not to get carried away when using the joystick with a move. Not being careful can result in hitting one of the limits. The computer will not lock up, but damage can occur to the motors and a circuit breaker may be tripped.

The keypad is mainly used through prompting by the laser program. The STOP button, though, is useful for breaking a file that is running. There are two platforms available for use on the laser: one for Green Tape(TM) and one for any relatively thin substrate
(stainless steel, aluminum nitride, ...). The platforms can be switched by pressing the air controlled pedal located on the floor in front of the laser and spinning the arm on which the platforms are connected.
Curriculum Vitae

Ryp Richard Walters was born on June 24, 1991 at the Walter Reed Hospital in Washington, D.C. As a Boy Scout, Ryp achieved the rank of Eagle Scout and earned his fifth year Pipestone. In the Fall of 1987, Ryp began attending Virginia Polytechnic Institute and State University. On the 9th day of March in 1988, Ryp fell head over heels for Paige E. Curtis. Determined as ever, he proposed to her on December 9, 1989 and they married each other on June 9, 1990. Ryp completed his undergraduate degree of a Bachelor of Science in Electrical Engineering in December of 1990. That same Fall, he started his Master’s Degree in Electrical Engineering while completing his undergraduate degree through the Dual Status Program.

Nothing that Ryp accomplished was done without the support of his twin flame Paige. Together they make an undefeatable pair.

Ryp R. Walters