

AI/CS 4 Project  
Model-Driven Assembled  
Circuit Board Inspection

Neil W. Rumney

May 29, 1990



In this project, a model-driven inspection system for circuit board components is described. The system is designed to exploit a modular approach allowing it to be extended easily for larger subsets of electronic components. The project is based on work done by Chan [1] but extends the model description language used to introduce hierarchical descriptions and some labour saving language features which make defining models easier. It is designed to inspect four types of component features - integrated circuit pins and bodies, capacitors and resistors.



# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Overview of Implementation</b>	<b>7</b>
2.1	Model Data and Loader . . . . .	7
2.2	Inspection Control Mechanism . . . . .	8
2.3	Inspection Tests . . . . .	9
2.4	Windowing System . . . . .	9
2.4.1	Transforming Co-ordinate Systems . . . . .	10
2.4.2	Displaying the Windows . . . . .	11
2.4.3	Preparing the Window Area for Applying Inspection Tests . . . . .	11
2.4.4	Example of Windowing System . . . . .	12
<b>3</b>	<b>Test Specification</b>	<b>15</b>
3.1	Circuit Board Contents . . . . .	15
3.2	Model Data . . . . .	16
3.3	Error Recovery . . . . .	19
3.4	Extending the Model . . . . .	19
3.4.1	Associating Tests With Features . . . . .	20
3.4.2	Pairwise Tests . . . . .	20
3.5	Example Model . . . . .	21
<b>4</b>	<b>Specific Tests</b>	<b>24</b>
4.1	Pin Test . . . . .	25
<b>5</b>	<b>Test Plan</b>	<b>28</b>
5.1	Modelling Test Plan . . . . .	28
5.1.1	Testing The Model Loading . . . . .	28
5.1.2	Example Test Models . . . . .	32

Faint, illegible text, possibly bleed-through from the reverse side of the page.



5.1.3	Model Testing Results . . . . .	37
5.2	Inspection Test Plan . . . . .	37
5.2.1	Translation and Rotation of Image . . . . .	37
5.2.2	Lighting Conditions . . . . .	37
5.2.3	Inspection Test Plan Results . . . . .	38
6	Achievements and Conclusions	40
7	Acknowledgements	43
A	Grammar BNF	45
B	User Instructions	47
C	Program	48
C.1	Extending the Program . . . . .	49
D	Program Listing	52





# 1 Introduction

With the development of robots being used on the production line, it has been increasingly necessary in terms of quality control to automate the inspecting processes of items produced in this way. This results in less time being spent by human inspectors on these tedious jobs. The introduction of automatic inspection also has the benefit of being much quicker and less prone to mistakes that human inspectors might make after a long shift. One particular inspection task that would be particularly useful if done automatically is circuit board inspection. Increasingly, circuit boards are assembled by robot rather than by hand. The layout of circuit boards are becoming more densely packed and therefore more difficult to inspect. Assuming that a suitably high picture resolution can be achieved then automatic inspection is a good candidate.

The problem being considered in this project is to carry out a series of tests to check that a circuit board matches a predefined model. A lot of inspection systems that have been developed previously have worked with binary images [1]. This is not possible for circuit board inspection because the features being examined vary greatly in intensity and too much information would be lost if the work was done using a binary image.

The inspection tests to be carried out are designed to see that the circuit board has no structural defects after being assembled. The defects that the tests will be designed to find include bent or missing chip legs, missing components, misplaced components and extra components that should not be there. Simple tests can be defined that will check for the existence of objects in the image of the circuit board. The problem is to be able to reliably measure the properties of the component and compare them against those specified for the component in the model.

The project can be readily split into two sections. The first is the design of the controlling framework. This will use a model definition of a circuit board to guide the inspection process, calling the appropriate tests where needed. The second section is the design of reliable tests. Inspecting a circuit board involves the following five steps:

- Capturing an image of the circuit to be inspected.
- Creating a model definition of the circuit.
- Parsing the model definition.
- Checking the semantics of the model.
- Applying the tests referenced in the model to the image features. This requires transforming from model to image co-ordinates.

The first part of the document discusses the importance of maintaining accurate records. It highlights the need for regular updates and the role of technology in streamlining the process. The text emphasizes that proper record-keeping is essential for compliance and operational efficiency.

In the second section, the author explores various methods for data collection and analysis. It compares traditional manual entry with modern automated systems, noting the benefits of reduced error rates and faster processing times. The text also touches upon the importance of data security and access control.

The third part of the document focuses on the integration of different systems. It discusses how data from various sources can be consolidated to provide a comprehensive view of the organization's performance. The text mentions the challenges of interoperability and the need for standardized protocols.

The final section provides a summary of the key findings and recommendations. It reiterates the importance of a proactive approach to record management and suggests specific steps for implementation. The author concludes by expressing confidence in the future of digital record-keeping.

Yours faithfully,

[Signature]

[Name]

[Title]

For more information, please contact the records management department at [Phone Number].

The structure of the controlling framework is based on the work of Chan [1]. In his paper he describes a method for the inspection of assemblies according to a geometric model. He developed a language for creating models using features consisting of lines, holes, corners and arcs. I have taken this as a basis for an extended language which is more suitable for describing circuit boards. The extended language allows for hierarchical object definitions that would not have appeared in Chan's assemblies.

An important consideration with the design of this project is that it should be easily extended to take account of more component features and feature tests. Therefore the framework should not depend upon some static definition of the modelling language being used.

A small example of what has been accomplished is shown below. The model below describes an IC positioned at the left hand side of the circuit board (see figure 1). The right hand row of pins is to be inspected. When this is done the window around the pins is highlighted and a copy of the window is positioned in the upper left hand corner of the image. This copy is how the test *sees* the area to be inspected. The scan line along which the pin test is done is shown in the upper left hand window.

```
CIRCUIT memory
WITH
    COMPONENT chip ic
        POSITION(15,200)
        ORIENT(-90)
ENDCIRCUIT

COMPONENT chip
WITH
    IClegs: iclegs2 WINDOW(135,15) AT(135,50,180)
    PINCOUNT(135,15,8,5)
ENDCOMPONENT

END
```

There are three other features available in the component library - ICbody, RESISTOR and CAPACITOR. These can all be used in the same way as IClegs except that different tests would be used than PINCOUNT.

*[Faint, illegible text, likely bleed-through from the reverse side of the page]*

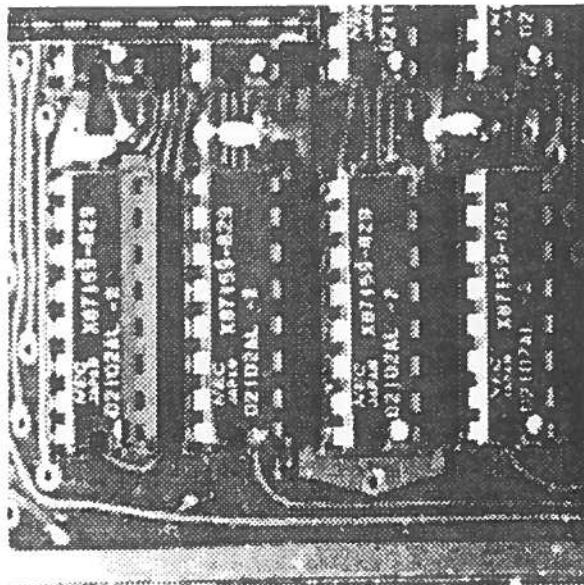
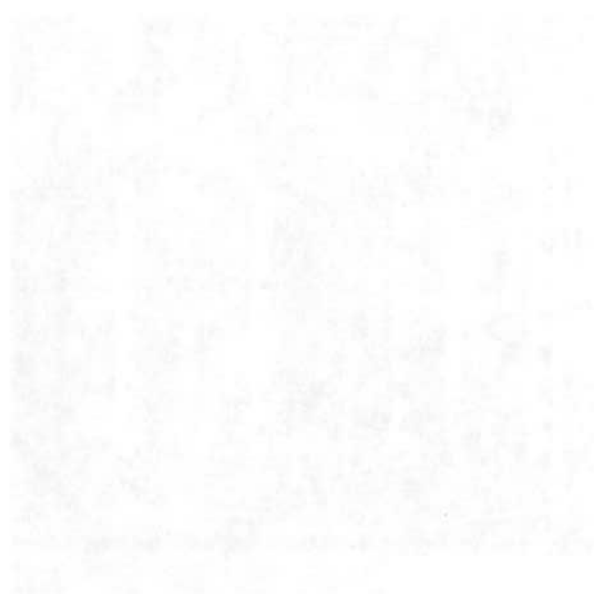


Figure 1: Display on Framestore



## 2 Overview of Implementation

This chapter describes the modules that constitute the inspection system. There are four modules:

- Model Data and Loader
- Inspection Control Mechanism
- Inspection Tests
- Windowing System

### 2.1 Model Data and Loader

The model data describes the circuits that are going to be inspected. The information given in the model descriptions gives the contents of the circuit, positional data for the components within the circuit and also information on how to identify whether particular components are correctly positioned and undamaged. The model structure and examples are discussed in more detail in chapter 3.

The model loader consists of a parser that has been written using the UNIX©tools YACC and LEX, and a model linker. The parser has been written so that all the relevant information held in the model data files is stored in a more convenient and compact form for use during the inspection process. The parser does not attempt to try any error recovery. If any syntax errors occur then the program will stop.

The output from the parser is a tree that has two main types of node. The first describes a type of component and what subcomponents it contains. The second type gives instances of component definitions. These instances contain information relating to the position and orientation of the particular components that appear in the circuit.

The inspection process requires access to both the definition of a component and also the instance of the component before it is able to inspect the components. To make these accesses easier the parse tree produced earlier is modified slightly by linking up each instance with its corresponding definition using the model linker. This is carried out after the parsing has taken place. It merely re-organises the layout of the data.

The first part of the document discusses the importance of maintaining accurate records.

This section describes the various methods used to collect and analyze data.

The results of the study are presented in the following table.

The data shows a significant increase in the number of participants over time.

It is concluded that the study has provided valuable insights into the topic.

The findings suggest that there is a need for further research in this area.

The authors would like to thank the participants for their contribution to the study.

The study was supported by the National Science Foundation.

The data was collected over a period of six months.

The results are consistent with previous research on the subject.

The study has implications for the development of new policies.

The authors are grateful to the reviewers for their helpful comments.

The study was conducted in accordance with the ethical guidelines.

The data is available upon request.



## 2.2 Inspection Control Mechanism

The inspection process applies tests to the specified features of the components specified. All of these components may not necessarily be referenced at the top level of the model definition. This is because components can be defined to contain subcomponents in their definition. The process requires that all the components appearing in the circuit definition be inspected.

Each component can contain instances of three forms of feature - a subcomponent, a subcomponent defined inside a repeat loop, and one of a number of predefined features. This means that one of three things must happen depending upon the type of feature in the component.

- If a non-repeating component is found then this subcomponent can then be inspected using the same process.
- A repeating component requires that the component referenced must be inspected the appropriate number of times with varying positional parameters given each time.
- A predefined feature has a list of tests which have to be applied to it. If all the tests succeed then the feature is said to have been tested correctly.

The inspection process can be viewed in the following way:

```
Select circuit model definition

repeat
    get component
    if component is in repeat loop then
        inspect component once for each position
    else
        inspect component once
until no more components
```

where *inspect* does the following:

```
repeat
    get feature
    if feature is in a repeat loop then
        inspect subcomponent once for each position
```

The first part of the report deals with the general situation in the country. It is noted that the economy is still in a state of depression, and that the government has taken various measures to stimulate it. The report then goes on to discuss the various sectors of the economy, including agriculture, industry, and commerce. It is noted that agriculture is still the main source of income for the population, and that the government has taken various measures to improve the situation of the farmers. The report also discusses the situation in the industrial sector, and notes that there has been a decline in industrial production since the end of the war. The report concludes by noting that the government has taken various measures to improve the situation of the population, and that it is hoped that the economy will soon be on a recovery path.



```
    if feature is a subcomponent then
        inspect subcomponent
    else
        apply tests to feature
until no more features
```

The recursive nature of the process can be seen in the inspection of subcomponents. Each definition of a component can have subcomponent definitions within it. It does not matter in which order the inspection is done since all the inspections have to be carried out.

### 2.3 Inspection Tests

The inspection process does not search over the whole image to find the features that it is inspecting. The model definition gives windows to direct the inspection of the particular features being referenced. A series of tests are then applied to this area of the image. If all succeed then it is assumed that the feature has been found and is not defective. The whole process therefore depends on the tests being sufficiently rigorous to identify the presence and correct composition of the component in question. The tests to be applied are detailed in the model definition for that circuit board. Each test will be used to find out specific things about the feature such as its orientation. The tests themselves will convey any necessary information back about the reason it might have failed.

### 2.4 Windowing System

The windowing system developed for the program is used when inspecting the features of each component. It converts from the model co-ordinate system to a world or screen co-ordinate system. The complication encountered here is that more than one transformation may be required. Each component defined has its own co-ordinate system as does the circuit board itself. Therefore if a particular component being inspected is the subcomponent of another then two transformations have to be carried out rather than a single one for the local co-ordinate system. A final transformation is then required to get to screen co-ordinates.

For each component there is a window with origin  $(x, y)$  and orientation  $\theta$  relative to its super-component definition. At the top level we have the circuit board with position and orientation relative to the screen. At the next level down we have component window positions relative to the circuit board. The subcomponents are positioned relative to the components (super-components) that are defined in terms of the subcomponents.

MEMORANDUM FOR THE RECORD

DATE: 10/15/54

RE: [Illegible]

[Illegible text follows, appearing to be a summary of a meeting or report.]

Very truly yours,

[Illegible text, likely the body of the memorandum.]

[Illegible signature]

The aim of the windowing system is threefold:

- Transform from component co-ordinates to screen co-ordinates
- Displaying the windows on the screen for debugging and user feedback
- Putting the window data into an appropriate form for applying tests

#### 2.4.1 Transforming Co-ordinate Systems

Transforming from model co-ordinates to screen co-ordinates may require multiple transformations, one for each level of subcomponent nesting. For  $n$  components each of which is a subcomponent of the next we have  $n+1$  transformations to get from component  $n$ 's co-ordinate system to screen co-ordinates. The extra transformation is required to get from circuit co-ordinates to screen co-ordinates. This process can be viewed as the following matrix multiplications combined together to give a single transformation:

Take the transformation matrix for moving from one co-ordinate system to another to be

$$\begin{pmatrix} \cos\theta & -\sin\theta & x \times s \\ \sin\theta & \cos\theta & y \times s \\ 0 & 0 & 1 \end{pmatrix}$$

for a window with origin  $(x, y)$ , orientation  $\theta$  and scale factor  $s$ . This implementation uses a single scale factor which converts from the model co-ordinates to screen pixels. It is therefore assumed that all the models have the same scale factor, i.e. are defined in terms of millimeters.

Let  $T_c$  to be the transformation matrix from circuit co-ordinates to screen co-ordinates. This matrix has a slightly different form from the others. A scaling factor is not used when defining the position of the circuit board to the screen origin. This is because the information is not held in the model definition and has to be given at run-time by finding out the co-ordinate of the reference point of the circuit board on the screen. Since this figure is already a screen co-ordinate then no scaling need take place. The resulting combined transformation  $T_{final}$  is given by

$$T_{final} = T_c T_1 T_2 \dots T_n$$

where  $T_1$  is the matrix for the outermost component definition,  $T_2$  is the matrix for a



subcomponent of  $T_1$  etc.

$T_{final}$  can then be applied to the corner points of the window defined for component  $n$  giving the desired screen co-ordinates.

#### 2.4.2 Displaying the Windows

The graphics package being used for display purposes on a framestore is a very basic one. It is not able to cope with displaying a rectangular window at all angles on the screen. It is therefore necessary to produce a copy of the screen with the appropriate changes made where the window is to be displayed and then dump the whole of this copy out to the framestore. To achieve this it is necessary to know more than just the corner points of the window. All the boundary points of the window must also be calculated.

Calculating the boundary points of a window is achieved by finding out the equations of the four lines that make up the window. From these equations it is simple enough to find all the possible  $x$  co-ordinates for each  $y$  co-ordinate of the window.

Once the boundary of the window has been calculated it is then necessary to create an image of the whole screen to be displayed. The original graphical output was developed on a framestore that could display three different pictures at once by displaying each one on a different colour plane. This was useful because the image of the circuit board could be displayed on one image plane with the windows showing where the inspection process is working could be showed on a second colour plane. This method had to be modified later to work on a different framestore that was only able to display one colour plane at any one time. Obviously there could not be two separate images used this time. It is not possible to display the window as a solid block of colour because this would blot out the part of the circuit board that is being scrutinised. The solution used was to take a copy of the circuit image and then invert the parts of the image that correspond to the window being displayed. This does not have such a desirable effect as using a different colour but it does highlight the region sufficiently.

#### 2.4.3 Preparing the Window Area for Applying Inspection Tests

Once the outline of the window has been identified it is then necessary to get the appropriate data into a form suitable for applying tests. If the data was used directly from the screen image then the result would be that each window area tested could be at a different angle. This would mean that tests would have to be able to cope with the varying angles that the data could be given to them. It was therefore decided to rotate all the screen co-ordinates of

Faint, illegible text, possibly bleed-through from the reverse side of the page. The text is too light to transcribe accurately.



the windows round until they were parallel to the co-ordinate axes of the model co-ordinate system. The only information required by the tests themselves is the size of the window plus any scaling factor that might be involved.

#### 2.4.4 Example of Windowing System

An example of the windowing system is detailed below. Figure 2 shows how two components, A and B, are related to each other and also how component A is related to the screen origin. The angles that are marked show the rotation of the component in relation to its super-component. The co-ordinates show the translation. The diagram shows that component B is positioned at point (20, 40) inside component A at an angle of  $-90^\circ$ . Component A is in turn positioned at (35, 5) with rotation  $45^\circ$  to the screen's x-axis.

The co-ordinate frames for these two objects are given by the tuples

$$[35, 5, 45^\circ]$$

for component A relative to the screen

$$[20, 40, -90^\circ]$$

for component B relative to component A

Figure 1 shows the co-ordinate axes used in defining objects. It was decided to use the same axes as used with the graphical display to keep consistency. This means that the origin is the upper left hand corner of the co-ordinate system. To obtain the transformation from the co-ordinate system in component B to screen co-ordinates we have to apply the technique show earlier. Take the two matrices formed from the co-ordinate frames and multiply together to give a combined transformation. The leftmost of the matrices should belong to the outermost component definition. Thus we have

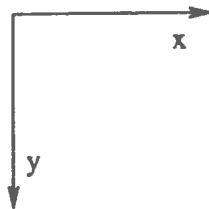


Figure 1: Co-ordinate Axes for Models and Screen

Handwritten text at the top of the page, possibly a header or title.

Handwritten text in the upper middle section of the page.

Handwritten text in the middle section of the page.

Handwritten text in the lower middle section of the page.

Handwritten text in the lower section of the page.

Handwritten text at the bottom of the page, possibly a footer or signature.

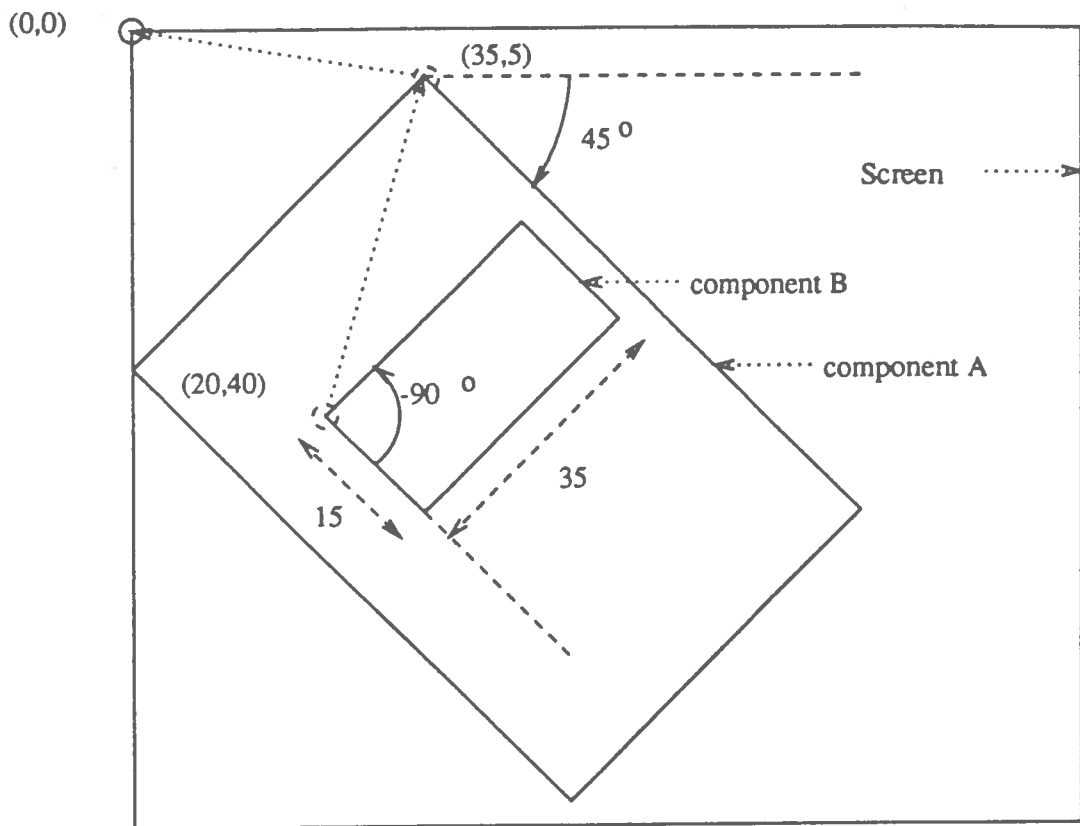


Figure 2: Pictorial View of Model

$$T_{final} = T_A * T_B$$

or in matrix form we have

$$T_{final} = \begin{pmatrix} \cos(45^\circ) & -\sin(45^\circ) & 35 \\ \sin(45^\circ) & \cos(45^\circ) & 5 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(-90^\circ) & -\sin(-90^\circ) & 20 \\ \sin(-90^\circ) & \cos(-90^\circ) & 40 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 0.7 & 0.7 & 20.9 \\ -0.7 & 0.7 & 47.4 \\ 0 & 0 & 1 \end{pmatrix}$$

We now have the matrix for applying to the four corner points of component B. Remember



that these corner points are defined in relation to the reference point of component B. So for length 35 and width 15, we have the corner points

$$(0,0) (35,0) (35,15) (0,15)$$

Translating to screen co-ordinates we get

$$= \begin{pmatrix} 0.7 & 0.7 & 20.9 \\ -0.7 & 0.7 & 47.4 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 35 & 35 & 0 \\ 0 & 0 & 15 & 15 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

which gives the points

$$(21,47) (46,23) (56,33) (32,58)$$

Once the screen co-ordinates have been calculated then the windowing system places the block of screen bounded by these corner points into a more suitable form for the inspection process.

... ..

... ..

... ..

... ..

... ..

... ..

... ..

... ..

... ..

... ..

... ..

... ..

... ..

... ..

... ..



### 3 Test Specification

A description of the circuit being tested is required so that the program is able to focus on the correct aspects of the image data. This chapter describes what kind of description is required and how the program receives it.

Section 5.1 describes a test plan that was used to evaluate the object modelling capabilities according to the specification given below.

#### 3.1 Circuit Board Contents

The aim of this program is to obtain a reliable inspection process for different circuit boards. The inspection process is designed to examine the components that make up the circuit board. There are a large number of different type of components that can appear in a circuit board. It was decided to concentrate on only three types to demonstrate the approach:

- Integrated Circuit
- Resistor
- Capacitor

Modelling these components should allow a wide variety of circuit boards to be described. The resistor and capacitor have simple bodies. The integrated circuit is an example of a more complex component that is made up from a number of smaller parts.

Modelling a circuit board using only descriptions of these three components restricts the extent of the modelling capabilities. Other components that may be present have to be ignored. There may be tracks visible on the surface of the board which are not modelled. All of this adds up to what appears as noise to the inspection process.

What information must be modelled? To model a circuit board there are two pieces of information that must be given - a position and what to expect at that position.

The components which are to be modelled come in standard forms for each type of component that appear on circuit boards. The differences come in where they are positioned. It is necessary to have a reference point on a component by which you can define a position and orientation that it will appear on a circuit board.

Describing the component in terms of a geometric description involving lines and arcs does not give enough information. It may describe the outline of components well enough

1. The first part of the document discusses the importance of maintaining accurate records of all transactions. It emphasizes that proper record-keeping is essential for the integrity of the financial system and for the ability to detect and prevent fraud.

2. The second part of the document outlines the specific requirements for record-keeping, including the need for clear, legible entries and the requirement to retain records for a minimum of seven years. It also discusses the importance of regular audits and the role of internal controls in ensuring the accuracy of the records.

3. The third part of the document provides a detailed description of the record-keeping system, including the types of records that must be maintained and the methods used to collect, process, and store the data. It also discusses the importance of data security and the need to protect the records from unauthorized access and loss.

4. The fourth part of the document discusses the role of the record-keeping system in the overall financial management process. It emphasizes that the system is not only a tool for record-keeping but also a means of providing valuable information to management for decision-making purposes. It also discusses the importance of regular reporting and the need to ensure that the records are up-to-date and accurate.

5. The fifth part of the document provides a summary of the key points discussed in the document and offers recommendations for improving the record-keeping system. It emphasizes that the system should be regularly reviewed and updated to reflect changes in the financial environment and to ensure that it remains effective and efficient.



but does not give any information about the internal appearance of the object such as how much light it reflects or what surface markings it has. A grey level image is required rather than a binary image otherwise too much information would be lost. This makes recognition of geometric features much more difficult. A line in a grey level image is much more difficult to pinpoint than it would be in a binary image.

Rather than have a geometric description of a component it was decided to use the tests themselves to describe the components. It is more of an implicit description than that given by a geometric description. The idea is to have a label for a particular feature that might belong to a component and then associate a number of tests with it. These tests are used to identify the presence of the feature. Using labels for features allows new features to be modelled by using different combinations of tests. The test can be considered as primitives that model the features.

### 3.2 Model Data

The data required to describe a circuit board well enough for inspection purposes has been discussed above. The position and composition of a component is required. What form should this be supplied in? The aim in defining a language to describe models is not only to get the required information but also to allow the information to be described in a concise form and without too much outlay in effort for the person describing the circuit. There are a number of points which should be considered to help with producing an easily used language:

- **Many Components:** There can be a wide diversity of components on a particular circuit board. Therefore it would be sensible to consider using a library of components for defining new circuit board models. It would save having to redefine components for new circuit boards when they have already been used previously.
- **Repeated Components:** A certain component does not necessarily appear only once on the circuit board. It would be tedious to make two or more definitions for a component just because it appears in more than one place on the board. It is much better to define it once and just reference the definition each time it is required.
- **Subcomponents:** A particular component might consist of different parts joined together. An example of this might be an integrated circuit. This has a body and pins. For such an occurrence it would be tedious to design it all as one component consisting of a definition of the body plus a number of pins. This would lead to repetition during the definition of the pins. It would be sensible to allow a component to be described in terms of features and also of other subcomponents.



Repeated components within a circuit are coped with in two ways. If there are a number of the same component which are positioned at unrelated places round the circuit then it is desirable to set up a single definition and use references to that definition within the model file. The second possibility is that a component may appear at regularly spaced intervals in columns or rows. In this case it would be sensible to define where the first of the components appear and then give the spacing between the components.

The two forms of repeated components discussed above are implemented at the top level in a model description. The top level describes what components the circuit board contains. The example below shows how the components can be described.

```
CIRCUIT <name>
WITH
    COMPONENT <definition_name> <instance_name>
        POSITION(x,y)
        ORIENT(angle)

    REPEAT <number_of_times> STEP <interval> ALONG <axis> FOR
        COMPONENT <definition_name> <instance_name>
            POSITION(x,y)
            ORIENT(angle)

    ENDREPEAT

ENDCIRCUIT
```

The first entry in the circuit definition is a reference to a component that must be defined elsewhere in the model file. A position and orientation is given to show where the component is to appear on the circuit board. The positioning of the component is in relation to an imaginary reference point on the circuit board. This reference point is usually taken to be at the top left point of the circuit board with a zero orientation being parallel to the x-axis. Each component has a similar reference point. This means that the position and orientation of a component is where the component's reference point will be placed in relation to the circuit's reference point.

The second entry of the circuit definition shows how to define a component that appears in regularly spaced intervals along one axis. The number of components to be positioned, the spacing between them and the axis along which they appear are all given. The reference to the component below that is used to show where the first component should appear. The others can be found by adding the interval value to the starting point. This method for defining a row of the same component is much easier than having to explicitly mention each component's position.

The first part of the document discusses the importance of maintaining accurate records of all transactions. It emphasizes that proper record-keeping is essential for the integrity of the financial system and for the ability to detect and prevent fraud. The text also mentions the need for regular audits and the role of internal controls in ensuring the reliability of financial data.

The second part of the document focuses on the role of the accounting profession. It highlights the need for accountants to adhere to high standards of ethical conduct and to maintain their professional competence through continuous education and training. The text also discusses the importance of transparency and accountability in the financial reporting process.

The third part of the document addresses the challenges faced by businesses in the current economic environment. It discusses the impact of global economic uncertainty and the need for businesses to adapt to changing market conditions. The text also mentions the importance of innovation and the role of technology in driving business growth and efficiency.

The fourth part of the document discusses the role of government in the financial system. It highlights the need for strong regulatory frameworks and the importance of government oversight in ensuring the stability and integrity of the financial system. The text also mentions the role of government in promoting economic growth and development.

The fifth part of the document discusses the role of investors in the financial system. It highlights the need for investors to conduct thorough due diligence and to make informed investment decisions. The text also mentions the importance of diversification and the role of investors in driving innovation and growth in the economy.

The sixth part of the document discusses the role of financial institutions in the financial system. It highlights the need for financial institutions to maintain high standards of risk management and to provide sound financial advice to their clients. The text also mentions the importance of financial institutions in promoting economic growth and development.

The seventh part of the document discusses the role of the media in the financial system. It highlights the need for the media to provide accurate and unbiased reporting on financial matters. The text also mentions the importance of the media in promoting transparency and accountability in the financial system.

The eighth part of the document discusses the role of the public in the financial system. It highlights the need for the public to be informed and to participate in the financial system. The text also mentions the importance of the public in promoting economic growth and development.

The ninth part of the document discusses the role of the future in the financial system. It highlights the need for the financial system to be resilient and to adapt to changing market conditions. The text also mentions the importance of innovation and the role of technology in driving business growth and efficiency.

The tenth part of the document discusses the role of the conclusion in the financial system. It highlights the need for the financial system to be transparent and accountable. The text also mentions the importance of the financial system in promoting economic growth and development.

So far only the components that appear on the circuit board have been positioned. It is now necessary to define what these components are. A component definition is made up of features which can be in one of three forms. Two of the forms are the same as those that can appear in a circuit definition. A component can appear in the definition of another component. This is known as a subcomponent. A subcomponent can also appear inside a repeat statement.

The third form of feature that can make up a component is a predefined feature. There are four types of feature in the component library at the moment:

- ICbody
- IClegs
- RESISTOR
- CAPACITOR

These types of feature are used as labels to which groups of tests can be associated. The features themselves have no meaning to the program. A name, window and position are associated with the feature. The name is used for the purposes of user feedback. The window is defined as having a length and a width and corresponds to area of the image which will be fed to the tests during the inspection phase. The position shows where the window will be placed. It has both a co-ordinate origin and an orientation. These are relative to the component in which the feature is defined. Finally we have some number of tests which are associated with the feature. These are used to determine whether the feature is present within the window in an undamaged form. The description of the feature is effectively tied up in the definition of the tests. These tests are chosen by the person making up the model file. A feature might appear as below:

```
IClegs: iclegs1 WINDOW(135,15) AT(0,0,0)
        PINCOUNT(135,15,8,5)
```

This defines a window starting at the top left hand corner of the component. The test PINCOUNT is associated with the feature *iclegs1* and looks for eight pins with a width of at least five units. The unit length will depend upon the scaling factor for converting from model co-ordinates to screen pixels.

The overall form of a component would consist of some combination of the three features:

...the ... of ...

...the ... of ...

...the ... of ...

...the ... of ...

...the ... of ...

...the ... of ...

...the ... of ...

...the ... of ...

...the ... of ...

```

COMPONENT <definition_name>
WITH
    COMPONENT <definition_name> <instance_name>
        POSITION(x,y)
        ORIENT(angle)

    REPEAT <number_of_times> STEP <interval> ALONG <axis> FOR
        COMPONENT <definition_name> <instance_name>
            POSITION(x,y)
            ORIENT(angle)
    ENDREPEAT

<feature_type>: <name> WINDOW(length,width) AT(x,y,angle)
                test_1(parameters)
                :
                test_n(parameters)

ENDCOMPONENT

```

### 3.3 Error Recovery

The language described above has been implemented using YACC and LEX. The BNF of the grammar is shown in appendix A. At present there is no provision for error recovery. When a syntax error is encountered during the parsing of a model then the program will stop. The data being parsed is displayed during the parsing process and so when an error occurs the place where the problem appears will be shown on the screen. A small amount of semantic checking is carried out during the parsing. This involves a repeat statement where at least one component must appear, and it must be either along the x or y axis. The only other checking done is for the number of parameters given to a feature test. If the correct number is not given then the program will report this and stop.

### 3.4 Extending the Model

Two extensions to the modelling capabilities were considered. One simplifies the creation of models whereas the other increases the power of the system.

THE EFFECTS OF PRACTICE ON THE LEARNING OF A MOTOR SKILL

1952

The purpose of this study was to determine the effects of practice on the learning of a motor skill. The subjects were 20 college students who were divided into two groups of 10. One group practiced the skill for 10 days, while the other group practiced for 20 days. The results showed that the group that practiced for 20 days performed significantly better than the group that practiced for 10 days.

The results of this study indicate that practice has a significant effect on the learning of a motor skill. The group that practiced for 20 days performed significantly better than the group that practiced for 10 days. This suggests that more practice leads to better performance.

The study also found that the rate of improvement was higher in the first few days of practice and then leveled off. This is consistent with the idea of a learning curve, where the rate of learning is highest at the beginning and then decreases as the skill is mastered.

These findings have important implications for the design of training programs. It suggests that providing more practice opportunities can lead to better performance. Additionally, it highlights the importance of the initial stages of practice, as this is when the most rapid improvement occurs.

The study was limited to a single motor skill and a short period of practice. Future research could explore the effects of practice on other motor skills and over a longer period of time. It would also be interesting to investigate the effects of different types of practice, such as spaced practice versus massed practice.

In conclusion, this study demonstrates that practice significantly improves performance on a motor skill. The group that practiced for 20 days performed significantly better than the group that practiced for 10 days. These findings support the idea that more practice leads to better performance.



### 3.4.1 Associating Tests With Features

As it stands at the moment, in order to create a model file it is necessary for the user to know what inspection tests are required for each component that is used. It is up to the user to put the appropriate tests in. Rather than giving tests it would be better if the user could give a description of the features. These descriptions could be formed from functions relating to the features, such as SIZE(a,b) denoting the length and width of the feature. This allows the user to give a more intuitive description. The parameters of these functions could be extracted and then given as arguments to a specific batch of tests associated with the particular feature. A definition might look like the following:

```
COMPONENT chip
WITH
    ICbody: ic1 WINDOW(length,width) AT(x,y,angle)
           SIZE(a,b)

    IClegs: p1 WINDOW(length,width) AT(x,y,angle)
           NUMBER(8)

ENDCOMPONENT
```

This is not too difficult to achieve because the features, ICbody etc. are effectively labels to which tests are associated. Rather than having the user specify them for the features this extension would add the tests on before hand. The parameters of the functions describing the features would then be given to the tests.

### 3.4.2 Pairwise Tests

An extra set of tests could be implemented which would allow comparisons between features. These might test the separation or relative orientation between features in the image. This would require standard parameters to be passed back such as the position and orientation of the feature. These could be stored away ready for pairwise comparisons to be carried out. A definition might now look like the following:

```
COMPONENT chip
WITH
    IClegs: p1 WINDOW(135,15) AT(0,0,0)
           PINCOUNT(135,15,8,5)
```

Faint, illegible text in the upper section of the page, possibly containing a title or introductory paragraph.

Main body of faint, illegible text, likely the primary content of the document.

Page 1 of 1

```
IClegs: p2 WINDOW(135,15) AT(135,50,180)
        PINCOUNT(135,15,8,5)
```

```
COMPARE
        RELORIENT(p1,p2,180,2)

ENDCOMPONENT
```

This describes a component which consists of two rows of IC pins. A pairwise test is made between these features which is used to compare the orientation of them. They should be 180° apart within a tolerance value of 2°.

Some standard tests would have to be applied to all the features regardless of what tests would normally be associated with them. These could be hidden away using the method discussed in the first extension (3.4.1).

### 3.5 Example Model

To show how a model file looks an example is shown in Figure 4 for the image in Figure 3. Four integrated circuits are modelled along with two resistors. The image has these components outlined to show what areas of the screen the inspection process actually deals with.

The resistors appear horizontally in the image and so the model file reflects this by positioning the parts as they are with no orientation. The chips, however, are shown rotated round by 90°. The model file also shows this (Figure 4). The four chips are grouped together into one unit. This unit is rotated through 90° so that as each chip will be correctly orientated when positioned. A repeat statement is used to define the positioning for the four chips. The first starts off at (15,200) and there is a gap of 60 units between each. The chips are positioned along the x-axis of the component *chips*. The individual chips are defined to consist of two rows of pins, i.e. two IClegs features. One feature is rotated through 180° in relation to the other. This is so that the window given to the inspection mechanism will have the pins pointing towards the bottom of the window, away from the body of the chip.

...the ... of ...  
...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...  
...the ... of ...

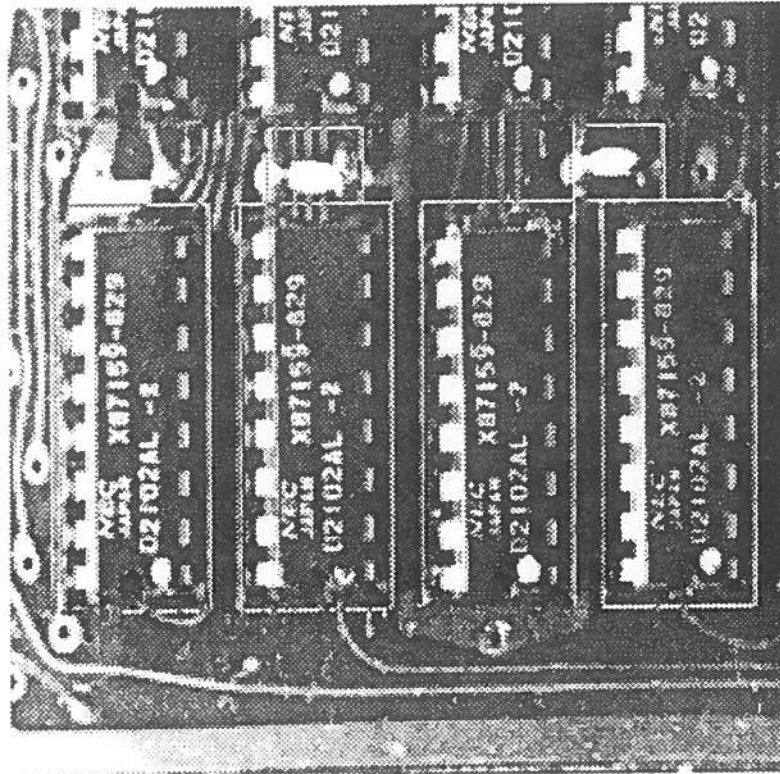
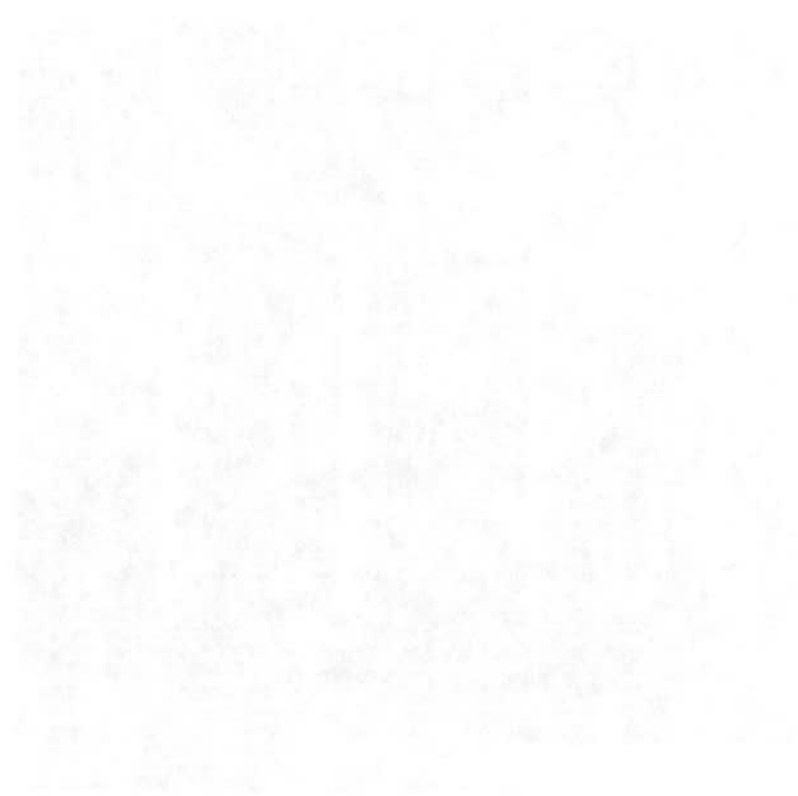


Figure 4: A Digitized Image of a Circuit Board With Components Outlined



```

CIRCUIT memory
WITH
    COMPONENT chips memchip
        POSITION(15,200)
        ORIENT(0)

    COMPONENT resistors res1
        POSITION(90,40)
        ORIENT(0)
ENDCIRCUIT

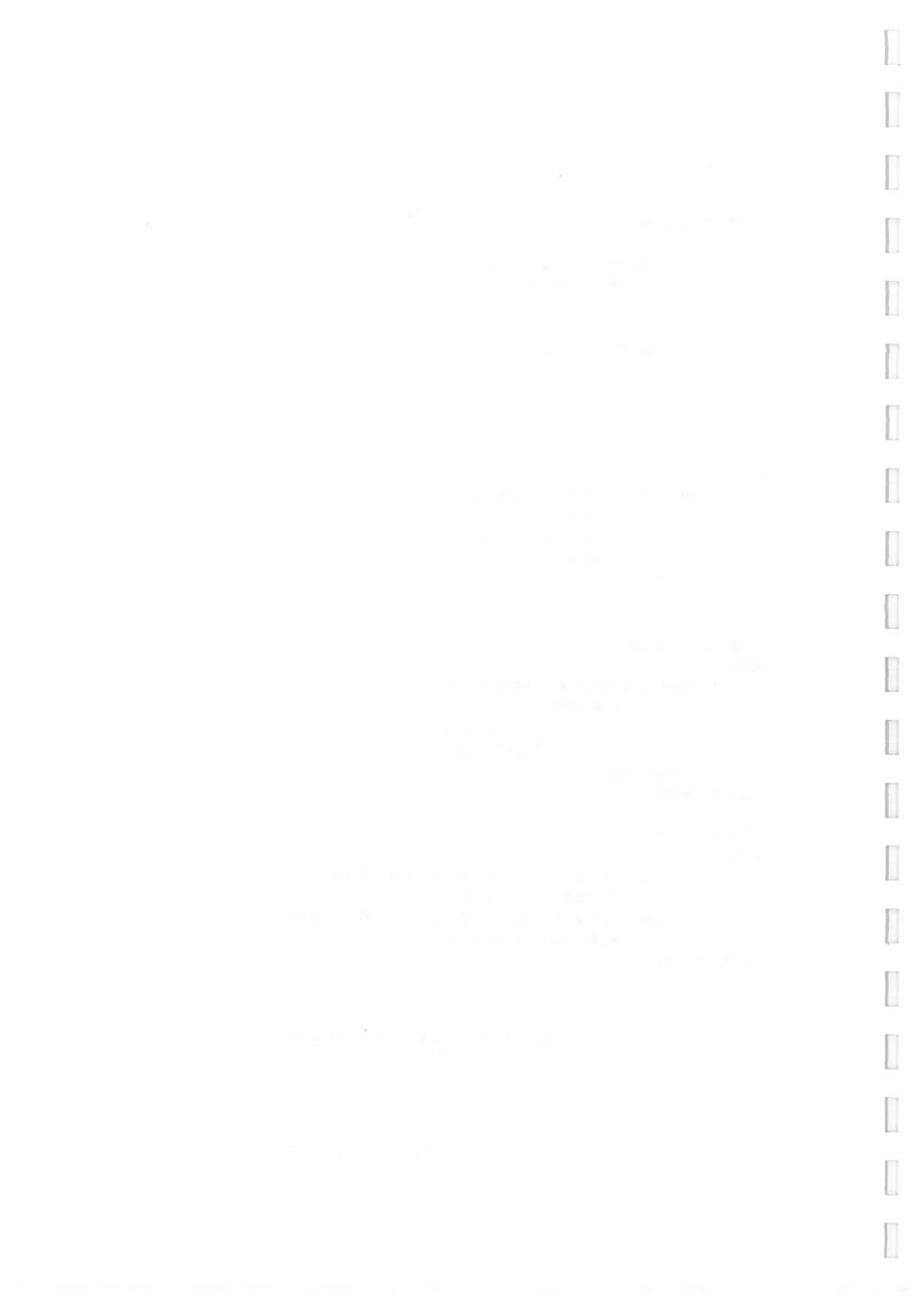
COMPONENT resistors
WITH
    RESISTOR: r1 WINDOW(30,25) AT(-5,0,0)
        LENGTH(20,5)
    RESISTOR: r2 WINDOW(30,25) AT(95,0,0)
        LENGTH(20,5)
ENDCOMPONENT

COMPONENT chips
WITH
    REPEAT 4 STEP 60 ALONG x FOR
        COMPONENT chip ic
            POSITION(0,0)
            ORIENT(-90)
    ENDREPEAT
ENDCOMPONENT

COMPONENT chip
WITH
    IClegs: iclegs1 WINDOW(135,15) AT(0,0,0)
        PINCOUNT(135,15,8,5)
    IClegs: iclegs2 WINDOW(135,15) AT(135,50,180)
        PINCOUNT(135,15,8,5)
ENDCOMPONENT
END

```

Figure 4: Model Definition for Circuit





## 4 Specific Tests

The inspection process is where the user sees the bulk of the work being done. All the previous work has been done to set up a framework ready for applying the actual inspections. The choice of tests to apply depends on what the user chooses. These are given in the model file and are associated with each feature that is to be inspected.

Each of the tests are independent of the others. No information is passed between them about the form of the feature being tested. This means that it is up to the user to give a sensible batch of tests to apply. Each of the tests should be designed to identify the presence of particular traits in the feature being inspected. These can vary widely from simple ones that might measure the length of some body to those that do a more complex analysis of the image such as finding the positions and counting the number of features.

The data that each test has to work with has a standardised form. For each feature there is an associated window within which it should be contained. The test requires only this area of the image plus any parameters that are necessary to describe the feature that it is trying to inspect. The parameters might describe the feature's length, orientation or relative illumination compared to its background. These parameters are test specific.

The success or failure of the tests shows whether a feature has been identified, found to be in its correct position and undamaged. This is why it is important to have a sensible choice of tests that will cover all eventualities that might result from incorrect assembly. Incorrect assembly can take the form of badly positioned or missing parts. Handling during assembly might also cause parts to be damaged. Failing a test cannot be used to identify the cause of the failure. A test may fail for any of the three reasons mentioned above: bad positioning, missing part and damaged part. The number of tests failed for a feature being inspected can be used to judge the cause but only to give a rough indication. The smaller the number of tests failed then the more likely that the part is only damaged rather than missing. A badly positioned part may pass tests that depend simply upon presence as compared to position within the window. The final result of identifying the cause of tests failing has to be left for the user to decide.

The window being tested is highlighted when using the display option. The window is also shown in the upper left hand corner of the screen. This addition allows the test routine to add any extra information to show how it is applying its test such as a scan line along which it is making some examination. Each of the tests are also left to identify the reason for the particular test failing. This removes the necessity for the main program to understand the workings of the tests. Both of these features, display access and error

Section 10

The first part of the document discusses the importance of maintaining accurate records of all transactions. It emphasizes that every entry should be supported by a valid receipt or invoice to ensure transparency and accountability.

Furthermore, it is noted that the records should be kept in a secure and accessible format, such as digital spreadsheets or specialized accounting software, to facilitate easy review and analysis.

The document also highlights the need for regular audits to verify the accuracy of the recorded data. This process helps in identifying any discrepancies or errors early on, allowing for prompt corrections and ensuring the integrity of the financial information.

In addition, it is stressed that all personnel involved in the financial operations should be trained and aware of the proper procedures for recording and reporting. This ensures consistency and adherence to the established protocols.

The final section of the document provides a summary of the key points discussed. It reiterates the importance of diligent record-keeping and the role of regular audits in maintaining the accuracy and reliability of the financial records.

Overall, the document serves as a comprehensive guide for anyone responsible for managing financial data. It offers practical advice and best practices to ensure that all financial activities are properly documented and reported.

By following the guidelines outlined in this document, organizations can ensure that their financial records are accurate, complete, and compliant with all relevant regulations and standards.

This document is intended to provide a clear and concise overview of the requirements for financial record-keeping. It is not intended to constitute professional advice, and users should consult with their respective accountants or legal advisors for more detailed information.

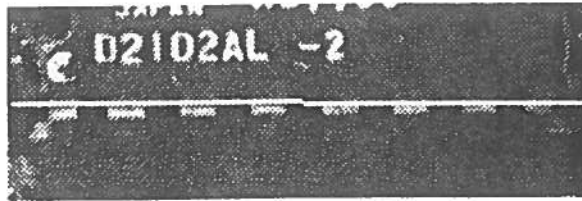


Figure 5: View of IC Pins

feedback, are in keeping with the modularity of the program. It is easy to interchange the tests without affecting other parts of the program. To show how well the overall inspection process managed, a final display is shown with all the features that failed tests outlined. This allows the user to see at a glance how well everything has proceeded.

A test plan is outlined in section 5.2 for evaluating the effectiveness of the inspection process.

#### 4.1 Pin Test

This particular test is designed to identify the number of pins that an IC has and also that none of them have been bent. This is really two separate tests but the intermediate data created doing one test is also used by the other test. It was therefore decided to combine them. To get a suitable method for achieving this it is necessary to look at an image corresponding to the pins of an IC. Figure 5 shows a view of eight pins on an integrated circuit.

If we take an intensity profile of a scanline travelling through all of the pins then it is clearly visible that the peaks in intensity are coincident with the pins (Figure 6). Figure 5 has a line down the centre of the image showing where the scanline is. The metal pins reflect much more light than the surroundings and so it would seem that counting the pins would involve finding peaks in the intensity. It is not quite that simple. The background behind the pins can be picked up as well as the pins. If there is any metal or other reflective material then that can cause unwanted peaks in intensity. To stop this fooling the count it is necessary to look for consecutive high values in the intensity profile. At least five consecutive units of a high intensity was used as a guide when testing the program. This is scaled into screen pixels. For five units with a scaling factor of two this would be ten pixels. The value can be varied at the discretion of the user by altering the model file.

As well as deciding how many consecutive pixels to look for it is also necessary to decide what value constitutes a peak in the intensity. It was decided to find the average intensity



[Illegible Title]

[Illegible text block 1]

[Illegible text block 2]

[Illegible text block 3]

[Illegible text block 4]

[Illegible text block 5]

[Illegible text block 6]

[Illegible text block 7]

[Illegible text block 8]

[Illegible text block 9]

[Illegible text block 10]

[Illegible text block 11]

[Illegible text block 12]

[Illegible text block 13]

[Illegible text block 14]

[Illegible text block 15]

[Illegible text block 16]

[Illegible text block 17]

[Illegible text block 18]

[Illegible text block 19]

[Illegible text block 20]

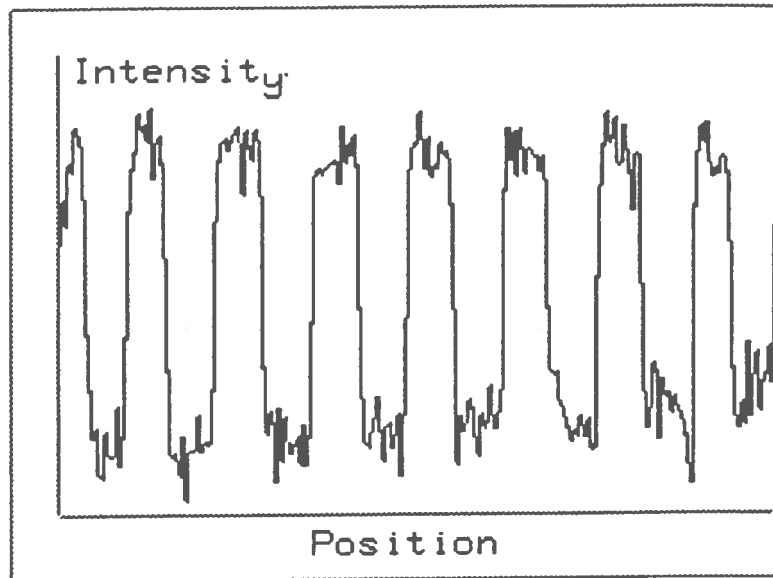


Figure 6: Intensity Profile Across Pins

along the scanline and add a small value on to that to give a threshold. The reason for using a threshold above the average is to try and stop noise from affecting the results. Since the pins are made of metal then they will reflect much more light and so will show as peaks well above the average rather than just above the average. A value of ten is added on to the average to give the threshold. This adds between seven and twelve percent on to the threshold depending upon the average intensity across the scan line. It is not necessary to allow this value to be changed since a different value would only be of advantage at either very low or very high average intensities. The high and low intensity profiles would be produced by either too much light or too little. In these conditions it would be too difficult to produce a reliable inspection process anyway.

Counting the pins now involves scanning across the window looking for five or more consecutive pixels above the threshold value. Each time such a peak<sup>1</sup> in intensity has been found a counter can be incremented.

Counting the pins is only half of the inspection process that is carried out. The other half involves checking to see that no pins are bent. This is a bit of a limiting case because there are many ways in which a pin could be bent. If a pin is bent back on itself then this

---

<sup>1</sup>A peak must have a beginning and an end. Therefore the counter is only incremented once the intensity drops below the threshold. This stops spurious peaks at the edges of the image window being picked up.

The first part of the report discusses the importance of maintaining accurate records of all transactions. It emphasizes that proper record-keeping is essential for the success of any business and for the protection of the interests of all parties involved.

The second part of the report describes the various methods used to collect and analyze data. It details the procedures followed to ensure the reliability and validity of the information gathered, and discusses the challenges encountered during the process.

The third part of the report presents the findings of the study. It provides a comprehensive overview of the results, highlighting the key trends and patterns observed. The data is presented in a clear and concise manner, using tables and graphs to illustrate the findings.

The final part of the report discusses the implications of the findings and offers recommendations for future research. It provides a detailed analysis of the strengths and weaknesses of the study, and suggests ways in which the results can be applied to improve business performance.

In conclusion, this report provides a thorough and detailed analysis of the data collected. It highlights the importance of accurate record-keeping and offers valuable insights into the various methods used to collect and analyze data. The findings of the study are presented in a clear and concise manner, and the implications of the results are discussed in detail.

The report is a valuable resource for anyone interested in the field of business and data analysis. It provides a comprehensive overview of the current state of the field, and offers a detailed analysis of the challenges and opportunities facing businesses today.

The findings of the study are particularly relevant in the current economic environment, where businesses are facing increasing competition and pressure to improve their performance. The report offers a range of practical recommendations that can be used to address these challenges and improve business outcomes.

Overall, this report is a well-written and informative study that provides a wealth of valuable information. It is a must-read for anyone looking to gain a better understanding of the current state of the business world and the role of data in its success.

cannot be seen using any inspection method viewing from above. A pin bent straight out can be spotted. This is what is being looked for with this test. This is done by making two scans across the window. The first of these will be along where the pins should be. The second scan will be further out from the IC at the edge of the window. If there is a matching peak at the same position during both scans then it is assumed that a bent pin has been discovered.

To find matches it is necessary to store the positions in which pins are found. For each intensity peak found in the second scan there could be a possibility that it is a part of a bent pin. If there is a peak of the same size and in the same position in the first scan then it is assumed that a bent pin has been found. If the second scan was carried out part of the way along a track in the board then this would produce an intensity peak that should be ignored. Even though there may be a matching intensity peak at a pin position it is caused by an object that is much wider than a pin and so will be discarded.

A simple extension to the pin test would improve its reliability. Using a straight line threshold copes with the ideal lighting situation where the intensity gradient across the image is negligible. If there is a large intensity gradient, which may be caused by a single light source close to the circuit board, then the present pin test will fail to work properly. To remedy this a least squares fit straight line could be used in place of the threshold value.

Faint, illegible text at the top of the page, possibly a header or introductory paragraph.

Main body of faint, illegible text, appearing to be several paragraphs of a document.





## 5 Test Plan

Once a working system has been developed it is necessary to comprehensively test the system. There now follows a test plan to achieve this aim.

There are two parts in the system which must be tested. The first of these is the model loader. It must be shown that the model data is correctly interpreted. The second part of the testing is to check that the specific inspection processes are carried out correctly.

### 5.1 Modelling Test Plan

Testing the model interpretation requires that the parsing of the model and subsequent modification of the parse tree be examined. This testing is aided by the inclusion of a run-time flag that will display the components within the circuit and how they are related. Looking at this output it is possible to see that the components have been properly linked together according to the model definition.

It is now necessary to define a series of models that will test the model loader. To do this each of the productions that define the grammar of the language for model descriptions must be tested. The grammar is shown in appendix A. The points that have to be considered in the grammar testing are discussed below in section 5.1.1. Example test models that cover the points discussed are given in section 5.1.2.

#### 5.1.1 Testing The Model Loading

The top level production

$$\text{grammar} ::= \text{list END} \tag{1}$$

has only the one possible interpretation and so displays its validity every time that a model is parsed.

$$\begin{array}{l} \text{list} ::= \text{circuit} \\ \qquad \text{component\_def} \\ \qquad \text{circuit list} \\ \qquad \text{component\_def list} \end{array} \tag{2}$$

The first part of the document discusses the importance of maintaining accurate records of all transactions. It emphasizes that every entry should be supported by a valid receipt or invoice. This ensures transparency and allows for easy verification of the data.

Furthermore, it is crucial to review the records regularly to identify any discrepancies or errors. This proactive approach helps in catching mistakes early and prevents them from escalating into larger issues. Consistent auditing is a key component of a robust financial management system.

In addition, the document highlights the need for clear communication between all parties involved. Regular meetings and reports should be used to keep everyone informed about the current status and any changes that may affect the records. This collaborative effort is essential for the success of the project.

Finally, the document concludes by stating that maintaining accurate records is not just a task, but a responsibility. It is the foundation upon which all other financial decisions are made. By adhering to these guidelines, the organization can ensure its financial health and long-term sustainability.

The following table provides a summary of the key points discussed in the document.

Key Point	Description
Record Accuracy	Ensure all transactions are supported by valid receipts.
Regular Review	Conduct regular audits to identify and correct errors.
Clear Communication	Use meetings and reports to keep all parties informed.

defines a list to contain a mixture of `circuit` and `component_def`, i.e. a mixture of circuit and component definitions. This can be tested by using combinations of more than one, one and zero of these instances. Zero instances should report an error since there will be insufficient information for creating a model.

```
circuit ::= CIRCUIT NAME WITH component_list ENDCIRCUIT |
          CIRCUIT NAME WITH component_list COMPARE
          inspection_list ENDCIRCUIT
```

(3)

Pairwise comparison tests have not yet been implemented but models can be created containing them. Therefore to test this production a dummy test has to be created that will inform the user that the test has executed. It will not actually carry out any inspections.

```
component_list ::= repeatloop component ENDREPEAT
                 repeatloop component ENDREPEAT component_list
                 component
                 component component_list
```

(4)

defines that a `component_list` can consists of one or more components and one or more components defined inside a repeat loop. Again this should be tested with zero, one or more instances of each type.

```
repeatloop ::= REPEAT value STEP value ALONG NAME FOR
```

(5)

has three user defined parameters. The first value defines how many times the component should appear. The second value gives the spacing between components and the `NAME` shows which axis the components should be placed along. An error results if the axis name given does not begin with x or y.

The definition of `component` uses two names in its definition.

```
component ::= COMPONENT NAME NAME posn
```

(6)

1. The first part of the document discusses the importance of maintaining accurate records of all transactions. This is essential for ensuring the integrity of the financial statements and for providing a clear audit trail.

2. The second part of the document outlines the various methods used to collect and analyze data. These methods include interviews, surveys, and focus groups, each of which has its own strengths and limitations.

3. The third part of the document describes the process of data analysis, which involves identifying patterns and trends in the data. This is a complex task that requires a high level of statistical expertise.

4. The fourth part of the document discusses the importance of communication in the research process. Researchers must be able to clearly and concisely communicate their findings to a wide range of stakeholders.

5. The fifth part of the document concludes by emphasizing the need for ongoing evaluation and improvement of the research process. This is a continuous process that requires a commitment to excellence and a willingness to learn from experience.

6. The sixth part of the document provides a detailed overview of the research methodology used in this study. This includes a description of the study design, the data collection methods, and the data analysis techniques.

7. The seventh part of the document presents the results of the study. These results are presented in a clear and concise manner, with a focus on the key findings and their implications for practice.

8. The eighth part of the document discusses the limitations of the study and the need for further research. This is an important part of the research process that helps to identify areas for future investigation.

The first name relates to the component definition name. The second relates to the instance name. The component definition name is used to link the component instance with its definition since this is the only common factor between them. Correct usage can be shown by each component instance having the correct tests applied to it, i.e. the link between component instance and definition has been made correctly.

To show that the next two productions work as expected then it is necessary to see that the parameters read in are stored and used correctly.

```

posn      ::= POSITION(value,value)
           ORIENT(value)
window    ::= WINDOW(value,value) AT(value,value,value)

```

(7)

A model must be set up with appropriate parameter values so that the program will display a predictable action.

```

component_def ::= COMPONENT NAME WITH feature_list
               ENDCOMPONENT
               COMPONENT NAME WITH feature_list
               COMPARE inspection_list ENDCOMPONENT

```

(8)

This definition again requires the use of a dummy test for the same reasons as production 3 to show that the list of pairwise tests *inspection\_list* is properly utilised.

The following two productions all have the same form:

```

feature_list ::= feature
              feature feature_list
inspection_list ::= inspection
                inspection inspection_list

```

(9)

Testing these requires that zero, one or more instances of the items making up the list are included in the model file. Zero instances should not be accepted but any other number should be handled correctly.

A feature in a feature list is defined as the following:

1. The first part of the document discusses the importance of maintaining accurate records of all transactions and activities. It emphasizes that this is crucial for ensuring transparency and accountability in the organization's operations.

2. The second part of the document outlines the various methods and tools used to collect and analyze data. It highlights the need for consistent and reliable data collection processes to support effective decision-making.

3. The third part of the document focuses on the role of technology in data management and analysis. It discusses how modern software solutions can streamline data collection, storage, and reporting, thereby improving efficiency and accuracy.

4. The fourth part of the document addresses the challenges associated with data management, such as data quality, security, and privacy. It provides strategies to mitigate these risks and ensure that data is used responsibly and ethically.

5. The fifth part of the document concludes by summarizing the key findings and recommendations. It stresses the importance of ongoing monitoring and evaluation to ensure that data management practices remain effective and aligned with the organization's goals.

6. The sixth part of the document provides a detailed overview of the data collection process, including the identification of data sources, the design of data collection instruments, and the implementation of data collection procedures.

7. The seventh part of the document discusses the various methods used for data analysis, such as descriptive statistics, inferential statistics, and qualitative analysis. It explains how these methods are used to interpret the data and draw meaningful conclusions.

8. The eighth part of the document focuses on the importance of data visualization in presenting complex information in a clear and concise manner. It discusses various visualization techniques and their applications in data analysis.

9. The ninth part of the document addresses the ethical considerations surrounding data management and analysis. It discusses the need for transparency, accountability, and respect for individual privacy and data rights.

10. The tenth part of the document provides a final summary and concludes the report. It reiterates the key findings and emphasizes the importance of data management and analysis in achieving organizational success.

```

feature ::= FEATURE : NAME window inspection_list |
             repeatloop component ENDREPEAT |
             component

```

(10)

There are three possibilities for a feature. It can be a predefined feature, a component within a repeat statement or a component.

Each predefined feature has a type *FEATURE* and a name associated with it. It is positioned according to the specification *window* and has a list of tests after it (*inspection\_list*). It is necessary to check that the correct inspection tests are applied for the appropriate feature.

The component appearing as a feature can be tested by defining a subcomponent to appear inside another component. A component inside a repeat statement can be tested in the same way as a subcomponent except that there should be the designated number tested.

```

inspection ::= TYPE (parm_list)

```

(11)

defines either a feature test plus a number of parameters (10) or a pairwise test plus parameters (8) & (3). It is necessary to show that the correct parameters are passed to the particular test. The parameters can consist of a mixture of numbers and names. At present only the pairwise comparison tests would take names in the parameter list. Checking to see that the correct parameters are used with the test would require that the test print them out. This was done at the early development stage but has now been removed.

```

parm_list ::= value |
              value ' ' parm_list |
              NAME |
              NAME ' ' parm_list

```

(12)

Numbers can be either integers or floating point. They are stored as floating point so that the individual tests can use either form.

THE UNIVERSITY OF CHICAGO  
DEPARTMENT OF POLITICAL SCIENCE  
1100 EAST 58TH STREET  
CHICAGO, ILLINOIS 60637

Dear Mr. [Name]:

I am writing to you regarding the [Topic] that you mentioned in your letter of [Date]. I have reviewed the information you provided and I am pleased to hear that you are interested in [Topic].

[Detailed paragraph of text, mostly illegible]

I am sure that you will find this information helpful. If you have any further questions, please do not hesitate to contact me at [Phone Number] or [Email Address].

Sincerely,  
[Name]

[Large block of illegible text, possibly a second letter or a very faded one]

[Final block of illegible text]



*value* ::= *INTEGER* |  
*FLOAT*

(13)

### 5.1.2 Example Test Models

There are two stages required for testing the model grammar. The first of these is to check that invalid model files will not be accepted. The second stage is to check that valid model files are correctly stored.

The type of invalid model files that will be discussed here cover the following features of the grammar:

1. Where a list of items is expected, none are given
  2. Missing parameters given to inspection tests
  3. Invalid feature types used
  4. Missing component and circuit definitions
1. There are four cases where a list of items is expected. These involve the productions at 2, 4, 9 and 12. To show that these invalid cases are not accepted by the parser the following sections of model files could be included in a syntactically correct model file such as the one shown in figure 5.

The first model is simply just the keyword **END**. This tests production rule 2.

**CIRCUIT** memory **WITH** **ENDCIRCUIT**

There is no list of components on the circuit board and so production 4 cannot accept this.

# Appendix 1

The following table provides a summary of the data collected during the field study. The data is organized into three main sections: General Information, Environmental Data, and Biological Data. Each section contains a list of variables measured and the methods used for data collection.

Section	Variable	Method
General Information	Date	Field notes
	Time	Field notes
	Location	GPS
	Observer	Field notes
	Weather	Weather station
Environmental Data	Temperature	Thermometer
	Humidity	Humidity gauge
	Wind speed	Anemometer
	Light intensity	Light meter
	Soil moisture	Soil moisture probe
Biological Data	Plant species	Field observations
	Animal species	Field observations
	Plant height	Measuring tape
	Plant biomass	Scale
	Plant density	Quadrat sampling

The data collected during the field study was analyzed using statistical software. The results of the analysis are presented in the following sections. The first section discusses the overall trends in the data, while the second section provides a detailed analysis of the individual variables. The third section discusses the implications of the findings for the study of the ecosystem.

- COMPONENT memory WITH ENDCOMPONENT
- COMPONENT memory WITH feature\_list COMPARE ENDCOMPONENT
- IClegs: p1 WINDOW(10,10) AT(0,0,0)

These are examples of a missing feature list for a component, missing test list for a pairwise comparison, and missing tests for a predefined feature (see production 9).

```
IClegs: p1 WINDOW(10,10) AT(0,0,0)
PINCOUNT()
```

This is an example of an empty parameter list which would not get past the parser. It is a slightly different case from missing parameters because just so long as there is at least one parameter then the parameter list will parse (production 12).

2. Missing parameters are checked for within production (11). Each type of test has a requirement for a certain number of parameters. This is checked against the actual number given before the parsing can continue.
3. There are four types of predefined feature: ICbody, IClegs, RESISTOR and CAPACITOR. Only these names can appear in the position designated for a predefined feature. The same follows for the test types.
4. Missing component and circuit definitions are not coped with at parse time. They can only be checked for once the model has been loaded in. The program will then look for the circuit definition that it is to use. If this is not found then an error is given. The same is done for each component that appears within the circuit definition. If one of these does not exist then an error is given. Since subcomponents can appear within components then for each component that appears in the circuit, all of its subcomponents must also exist. A model file such as the one shown in figure 5 will show what happens for missing definitions. Each component definition can in turn be removed from the file to show that its removal will trigger an error. The same can be done for the circuit definition.

Once the above testing has taken place it is necessary to look at how the inspection process uses a syntactically correct model definition. The following points are taken into consideration at this stage:

1. A correct circuit definition is specified if the circuit exists within model definition.
2. Components are correctly related to other components as subcomponents.



3. Each feature has the correct tests applied to it.
  4. The positioning of the components appear as the model says they should.
  5. Repeat statements work correctly.
  6. Recursively defined components work correctly.
1. An optional check flag can be given when invoking the program (see appendix B). This will display which circuit is being inspected. It should be checked when there is one or more circuit definitions in the model.
  2. The check flag will also invoke the program to display the component names in a roughly tree-like structure. This allows the user to see that all the components, subcomponents and features belong to the correct component definitions. The top level components (those that appear in the circuit definition) are shown leftmost in the output. If the model shown in figure 5 is used with the check flag on then the output would look like the following:

Circuit part list for memory:

```

memchip
    ic-a
        iclegs1
        iclegs2
    ic-b
        iclegs1
        iclegs2
    ic-c
        iclegs1
        iclegs2
    ic-d
        iclegs1
        iclegs2
res1
    r1
    r2

```

3. Each feature should have the correct tests applied to it. As the inspection process proceeds each feature name is displayed along with all the tests that are applied to it. This will show that production (10) is applied correctly along with the production

1. The first part of the document discusses the importance of maintaining accurate records of all transactions and activities. It emphasizes that this is essential for ensuring transparency and accountability in the organization's operations.

2. The second part of the document outlines the various methods and tools used to collect and analyze data. It highlights the need for consistent data collection procedures and the use of advanced analytical techniques to derive meaningful insights from the data.

3. The third part of the document focuses on the role of data in decision-making. It explains how data-driven insights can help identify trends, anticipate challenges, and optimize resource allocation, leading to more informed and effective strategic decisions.

4. The fourth part of the document addresses the challenges associated with data management, such as data quality, security, and privacy. It provides recommendations for implementing robust data governance frameworks to mitigate these risks and ensure the integrity and confidentiality of the data.

5. The fifth part of the document discusses the future of data analytics and the impact of emerging technologies like artificial intelligence and machine learning. It suggests that these technologies will further enhance the capabilities of data analysis, enabling organizations to uncover deeper insights and drive innovation.

6. The sixth part of the document concludes by summarizing the key points and emphasizing the overall importance of a data-driven approach in today's competitive business environment. It encourages organizations to embrace data as a strategic asset and invest in the necessary infrastructure and talent to harness its full potential.

7. The seventh part of the document provides a detailed overview of the data collection process, including the identification of data sources, the design of data collection instruments, and the implementation of data collection protocols. It also discusses the importance of ensuring data accuracy and reliability throughout the collection process.

8. The eighth part of the document describes the various data analysis techniques used to process and interpret the collected data. It covers both descriptive and inferential statistics, as well as more advanced methods like regression analysis and data mining. It also discusses the importance of visualizing data to facilitate understanding and communication of findings.

9. The ninth part of the document discusses the ethical considerations surrounding data collection and analysis. It highlights the need for transparency, informed consent, and data protection, and provides guidelines for ensuring that data is used responsibly and in compliance with relevant regulations and standards.

10. The tenth part of the document provides a final summary and reiterates the key takeaways from the document. It emphasizes the importance of a holistic data strategy that integrates data collection, analysis, and governance into the organization's overall operations and decision-making processes.

that called it (8), and the corresponding rule for the component instance (6) was linked correctly with the component's definition.

4. The positioning of the components (productions at (7)) requires a carefully designed model to show that the positions are as they should be. A model such as the one below is suitable.

```
CIRCUIT c1
WITH
    COMPONENT chip ic1
        POSITION(200,300)
        ORIENT(45)
ENDCIRCUIT
COMPONENT chip
WITH
    ICbody: b1 WINDOW(60,75) AT(0,10,-90)
        LENGTH(11,2)
ENDCOMPONENT
END
```

Notice that all of the positional parameters are different from each other. This is to try and stop a coincidental error corrections should there be any incorrectly stored positional data. The area that is tested can be viewed on the framestore by setting the display flag (see appendix B). It can be compared with the expected results.

5. Repeat statements have three parameters (production (5)) that should be checked. The first of these is the number of times the component should be repeated. It is not allowed to be less than one as this would not make any sense. The number of times the repeat is carried out using an example file will show whether it is being used correctly. The step size between components is the next parameter. This has to be a visual test. The third parameter is the axis along which the components are placed. This can only be the x-axis or the y-axis. As well as checking that only these are accepted, a visual check should be made to see that the correct axis is being used.
6. The final language feature to test is the use of recursive subcomponents. It is necessary to see that the windowing system is using all the co-ordinate frames of each component correctly.

The model in figure 8 was used to carry out the testing discussed above. It has all the language features included that are necessary for testing. For all of the tests above the model was used with minor modifications as necessary such as trying different feature names, using different axes for the repeat statements etc.

The first part of the document discusses the importance of maintaining accurate records of all transactions. It emphasizes that proper record-keeping is essential for the integrity of the financial system and for the ability to detect and prevent fraud. The text also mentions the need for regular audits and the role of independent auditors in ensuring the accuracy of financial statements.

The second part of the document focuses on the role of the accounting profession. It highlights the need for accountants to adhere to high standards of ethical conduct and to maintain their professional competence through continuous education. The text also discusses the importance of transparency and accountability in the accounting process.

The third part of the document addresses the challenges faced by businesses in the current economic environment. It discusses the impact of global economic uncertainty and the need for businesses to adapt to changing market conditions. The text also mentions the importance of innovation and the role of technology in driving business growth.

The fourth part of the document discusses the role of government in the economy. It highlights the need for government to provide a stable and predictable regulatory environment for businesses. The text also mentions the importance of government in addressing social and environmental issues and in promoting economic development.

The fifth part of the document discusses the role of international organizations in the global economy. It highlights the need for these organizations to promote international trade and investment and to address global economic challenges. The text also mentions the importance of cooperation and coordination between countries in addressing these challenges.

The final part of the document discusses the role of individuals in the economy. It highlights the need for individuals to be informed and active participants in the economic process. The text also mentions the importance of education and training in preparing individuals for the workforce.



```

CIRCUIT
WITH
    REPEAT 3 STEP 60 ALONG x FOR
        COMPONENT IC chip
            POSITION(15,200)
            ORIENT(-90)
    ENDREPEAT
    COMPONENT IC chip1
        POSITION(195,200)
        ORIENT(-90)
ENDCIRCUIT

COMPONENT IC
WITH
    IClegs: iclegs1 WINDOW(135,15) AT(0,0,0)
        PINCOUNT(135,15,8,5)
    IClegs: iclegs2 WINDOW(135,15) AT(135,50,180)
        PINCOUNT(135,15,8,5)
    REPEAT 2 STEP 10 ALONG y FOR
        COMPONENT resistor r1
            POSITION(50,15)
            ORIENT(0)
    ENDREPEAT
    COMPONENT resistor r2
        POSITION(-20,-30)
        ORIENT(90)
ENDCOMPONENT

COMPONENT resistor
WITH
    RESISTOR: r1 WINDOW(30,25) AT(0,0,0)
ENDCOMPONENT
END

```

Figure 8: Model Used For Testing Program



### 5.1.3 Model Testing Results

The tests outlined in the previous section were carried out using the model given. It was found that the program behaved as expected. It should be pointed out that the implementation does have limits built into it. The depth of subcomponent definitions is limited to nine. The maximum length of filenames is thirty characters. These limits will not be exceeded in normal circumstances.

## 5.2 Inspection Test Plan

The inspection test plan shows that the tests used during the inspection process are applied correctly under varying conditions. The varying conditions that will be examined are the translation and rotation of the circuit board within the image, and varying lighting conditions. Only variations in lighting should affect the reliability of the tests.

### 5.2.1 Translation and Rotation of Image

The program has been tested using the image shown in figure 4 as a reference. The next step is to show that the inspection process is invariant under translation and rotation of the image.

Applying the inspection process with the model file in figure 5 to the image translated should still get the same test results. The new circuit reference point is required in the command line (see appendix B). The same model should then be used with the image rotated about its centre. Finally both a translation and rotation should be tried.

### 5.2.2 Lighting Conditions

The previous test plan should not affect the results in any way. This next series of tests could cause the results to change. Using varying lighting conditions will show how durable the tests are. There are three lighting conditions to try which should give a feeling for how resilient the tests are.

Using daylight as the light source will give an even light cast over the circuit board without causing too much shadowing and no highlighting. A single spotlight on the circuit board will give a much brighter image than that achieved with daylight. It will cast shadows into the image since the light cannot be directly above all the components<sup>2</sup>. Using a second

---

<sup>2</sup>The image is taken by a camera directly above the centre of the circuit board and hence the light cannot

The first part of the report deals with the general situation of the country and the progress of the various branches of industry and commerce. It is found that the country is generally prosperous and that the various branches of industry and commerce are all making rapid progress.

The second part of the report deals with the various branches of industry and commerce in detail. It is found that the various branches of industry and commerce are all making rapid progress and that the country is generally prosperous.

The third part of the report deals with the various branches of industry and commerce in detail. It is found that the various branches of industry and commerce are all making rapid progress and that the country is generally prosperous.

The fourth part of the report deals with the various branches of industry and commerce in detail. It is found that the various branches of industry and commerce are all making rapid progress and that the country is generally prosperous.

The fifth part of the report deals with the various branches of industry and commerce in detail. It is found that the various branches of industry and commerce are all making rapid progress and that the country is generally prosperous.

The sixth part of the report deals with the various branches of industry and commerce in detail. It is found that the various branches of industry and commerce are all making rapid progress and that the country is generally prosperous.

The seventh part of the report deals with the various branches of industry and commerce in detail. It is found that the various branches of industry and commerce are all making rapid progress and that the country is generally prosperous.

light from the opposite side of the circuit board will help reduce the shadowing and still give a bright image.

### 5.2.3 Inspection Test Plan Results

It was found that translating the circuit board within the image taken by the camera did not affect the results achieved during inspection except in one aspect. Translating the image, or rotating for that matter, may result in parts of the circuit board that are to be inspected not being visible. The program will reply with a "Window Outside Boundary" error in these cases. Those parts of the image which were visible were found to be unaffected by the translation.

Rotating the image did cause some problems. Due to rounding errors when working with floating point arithmetic and also due to the small error margins involved it was found that some tests failed. At this time only the one test is implemented. This is the pin test described in chapter 4. The scan line across the pins is only a few pixels wide. This results in the pincount test missing some pins when using the rotated image.

The same problem appeared when using the image that was both translated and rotated. No other problems were encountered due to anything other than inaccuracies caused by using a rotated image.

The aim of using different lighting conditions was not carried out to the full specifications due to restrictions in the time available. The following results would have been expected for the pin test if they had been carried out.

- **Daylight:** Since the variation in intensity across the circuit board would have been minimal then I would have expected the pin tests to work satisfactorily.
- **Single Light:** This was carried out at one angle where the pins were perpendicular to the direction of the light source. This minimalised the intensity variation across the scan lines through the pins. The results from the pin tests worked correctly. Since the pin test uses local information to decide upon a threshold (see chapter 4.1) then the average intensity variations between different sets of pins did not make any difference. If the scan lines across the pins had been in the direction of the light then there would have been a visible intensity gradient across the pins. The single threshold value used in the test would not have been able to cope with this. To cope with this a best line fit to the average intensity gradient would be required.

---

be positioned in the ideal place

1. The first step in the process of the scientific method is to ask a question.

### 2. Formulating a Hypothesis

A hypothesis is a statement that can be tested. It is a prediction about the outcome of an experiment. The hypothesis should be based on previous knowledge and observations.

For example, if you observe that plants grow better in sunlight than in shade, you might formulate a hypothesis that "Plants grow taller when they receive more sunlight."

The hypothesis should be specific and measurable. It should also be testable through an experiment.

Once you have formulated a hypothesis, you can design an experiment to test it. The experiment should be controlled and repeatable.

During the experiment, you should collect data and record your observations. This data will be used to evaluate the hypothesis.

After the experiment is complete, you should analyze the data and draw a conclusion. The conclusion should state whether the hypothesis was supported or refuted by the data.

If the hypothesis is supported, it may lead to further research. If it is refuted, you may need to revise the hypothesis and repeat the experiment.

- **Double Lighting System:** The problems with intensity gradients would be reduced when using two lights from opposite sides. The specular reflections would be intensified with the greater amount of light present. All the metal surfaces would reflect the light back and cause parts of the image to wash out. This effect can be seen slightly in the leftmost set of pins in figure 5.

1. The first part of the document is a letter from the author to the editor, dated 10/10/1954. The letter discusses the author's interest in the subject of the journal and the author's hope that the journal will be a valuable contribution to the field.



## 6 Achievements and Conclusions

The aim of the project was to produce a reliable system for the inspection of the components on a variety of circuit boards. This required the development of a language which could be used to describe circuit boards. A framework was then built which could take the language descriptions of circuit boards and use them to guide an inspection process.

The grammar for the modelling language included a number of features that simplified the definition of models for describing circuits. The main features which extended the language beyond that originally developed by Chan [1] are:

- Referencing of components. Rather than repeatedly defining a component each time it is needed, a reference to it can be used.
- Hierarchical component definitions. Commonly occurring features can be grouped together into component definitions. These definitions can then be used inside other component definitions. This helps to keep the model readable.
- Repeat statements. If a number of the same component appear at regular intervals along the circuit board then they can be replaced by a single statement. This saves repetition during the modelling.

The development of a series of tests for feature recognition was not accomplished as had originally been anticipated. Too much time had been spent on the development of the program framework. This was caused by an unfamiliarity with the programming language used. One test was developed for the ICpins feature. The remaining features, ICbody, CAPACITOR and RESISTOR, do not yet have any tests designed for them.

The modularity of the program's framework lends itself well to an easily extendible system. The inspection mechanism which calls the individual tests does not know anything about these tests except the names of them. This means that the number of tests can be extended by increasing the list of tests which the inspection mechanism knows about. The grammar would also have to be altered slightly so that the test names would be recognised by the lexical analyser. It is also just as easy to extend the number of component features that can be used within the modelling language.

Two test plans were given which were used to evaluate the effectiveness of the program as a whole. The first test plan was designed to see that the model loading procedure was carried out correctly. This was found to be so. The second test plan was used to find out how resilient the feature tests were to varying conditions. These varying conditions consisted

The first part of the document discusses the importance of maintaining accurate records of all transactions. It emphasizes that every entry should be supported by a valid receipt or invoice. This ensures transparency and allows for easy verification of the data. The text also mentions that regular audits are necessary to identify any discrepancies or errors in the accounting process.

In addition, the document highlights the need for a clear and concise chart of accounts. This tool is essential for organizing financial data and providing a comprehensive overview of the company's financial health. It is recommended that the chart of accounts be updated regularly to reflect any changes in the business structure or operations. Furthermore, the text stresses the importance of proper classification of expenses and revenues to ensure accurate financial reporting.

The second part of the document focuses on the implementation of internal controls. These controls are designed to prevent fraud, reduce the risk of errors, and ensure the integrity of the financial information. Key elements of an effective internal control system include segregation of duties, authorization of transactions, and regular reconciliations. The document provides detailed guidelines on how to establish and maintain these controls, as well as the importance of training employees on their roles and responsibilities.

Finally, the document concludes by emphasizing the role of management in overseeing the financial reporting process. Management should ensure that the accounting system is properly designed and maintained, and that all financial statements are prepared in accordance with the applicable accounting standards. Regular communication and collaboration between management and the accounting department are essential for the success of the financial reporting process.

of circuit boards which appeared at rotated and translated positions within the digitized image, and different lighting conditions. The program was found to work satisfactorily under rotations and translations. The different lighting conditions were not investigated. Only the one lighting condition was tried - a single light source to one side. This did highlight an improvement that could be made to the pin test (cf 4.1).

There is much work that can still be done on this project. It falls into three areas:

- Improving the modelling language.
- Developing more feature tests.
- Improving the interface.

Improvements have been suggested to the modelling language (cf 3.4). The first of these allows the user to abstract away from associating particular tests with features, and instead to use a more intuitive description. The second proposal is to introduce pairwise tests. The grammar used with the YACC parser already has the syntax for pairwise comparisons. The framework would have to be modified to call pairwise tests.

The main development that must be done concerns the feature tests. There are no tests which can be used for the **ICbody**, **CAPACITOR** and **RESISTOR** features. At present if these features appear in a model then the dummy test **LENGTH** must be added. The features will not be tested but if the display flag is set then the feature will be highlighted to show that they would have been tested had there been any to apply.

As it stands, the program applies tests and reports any failures when applying them to features. It does not attempt to reason why the failures occurred. It is up to the tests themselves to display helpful diagnostics for the user. It would be useful, for instance, if a misplaced part could be reported. This is something that is discussed by Barnard [4]. In his system he uses statistical tests to detect damaged, missing and misplaced parts.

There are two papers which have aspects that would be useful in this system. The first of these is a paper by Woods and Taylor [2]. They also use a model-driven system. One particular aspect is that the co-ordinate system is automatically calibrated. At present my program requires the user to give information relating to the position, orientation and scale of the circuit board in the image. Automatic calibration is also discussed in a paper by Bolles [3]. In it he describes the *local feature focus method* that is used when recognising partially visible objects. A reliable feature is used to determine a co-ordinate system from which other key features can be found so that an object's position and orientation can be determined. This would be useful if implemented to find the position and orientation of the

...the ... of ...  
...the ... of ...  
...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...  
...the ... of ...

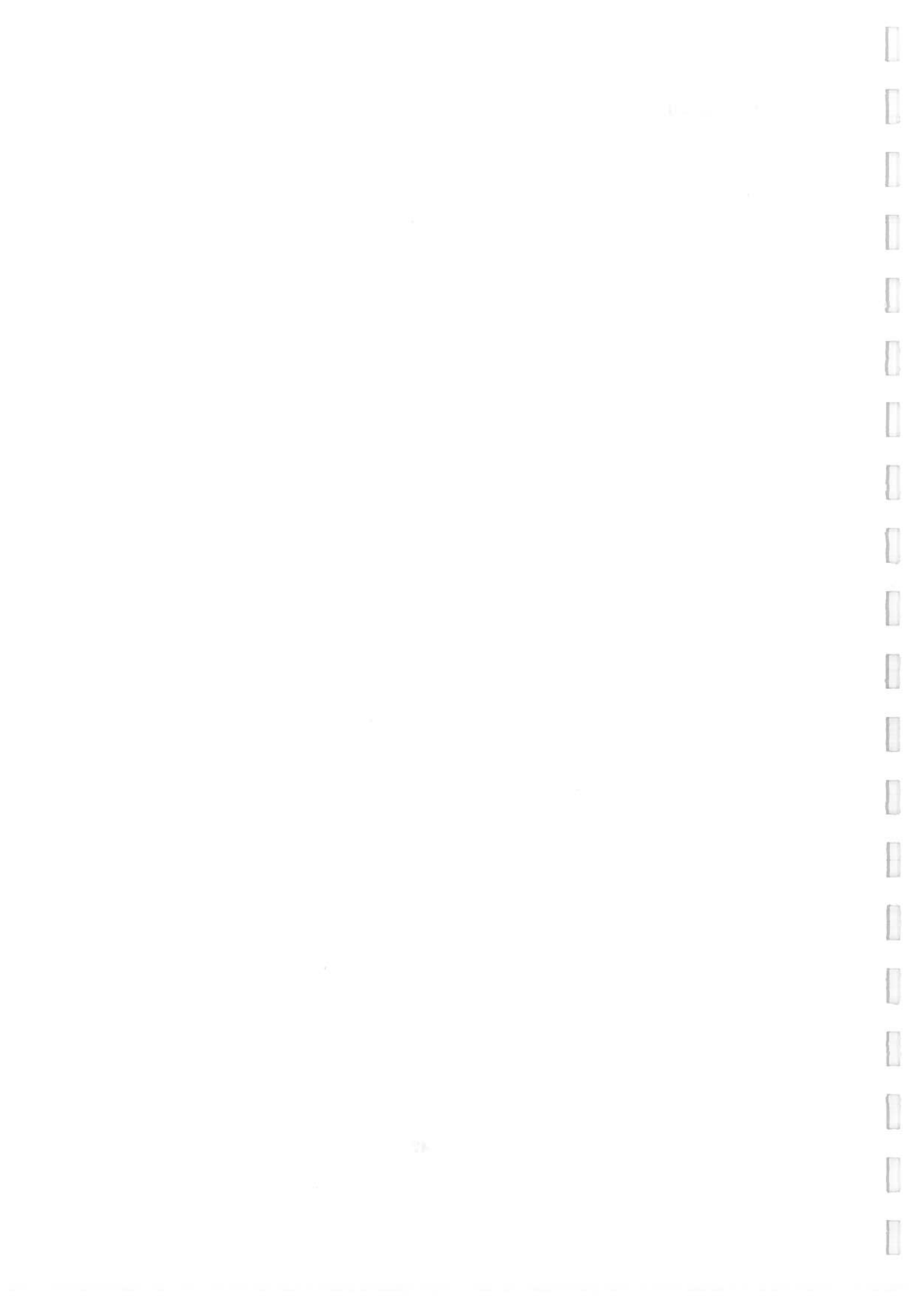
...the ... of ...  
...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...  
...the ... of ...

...the ... of ...  
...the ... of ...  
...the ... of ...

circuit board.



## 7 Acknowledgements

There are a number of people who have been of help to me during this project that I would like to thank:

- Bob Fisher, my supervisor, who helped with general aspects of how to approach the project and who also spent time reading over drafts of the project.
- Dave Croft who answered my queries about the graphical interface, and who managed to get the new framestore up and working just before the old one broke.
- Andrew Tait who read some early drafts of the report.

Main body of handwritten text, consisting of several lines of cursive script.



## References

- [1] Chan David *Grey Level Part Inspection*. Msc Thesis 1989
- [2] P W Woods & C J Taylor *Application of a User-Programmable Vision System to Inspection of Complex Assemblies*. Proceeding of 5th Alvey Vision Conference p31-36 1989
- [3] Bolles Robert C *Locating Partially Visible Objects: The Local Feature Focus Method*. AAAI National Conference on Artificial Intelligence p41-44 1980
- [4] Barnard Stephen T *Automated Inspection Using Gray-Scale Statistics*. AAAI National Conference on Artificial Intelligence p49-52 1980
- [5] Lex and Yacc Sun© *User Manual*

THE HISTORY OF THE UNITED STATES

The first part of the book is devoted to the early history of the United States, from the time of the first European settlement to the end of the Revolutionary War. It covers the discovery of the continent, the establishment of the first colonies, and the struggle for independence.

The second part of the book deals with the period from the end of the Revolutionary War to the beginning of the Civil War. It discusses the growth of the nation, the expansion of territory, and the political and social changes that took place during this time.

The third part of the book is devoted to the Civil War and Reconstruction. It covers the causes of the war, the course of the conflict, and the efforts to rebuild the South and integrate the freed slaves into society.

The fourth part of the book deals with the period from the end of Reconstruction to the present. It discusses the Gilded Age, the Progressive Era, and the modern era, including the two world wars and the civil rights movement.

## A Grammar BNF

The following section shows the BNF of the grammar for the modelling language described in chapter 3. Upper case words denote keywords which the lexical analyser recognises. There are three exceptions to this:

- **NAME** refers to any string of alpha-numeric characters.
- **INTEGER** and **FLOAT** refer to integer and floating point numbers respectively.

These three tokens are special cases for the lexical analyser used along with the parser. The parser is written using YACC and the following BNF grammar.

Faint, illegible text, possibly bleed-through from the reverse side of the page.



grammar	::=	list END	
list	::=	circuit	
		component_def	
		circuit list	
		component_def list	
circuit	::=	CIRCUIT NAME WITH component_list ENDCIRCUIT	
		CIRCUIT NAME WITH component_list COMPARE	
		inspection_list ENDCIRCUIT	
component_list	::=	repeatloop component ENDREPEAT	
		repeatloop component ENDREPEAT component_list	
		component	
		component component_list	
repeatloop	::=	REPEAT value STEP value ALONG NAME FOR	
component	::=	COMPONENT NAME NAME posn	
posn	::=	POSITION('value','value')	
		ORIENT('value')	
component_def	::=	COMPONENT NAME WITH feature_list	
		ENDCOMPONENT	
		COMPONENT NAME WITH feature_list	
		COMPARE inspection_list ENDCOMPONENT	
feature_list	::=	feature	
		feature feature_list	
feature	::=	FEATURE':' NAME window inspection_list	
		repeatloop component ENDREPEAT	
		component	
window	::=	WINDOW('value','value') AT('value','value','value')	
inspection_list	::=	inspection	
		inspection inspection_list	
inspection	::=	TYPE '('parm_list')	
parm_list	::=	value	
		value ',' parm_list	
		NAME	
		NAME ',' parm_list	
value	::=	INTEGER	
		FLOAT	

The following information is provided for your reference:  
 1. The project was completed on a timely basis.  
 2. The budget was maintained throughout the project.  
 3. The quality of the work was consistently high.  
 4. The team worked well together and communicated effectively.  
 5. The client was satisfied with the final results.  
 6. The project was completed within the agreed-upon timeline.  
 7. The team demonstrated strong problem-solving skills.  
 8. The project was completed with a high level of professionalism.  
 9. The team exceeded expectations in all areas.  
 10. The project was a significant success for the organization.

## B User Instructions

This program is designed to take a digitized image of a circuit board and apply an inspection process to it according to a model definition of what should be contained in the circuit board. It therefore requires two input files:

- Model Definition File
- Digitized Image File

Before the program can be run, the digitized image has to be in HIPS format and compressed from a 4:3 horizontal:vertical scaling to a 1:1 scaling. Some information also has to be given to the program about where the circuit board appears in the image. An (x,y) co-ordinate along with an orientation has to be stated. This will give the program a reference point relative to which the inspections will take place. This point has to be the same point which was used when the model file was created. It is therefore best to choose a prominent feature on the circuit board. As well as this it is necessary to give a scaling factor for converting from model co-ordinates to screen pixels.

There are two flags which can be invoked to get more feedback about what the program is doing. The first of these is the (c)heck flag. This will make the program print out a tree like diagram of how the components within the model relate to each other. The program does not run the tests when this flag is set. It will stop after the tree has been displayed. The second option is the (d)isplay flag. This will make use of a connected framestore to inform the user of how the inspection is proceeding. As each feature in the circuit is inspected, the area of the image that is being inspected will be inverted to show this. These areas of the screen are not necessarily going to be parallel with the x-axis of the display so another copy of the area is shown in the top left hand corner of the screen. Any actions that a particular test may be doing can also be displayed in the top left hand portion of the screen.

The command to invoke the inspection program looks like the following:

```
ci -c -d modelfile circuit_name imagefile X Y Rot Scale
```

The -c and -d flags are optional arguments. The circuit\_name is the circuit definition from the model file that is to be used for the inspection process. The arguments X Y and Rot define the reference point for the circuit board and take integer values. The scaling factor converting from model to screen co-ordinates is a floating point value.

The first part of the report discusses the current state of the world economy and the impact of the Asian crisis. It notes that the crisis has led to a sharp decline in growth rates in many Asian countries, and has had a significant impact on the global economy.

The second part of the report discusses the impact of the crisis on the global economy. It notes that the crisis has led to a sharp decline in growth rates in many Asian countries, and has had a significant impact on the global economy.

The third part of the report discusses the impact of the crisis on the global economy. It notes that the crisis has led to a sharp decline in growth rates in many Asian countries, and has had a significant impact on the global economy.

The fourth part of the report discusses the impact of the crisis on the global economy. It notes that the crisis has led to a sharp decline in growth rates in many Asian countries, and has had a significant impact on the global economy.

The fifth part of the report discusses the impact of the crisis on the global economy. It notes that the crisis has led to a sharp decline in growth rates in many Asian countries, and has had a significant impact on the global economy.

The sixth part of the report discusses the impact of the crisis on the global economy. It notes that the crisis has led to a sharp decline in growth rates in many Asian countries, and has had a significant impact on the global economy.

The seventh part of the report discusses the impact of the crisis on the global economy. It notes that the crisis has led to a sharp decline in growth rates in many Asian countries, and has had a significant impact on the global economy.

The eighth part of the report discusses the impact of the crisis on the global economy. It notes that the crisis has led to a sharp decline in growth rates in many Asian countries, and has had a significant impact on the global economy.

The ninth part of the report discusses the impact of the crisis on the global economy. It notes that the crisis has led to a sharp decline in growth rates in many Asian countries, and has had a significant impact on the global economy.

The tenth part of the report discusses the impact of the crisis on the global economy. It notes that the crisis has led to a sharp decline in growth rates in many Asian countries, and has had a significant impact on the global economy.



## C Program

The program consists of seventeen files. Here is a short description of what each one does:

- **Makefile** The bulk of this makefile is used for setting up the libraries required for using the framestore. Otherwise it is a standard makefile that will compile the following code fragments together.
- **ci.c** This is the main section of code which calls the various program sections as required. It reads in the command line, calls the parser to parse the model file and starts the control procedure.
- **ci.extern** All the external variables are defined in here.
- **ci.h** It contains some general constant definitions and all of the structures that are used by the YACC parser.
- **feed.c** is used by the windowing system. It rotates an area of the screen round until it is parallel with the axes of the screen.
- **halloc.c** contains a memory allocation function used by the windowing system when storing screen images.
- **length.c** contains a dummy inspection test which was used during the testing of the program.
- **lex.l** is the lexical analyser code which is written using LEX.
- **model.h** defines constants for each of the features and test names.
- **pin.c** is the pin test code.
- **pread.c** is a small piece of code for loading in images from a file. It stores them in a single dimensional array.
- **read\_header.c** is used to load in the header information from a HIPS image in preparation for reading in the actual image.
- **show.c** This is used when the display flag is set. If any inspections of features fail then the showerrors function is called to display a box around the offending feature on the framestore.
- **test.c** The main inspection control mechanism resides here. The appropriate tests are called for each feature in the circuit.

The first part of the report deals with the general situation in the country. It is noted that the economy is still in a state of depression, and that the government is struggling to find ways to improve the situation. The report also discusses the political situation, and the role of the various political parties.

The second part of the report deals with the social situation. It is noted that the population is still suffering from the effects of the war, and that there is a need for social reforms. The report also discusses the role of the various social organizations, and the need for a more equitable distribution of resources.

The third part of the report deals with the economic situation. It is noted that the economy is still in a state of depression, and that there is a need for economic reforms. The report also discusses the role of the various economic organizations, and the need for a more diversified economy.

The fourth part of the report deals with the political situation. It is noted that the government is struggling to find ways to improve the situation, and that there is a need for political reforms. The report also discusses the role of the various political parties, and the need for a more democratic system.

The fifth part of the report deals with the social situation. It is noted that the population is still suffering from the effects of the war, and that there is a need for social reforms. The report also discusses the role of the various social organizations, and the need for a more equitable distribution of resources.

The sixth part of the report deals with the economic situation. It is noted that the economy is still in a state of depression, and that there is a need for economic reforms. The report also discusses the role of the various economic organizations, and the need for a more diversified economy.

The seventh part of the report deals with the political situation. It is noted that the government is struggling to find ways to improve the situation, and that there is a need for political reforms. The report also discusses the role of the various political parties, and the need for a more democratic system.

The eighth part of the report deals with the social situation. It is noted that the population is still suffering from the effects of the war, and that there is a need for social reforms. The report also discusses the role of the various social organizations, and the need for a more equitable distribution of resources.

The ninth part of the report deals with the economic situation. It is noted that the economy is still in a state of depression, and that there is a need for economic reforms. The report also discusses the role of the various economic organizations, and the need for a more diversified economy.

The tenth part of the report deals with the political situation. It is noted that the government is struggling to find ways to improve the situation, and that there is a need for political reforms. The report also discusses the role of the various political parties, and the need for a more democratic system.

- `wframe.c` is responsible for displaying the windows which signify the features being tested.
- `window.c` contains the mechanism which translates a feature's model co-ordinates to screen co-ordinates.
- `yacc.y` contains the grammar describing the modelling language. It is written using YACC.

## C.1 Extending the Program

An important feature of this program is the ease with which it can be extended to include more inspection tests and features. There now follows a description of how to do this:

Adding new features to the language does not require much effort. This is because a feature is merely a label to which tests are associated. Therefore it is only necessary to give the program the name of the feature. This requires changes to two files - `lex.l` and `model.h`. A new token has to be added to the lexical analyser in the form

```
Feature_name yylval.ival=Feature_name; return(FEATURE);
```

A constant definition for that feature has to be entered in the `model.h` file:

```
#define Feature_name <number>
```

where *number* is a numeric value that is not already used within the file. The numbering system presently used is that features are numbered from 1000 up to 1999. Test names are numbered from 2000 upwards.

Adding new tests obviously requires a definition to be entered into the `model.h` file as mentioned above. A similar change is made to the lexical analyser, `lex.l`, except that `TYPE` is returned rather than `FEATURE`, i.e.

```
Test_name yylval.ival=Test_name; return(TYPE);
```

The parser, `yacc.y`, has to be informed of the new test. In the production

```
inspection ::= TYPE(parm_list)
```

there is a case statement that holds a list of all the tests. A new case has to be entered for the new test. The case statement is used to determine whether the correct number of arguments have been given to the test. So a new entry might look like the following for a new test **ORIENT**:

1. The first part of the document discusses the importance of maintaining accurate records of all transactions.

2. It is essential to ensure that all entries are supported by appropriate evidence and documentation.

3. The second part of the document outlines the various methods used to collect and analyze data.

4. These methods include both qualitative and quantitative approaches, each with its own strengths and limitations.

5. The third part of the document provides a detailed overview of the theoretical framework underlying the research.

6. This framework is based on a combination of established theories and new insights from recent research.

7. The fourth part of the document describes the specific research design and methodology employed in the study.

8. This includes a discussion of the sample selection process, data collection procedures, and analysis techniques.

9. The fifth part of the document presents the results of the study, including a detailed description of the findings.

10. These results are then compared against the theoretical framework and previous research to identify key insights.

11. The sixth part of the document discusses the implications of the findings for practice and policy.

12. It highlights the potential benefits of the research and offers suggestions for further exploration.

13. The seventh part of the document provides a concluding summary of the study and its contributions.

14. Finally, the eighth part of the document includes a list of references and a list of figures and tables.

15. The list of references includes a comprehensive review of the literature related to the study.

16. The list of figures and tables provides a visual representation of the data and results.

17. The list of figures and tables includes a detailed description of each figure and table.

18. The list of figures and tables also includes a list of captions for each figure and table.

19. The list of figures and tables is organized in a clear and concise manner.

inspection:

```
TYPE '(' parm_list ')'
{
    arity=count($3);
    switch($1) {
    case PINCOUNT:
        if (arity!=4)
            error("Wrong Arity");
        else
            break;
        :
    case ORIENT:
        if (arity!=2)
            error("Wrong Arity");
        else
            break;
        :
    }
    if (($$=(INSP_PTR) malloc(sizeof(INSP)))==NULL)
        error("Can't allocate memory");
    $$->insptype=$1;
    $$->insparm=$3;
}
;
```

Finally test.c has to be changed so that it will call the new test when it is found in the model definition. The function inspect() has a case statement that calls the appropriate test. A new case could be entered which calls the test function. The needs to take four arguments - the feature name being tested, the parameters that are required for the test to carry out the inspection, the scale for applying to any of the parameters that require it, and a display flag which informs the test if the framestore is available for use. The function also has to return a value (TRUE/FALSE or 1/0) which signifies whether the test succeeded. The new entry in the inspect function would look like the following:

1. The first part of the document is a letter from the author to the editor.

2. The second part is a letter from the editor to the author.

3. The third part is a letter from the author to the editor.

4. The fourth part is a letter from the editor to the author.

5. The fifth part is a letter from the author to the editor.

6. The sixth part is a letter from the editor to the author.

7. The seventh part is a letter from the author to the editor.

8. The eighth part is a letter from the editor to the author.

9. The ninth part is a letter from the author to the editor.

10. The tenth part is a letter from the editor to the author.

11. The eleventh part is a letter from the author to the editor.

12. The twelfth part is a letter from the editor to the author.

13. The thirteenth part is a letter from the author to the editor.

14. The fourteenth part is a letter from the editor to the author.

15. The fifteenth part is a letter from the author to the editor.

16. The sixteenth part is a letter from the editor to the author.

17. The seventeenth part is a letter from the author to the editor.

18. The eighteenth part is a letter from the editor to the author.

The following is a list of the names of the authors of the letters in the document. The names are listed in the order in which they appear in the document. The names are: 1. John Doe, 2. Jane Smith, 3. Bob Johnson, 4. Alice Brown, 5. Charlie White, 6. David Green, 7. Emily Black, 8. Frank Gray, 9. Grace Pink, 10. Henry Blue, 11. Irene Yellow, 12. Jack Purple, 13. Karen Red, 14. Larry Orange, 15. Mary Silver, 16. Norman Gold, 17. Olivia Bronze, 18. Paul Copper.

```
switch(tst->insptype)
{
case PINCOUNT:
    ok=pincount(feature_name,tst->insparm,scale,display);
    break;
case ORIENT:
    ok=orient(feature_name,tst->insparm,scale,display);
    break;
    :
default:
    ok=TRUE;
    break;
}
```

1. [Faint text]

2. [Faint text]

3. [Faint text]



**D Program Listing**





1. Name: \_\_\_\_\_  
2. Address: \_\_\_\_\_  
3. City: \_\_\_\_\_  
4. State: \_\_\_\_\_  
5. Zip: \_\_\_\_\_  
6. Phone: \_\_\_\_\_  
7. Email: \_\_\_\_\_  
8. Date: \_\_\_\_\_  
9. Signature: \_\_\_\_\_  
10. Title: \_\_\_\_\_

```

error("Cannot open image file".imagefile);
fread_header(image_fp,abd);
if (bd.pixel_format!=PFBYTE)
{
    fprintf(stderr,"Frame must be in byte format");
    exit(1);
}
for (row=0; row<bd.rows; row++)
{
    if ((indata[row]=(unsigned char*)calloc(bd.cols,1))==NULL)
    {
        fprintf(stderr,"Cannot allocate array");
        exit(0);
    }
    if ((dest[row]=(unsigned char*)calloc(bd.cols,1))==NULL)
    {
        fprintf(stderr,"Cannot allocate array");
        exit(0);
    }
    fread(image_fp,indata[row],bd.cols);
    empty(dest[row],bd.cols);
}
if (display) /* set up framebuffer for display Show circuit image */
{
    iznum=1;
    vb_plane(1);
    vb_open();
    if ((outdata=(unsigned char *)malloc(bd.rows*bd.cols))==(unsigned char*)NULL)
    {
        fprintf(stderr,"Can't allocate array\n");
        exit(0);
    }
    for (y=0; y<bd.rows; y++)
        for (x=0; x<bd.cols; x++)
            outdata[x*y+bd.cols]=indata[y][x];
    vb_putblock(outdata,0,0,bd.cols,bd.rows);
    vb_plane(0);
    vb_putblock(outdata,0,0,bd.cols,bd.rows);
}
/**** Call to Control Routine ****/
feature = object->circptr->compls;
if (check) printf("Circuit part list for %s:\n",object->circptr->circum);
control(feature,display,check);
vb_plane(1);
vb_close();
}
/******
/* This function calls the test function for each component defined in */
/* the circuit being inspected */
/******
control(feature,display,check)

```

```

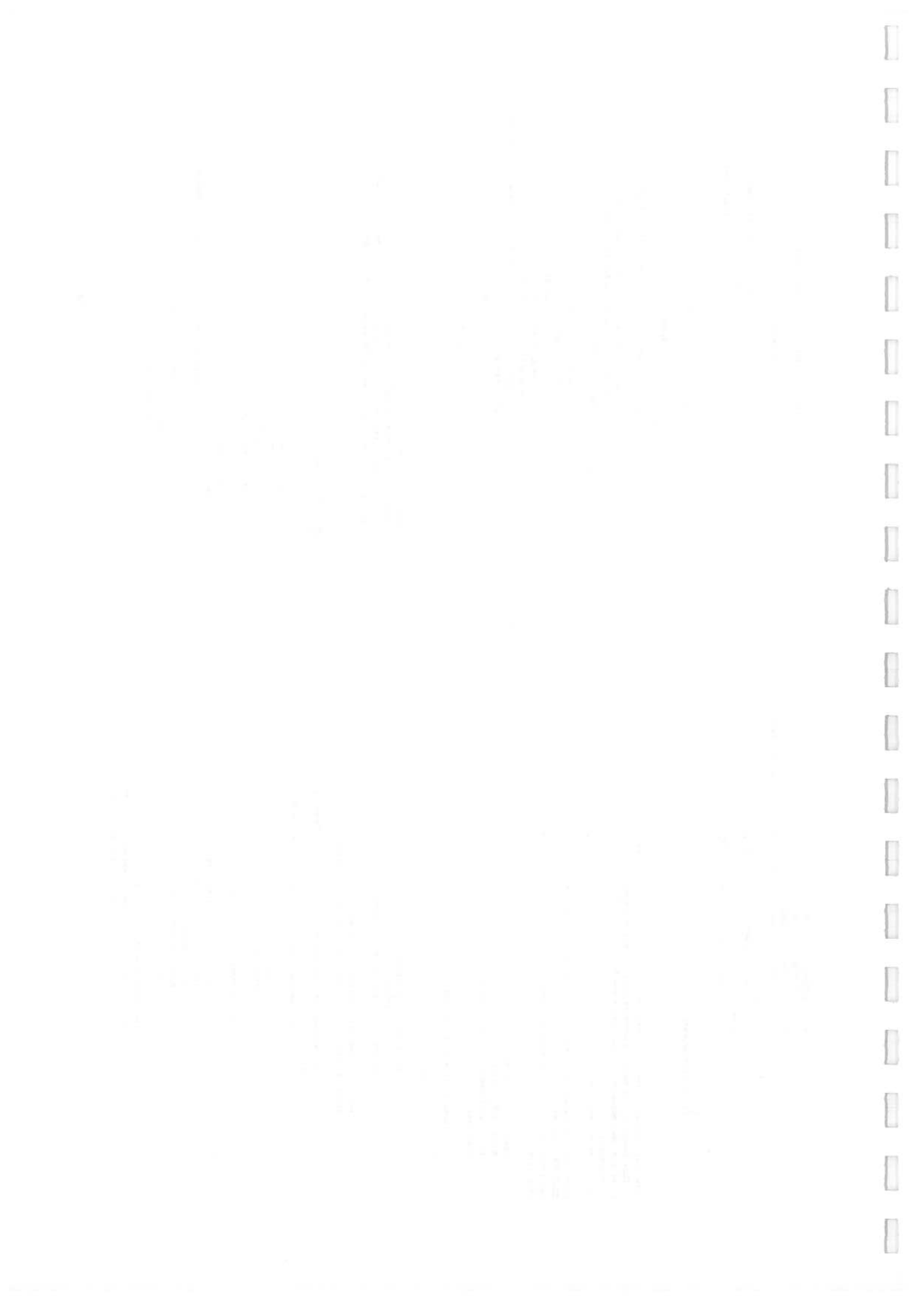
COMP_LS_PTR *feature;
int display,check;
{
    COMP_LS_PTR feat;
    FEAT_LS_PTR subcomp;
    REPEAT_PTR rptinfo;
    int count,step,loop,repeat,lan;
    char axis, *name;
    feat=(*feature);
    while (feat!=NULL)
    {
        if (feat->rpt==NULL) /* simple component */
        {
            if (check)
                printf("%s\n",feat->compenry->compnm);
            subcomp=feat->compenry->definition->feats;
            while (subcomp!=NULL)
            {
                if (subcomp->feat->featnm==NULL)
                {
                    if (subcomp->feat->rpt == NULL)
                    {
                        repeat=FALSE,count=0;
                        step=0,axis=NULL;
                    }
                    else
                    {
                        rptinfo=subcomp->feat->rpt;
                        count=rptinfo->times;
                        step=rptinfo->step;
                        axis=rptinfo->axis[1];
                        repeat=TRUE;
                    }
                    subcontrol(s(subcomp->feat->compefeat),
                        Repeat,axis,count,step,1,display,check);
                }
                else
                {
                    if (check)
                        printf("\t");
                }
                subcomp=subcomp->feateext;
            }
            /* call to test component */
            if (check)
                test(feat->compenry,bd.rows,bd.cols,display);
        }
        else /* repeat component */
        {
            rptinfo=feat->rpt;
            count=rptinfo->times;
            step=rptinfo->step;
            axis=rptinfo->axis[1];
            lan=atrian(feat->compenry->compnm);
            for (loop=1; loop<count; loop++) {

```

Faint, illegible text, possibly bleed-through from the reverse side of the page.









```

/* search for component definition. If not found then return */
/* from function call. */
while (strcmp(nextobject->componentptr->compdefnm, name))
{
    nextobject=nextobject->objnext;
    if (nextobject == NULL)
        return(name);
    while (nextobject->componentptr == NULL)
    {
        nextobject=nextobject->objnext;
        if (nextobject == NULL)
            return(name);
    }
    (-defn) = nextobject->componentptr;
    flist= (-defn)->featls;
    while (flist != NULL)
    {
        if (flist->feat->featnm == NULL)
        {
            flist->feat->parent=component;
            errorname=findcomponent(&newcomp, flist->feat->compfeat);
            if (!strcmp(errorname, ""))
            {
                flist->feat->compfeat->definition=newcomp;
                flist->feat->featnext;
            }
            else
            {
                name2=strcmp(":", name);
                name=strcmp(errorname, name2);
                return(name);
            }
        }
        else flist=flist->featnext;
    }
    return("");
}

/* The following three functions are error routines. yerror() is used */
/* by YACC, but the others are used throughout the code. */
/*.....*/
error(as, bs)
char *as;
char *bs;
{
    fprintf(stderr, "%s\n\n", as, bs);
    exit(-1);
}

yerror(s)
char *s;
{
    fprintf(stderr, "%s\n", s);
}

```





1. The first part of the document discusses the importance of maintaining accurate records of all transactions. It emphasizes that this is crucial for ensuring the integrity of the financial statements and for providing a clear audit trail.

2. The second part of the document outlines the various methods used to collect and analyze data. It describes how different types of information are gathered and how they are processed to identify trends and anomalies.

3. The third part of the document details the specific procedures followed during the audit process. It includes a description of the sampling techniques used and the criteria for selecting items for review.

4. The fourth part of the document provides a summary of the findings and conclusions reached. It discusses the overall health of the organization's financial operations and highlights any areas that require further attention.

5. The final part of the document contains the auditor's report and recommendations. It provides a clear statement of the auditor's opinion and offers suggestions for improving internal controls and financial reporting practices.

```

width = *(param_ptr++) * scale;
wind_x = *(param_ptr++) * scale;
wind_y = *(param_ptr++) * scale;
wind_orient = *(param_ptr++);

sin_theta = sin(wind_orient * PI / 180.0);
cos_theta = cos(wind_orient * PI / 180.0);

/* work out window corner points in screen coords */
vertex[0].x = wind_x;
vertex[0].y = wind_y;
vertex[1].x = wind_x + cos_theta * length;
vertex[1].y = wind_y + sin_theta * length;
vertex[2].x = wind_x + cos_theta * length * sin_theta * width;
vertex[2].y = wind_y + sin_theta * length * cos_theta * width;
vertex[3].x = wind_x + sin_theta * width;
vertex[3].y = wind_y + cos_theta * width;

for (i=0; i<4; i++)
{
temp_x = vertex[i].x;
temp_y = vertex[i].y;

vertex[i].x =
transform[0][0] * temp_x +
transform[0][1] * temp_y +
transform[0][2];

vertex[i].y =
transform[1][0] * temp_x +
transform[1][1] * temp_y +
transform[1][2];

}

/* find the 4 linear equations which model the ...
/* window boundary
/* This following piece of code was originally
/* written by David Chan.
for(i=0; i<4; i++)
{
j = (i+1) % 4;
temp_x = vertex[j].x - vertex[i].x;
temp_y = vertex[j].y - vertex[i].y;
if (temp_x==0)
{
equation[i].ok = FALSE;
equation[i].m = 0.0;
equation[i].c = vertex[i].x;
}
else
{
equation[i].ok = TRUE;
equation[i].m = temp_y / temp_x;
equation[i].c = vertex[i].y - equation[i].m * vertex[i].x;
}
}
equation[i].ep1.x = vertex[i].x;
equation[i].ep1.y = vertex[i].y;
equation[i].ep2.x = vertex[j].x;

```

```

/* .....
/* window c
/* This is the main part of the windowing system. It converts feature
/* windows from model co-ordinates to screen co-ordinates.
/* .....
#include <stdio.h>
#include <math.h>
#include "ci.h"
#include "ci.extern"

struct mat_data
{
double x, y, orient;
};

void mult_mat():
/* .....
/* getwindow takes the parameters for a feature window. It returns the
/* screen coordinates of the window's corner points and also fills the
/* global array 'scanline' with the boundary points of the window.
/* .....
int getwindow(rows, cols, param, scale, comp, vertex)
int rows, cols;
int *param;
float scale;
COMP_PTR comp;
struct coordinate vertex[4];
{
struct mat_data frame[10];
int num_wind_x, wind_y, wind_orient;
float transform[3][3];
int i, j, x, temp[4];
int *param_ptr;
float temp_x, temp_y, length, width;
float sin_theta, cos_theta;
struct linear_eq {
float m;
float c;
int ok;
} equation[4];
struct coordinate ep1;
struct coordinate ep2;
} equation[4];

/* get transformation matrix */
param_ptr = param;
num = getframes(comp, frame, scale);
mult_mat(frame, num+1, transform);

/* get window parameters */
length = *(param_ptr++) * scale;

```

Faint, illegible text, possibly bleed-through from the reverse side of the page. The text is arranged in several paragraphs and appears to be a formal document or report.



x\_temp[j++]=(y-equation[i] c)/equation[i] m.

```

    }
    x_temp[j++]=(y-equation[i] c)/equation[i] m.
  }
}
scanline[y].left=scanline[y] right=x_temp[0];
for(i=0; i<j; i++)
{
  if(x_temp[i] < scanline[y].left)
    scanline[y].left = x_temp[i];
  else if(x_temp[i] > scanline[y].right)
    scanline[y].right = x_temp[i];
}
}
return(0);
}
}

```

```

/*.....
/* The co-ordinate frames required for the component are gathered */
/* together here.
/*.....

```

```

int getframes(comp,frameinfo, scale)
COMP_PTR comp;
struct mat_data *frameinfo;
float scale;
{
  int number=0, loop;
  COMP_PTR component;
  /* get components coordinate frame */
  component=comp;
  frameinfo[number].x = (double) component->wind[0] * scale;
  frameinfo[number].y = (double) component->wind[1] * scale;
  frameinfo[number].orient = (double) component->wind[2];
  number++;
}

```

```

/* get each supercomponents coordinate frame */
while (component->parent != NULL)
{
  frameinfo[number].x = (double)
    component->parent->wind[0] * scale;
  frameinfo[number].y = (double)
    component->parent->wind[1] * scale;
  frameinfo[number].orient = (double)
    component->parent->wind[2];
  component=component->parent;
  number++;
  if (number<1)
    error("Recursive Definition too deep for ", comp->compnum);
}
frameinfo[number].x= Xref;
frameinfo[number].y= Yref;
frameinfo[number].orient= rotation;
/* have to copy coordinate frame information so that it */
/* starts at index 0 in the array

```

```

equation[i].ep2.y = vertex[j].y;
}
/*... Find the y-min and y-max coordinate of the window corners ...*/
y_min=y_max=equation[0].ep1.y;
for (i=0; i<4; i++)
{
  if (equation[i].ep1.y > y_max)
    y_max = equation[i].ep1.y;
  if (equation[i].ep2.y > y_max)
    y_max = equation[i].ep2.y;
  if (equation[i].ep1.y < y_min)
    y_min = equation[i].ep1.y;
  if (equation[i].ep2.y < y_min)
    y_min = equation[i].ep2.y;
}

```

```

/*... Remove those linear equations which do not intersect .../
/*... the current line

```

```

x_min=x_max=equation[0].ep1.x;
for (i=0; i<4; i++)
{
  if (equation[i].ep1.x > x_max)
    x_max = equation[i].ep1.x;
  if (equation[i].ep2.x > x_max)
    x_max = equation[i].ep2.x;
  if (equation[i].ep1.x < x_min)
    x_min = equation[i].ep1.x;
  if (equation[i].ep2.x < x_min)
    x_min = equation[i].ep2.x;
}
if(x_min < 0 || x_max > cols || y_min < 0 || y_max > rows)
return(i);

```

```

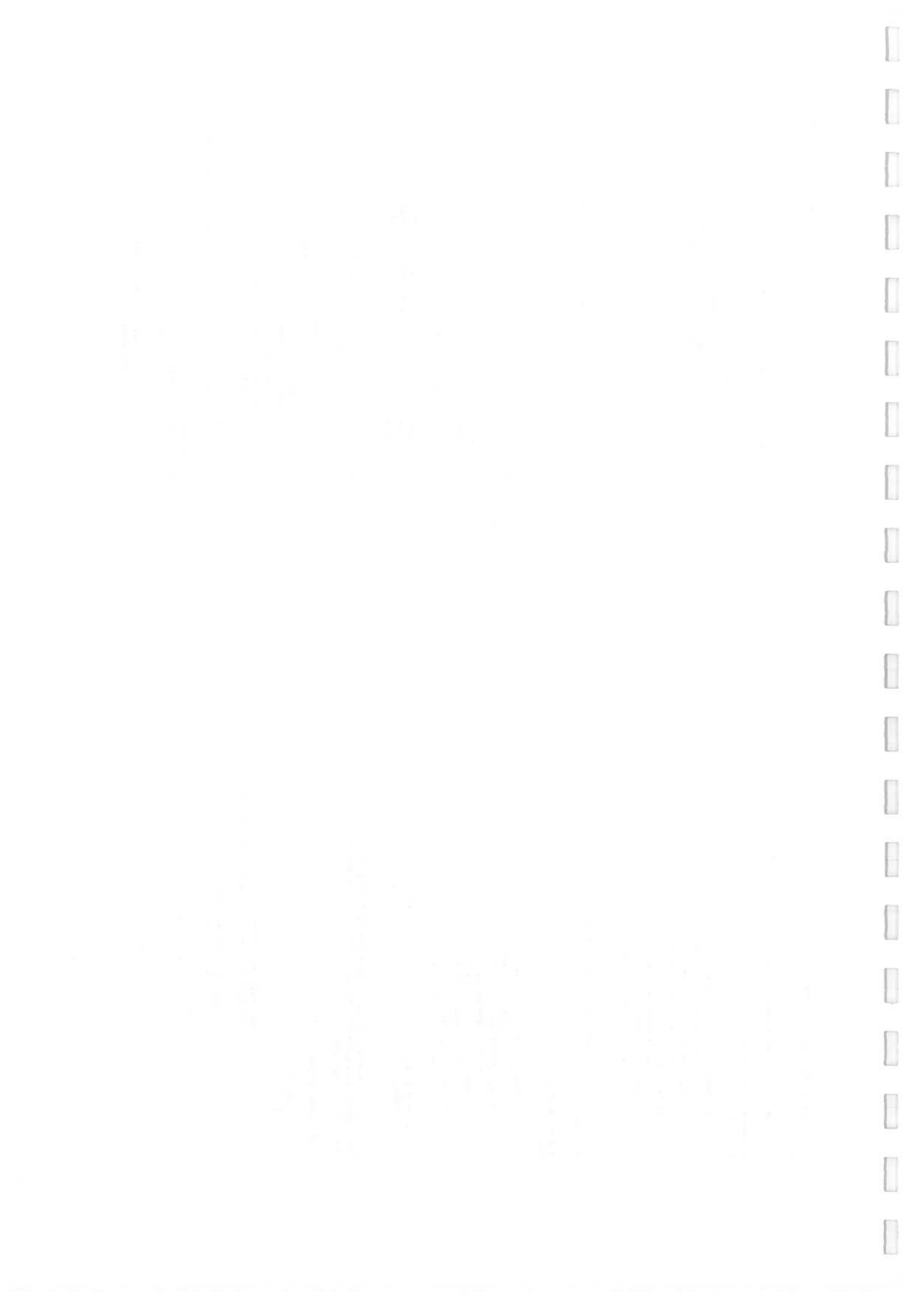
/*... find the boundary points of the window by finding the .../
/*... intersection points of the current scan line with the .../
/*... 4 linear equations

```

```

for(y=y_min; y<y_max+1; y++)
{
  for(i=0; i<4; i++)
  {
    if((y>equation[i].ep1.y&&y<equation[i].ep2.y)||
      (y<equation[i].ep1.y&&y>equation[i].ep2.y))
    {
      if(equation[i].ok == FALSE)
        x_temp[j++] = equation[i].c;
      else {
        if (equation[i] m == 0 0)
        {
          x_temp[j++] = equation[i].ep1.x;
          x_temp[j++] = equation[i].ep2.x;
          break;
        }
        else

```





```

}
for (loopnumber; loop<10; loop++)
    frameinfo[loop-number]=frameinfo[loop];
return(0-number); /* number of frames found */
}

/*****
/* This function takes all of the coordinate frames, forms matrices
/* out of them and multiplies them all together to obtain a single
/* transformation matrix
*****/

void mult_mat(frames, numb, comp_mat)
struct mat_data *frames;
int numb;
float comp_mat[3][3];
{
    int h, i, j;
    float c, s, x, y, tmp_mat[3][3];
    for (i = 0; i < 3; i++)
    {
        for (j = 0; j < 3; j++)
        {
            comp_mat[i][j] = 0.0;
        }
        comp_mat[i][i] = 1.0;
    }
    for (h = 0; h < numb; h++)
    {
        c = cos((frames[h].orient)*(PI/180.0));
        s = sin((frames[h].orient)*(PI/180.0));
        x = frames[h].x;
        y = frames[h].y;
        tmp_mat[0][0] = c*comp_mat[0][0]+s*comp_mat[0][1];
        tmp_mat[0][1] = c*comp_mat[0][1]-s*comp_mat[0][0];
        tmp_mat[0][2] = x*comp_mat[0][0]+y*comp_mat[0][1]+
            comp_mat[0][2];
        tmp_mat[1][0] = c*comp_mat[1][0]+s*comp_mat[1][1];
        tmp_mat[1][1] = c*comp_mat[1][1]-s*comp_mat[1][0];
        tmp_mat[1][2] = x*comp_mat[1][0]+y*comp_mat[1][1]+
            comp_mat[1][2];
        tmp_mat[2][0] = c*comp_mat[2][0]+s*comp_mat[2][1];
        tmp_mat[2][1] = c*comp_mat[2][1]-s*comp_mat[2][0];
        tmp_mat[2][2] = x*comp_mat[2][0]+y*comp_mat[2][1]+
            comp_mat[2][2];
        for (i = 0; i < 3; i++)
            for (j = 0; j < 3; j++)
                comp_mat[i][j] = tmp_mat[i][j];
    }
}

/*****
/* This function will take the window defined in the scanline array and
/* put it into a block array in a suitable form for applying tests
*****/

#include "stdio.h"
#include "math.h"
#include "ci.h"
#include "model.h"
#include "ci_extern"

/*****
/* This function produces a pointer to the array containing the area
/* of the screen to be used for inspection purposes.
*****/

convert(rows, cols, para_ptr, SF, vertex)
int *para_ptr;
float SF;
{
    int wind_x, wind_y, wind_orient, length, width;
    int destx, desty, refpt_x, refpt_y;
    int xi, yi, x2, y2;
    double X, Y, dist, rot;
    /* Get reference line for calculating angle of window */
    xi = vertex[0].x; x2 = vertex[1].x;
    yi = vertex[0].y; y2 = vertex[1].y;
    /* get window parameters */
    length = *(para_ptr++);
    width = *(para_ptr++);
    wind_x = *(para_ptr++);
    wind_y = *(para_ptr++);
    wind_orient = *(para_ptr++);
    X = (x2-x1);
    Y = (y2-y1);
    if (X==0)
        rot=PI/2;
    else
        rot = atan2(Y, X);
    test_length=length*SF;
    test_width=width*SF;
    map(rows, cols, test_length, test_width, xi, yi, rot);
}

/*****
/* This function actually maps the window defined by x,y,width,length
/* and orient to the origin with the window length parallel to the
*****/

```

1. The first part of the paper discusses the importance of the study.

2. The second part of the paper discusses the methodology used.

3. The third part of the paper discusses the results of the study.

4. The fourth part of the paper discusses the conclusions.

5. The fifth part of the paper discusses the implications.

6. The sixth part of the paper discusses the limitations.

7. The seventh part of the paper discusses the future research.

8. The eighth part of the paper discusses the references.

9. The ninth part of the paper discusses the appendix.

10. The tenth part of the paper discusses the bibliography.

11. The eleventh part of the paper discusses the index.

12. The twelfth part of the paper discusses the glossary.

13. The thirteenth part of the paper discusses the acknowledgments.

14. The fourteenth part of the paper discusses the disclaimer.

1. The first part of the paper discusses the importance of the study.

2. The second part of the paper discusses the methodology used.

3. The third part of the paper discusses the results of the study.

4. The fourth part of the paper discusses the conclusions.

5. The fifth part of the paper discusses the implications.

6. The sixth part of the paper discusses the limitations.

7. The seventh part of the paper discusses the future research.

8. The eighth part of the paper discusses the references.

9. The ninth part of the paper discusses the appendix.

10. The tenth part of the paper discusses the bibliography.

11. The eleventh part of the paper discusses the index.

12. The twelfth part of the paper discusses the glossary.

13. The thirteenth part of the paper discusses the acknowledgments.

14. The fourteenth part of the paper discusses the disclaimer.

```

/* x-axis of the framstore display.
/.....*/
map(rows,cols,length,width,x,y,orient)
int length,width,x,y;
double orient;
{
    int a,b,xloop,yloop,xpos,ypos;
    double cos_theta,sin_theta;
    cos_theta=cos(orient);
    sin_theta=sin(orient);
    for (yloop=0;yloop<width;yloop++)
    {
        for (xloop=0;xloop<length;xloop++)
        {
            a=x*cos_theta+xloop - sin_theta*yloop;
            b=y*sin_theta+xloop + cos_theta*yloop;
            if (b<512 && b>=0 && a<512 && a>=0)
                dest[yloop][xloop]=indata[b][a];
        }
    }
}

/.....*/
/* File: wframe.c
*/
/* This is the program which is responsible for
*/
/* displaying the highlighted feature windows that are
*/
/* being tested.
/.....*/

#include <stdio.h>
#include <hip_format.h>
#include "vision_box.h"
#include <math.h>
#include "ci.h"
#include "ci_extern"

extern struct header hd;

wframe(rows,cols)
int rows,cols;
{
    unsigned char *outdata;
    int i, strength;
    int left, right;
    int x,y;
    long picsize;

    picsize=cols*rows;
    strength = 255;

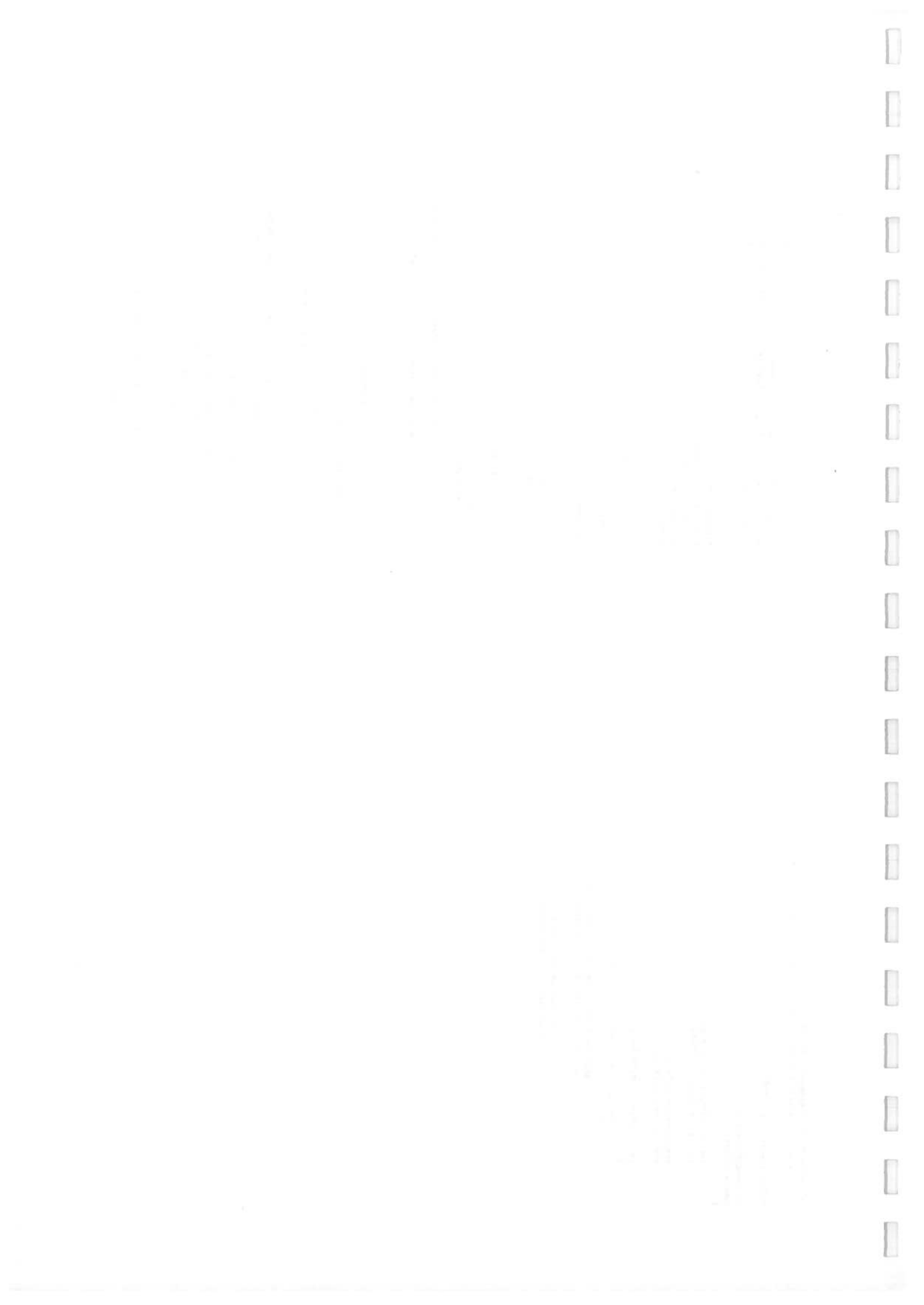
    if ((outdata=(unsigned char *)malloc(picsize))!=(unsigned char *)NULL)
    {
        fprintf(stderr,"can't allocate array\n");
        exit(0);
    }

    /* initialize the frame */
    for (y=0;y<rows;y++)
    {
        for (x=0;x<cols;x++)
            outdata[x+y*cols]=indata[y][x];
    }

    /* set all the pixels between the two intersection points ...
    /* Left and Right of the current scanline line to 255 ...
    for (y=y_min; y<y_max; y++)
    {
        left = scanline[y].left;
        right = scanline[y].right;
        for (x=left; x<right; x++)
            outdata[x+y*cols]=255-indata[y][x];
    }

    /* invert area of screen where window is to be displayed */
    for (y=0; y<test_width; y++)

```



```

{
    for (x=0; x<test_length; x++)
        outdata[x*ycols]-dest[y][x];
}
for (y=0; y<test_width; y++)
    outdata[test_length*ycols]=255;
for (x=0; x<test_length; x++)
    outdata[x*test_width*cols]-255;

vb_putblock(outdata,0,0,cols,rows);
free(outdata);
}

/*****
/* show c
/* showerrors() is used to display a rectangular window outline on the c
/* framstore's second image plane. This is used to build up an image c
/* containing all the components that failed a test outlined c
/*****/

#include <stdio.h>
#include "vision_box.h"
#include "ci.h"
#include "modal.h"
#include "ci_extern"

showerrors(vertex)
struct coordinate vertex[4];
{
    int x0,y0,x1,y1,x2,y2,x3,y3;

    fanum=1;
    vb_plane(fanum);

    x0=vertex[0].x; x1=vertex[1].x; x2=vertex[2].x; x3=vertex[3].x;
    y0=vertex[0].y; y1=vertex[1].y; y2=vertex[2].y; y3=vertex[3].y;

    if ((x0<x1) || ((x0==x1) && (y0<y1))) vb_line(x0,y0,x1,y1,255);
    else vb_line(x1,y1,x0,y0,255);

    if ((x1<x2) || ((x1==x2) && (y1<y2))) vb_line(x1,y1,x2,y2,255);
    else vb_line(x2,y2,x1,y1,255);

    if ((x2<x3) || ((x2==x3) && (y2<y3))) vb_line(x2,y2,x3,y3,255);
    else vb_line(x3,y3,x2,y2,255);

    if ((x3<x0) || ((x3==x0) && (y3<y0))) vb_line(x3,y3,x0,y0,255);
    else vb_line(x0,y0,x3,y3,255);

    fanum=0;
    vb_plane(fanum);
}

```

1998-1999  
at the University of  
California, Berkeley  
Department of  
Psychology  
101

```

/.....
/* pin c
/* This file contains the pin test code
/.....

#include "ci.h"
#include "ci_extern"
#include "vision_box.h"

#define THRESH=10
/.....
/* pincount() will count the number of intensity peaks along a scan
/* line which goes through the middle of the feature window. It then
/* compares the peak positions with those found on a second scan line
/* going across the bottom of the window. If any matches are found
/* then it is assumed that a bent pin has been found
/.....

pincount(name,params, scale, display)
char *name;
PARAM_LS_PTR params;
float scale;
int display;
{
    int scan_ypos, pincount=0, average=0, threshold;
    int pin_pos[512], nopin_pos[512];
    int length, width, count, size, loop, temp1, temp2;
    int ok=TRUE;

    /* get paramaters required by test and scale those that require it */
    length = (params->para)*scale;
    params=params->parameter;
    width = (params->para)*scale;
    params=params->parameter;
    count = params->para;
    params=params->parameter;
    size = params->para*scale;

    /* work out average intensity along first scan line */
    scan_ypos=width/2;
    for (loop=0; loop<length; loop++)
    {
        if (display) vb_putpix(loop,scan_ypos,255);
        average+=dest[scan_ypos][loop];
    }
    average=average/length;
    threshold=average*THRESH;

    /* make first scan */
    pincount = pinscan(length,scan_ypos,threshold,size,pin_pos);
    if (pincount != count)
    {
        ok=FALSE;
        printf("Pin count failed: count=%d\n",pincount);

```

```

}
else
    printf("Pin count succeeded\n\n");
/* make second scan at bottom of window */
pincount = pinscan(length,width-1,threshold,size,nopin_pos);
/* compare peak positions from two scans */
/* If size of peak and position match then give warning */
loop=0;
while (loop<length)
{
    if (pin_pos[loop]==TRUE)
    {
        if (nopin_pos[loop]==TRUE)
        {
            temp1=temp2-loop;
            do {
                temp1++;
            } while (pin_pos[temp1]==TRUE);
            do {
                temp2++;
            } while (nopin_pos[temp2]==TRUE);
            if (abs(temp1-temp2)<2)
            {
                if (display) vb_line(loop,0,loop,width-1,255);
                printf("Possible Bent Pin at pos %d\n",loop);
                ok=FALSE;
            }
            loop = (temp1>temp2) ? temp1 : temp2;
        }
        else if (nopin_pos[loop+1] != TRUE)
        {
            do {
                loop++;
            } while (pin_pos[loop]==TRUE);
        }
        else
            loop++;
    }
    else
        loop++;
}
return(ok);
/.....
/* This function makes the scan along the position given to it in ypos
/* If any peaks are found then they are stored in the array peakpos
/* The peak has to last for at least 'size' pixels and be above the
/* intensity value of threshold.
/.....
int pinscan(xsize,ypos,threshold,size,peakpos)
int xsize,ypos,threshold,size,peakpos[512];

```

Faint, illegible text at the top of the page, possibly a header or title.

Faint, illegible text at the bottom of the page, possibly a footer or a list of items.





```

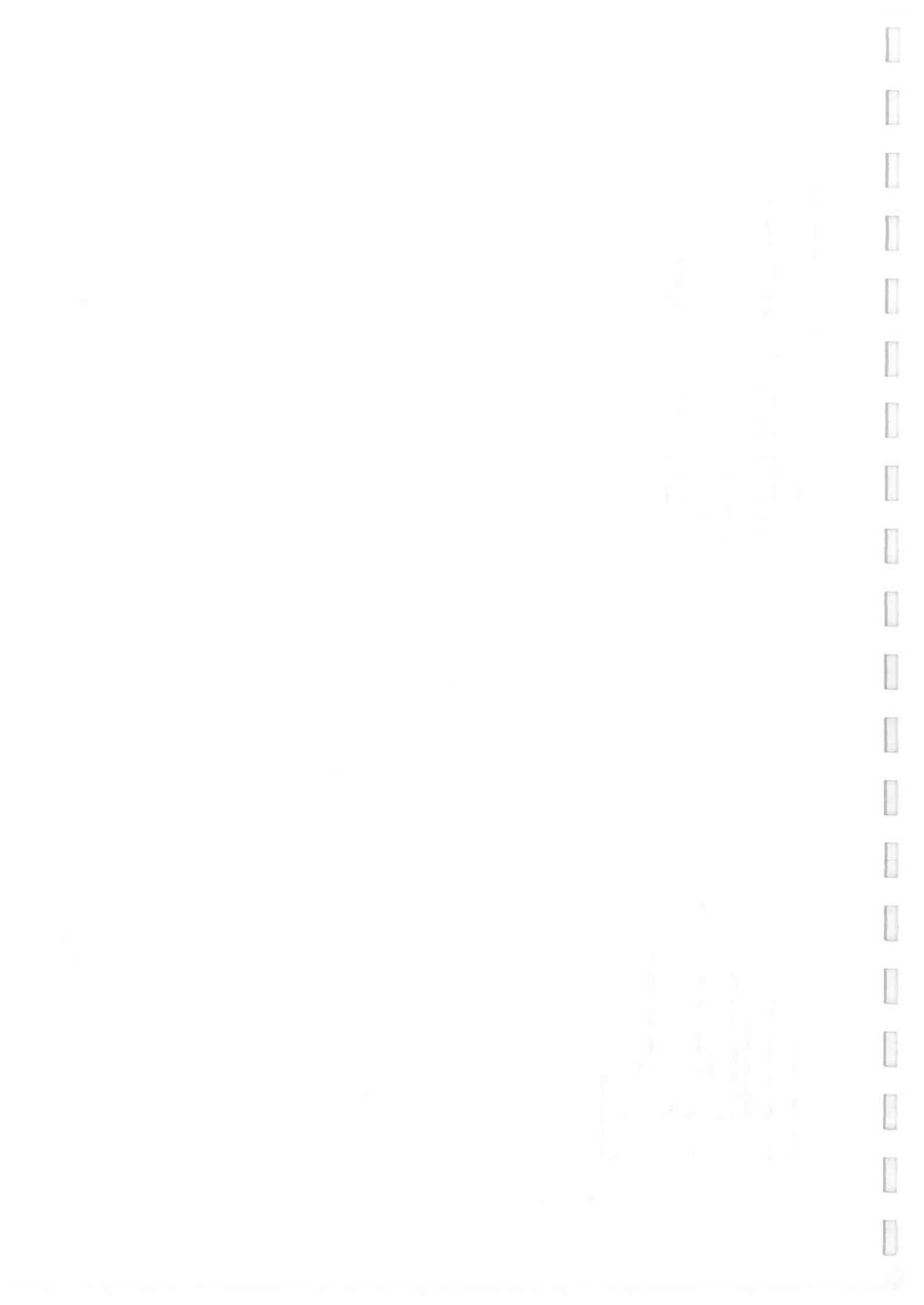
(
    int loop, peak=0, pincount=0;
    for (loop=0, loop<ysize; loop++)
    {
        peakpos [loop]=FALSE;
        if (dest[ypos] [loop]>threshold)
        {
            peak++;
            if (peak>size) peakpos [loop]=TRUE;
        }
        else
        {
            if (peak>size) pincount++;
            peak=0;
        }
    }
    return(pincount);
}
)

/*.....
/* length.c
/*
/* This is a dummy inspection test which was used during the
/* testing of the program. It prints out a message and then terminates. */
/*.....

#include "c1.h"

int length()
{
    printf("Calling length test: Not yet implemented\n");
    return(TRUE);
}

```



```

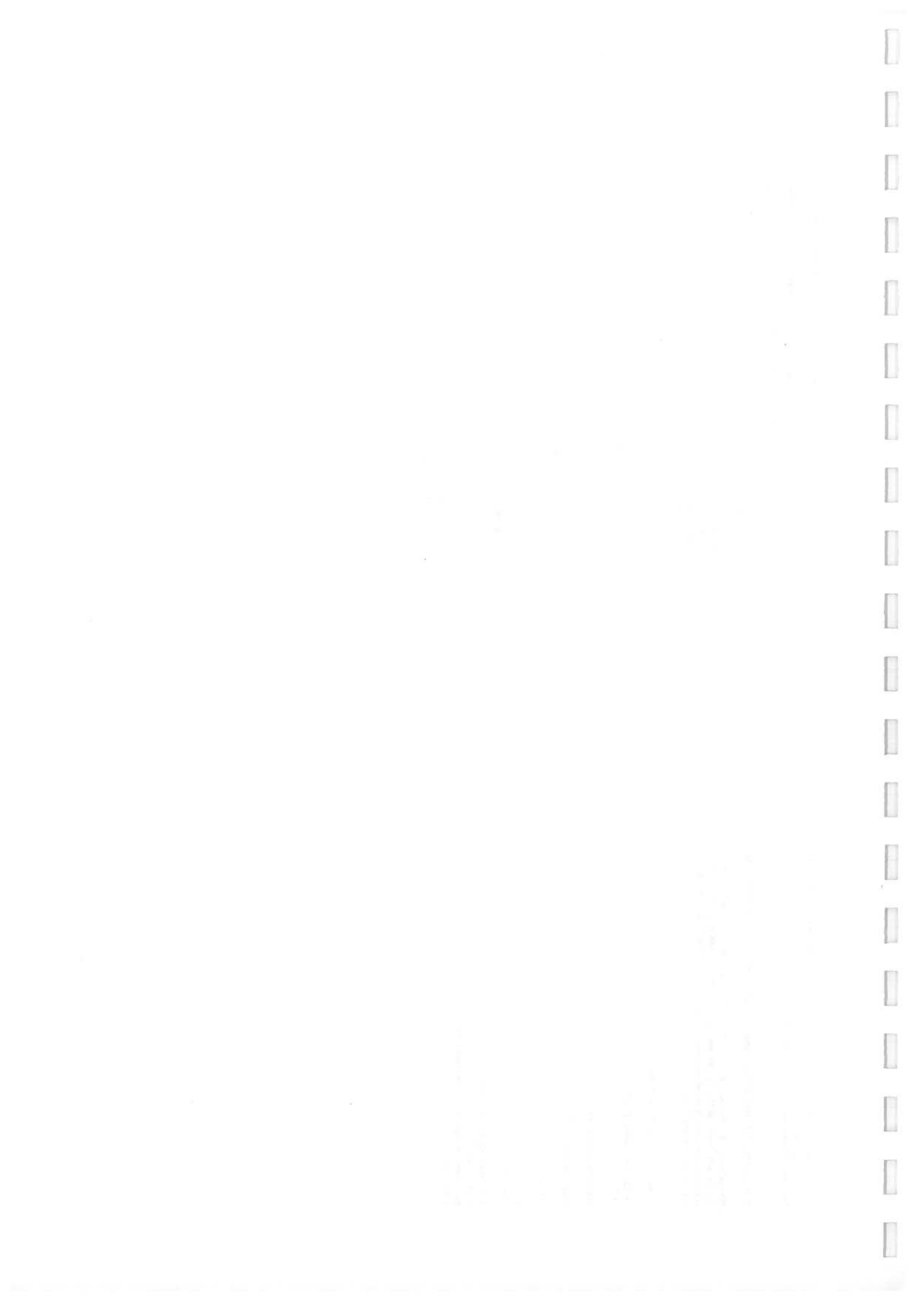
/*****
/*  malloc.c
/*****
/* Copyright (c) 1982 Michael Landy, Yoav Cohen, and George Sperling
Disclaimer: No guarantees of performance accompany this software,
nor is any responsibility assumed on the part of the authors. All the
software has been tested extensively and every effort has been made to
insure its reliability. */
/* malloc.c - HIPPL Picture Format core allocation
* Michael Landy 2/1/82
*/
#include <stdio.h>
char *malloc(i,j)
int i,j;
{
char *k;
k = (char *) malloc(i,j);
if(k == NULL)
perr("Not enough core available.");
return(k);
}

```

```

/*****
/*  pread.c
/*****
/* This contains functions for reading a byte format file(image) into */
/* an array. This was taken from David Chan's code.
/*****
#include <stdio.h>
pread(fd,buf,count)
FILE *fd;
char *buf;
int count;
{
int cnt,r;
cnt=count;
while (cnt>0)
{
*(buf++)=getc(fd);
cnt--;
}
return(count-cnt);
}
empty(buf,count)
char *buf;
int count;
{
int cnt;
cnt=count;
while (cnt>0)
{
*(buf++)='\0';
cnt--;
}
return(count-cnt);
}

```



```

/*****
/* read_header.c
/* This is used to read the header information on an image file. This */
/* was not written by me.
/*****

#include "ipl_format.h"
#include <stdio.h>
#define LINES 100
#define LINELENGTH 132

static char *save[LINES];
static int s1max[LINES];
static int lalloc = 0;
extern char *malloc();

fread_header(fd,hd)
struct header *hd;
FILE *fd;
{
    int lineno, len, i;
    char *s;
    char *getline();

    if(!lalloc) {
        save[0] = malloc(LINELENGTH, sizeof (char));
        s1max[0] = LINELENGTH;
        lalloc = 1;
    }
    getline(fd,save[0],s1max[0]);
    hd->orig_name = malloc(strlen(save[0])*i, sizeof (char));
    strcpy(hd->orig_name,save[0]);
    getline(fd,save[0],s1max[0]);
    hd->seq_name = malloc(strlen(save[0])*i, sizeof (char));
    strcpy(hd->seq_name,save[0]);
    hd->num_frames = fscanf(fd);
    getline(fd,save[0],s1max[0]);
    hd->orig_data = malloc(strlen(save[0])*i, sizeof (char));
    strcpy(hd->orig_data,save[0]);
    hd->rows = fscanf(fd);
    hd->cols = fscanf(fd);
    hd->bits_per_pixel = fscanf(fd);
    hd->bit_packing = fscanf(fd);
    hd->pixel_format = fscanf(fd);
    lineno = 0;
    len = 1;
    getline(fd,save[0],s1max[0]);
    s = save[0];
    while((c = *s) != '\0') {
        len += strlen(save[lineno]);
        lineno++;
        if (lineno >= LINES)
            perror("Too many lines in header history");
        if(lineno >= lalloc) {
            save[lineno] = malloc(LINELENGTH, sizeof (char));
            s1max[lineno] = LINELENGTH;
            lalloc++;
        }
        hd->seq_desc = malloc(len, sizeof (char));
        *hd->seq_desc = '\0';
        for (i=0;i<lineno;i++)
            strcat(hd->seq_desc,save[i]);
    }

    char *getline(fd,s,1)

    char *s;
    FILE *fd;
    int *i;

    {
        int i,m;
        char c,*s1,*s2;

        i = 0;
        s1 = *s;
        m = *i;
        while((c =getc(fd)) != EOF && c != '\n') {
            if (m-- <= 2) {
                s2 = malloc(LINELENGTH+1, sizeof (char));
                strcpy(s2,*s);
                *s = *s2;
                *i += LINELENGTH;
                m = LINELENGTH;
                s1 = s2 + strlen(s2);
            }
            *s1++ = c;
        }
        if (c == '\n') {
            *s1++ = '\n';
            *s1 = '\0';
            return(*s);
        }
        perror("Unexpected EOF while reading header.");
    }
}

```



```

}
dfscanf(fd
FILE *fd,
{
    int i;
    getline(&line, &linebuf, fd);
    sscanf(line, "%d", &i);
    return(1);
}
}

/*
 * ci b
 * All the structures that are used by YACC for creating the
 * parse trees are defined here. There are a few general definitions too.
 */
#include <stdio.h>

#define TRUE 1
#define FALSE 0
#define PI 3.141592654
#define POBN_ENTRIES 3
#define WIN_ENTRIES 5
#define ROWS 512

typedef struct parm_ls {
    float parm;
    char *name;
    struct parm_ls *parnext;
} PARM_LS;

typedef PARM_LS *PARM_LS_PTR;

typedef struct insp {
    int insptype;
    PARM_LS_PTR insparm;
} INSP;

typedef INSP *INSP_PTR;

typedef struct insp_ls {
    INSP_PTR testptr;
    struct insp_ls *inspnext;
} INSP_LS;

typedef INSP_LS *INSP_LS_PTR;

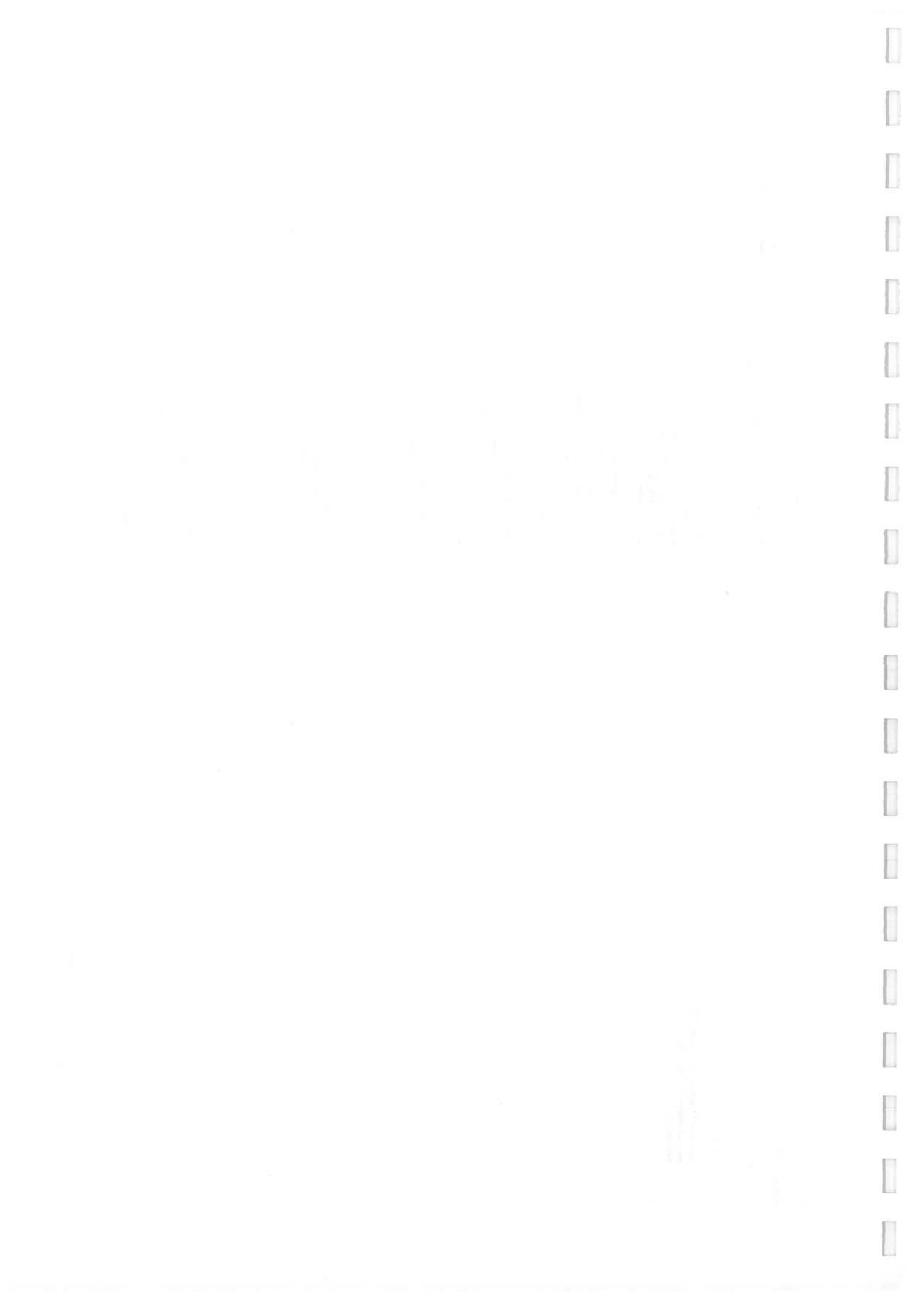
typedef struct feat {
    int feattype;
    char *featname;
    int wind[WIN_ENTRIES];
    INSP_LS_PTR featinsp;
    struct comp *compfeat;
    struct repeat *rpt;
} FEAT;

typedef FEAT *FEAT_PTR;

typedef struct feat_ls {
    FEAT_PTR feat;
    struct feat_ls *featnext;
} FEAT_LS;

typedef FEAT_LS *FEAT_LS_PTR;

```





```

typedef struct pair {
    char *name;
    int value;
    struct pair *pairnext;
} PAIR;

typedef PAIR *PAIR_PTR;

typedef struct comp_defn {
    char *compdefnm;
    FEAT_LS_PTR featls;
    INSP_LS_PTR testls;
} COMP_DEF;

typedef COMP_DEF *COMP_DEF_PTR;

typedef struct repeat {
    char *axis;
    int times,step;
} REPT;

typedef REPT *REPEAT_PTR;

typedef struct comp {
    char *comptype;
    char compnm[50];
    PAIR_PTR pairls;
    int wind[POSS_ENTRIES];
    COMP_DEF_PTR definition;
    struct comp *parent;
} COMP;

typedef COMP *COMP_PTR;

typedef struct comp_ls {
    COMP_PTR compentry;
    REPEAT_PTR rpt;
    struct comp_ls *component;
} COMP_LS;

typedef COMP_LS *COMP_LS_PTR;

typedef struct circ {
    char *circnm;
    COMP_LS_PTR compls;
    INSP_LS_PTR testls;
} CIRC;

typedef CIRC *CIRC_PTR;

typedef struct obj_ls {
    CIRC_PTR circptr;
    COMP_DEF_PTR componentptr;
    struct obj_ls *objnext;
} OBJ_LS;

```

```

typedef OBJ_LS *OBJ_LS_PTR;

OBJ_LS_PTR obj_head;

struct rastercan {
    int left;
    int right;
} scanline[ROWS];

struct bound {
    int x;
    int y;
};

unsigned char *indata[ROWS];
unsigned char *dest[ROWS];

int Xref,Yref;
int rotation;
int x_min,x_max,y_min,y_max;
int test_length,test_width;

```

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000

1000

```

/*****
/* model.h
/* Any new features or test names have to be put here.
/* Feature names are numbered from 1000-1999.
/* Test names are numbered from 2000 upwards
*****/

#define ICbody 1000
#define IClegs 1001
#define CAPACITOR 1002
#define RESISTOR 1003
#define LENGTH 2000
#define SEPARATION 2001
#define RELORIENT 2002
#define PINCOUNT 2003

```

```

/*****
/* yacc.y
/* This is the YACC grammar file.
*****/

%{
#include "ci.h"
#include "ctype.h"
#include "model.h"
#include <stdio.h>
#include <string.h>
int i,arity;
OBJ_LS_PTR obj_head;
char *malloc();
char name[80];
%}

%start grammar
%union {
int ival;
float fval;
char *sval;
CIRC_PTR circ;
OBJ_LS_PTR objt_ls;
COMP_PTR comp;
COMP_LS_PTR comp_ls;
COMP_DEF_PTR comp_defn;
REPEAT_PTR repeat;
FEAT_LS_PTR feat_ls;
FEAT_PTR feat;
INSP_LS_PTR insp_ls;
INSP_PTR insp;
PARAM_PTR para_ls;
int place[PUSH_ENTRIES];
int win[WIN_ENTRIES];
}

```

```

%token <fval> FLOAT
%token <ival> INTEGER FEATURE TYPE
%token <sval> NAME
%token END CIRCUIT ENDCIRCUIT COMPONENT ENDCOMPONENT POSITION ORIENT
%token COMPARE WITH WINDOW AT REPEAT STEP FOR ALONG ENDREPEAT

%type <objt_ls> list
%type <circ> circuit
%type <comp_ls> component_list
%type <comp> component
%type <comp_defn> component_def
%type <repeat> repeatloop
%type <feat_ls> feature_list
%type <feat> feature
%type <insp_ls> inspection_list
%type <insp> inspection
%type <para_ls> para_list
%type <fval> value
%type <place> place
%%

```

1. The first part of the document discusses the importance of maintaining accurate records of all transactions. It emphasizes that this is crucial for ensuring the integrity of the financial statements and for providing a clear audit trail. The text also mentions that proper record-keeping is essential for identifying trends and anomalies in the data.

2. The second part of the document focuses on the role of internal controls in preventing fraud and errors. It highlights that a strong internal control system is necessary to ensure that all transactions are properly authorized and recorded. The text also notes that internal controls should be designed to be effective and efficient, and should be regularly reviewed and updated.

3. The third part of the document discusses the importance of transparency and communication in financial reporting. It emphasizes that clear and concise communication is essential for ensuring that all stakeholders have a clear understanding of the company's financial performance. The text also mentions that transparency is a key factor in building trust and confidence in the company's financial statements.

4. The fourth part of the document discusses the importance of compliance with applicable laws and regulations. It emphasizes that the company must ensure that all financial reporting is done in accordance with the relevant accounting standards and regulations. The text also mentions that compliance is essential for avoiding legal and financial penalties.

```

error("Can't allocate memory");
##->circm=#2;
##->compl=#4;
##->testis=#6;
}

component_list:
repeatloop component ENDREPEAT
{
if (($= (COMP_LS_PTR) malloc(sizeof(COMP_LS)))==NULL)
error("Can't allocate memory");
##->rpt=#1;
##->compeny=#2;
##->compxnt=NULL;
}
repeatloop component ENDREPEAT component_list
{
if (($= (COMP_LS_PTR) malloc(sizeof(COMP_LS)))==NULL)
error("Can't allocate memory");
##->rpt=#1;
##->compeny=#2;
##->compxnt=#4;
}
component
{
if (($= (COMP_LS_PTR) malloc(sizeof(COMP_LS)))==NULL)
error("Can't allocate memory");
##->compeny=#1;
##->compxnt=NULL;
##->rpt=NULL;
}
component component_list
{
if (($= (COMP_LS_PTR) malloc(sizeof(COMP_LS)))==NULL)
error("Can't allocate memory");
##->compeny=#1;
##->compxnt=#2;
##->rpt=NULL;
}
}

```

```

repeatloop:
REPEAT value STEP value ALONG NAME FOR
{
if (($= (REPEAT_PTR) malloc(sizeof(REPT)))==NULL)
error("Can't allocate memory");
##->times=#2;
if ($2<1) error("Cannot repeat component less than once");
##->axis=#6;
if ((strcmp($6,"x")!=0) && (strcmp($6,"y")!=0))
printf("%d.%d \n",strcmp($6,"x"),strcmp($6,"y"));
error("Invalid Axis for repeat: '$6');
##->step=#4;
}

```

88

```

Grammar:
list END
{
obj_head=#1;
}

list:
circuit
{
if (($= (OBJ_LS_PTR) malloc(sizeof(OBJ_LS)))==NULL)
error("Can't allocate memory");
##->circptr=#1;
##->compenyptr=NULL;
##->objnext=NULL;
}
component_def
{
if (($= (OBJ_LS_PTR) malloc(sizeof(OBJ_LS)))==NULL)
error("Can't allocate memory");
##->compenyptr=#1;
##->circptr=NULL;
##->objnext=NULL;
}
circuit list
{
if (($= (OBJ_LS_PTR) malloc(sizeof(OBJ_LS)))==NULL)
error("Can't allocate memory");
##->circptr=#1;
##->compenyptr=NULL;
##->objnext=#2;
}
component_def list
{
if (($= (OBJ_LS_PTR) malloc(sizeof(OBJ_LS)))==NULL)
error("Can't allocate memory");
##->compenyptr=#1;
##->circptr=NULL;
##->objnext=#2;
}
circuit:
CIRCUIT NAME WITH component_list ENDCIRCUIT
{
if (($= (CIRC_PTR) malloc(sizeof(CIRC)))==NULL)
error("Can't allocate memory");
##->circm=#2;
##->compl=#4;
##->testis=#6;
}
CIRCUIT NAME WITH component_list COMPARE inspection_list ENDCIRCUIT
{
if (($= (CIRC_PTR) malloc(sizeof(CIRC)))==NULL)

```

87

1. The first part of the document is a list of names and addresses. The names are listed in the first column, and the addresses are listed in the second column. The names are: [faint names]. The addresses are: [faint addresses].

2. The second part of the document is a list of names and addresses. The names are listed in the first column, and the addresses are listed in the second column. The names are: [faint names]. The addresses are: [faint addresses].

3. The third part of the document is a list of names and addresses. The names are listed in the first column, and the addresses are listed in the second column. The names are: [faint names]. The addresses are: [faint addresses].

4. The fourth part of the document is a list of names and addresses. The names are listed in the first column, and the addresses are listed in the second column. The names are: [faint names]. The addresses are: [faint addresses].

5. The fifth part of the document is a list of names and addresses. The names are listed in the first column, and the addresses are listed in the second column. The names are: [faint names]. The addresses are: [faint addresses].

```

feature feature_list
{
  if (($-(FEAT_LS_PTR) malloc(sizeof(FEAT_LS)))-=NULL)
    error("Can't allocate memory");
  $$->feat=$1;
  $$->featnext=$2;
}

feature:
FEATURE: NAME window inspection_list
{
  if (($-(FEAT_PTR) malloc(sizeof(FEAT)))-=NULL)
    error("Can't allocate memory");
  $$->featm=$3;
  $$->compfeat=NULL;
  $$->featinsp=$6;
  for(i=0; i<WIN_ENTRIES; i++)
    $$->wind[i]=$4[i];
}
|
repeatloop component ENDREPEAT
{
  if (($-(FEAT_PTR) malloc(sizeof(FEAT)))-=NULL)
    error("Can't allocate memory");
  $$->featm=NULL;
  $$->rpt=$1;
  $$->compfeat=$2;
}
|
component
{
  if (($-(FEAT_PTR) malloc(sizeof(FEAT)))-=NULL)
    error("Can't allocate memory");
  $$->featm=NULL;
  $$->compfeat=$1;
  $$->rpt=NULL;
}
;

window:
WINDOW('value','value'). AT('value','value','value').
{
  $$[0]=$3;
  $$[1]=$6;
  $$[2]=$9;
  $$[3]=$11;
  $$[4]=$13;
}
;

inspection_list:
inspection
{
  if (($-(INSP_LS_PTR) malloc(sizeof(INSP_LS)))-=NULL)
    error("Can't allocate memory");
  $$->testptr=$1;
  $$->inspnext=NULL;
}
;

```

```

}
;

component:
COMPONENT NAME NAME posn
{
  if (($-(COMP_PTR) malloc(sizeof(COMP)))-=NULL)
    error("Can't allocate memory");
  $$->comptype=$2;
  for(i=0; i<POSM_ENTRIES; i++)
    $$->wind[i]=$4[i];
  strcpy(name,$3);
  strcat(name," ");
  strcpy($$->compom.name);
  $$->pairis=NULL;
  $$->parent=NULL;
}
;

posn:
POSITION('value','value').
ORIENT('value').
{
  $$[0]=$3;
  $$[1]=$6;
  $$[2]=$9;
}
;

component_def:
COMPONENT NAME WITH feature_list ENDCOMPONENT
{
  if (($-(COMP_DEF_PTR) malloc(sizeof(COMP_DEF)))-=NULL)
    error("Can't allocate memory");
  $$->compdefm=$2;
  $$->featl=$4;
  $$->testle=NULL;
}
|
COMPONENT NAME WITH feature_list COMPARE inspection_list ENDCOMPONENT
{
  if (($-(COMP_DEF_PTR) malloc(sizeof(COMP_DEF)))-=NULL)
    error("Can't allocate memory");
  $$->compdefm=$2;
  $$->featl=$4;
  $$->testle=$6;
}
;

feature_list:
feature
{
  if (($-(FEAT_LS_PTR) malloc(sizeof(FEAT_LS)))-=NULL)
    error("Can't allocate memory");
  $$->feat=$1;
  $$->featnext=NULL;
}
;

```

1. The first part of the document discusses the importance of maintaining accurate records of all transactions.

2. It is essential to ensure that all entries are supported by proper documentation and receipts.

3. Regular audits should be conducted to verify the accuracy of the records and identify any discrepancies.

4. The second part of the document outlines the procedures for handling and storing financial records.

5. All records should be stored in a secure and accessible location, and backed up regularly.

6. It is also important to establish a clear policy regarding the retention and disposal of financial records.

7. The third part of the document provides a detailed overview of the accounting system used by the organization.

8. This includes a description of the software used, the chart of accounts, and the reporting structure.

9. The fourth part of the document discusses the role of the accounting department in supporting the organization's operations.

10. The department is responsible for providing accurate financial information to management and other stakeholders.

11. Finally, the document concludes with a summary of the key points and a call to action for all employees.

12. The first part of the document discusses the importance of maintaining accurate records of all transactions.

13. It is essential to ensure that all entries are supported by proper documentation and receipts.

14. Regular audits should be conducted to verify the accuracy of the records and identify any discrepancies.

15. The second part of the document outlines the procedures for handling and storing financial records.

16. All records should be stored in a secure and accessible location, and backed up regularly.

17. It is also important to establish a clear policy regarding the retention and disposal of financial records.

18. The third part of the document provides a detailed overview of the accounting system used by the organization.

19. This includes a description of the software used, the chart of accounts, and the reporting structure.

20. The fourth part of the document discusses the role of the accounting department in supporting the organization's operations.

21. The department is responsible for providing accurate financial information to management and other stakeholders.

22. Finally, the document concludes with a summary of the key points and a call to action for all employees.



```

NAME
{
  if (($-(PARM_LS_PTR) malloc(sizeof(PARM_LS)))==NULL)
    error("Can't allocate memory");
  $$->parm=0;
  $$->name=$1;
  $$->parnext=NULL;
  }
|
  NAME '.' param_list
{
  if (($-(PARM_LS_PTR) malloc(sizeof(PARM_LS)))==NULL)
    error("Can't allocate memory");
  $$->parm=0;
  $$->name=$1;
  $$->parnext=$2;
  }
;

value:
  INTEGER = {$0-$1;}
  |
  FLOAT = {$0-$1;}
  ;

%%
#include "lex.c"

count(head)
PARM_LS_PTR head;
{
  if (head==NULL)
    return(0);
  else
    return(1+count(head->parnext));
}

```

```

|
inspection inspection_list
{
  if (($-(INSP_LS_PTR) malloc(sizeof(INSP_LS)))==NULL)
    error("Can't allocate memory");
  $$->testptr=$1;
  $$->insexpnext=$2;
  }
|
  error
  {yyerror;}
;

inspection:
  TYPE '(' param_list ')'
  {
    arity=count($2);
    switch($1) {
      case SEPARATION:
      case RELORIENT:
      case PINGCOUNT:
        if (arity!=4)
          error("Wrong Arity");
        else
          break;
      case LENGTH:
        if (arity!=2)
          error("Wrong Arity");
        else
          break;
      default:
        error("Invalid test");
    }
  }
  if (($-(INSP_PTR) malloc(sizeof(INSP)))==NULL)
    error("Can't allocate memory");
  $$->insptype=$1;
  $$->insexp=$2;
  }
;

param_list:
  value
  {
    if (($-(PARM_LS_PTR) malloc(sizeof(PARM_LS)))==NULL)
      error("Can't allocate memory");
    $$->parm=$1;
    $$->name=NULL;
    $$->parnext=NULL;
  }
|
  value '.' param_list
  {
    if (($-(PARM_LS_PTR) malloc(sizeof(PARM_LS)))==NULL)
      error("Can't allocate memory");
    $$->parm=$1;
    $$->name=NULL;
    $$->parnext=$2;
  }
|

```

1. 1948

2. 1949

3. 1950

4. 1951

5. 1952

6. 1953

7. 1954

8. 1955



```

) (\n) ;
return((int) yytext);
%%

char *makename(p)
char *p;
{
    char *start;
    start = nameend;
    while (*p != '\0')
        { *nameend++ = *p++;
          if (nameend >= name + NAMELENGTH)
              error("name table overflow");
        }
    *nameend++ = '\0';
    return(start);
}

```

```

/*****
/ * lex.l
/ * This is the lexical analyser written using lex
/ *****/

%{
double stof();

int yytppchr;
# undef input
# define ngetc() (yytppchr=getc(yyin), putc(yytppchr, stdout), yytppchr)
# define input() (((yytchar=yytpptr>yyabuf7U(---yytpr):ngetc())==10?yylineno++:yytchar):yytchar)--EDF?

#define NAMELENGTH 10000
char names[NAMELENGTH];
char *nameend = names;

char *makename();
%}

CIRCUIT;
return(CIRCUIT);
return(ENDCIRCUIT);
return(COMPONENT);
return(ENDCOMPONENT);
return(REPEAT);
return(STEP);
return(FOR);
return(ALONG);
return(ENOREPEAT);
return(COMPARE);
return(POSITION);
return(ORIENT);
return(WITH);
return(WINDOW);
return(AT);
return(EMD);
{yyival.ival = ICbody; return(FEATURE);}
{yyival.ival = IClegs; return(FEATURE);}
{yyival.ival = CAPACITOR; return(FEATURE);}
{yyival.ival = RESISTOR; return(FEATURE);}
{yyival.ival = LENGTH; return(TYPE);}
{yyival.ival = PINCOUNT; return(TYPE);}
{yyival.ival = SEPARATION; return(TYPE);}
{yyival.ival = RELORIENT; return(TYPE);}
{yyival.sival = makename(yytext); return(NAME);}
-? "[A-Za-z][A-Za-z0-9_]*"
-? "[0-9]+[A-Za-z0-9_]*"
-? "[0-9]+[A-Za-z0-9_]*"
/* ** */ /* delete comments */
int c;
do {
while (input() != '\n') {}
c = input();
if (c == '/') break;
ungetc(c);
} while (1);

```

1950

1951

1952

1953

1954

1955

1956

1957

1958

1959

1960

1961

1962

1963

1964

1965

1966

1967

1968

1969

1970

1971

1972

1973

1974

1975

1976

1977

1978

1979

1980

1981

1982

1983

1984

1985

1986

1987

1988

1989

1990

1991

1992

1993

1994

1995

1996

1997

1998

1999

2000

2001

2002

2003

2004

2005

2006

2007

2008

2009

2010

2011

2012

2013

2014

2015

2016

2017

2018

2019

2020

2021