

1999

## **Integrative Approach to Online Quality Management: Process Control and Packout Verification in an Intelligent Manufacturing Workcell.**

Turuvekere R. Sethumadhava  
*Louisiana State University and Agricultural & Mechanical College*

Follow this and additional works at: [https://digitalcommons.lsu.edu/gradschool\\_disstheses](https://digitalcommons.lsu.edu/gradschool_disstheses)

---

### **Recommended Citation**

Sethumadhava, Turuvekere R., "Integrative Approach to Online Quality Management: Process Control and Packout Verification in an Intelligent Manufacturing Workcell." (1999). *LSU Historical Dissertations and Theses*. 7057.

[https://digitalcommons.lsu.edu/gradschool\\_disstheses/7057](https://digitalcommons.lsu.edu/gradschool_disstheses/7057)

This Dissertation is brought to you for free and open access by the Graduate School at LSU Digital Commons. It has been accepted for inclusion in LSU Historical Dissertations and Theses by an authorized administrator of LSU Digital Commons. For more information, please contact [gradetd@lsu.edu](mailto:gradetd@lsu.edu).

## **INFORMATION TO USERS**

**This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.**

**The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.**

**In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.**

**Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.**

**Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.**

**Bell & Howell Information and Learning  
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA**

**UMI<sup>®</sup>**  
**800-521-0600**



**INTEGRATIVE APPROACH TO ONLINE QUALITY MANAGEMENT:  
PROCESS CONTROL AND PACKOUT VERIFICATION IN AN  
INTELLIGENT MANUFACTURING WORKCELL**

**A Dissertation**

**Submitted to the Graduate Faculty of the  
Louisiana State University and  
Agricultural and Mechanical College  
in partial fulfillment of the  
requirements for the degree of  
Doctor of Philosophy**

**in**

**The Interdepartmental Program in Engineering Science**

**by**

**Turuvekere R. Sethumadhava  
B.S., University of Mysore, 1985  
M.S., Louisiana State University, 1991  
M.S., Louisiana State University, 1994  
December, 1999**

**UMI Number: 9951617**

**UMI<sup>®</sup>**

---

**UMI Microform 9951617**

**Copyright 2000 by Bell & Howell Information and Learning Company.**

**All rights reserved. This microform edition is protected against  
unauthorized copying under Title 17, United States Code.**

---

**Bell & Howell Information and Learning Company  
300 North Zeeb Road  
P.O. Box 1346  
Ann Arbor, MI 48106-1346**

## **ACKNOWLEDGEMENTS**

I would like to express my sincere appreciation to all of the people that have helped me to make this dissertation possible. In particular, I gratefully acknowledge the guidance and support of my major advisor, Dr. Arthur Sterling. He has influenced and supported my career in uncountable occasions. I would also like to thank the members of my thesis committee: Dr. Ken Keys, Dr. Bush Jones, Dr. Donald Kraft, Dr. Lawrence Mann, Dr. Robert Holmes and Dr. Armando Corripio for their valuable input and support. In particular, I am indebted to Dr. Ken Keys for his professional and personal advice.

I would also like to thank Dr. Stephen Dodd and Cheryl Crowder for their support and understanding. I would like to thank all my friends at LSU. Their encouragement and true friendship made my years at LSU unforgettable.

I am grateful to my parents, my brothers, sister, sister-in-laws and brother-in-law. They are always there when I need them. I appreciate their support and understanding. Most importantly, I would like to thank my wife, Suparna and daughter Spandana for their love, patience, and encouragement throughout the years of my graduate study. I really could not have asked for anything more. It is to them that I dedicate this thesis.

## TABLE OF CONTENTS

ACKNOWLEDGEMENTS .....	ii
LIST OF TABLES .....	vi
LIST OF FIGURES .....	vii
ABSTRACT .....	ix
CHAPTER	
1. INTRODUCTION .....	1
Motivation .....	3
Research Objectives and Methodology .....	5
Intelligent Manufacturing Systems .....	6
Establishing Integrated Intelligent Systems .....	8
Benefits of CIMS and AMT .....	9
Direct benefits/Costs: .....	9
Indirect benefits/Costs: .....	9
Intangible benefits/Costs: .....	10
Overview of Process Control .....	10
Intelligent Manufacturing Control .....	12
Flexible Manufacturing System and Computer Integrated System .....	13
Organization of the Dissertation .....	14
2. INDUSTRY AND OPERATION BACKGROUND .....	15
Pilot Manufacturing Workcell Overview .....	16
Circuit Board Manufacturing Area .....	16
Final Assembly Module .....	20
3. MODELING AND SIMULATION .....	24
Background .....	24
The Manufacturing Cell Design Criteria .....	28
Methodology .....	31
Establish Objectives & Scope .....	31
Collect Data & Build Model .....	31
Sensitivity Analysis .....	36
Detailed analysis .....	36
Layout Design .....	36
Analysis and Recommendations .....	45
4. SYSTEM ARCHITECTURE, INTEGRATION AND OPERATION .....	49
Factory Information System Overview .....	49

The Base System.....	52
Modeling a factory .....	54
Modeling in-process material .....	54
Tracking in-process material .....	54
Viewing history of in-process material (WIP history).....	55
Tracking parametric data.....	55
Quarantining and retrieving in-process material .....	56
Statistical Process Control (SPC) .....	56
Data Collection .....	57
Charting .....	57
Alarming.....	58
Paperless Repair.....	58
Packing.....	58
Automatic Change Over .....	58
Equipment Monitoring.....	59
Component Traceability .....	60
Prior Step Verification.....	61
Scheduling Execution .....	62
Alarm Management .....	63
Operational Report.....	63
5. PACKOUT VERIFICATION SYSTEM.....	66
New Packout Process.....	66
Introduction to Bar Coding Systems.....	75
Bar Code Symbols .....	77
Printing Methods.....	78
Scanner.....	79
TimePoint computer .....	79
System Overview .....	80
Hardware Overview .....	80
Software Overview .....	87
Software Architecture.....	87
General Operation.....	89
User Interface.....	90
Scan Shipping Sheet Label .....	91
Shipping Sheet Data Entry.....	94
Scan Part Label .....	96
6. RESULTS AND DISCUSSION.....	102
7. SUMMARY, CONCLUSION, AND SUGGESTIONS FOR FUTURE RESEARCH.....	103
REFERENCES .....	105



APPENDIX: SOURCE CODE .....	109
VITA .....	164

## LIST OF TABLES

Table 1: Data Summary Sheet.....	35
Table 2: Simulation Model Summary .....	37
Table 3: Sensitivity Analysis.....	41
Table 4: Statistical Summary Report.....	46
Table 5: Statistical Summary Report.....	47
Table 6: Simulation Model Statistics .....	48
Table 7: Field Descriptions for Shipping Data Entry Screen .....	95
Table 8: Field Descriptions for Scan Part Label Screen.....	97
Table 9: Data Dictionary .....	100
Table 10: Part Number File .....	101
Table 11: Pull Signal Number File.....	101

## LIST OF FIGURES

Figure 1: Circuit Board Manufacturing Area .....	17
Figure 2: Final Assembly Area.....	21
Figure 3: Circuit Board Manufacturing and Final Assembly Process Flow.....	23
Figure 4: Simplified Factory Floor with Process Flow .....	26
Figure 5: WITNESS Simulation Inputs and Outputs .....	27
Figure 6: Production Process Operations .....	29
Figure 7: Schedule of Design Stages.....	32
Figure 8: Schematic View of the Circuit Board Soldering Process .....	33
Figure 9: Schematic View of the Circuit Board Surface Mount Process .....	34
Figure 10: Simulation Model to analyze the throughput, quality and bottlenecks-I ...	38
Figure 11: Simulation Model Element Flow Diagram.....	39
Figure 12: Simulation Model to analyze the throughput, quality and bottlenecks-II..	40
Figure 13: Sensitivity Analysis .....	42
Figure 14: Process Cells .....	43
Figure 15: Integration/Consolidation of Cells.....	44
Figure 16: Logical View of the FIS configuration .....	52
Figure 17: Typical Part Bar Code Label.....	66
Figure 18: Typical Shipping Scan Sheet .....	67
Figure 19: Shipping Bar Code Label.....	69
Figure 20: Packout Process Flow .....	70
Figure 21: TimePoint Computer Showing Circuit Board and Connectors.....	80

Figure 22: TimePoint Computer with Standard Mounting Plate.....	81
Figure 23: Packout Verification System.....	82
Figure 24: Data Collection Terminal and Printer Mounting Scheme.....	84
Figure 25: Hardware Configuration for the Packout Scanning System .....	84
Figure 26: Screen Flow Diagram.....	91

## **ABSTRACT**

The current global competition and the economic situation in the United States and the world is forcing industries to produce quality products quickly and at a competitive price. Many industries are aiming towards world class manufacturing objectives like responsive delivery, defect free product and declining cost. Industries in the present environment can survive and produce the quality products to customer expectations only if they implement new technologies. The key element in the success of industries is using the continuous process improvement strategies like reducing process variability and reducing response time to process deviations.

Achieving quality in manufacturing processes is an important part of the job description of everyone concerned with the manufacturing operation. The greatest savings can come when a quality system can immediately inform appropriate personnel when process problems occur, and can then assist in ensuring rapid response at the lowest possible level in the organization. This type of system adds not only to the bottom line but also to the job satisfaction of all concerned [John, 1992]. The quality tools of the future are those that operate under a different scenario: the computer systems that collect the data also automatically do the analysis, interpretation, detection and correction, along with exception-based alarming and reporting.

In this research, we examine the potential benefits of an integrative approach to on-line quality management. The motivation for this research comes from a field study with a printed circuit board assembly and instrumentation cluster manufacturer. The company made substantial investments in setting up elaborate systems for on-line data collection and monitoring of process status. While these modern quality control systems

provided a rich database, their application in quality management was rather limited. This is partially due to the lack of appropriate methodology for quality decisions. The objective of this research is to develop an integrative approach to process control and packout verification of products. When the developed approach was implemented, it enabled the company to reduce their Problem Resolution Requests (PRRs) by 25% and was a cost avoidance of approximately \$600,000 annually.

# **CHAPTER 1**

## **INTRODUCTION**

Increasing global competition is driving industries to adopt intelligent manufacturing control. Intelligent manufacturing control is the tight coupling of sensor technologies and microprocessor-based software systems that manifests intelligence by learning from experience and exhibiting some degree of synergy with a human interface [Black J.T, 1993].

Industries in the United States are taking steps to compete with the global competition. They are close on quality, but must reduce costs and improve time to market, and for many, this will hinge on their ability to compete by changing or reconfiguring their manufacturing systems. Companies in the U.S. have recently begun to reduce the number of vendors and workers, eliminating waste and practicing recycling in order to be competitive [The Competitive Edge, 1991]. Good labor, productivity and 'the white heat of technology' are no longer particularly relevant. Product design and process management skills are the new hallmarks of success [Underwood, 1994].

Concurrent or simultaneous engineering is part of the solution, but not the whole answer. What's needed is a linked-cell manufacturing system functionally integrated to support pull manufacturing, operating in a simplified environment [Black J.T, 1993].

The integrated manufacturing system is very efficient. It uses less space, time, people, and materials to make superior quality, low cost goods. It is very flexible to meet the customer demand and adapt to changes in product design.

The major characteristics of an integrated manufacturing system are:

1. Flexibility (adapting to changes in product design and demand)
2. Controllability (integrating the critical control functions of quality control, production control, inventory control, and process repeatability and reliability)
3. Efficiency (minimizing waste of space, time, and materials)
4. Uniqueness (developing and refining in-house manufacturing technology).

The new manufacturing strategy is first to design the manufacturing system to be flexible, controllable, efficient, effective and unique, then fully integrate those critical control functions defined above, and finally, design products simultaneously that can be made in this system and also support several generation of such products.

The principles of a new factory floor-operating model are: 1. Monitor the product but control the process; 2. Automated detection; 3. Proactive control not reactive analysis; and, 4. Pick the metrics carefully. The fact is, factories are complex and dynamic places. They can't respond well to complex systems built on rigid and deterministic models. The shop floor needs to react quickly to events as they happen. We need to automate the data collection and evaluation process and turn it into a powerful tool to alert operators to problems on the line. It's easy to automate paperwork, but it's much more difficult to develop revolutionary concepts and quality methods that will actually contribute to manufacturing success [John, 1992]. Computer tools such as inexpensive sensing devices, networking software, powerful local computing resources via workstations and PCs, fast real-time processing databases, and touch screens allow us unprecedented freedom to approach the reconfiguration/throughput problems in new and different ways in order to achieve greater quality and economy.



In most industrial processes, inherent disturbances are present that can propagate through the manufacturing system causing undesirable effects on outgoing process quality levels. If these disturbances are not controlled, they can often produce deviations from the required quality levels. The traditional process control methods are not applicable to many applications and thus can produce scrap since they are based on assumptions. CASA/SME envisions new intelligent controls and sensor technologies as the critical step to implement more flexible and adaptive production processes and systems. New processors, controllers, and sensors coupled with new real-time control technique holds the promise for the next generation of intelligent manufacturing systems [CASA/SME, 1992].

The National Research Council (NRC) committee on Analysis of Research Directions and Needs in U.S. Manufacturing has identified several important issues to be resolved for the success of Intelligent Manufacturing Control Systems. These issues are "data acquisition, correlation, presentation, quality control, simulation of control decisions, learning from process disruptions, understanding process complexity, standardization in system implementation, and more efficient user training." [The Competitive Edge, 1991; Robert, 1993]. A recent NRC review requested by the National Science Foundation (NSF) has Intelligent Manufacturing Control (IMC) ranked as first of five advanced manufacturing technology focuses.

### **Motivation**

The motivation for this research comes from a field study with a printed circuit board assembly and instrumentation cluster manufacturer. The company made substantial investments in setting up elaborate systems for on-line data collection and

monitoring of process status. While these modern quality control systems provided a rich database, their application in quality management was rather limited. This is partially due to the lack of appropriate methodology for quality decisions. The objective of this research was to develop an integrative approach to process control and packout verification of products.

Significant attention was placed on first-time-through quality and reducing the number of parts rejected by the company's customers measured by the number of Problem resolution requests (PRRs). The company used for this study indicated that, at their manufacturing facility, over 25 percent of all PRR's were due to mis-labeling of products. These mis-labeling accidents at the company's facility, in turn, were causing quality accidents at company's customer facilities. The reoccurrence of these errors resulted in a serious financial penalty from customer facilities (estimated at over \$600,000 per year) and the strong potential for losing business. Mis-labeling errors can have serious impact on assembly plants productivity and can cause considerable confusion in their material handling systems.

The objective of this study was to learn from, understand, and apply the use of Intelligent Information System to a workcell to control the processes. The underlying goal of this study was to develop a generic process control methodology for use with various sensor-assisted, computer-controlled automated industrial processes in the effort to understand, control, and improve their performance and also to implement a process to reduce and eventually eliminate labeling error occurrences.

## **Research Objectives and Methodology**

Compared with other types of production, the special problem of electronics production is the large number of series of production steps having strong inter-relationships with each other. Another difference is the fact that many components are assembled onto one board. Because the correct function of the assembly normally depends on correctness of all placed components and their leads, small connection errors usually lead to the failure of the whole assembly. To ensure a final board quality level of almost zero defects, usually visual touch up, incircuit and functional testing and, in some cases, thermal tests have to be performed. Almost 30-50% of fabrication costs in electronics production is caused by testing and repair operations. To have an impact on these cost intensive tests and repair operations, a strong commitment to robust process control is required. An integrated quality control system was designed to enable the company to engineer and manage processes effectively from the beginning of circuit board assembly to the finished product.

Considering the major impact that the mislabeling errors have on ability to build quality products, a series of solutions were looked into for implementation to avoid mislabeling errors. The objective was to develop a general process control methodology that assures that the assembly plants will always receive the correct shipping container, assembly and quantity of parts ordered. This is necessary because as the customer/supplier interface becomes lean, wrong parts and mixed parts cause assembly line stoppages resulting in additional cost to the assembly plant and delays in delivery of the product to the ultimate customer. Wrong assemblies installed in

vehicle result in costly rework. A process was implemented to reduce and eventually eliminate labeling error occurrences.

### **Intelligent Manufacturing Systems**

To compete in world markets, the United States must work more productively and compete with countries offering inexpensive labor; advanced technologies must be applied to all sectors of our economy. Manufacturing technologies in particular have been lagging behind other industries. The technologies do exist to create an integrated manufacturing environment that provides much greater amounts of flexibility and control. Developing these technologies allows the creation of intelligent manufacturing systems (IMS). An IMS is a system that applies powerful computer decision making control to the process of creating products. The increasing power and decreasing cost of PC and lower class processors makes them the ideal workhorse cell level and machine level control. The coordination of so many processors in one system creates a complex distributed industrial control system.

An Intelligent manufacturing system can be defined as a next generation, flexible manufacturing system which will integrate the entire spectrum of manufacturing activities - from R&D through design, manufacturing, process control, production control, quality control and management.

Intelligent manufacturing control has been the focus of recent reports and workshops addressing the intelligent processing of materials for design and manufacturing sponsored by the Department of Energy (DOE) and the National Institute of Science and Technology (NIST) [Wodley, 1989; Yolken, 1989]. Also, an international consortium of companies and agencies has been formed to address a series

pilot programs in the IMS area, under the umbrella of the International Intelligent Manufacturing Systems Coalition [Coleman, 1993; Mitchell, 1993].

The pressure to incorporate more and more intelligence into designs requires the integration of complex electronic control technology into products. Consumers expect very complex products such as automobiles to be reliable and thoroughly integrated with electronic controls. Reliability and electrical complexity have been at odds until the recent introduction of mechatronics and intelligent manufacturing systems technologies [Keys, 1991]. Indeed, the crux of the development has become balancing the trade-off between technological complexity and ease of use. This requires detailed systems analysis of the design technologies, skills, and organizations required to produce the best possible systems [Daniel, 1992; Keys, 1990]. Intelligent manufacturing systems have evolved as a method to reduce complexity through the adaptation of inter-disciplinary technologies. To accomplish this goal, IMS faces several challenges: creating new design and control tools, creating an engineering knowledge base to handle the new complexity, building inter-disciplinary teams, and educating engineers with the needed skills [Keys, 1993].

Meieran [1993] lists a number of possibilities to help reduce manufacturing costs and assist in managing increasing factory complexity. The possibilities are 1) automating routine decision-making processes involved in manufacturing, resulting in faster or more cost-effective decisions, or 2) capturing and applying manufacturing knowledge to reduce the time necessary to detect, analyze and solve manufacturing problems. Newer manufacturing systems have distinct advantages:

- 1.They can be used to produce many different products,
- 2.They are adaptable to changes in design or recipe, and
- 3.They can operate unattended.

### **Establishing Integrated Intelligent Systems**

The following suggestions should be borne in mind to help develop advanced approaches towards integrated intelligent systems [Hong, 1992]:

1. Make the best use of expert systems, neural networks, fuzzy systems, mathematical approaches, and other new technologies for the development of intelligent control, by enhancing the desirable properties of each other.
2. Establish a hybrid environment where multiple components can work together, share the databases, communicate more easily, reduce conflicts, and solve problems more professionally, in a user – friendly environment.
3. Enable an intelligent control system to conduct symbolic reasoning, numerical computation, algorithmic operation, heuristics utilizing, knowledge processing, monitoring, diagnosing, pattern recognition, language understanding, and learning.
4. Obtain efficient knowledge acquisition, representation and comprehension; information and data processing, as well as systems integration are particularly important to enrich and improve such an integrated intelligent system.
5. Pursue the best synergism and combinations of new technologies from control, computer, systems, neural science and engineering.

In this way, a real integrated intelligent system can be promised to accomplish the following tasks: high level controls; management of information and activities,

decision making; problem solving; and knowledge processing in uncertain and complex environments.

### **Benefits of CIMS and AMT**

Substantial, company-wide changes in the nature of the manufacturing tasks are intrinsic to implementations of Computer Integrated Manufacturing Systems (CIMS) and Advanced Manufacturing Technology (AMT). These basic changes affect all functional areas of the company including operations, engineering, marketing, accounting, and finance. This requires management to think of the benefits of CIMS and AMT differently than has been done traditionally for other capital expenditures in plant and equipment that may simply augment the current method of manufacturing [Alan, 1995]. A CIM system integrates all the major functions within a manufacturing enterprise via the use of a computer system [Joshi, 1992].

Tayyari and Kroll [1990] have constructed one of the most extensive lists of direct, indirect, and intangible benefits and costs for CIMS and AMT in the literature:

#### **Direct benefits/Costs:**

- Less labor in production
- Less setup time
- Less downtime
- Less work-in-process inventory
- Less finished-goods inventory
- Higher capacity

#### **Indirect benefits/Costs:**

- Faster deliveries

Increased utilization of equipment

Increased utilization of manpower

Less inspection

Less material handling

Less redundant data generation

Less rework

Less supervision

**Intangible benefits/Costs:**

Better product quality

Better production control

Consistent quality

Faster product introduction

Improved customer service

Improved product reliability

Improved lead times

**Overview of Process Control**

In most industrial processes, inherent disturbances are present that can propagate through the system causing undesirable effects on outgoing process quality levels. If left uncontrolled, these disturbances often produce deviations from a desired target level in product quality variables and, subsequently, can result in poor quality in the process. Therefore, industrial processes must be equipped to overcome the adverse effects of these disturbances to achieve acceptable quality levels in their output. Various methods for process control are often employed to minimize the variation



caused by disturbances present in the process. The overall requirements like speed, reduction of costs and flexibility of manufacturing are increasing in a very fast manner [Gausemeier, 1997].

Control over industrial processes is motivated by the desire to satisfy design and production specifications consistently while minimizing the variation within those specifications. Stated simply, variation is present in every industrial process [Western Electric, 1956]. Process control is necessary because this inherent variation can lead to undesirable effects in the process output. The variation is typically present in many forms: in the process equipment itself, in the raw materials used in the process, in the surrounding environment, in human operating procedures and individual decisions, and in the design of the product or assembly under construction. The capability of the process to overcome the undesirable effects of this variation and to consistently conform to the design and manufacturing specifications is a strong determinant in the quality level of the product [Yarling, 1993].

In practice, various types of process control methodologies are employed simultaneously to eliminate or compensate for variation. Methods such as factory quality assurance, statistical quality control, statistical process control, feedforward and feedback control, artificial intelligence, neural networks, and fuzzy logic have been used to gain control over and improve industrial processes [Yarling, 1993]. Among some of the most common of these methods is the use of traditional statistical process control charting techniques such as SPC charts [Shewhart, 1931], CUSUM charts [Page, 1961], and Exponentially Weighted Moving Average charts [Hunter, 1986].

## **Intelligent Manufacturing Control**

Intelligent Manufacturing Control (IMC) is a distributed, hierarchical approach to the control of manufacturing processes. It employs electrically coupled computer-based hardware controllers and process sensors in conjunction with a trained, self-directed work force to process physical state and historical data derived from the manufacturing environment [The Competitive Edge, 1991].

IMC has two main objectives:

1. To satisfy product quality and process control requirements for existing products and processes.
2. To be adaptable enough to do the same for future products and processes by providing a way not only to control the manufacturing process, but also to promote learning that will lead to process improvement.

IMC can be divided into three levels of traditional plant hierarchy, which permits a simpler organization and provides an avenue for establishing interactive links with manufacturing areas [The Competitive Edge, 1991].

1. Domain of real-time process control:

In the domain of process control, a precisely stated contingency procedure operates in real time at the machine level without human intervention.

2. Domain of observation and pattern recognition:

In the domain of observation and pattern recognition, the efficacy of procedures defined in the domain of process control is observed; contingencies in the behavior of

procedures are studied; and improvements are made. Problems are solved at the cell level.

### 3. Domain of learning and improvement:

The domain of learning and improvement is one of choice, where the options available for improving a system are assumed to be numerous and available resources to be limited. In this domain, at the plant level, those economic choices are made about which avenues of process improvement to pursue in view of supply and demand, resource utilization, and other production management functions.

According to Pierce [1991], integration in IMC involves data accumulated over time, including data on past disruptions, and implies the ability to relate current disruptions to earlier, similar disruptions. In this temporal context, IMC exists at three levels: (1) between machines and the flow of processes within a factory, (2) among different functions, such as design, engineering, and manufacturing, and (3) between human knowledge and machine intelligence.

### **Flexible Manufacturing System and Computer Integrated System**

Flexible manufacturing systems (FMS) attempt to bring the productivity of a dedicated flow line to small and medium batch manufacturing. The concept of flexible manufacturing was developed in response to demands for reductions in lead times without increases in production costs in low to medium volume discrete parts manufacturing [Smith, 1990]. Typically, an FMS is made up of a small number of general-purpose machine tools connected by an automated material handling system. Ideally, the entire system is controlled by a computer or a system of computers programmed to exploit the inherent flexibility of the individual machines in order to

process a variety of part types with little or no setup or changeover costs [Smith,1990]. FMS concepts and implementations have been described in many publications: [Dupont, 1982; Kimenia, 1983; Whitney, 1985; Cohen, 1985; Gupta, 1988].

Computer Integrated Manufacturing (CIM) is a much broader concept than flexible manufacturing. Where as FMS seeks to integrate a small set of shop floor machines, CIM integrates the entire manufacturing enterprise. Bravoco and Kasper [1985] define integration in the context of manufacturing as two or more things, which have common parts, which are leveraged to provide economies and benefits. In a CIM system, production status, purchasing reports, marketing information, and material availability information should all be accessible through a central computer system (the computer system is not necessarily physically centralized, only conceptually centralized) with well defined interfaces and communications standards.

### **Organization of the Dissertation**

This dissertation is divided into five chapters and one appendix. Chapter 1 has provided an overview and motivation for this research. Chapter 2 presents the industry and operation background. Modeling and Simulation details are presented in Chapter 3. Chapter 4 introduces the system architecture, interfaces and system operation. Packout verification system details are presented in Chapter 4. Results and discussion is presented in Chapter 5. Chapter 6 presents a summary of this research, draws conclusion, and provides suggestions for future research. Appendix presents a detailed documentation and application coding.

## **CHAPTER 2**

### **INDUSTRY AND OPERATION BACKGROUND**

The company used for this research manufactures instrument clusters for cars, vans and motorcycles at their facility. Instrument clusters consists of fuel, oil, temperature, and voltage gages, speedometers and tachometers. Instrument clusters are assembled from several components like spindles, armatures, upper and lower bobbins, terminals, windings, cans and resistors at this company's plant. These instrument clusters are supplied to customers in vehicle assembly plants and service operations. There are 35 cluster assembly stations. After assembly, instrument clusters are packed in twelve standard shipping containers on a single shipping skid. Each container consists of a standard quantity of instrument clusters, referred to as a standard pack, depending on the size of the cluster assembly. An exception is made for service operations where cluster assemblies may be packaged in containers with specified, but non-standard pack sizes. Each such individual container has a separate identification label listing part number, quantity, shipping serial number, and certain critical traceability data. All the shipping barcode labels are preprinted in a separate location. The traceability data are currently manually recorded and the labels are placed on the containers as the last plant operation. Each skid of twelve containers contains a single cluster part number. Two identical barcode shipping labels are located on two adjacent sides identify the skid. These labels conform to the AIAG-B3 (Automotive Industry Action Group) shipping identification label standard.

## **Pilot Manufacturing Workcell Overview**

### **Circuit Board Manufacturing Area**

Circuit Board manufacturing area is shown in Figure 1. Loading circuit board to screen print is the first step in the circuit board manufacturing process. The MPM screen printer is used to screen print solder paste onto a bare circuit board. The board transport system is automatically loaded from an upstream transfer conveyor. The boards are transported into the screen printer where they are automatically aligned to the stencil and then screen-printed with solder paste.

The boards are then delivered to the next conveyor. The board then goes into the pick and place machine. These machines are designed to automatically pick surface mount components from feeders and place them to printed circuit boards. The board handling system is automatically loaded from an upstream transfer conveyor. Once the start button has been pushed the machine will automatically cycle boards from the input conveyor, clamp them at the center conveyor, place components from each head and unload to the output conveyor until a stop button is pushed. The operator is required to load feeders with components and position the feeders in their proper locations on the machine when a feeder becomes empty.

Radial Component Inserters are then used to insert components to the circuit board. The radial component inserters are designed to automatically insert radial leaded components such as transistors, LEDs, resistors, capacitors, diodes and others. Radial inserters consist of a sequencer module, component conveyor system, X-T table, verifier module, insertion head, and cut and clinch assembly.

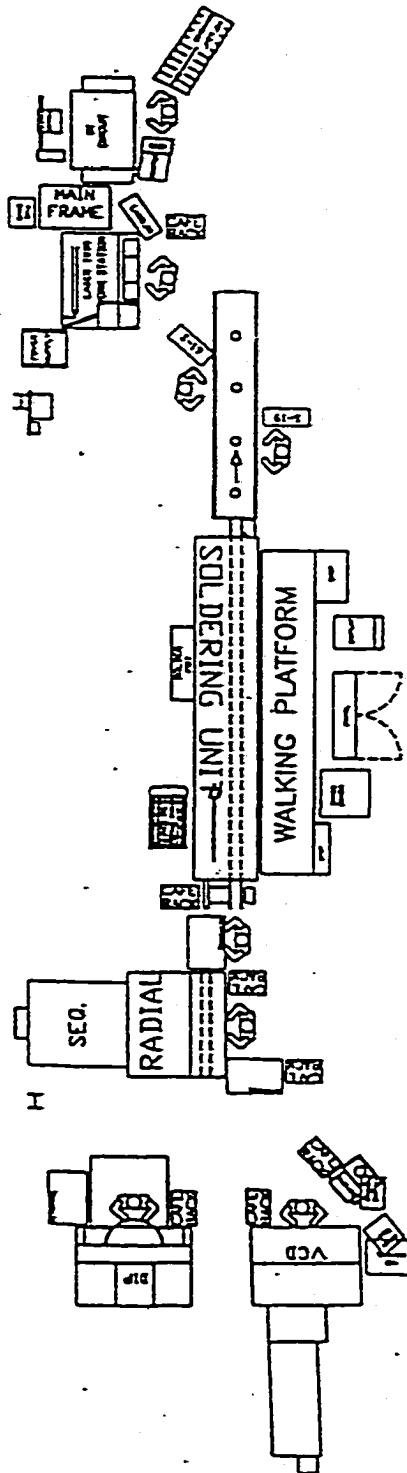


Figure 1: Circuit Board Manufacturing Area

The operator manually loads each board or array into a workboard holder on the X-Y table, pushes the start button to initiate the insertion cycle and manually unloads the board when the insertion is complete. The onboard computer controls the operation of the radial inserter via stored executive program and pattern programs. The executive program is a sequence of instructions, which allow the machine to interpret and execute position, and insert data provided in the pattern programs. A data terminal provides the means for communications between the operator and the controller.

Then the board is transferred into the clip insertion robot workcell by the work in process conveyor. The robot begins to pick and place the clips into the board. After the robot has completed its tasks, the pallet is released from the robot workcell and sent to the radial or odd form insertion machine. Then the circuit board is unloaded from the radial inserter and sent to the motor press where motors, pin header connector are placed on the board. Then choke, display and RTT connectors are hand inserted.

A singulation machine is used to cut the array rails away from the circuit board, and the circuit board is sent through the reflow system for soldering process. The reflow system is used to cure the adhesive and/or reflow solder paste which holds surface mounted parts on the top and bottom side of circuit boards through the wave solder process. The oven consists of twenty heater sections surrounding a board transport system, and a computer for controlling the process and interfacing with the operator. The board transport system is automatically loaded from an upstream transfer conveyor. The boards are transported through the heat zones of the oven



where the adhesive curing and/or solder paste reflow takes place. They then pass through a cooling section near the ovens exit and are delivered to the next conveyor.

Then, the wave-soldering machine solders electronic component leads to the printed circuit board pads. The machine is designed so many PCB connections can be soldered in one pass over the solder wave. The conveyor moves the PCB through three stations: spray fluxer, preheaters and dual solder waves. The flux is applied evenly to the bottom of the PCB as it passes over the fluxer. The flux removes the thin film of oxide present on the metal circuitry. The heaters reduce thermal shocking of the PCB and components and evaporate the flux solvent during the soldering process. A dual pump/dual wave module is contained within the solder pot.

The circuit board is transferred into the bulb insertion robot workcell by the work-in-process conveyor. The robot begins to pick and place bulb assemblies onto the circuit board. After the robot has completed its tasks, the circuit board is released from the robot workcell and sent back to the operator. Then the boards are placed on the fixture for incircuit testing.

The HP3070 series machine is an automated tester used for performing incircuit and sometimes functional testing of circuit board assemblies. The purpose of an incircuit test is to verify that the board is assembled correctly. It does this by checking for shorts, opens, and the presence/value/polarity of the components on the board. The purpose of a functional test is to confirm that the board operates correctly. It does this by providing stimulus to the inputs of the board and monitoring the outputs. An incircuit test verifies that the board is assembled correctly and should work, whereas a functional test verifies that the board works and is likely assembled

correctly. Once the test cycle is complete and the circuit board in the tester passes, the boards are autoloading to conformal coat. Here a conformal coating film is applied to printed circuit boards. The purpose of coating is to protect the circuit board assembly from certain outside environmental factors, which could degrade the board. Boards are sent to final assembly area after this process.

### **Final Assembly Module**

Final Assembly area is shown in Figure 2. Applique press/Coil assembly is the first station on the final assembly line. Here the operator will assemble the coils and applique to the light pipe. Case assembly is the next station on the line. The same operator is used for both this operation and the following one, the marriage press. The operator will assemble the circuit board and retainer to the case and use the press to assure positive engagement of the retainer snaps. At this point the part receives the unique bar code label via an Intermec bar code printer. After the operator completes the case press operation, the lightpipe is assembled and pressed into the case on the marriage press. The machine will cycle only if the bar code is read and the part is approved for pressing by the packout verification station. Fourth station on the line is the robotic staker. The same operator is used for both this operation and the following one; the close and verify station. The robot will perform the staking and inspection tasks only after receiving permission from the packout verification station. After the robot shuttles the finished product back to the operator, the final assembly of the part takes place at the close station. The part is inspected here both visually and electrically. Upon a good result of the tests, the part has the lens and retainer pressed into position to ensure snap engagement. The close station will not perform the testing

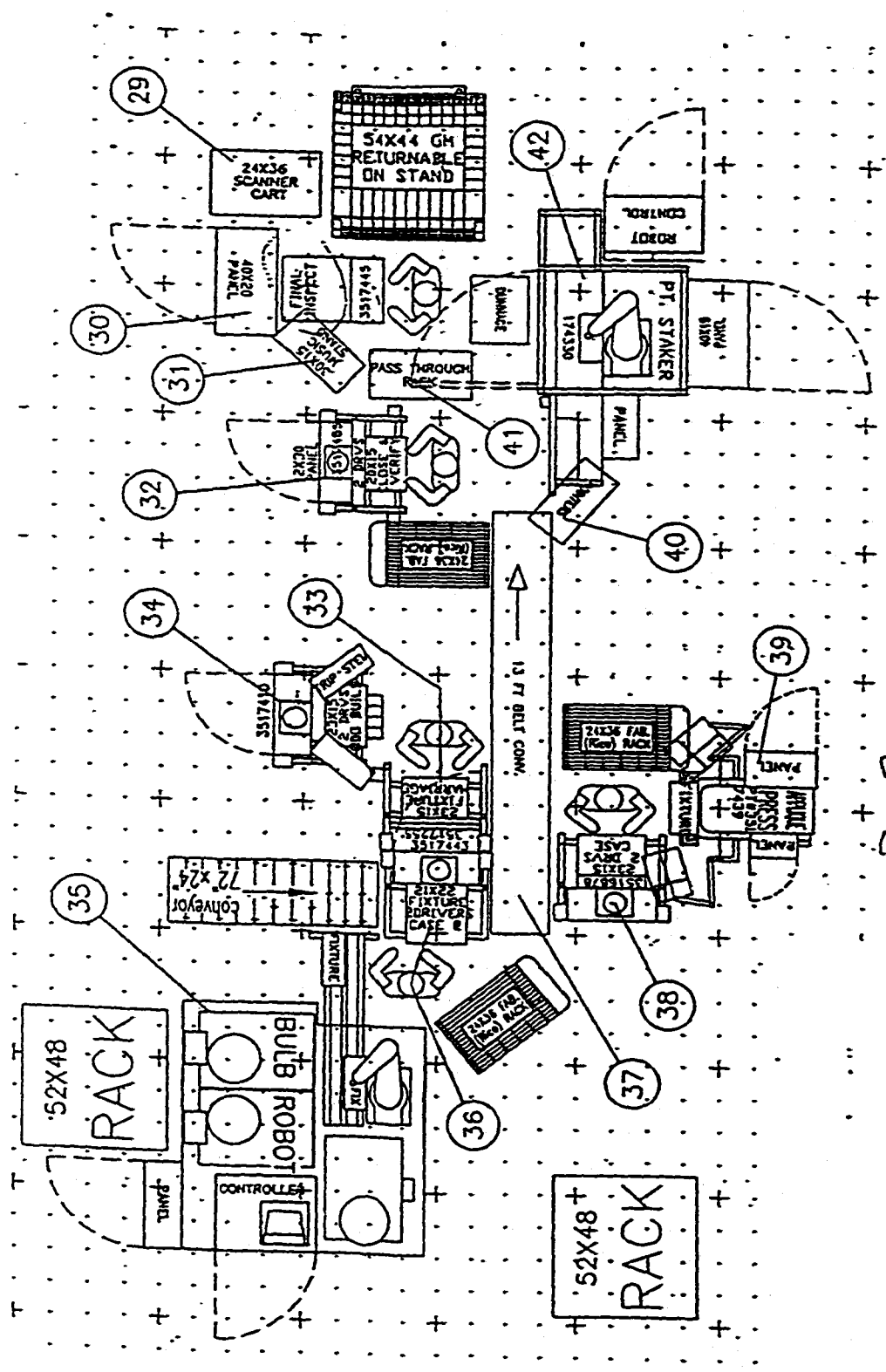


Figure 2: Final Assembly Area

or pressing until permission is granted from the packout verification station. The bar code is read and permission is granted to pack out the part if all the previous processes have been completed and passed. This will take the part out of the work-in-process queue.

Repair station is in the assembly area but not on the line. This station is where the computer running the packout verification system resides. The station is also equipped with a gun-style scanner and wedge. Any part that cannot pass through one of the stations is sent to the repair station. At the repair station the part is either repaired or scrapped. If scrapped, the part is removed from the work-in-process queue and archived. If repaired, the part is returned to the assembly line at the station at which it failed. An operator interface is provided for the repairperson to record data on the repair of each part and its status. Figure 3 shows the Circuit Board manufacturing and Final Assembly process flow.

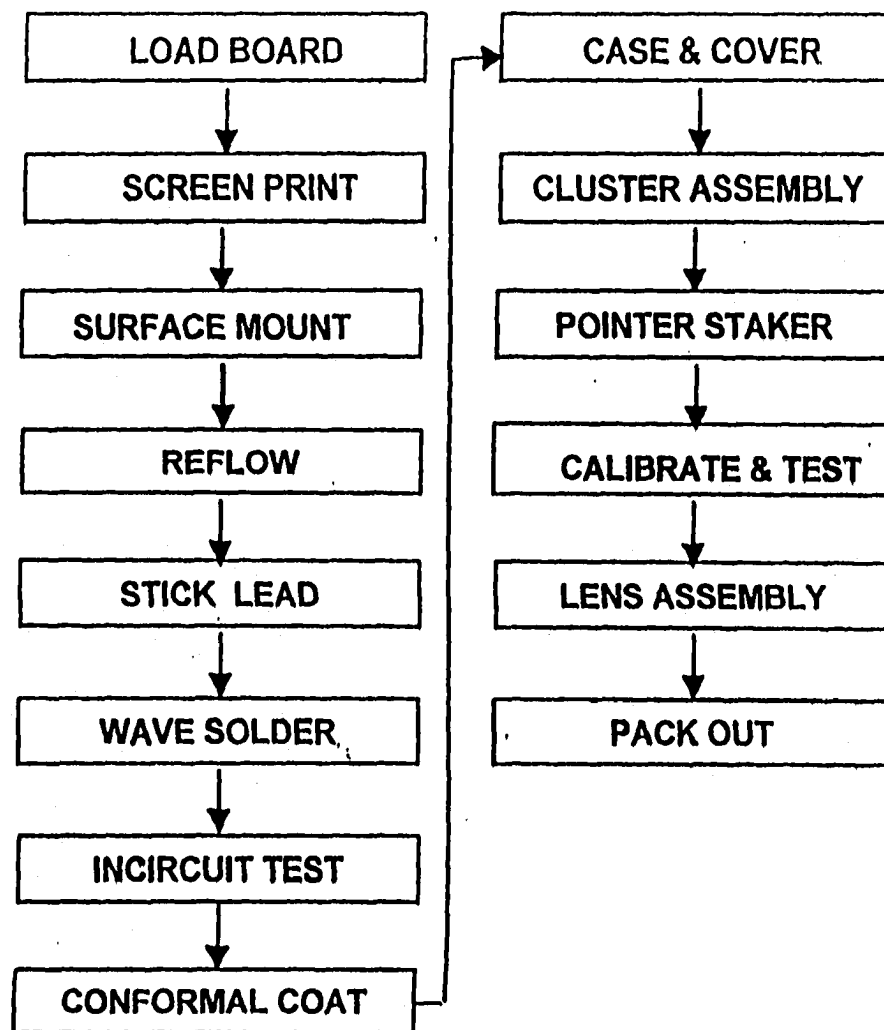


Figure 3: Circuit Board Manufacturing and Final Assembly Process Flow

## **CHAPTER 3**

### **MODELING AND SIMULATION**

#### **Background**

Simulation models represent an abstraction of real life systems in terms of their components, parameters, and relationships. To obtain an acceptable understanding of any simulated system, experiments must be performed on their elements through representative simulation runs [Hatami, 1990]. Statistically designed experiments can be employed to improve the efficiency and effectiveness of experimentation with systems-real life or simulated. This technique coupled with simulation modeling provides a systematic and scientific approach to system analysis. Modern manufacturing systems can be very complex, embodying many of the latest technologies such as automated material handling, as well as the ideas such as just-in-time (JIT) [Suri, 1990]. Designing and maintaining such facilities can involve a number of difficult decisions. Simulation software packages are used by industry to assist in making these complex decisions.

Recent advances in electronics factory automation have widened the role and increased the importance of factory modeling. In conjunction with computer aided manufacturing systems, factory models are being used for on-line control of material movement. This use, in factory control, imposes stringent requirements for accuracy and computational speed [Atherton, 1990]. A second need for factory modeling results from the trend in electronics factories toward the complex integration of machines, material, and people. For example, in semiconductor industry, equipment, instruments and robots are being tied together to create workcells.

As manufacturing modules are connected to produce complex systems, the cycle-time performance of the overall system becomes difficult to predict. Simple spreadsheet calculations can no longer represent the variety of dynamic behavior exhibited by the system. In order to analyze, design, and control these new complex-manufacturing systems, system specific models are required for performance analysis.

This research describes the design of a high volume circuit board manufacturing cell using a set of PC-based modeling and simulation tools. Figure 4 presents a schematic view of the simplified factory floor with process flows. The simulation was performed using Witness, a visual interactive simulation tool. Witness is a true simulation and modeling tool, for evaluating the interaction among different product lines that may share a facility's resources. Witness makes obvious any production bottlenecks, overly-idle resources, storage areas that are too small or too large and any potential issues with respect to labor availability [Rawles, 1998]. The tool has features for graphical creation of simulation models, dynamic display of the simulated system and user interaction with the running program. This tool was also used to perform a sensitivity analysis of the effect of breakdown factors. Simulation of the critical areas of the cell allowed to fine tune the buffer sizes and the number of machines. The inputs and outputs of Witness are summarized in Figure 5. Different layout alternatives were generated using PowerPoint program.

Simulation studies were conducted at a large corporation manufacturing electronic circuit boards for high volume automotive products. The company has decided to shift from functional layout to cellular production organization to improve its manufacturing operations.

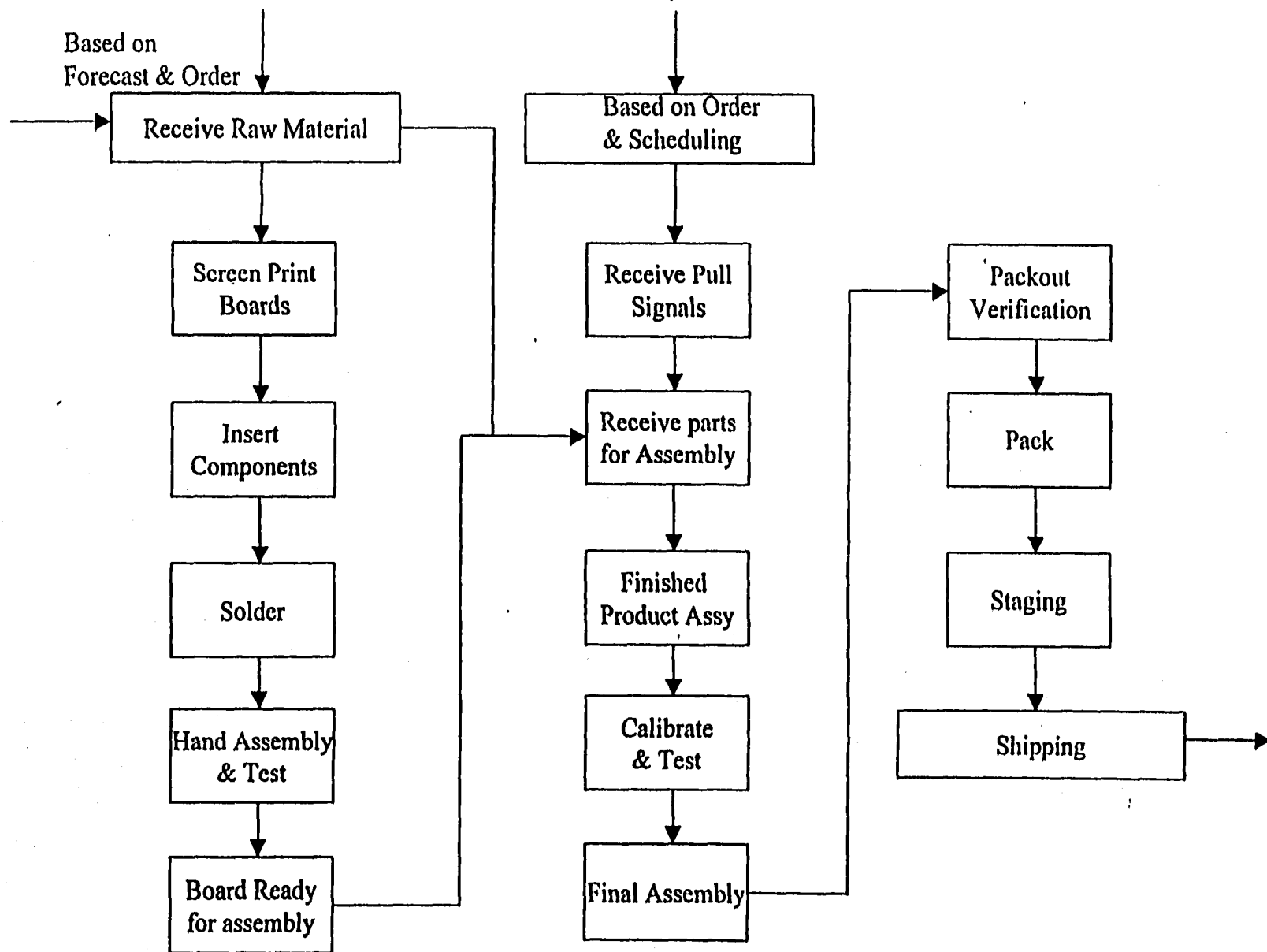


Figure 4: Simplified Factory Floor with Process Flows



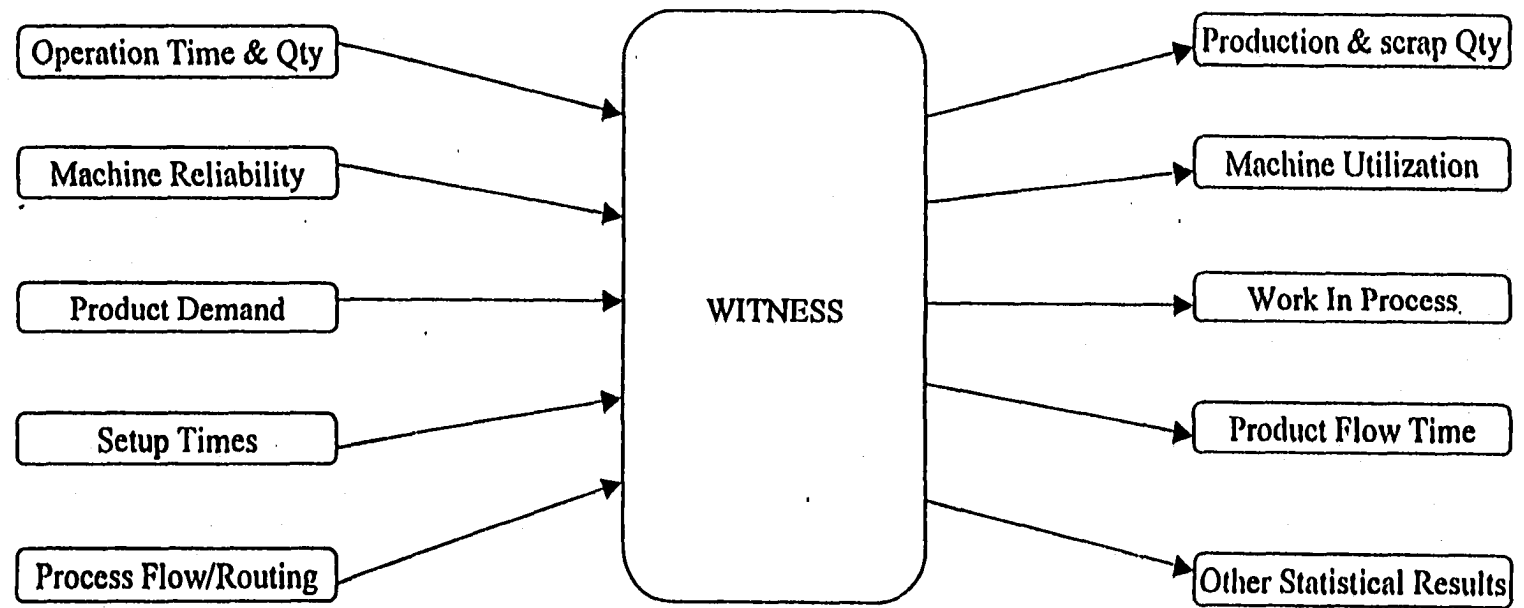


Figure 5; WITNESS Simulation Inputs and Outputs

Some of the problems faced by the management of the plant under consideration before the shift were:

- A finished product inventory
- A semi-finished product inventory
- Levels of scrapped products
- Machine uptime
- Throughput
- Productivity
- Process Control

To resolve these problems, simulation models were developed and analyzed to explore different options before making the strategic decision to alter the layout organized by process, and to regroup the equipment in manufacturing cells. Depending on the production volumes, some of the manufacturing cells are dedicated to a specific type of product or to a family of products.

### **The Manufacturing Cell Design Criteria**

The manufacturing cell analyzed is dedicated to two similar products. A forecasted demand of about one million units per year justifies the dedication to the two products. The operations involved were grouped in three main categories: circuit board assembly, testing and final assembly. Figure 6 presents a schematic view of the production process operations, which summarizes the entire system that was modeled through the use of different symbols for operations, storage spaces, and material movements.

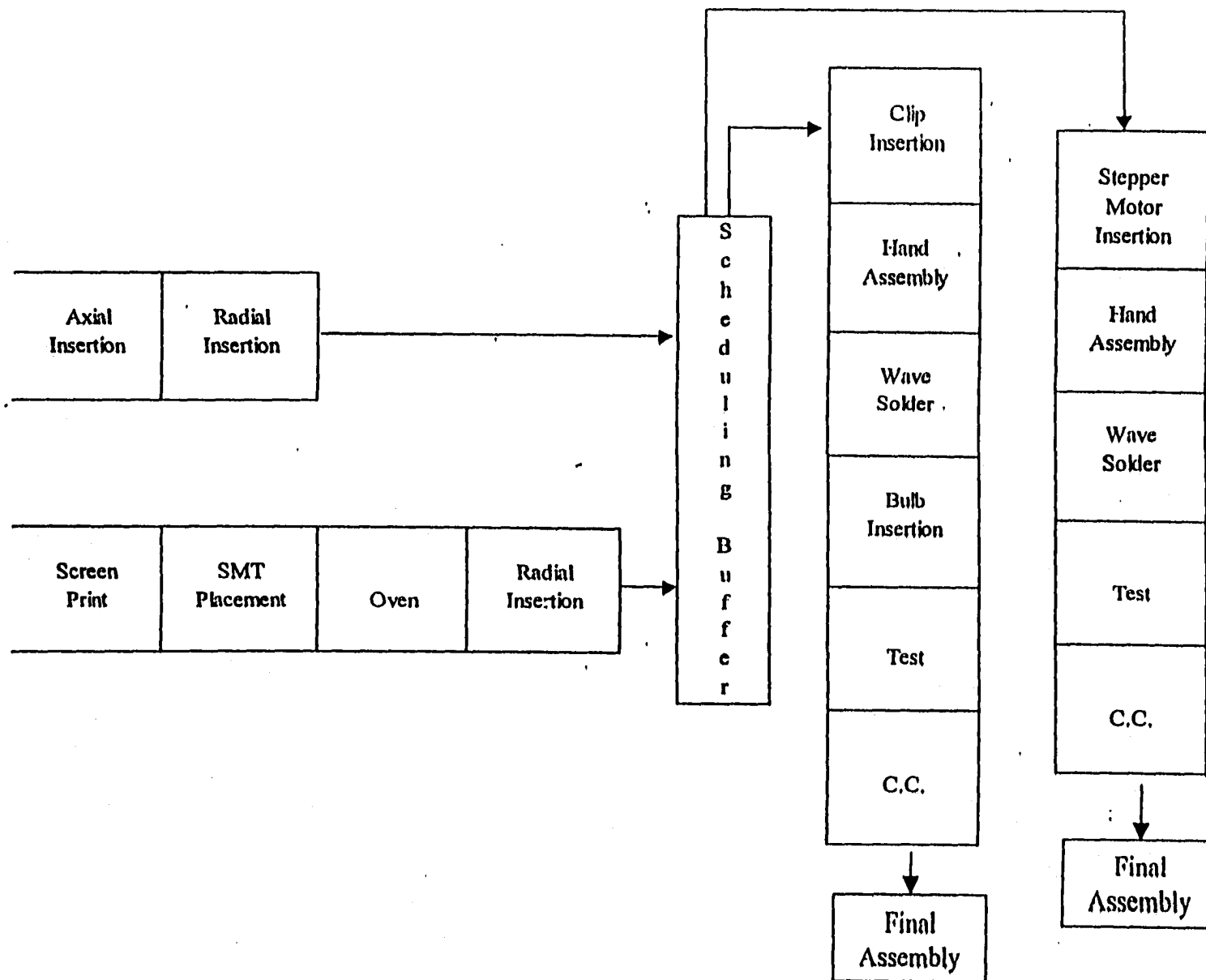


Figure 6: Production Process Operations

Every product is tested. If the test is successful the product proceeds to the next operation. If the test fails the product goes to a repair loop. Repair loops are dedicated to each testing operation and are integrated to the cell. The plant objectives were to operate the cell in a just-in-time and lean vision mode. That is, to produce exactly the quantity needed by the customers and be able to react quickly to short term demand variations.

The main goals and design criteria are listed below:

- Develop an efficient layout, in order to: meet the required output, minimize the work-in-process (WIP), minimize material flow
- Parallel synchronous process cell
- Multiple products through same equipment
- Cleaner processing
- Improved throughput & uptime
- Planned maintenance
- Minimized over time
- Reduced scrap
- Improved productivity/headcount Savings/flexibility
- Increased sharing of expertise in assembly
- Improved ability to manage circuit board assembly operations
- Improved material flow through plant
- Consolidate circuit board assembly and co-locate with cluster final assembly
- SPC improvements

## **Methodology**

Information obtained from the company relating to manufacturing operations (setup and cycle time) and workstation parameters (mean time to failure MTTF and mean time to repair MTTR) is used in the model. To design the manufacturing system in a short time frame, a consistent set of modeling, analysis and design tools were used which includes the following desktop computer based packages: Witness, Excel and PowerPoint. A diagram showing the different design stages is shown in Figure 7.

### **Establish Objectives & Scope**

During this phase benchmark values of number of machines were collected prior to detailed simulation analysis. The main concerns were to reduce Work in Process and flow time. Process flow and layout diagrams were documented to illustrate the scope of the project as well as the flow of the parts through the system. Figure 8 shows the schematic view of the circuit board soldering process and Figure 9 shows the schematic view of the circuit board surface mount process.

### **Collect Data & Build Model**

A simulation model is a surrogate for actually experimenting with a manufacturing system, which is often infeasible or not cost-effective. Thus, it is important for a simulation analyst to determine whether the simulation model is an accurate representation of the system being studied, i.e., whether the model is valid. It is also important for the model to be credible; otherwise the results may never be used in the decision-making process, even if the model is valid [Law, 1998].

During this phase, the Witness element selections were made to model the system. Data summary information is given in Table 1.

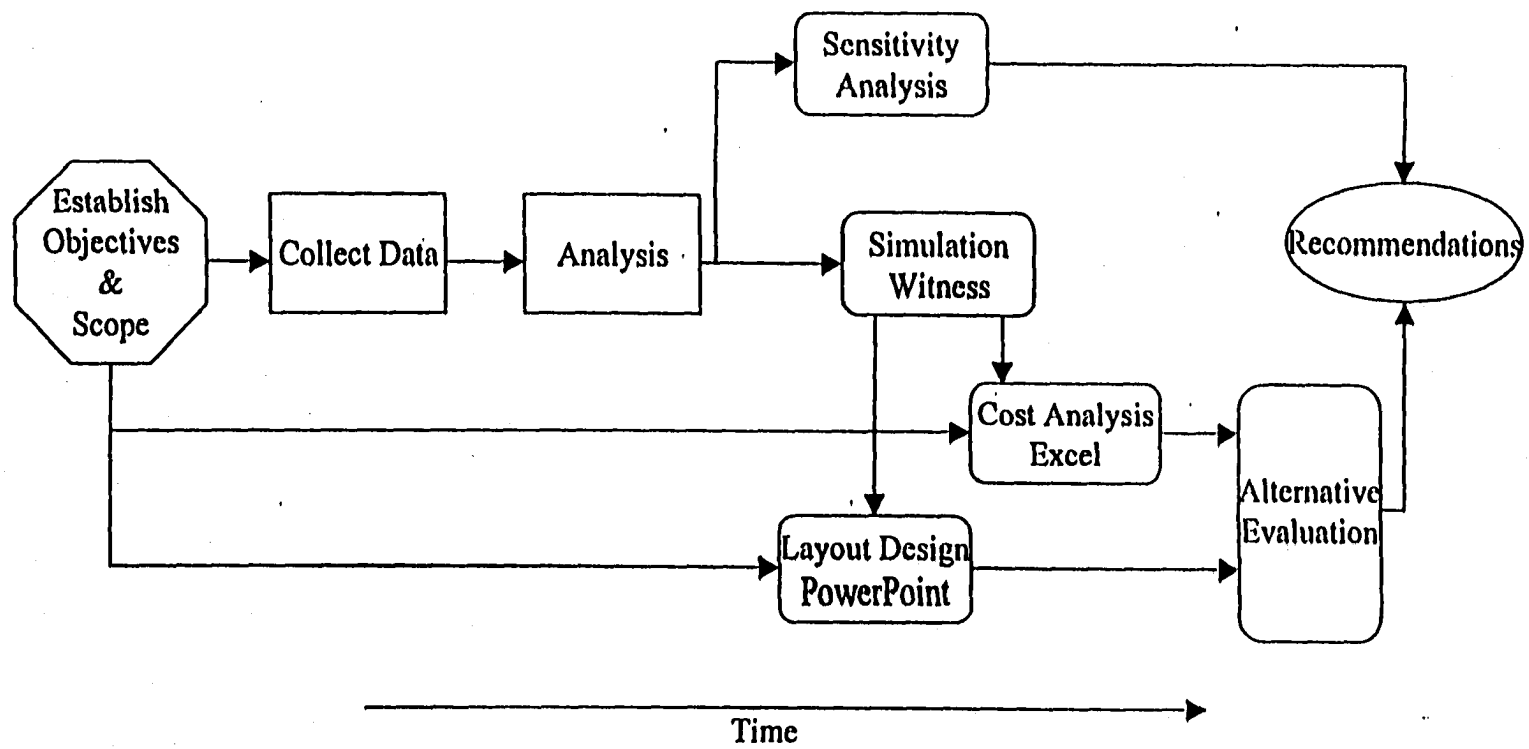


Figure 7: Schedule of design stages

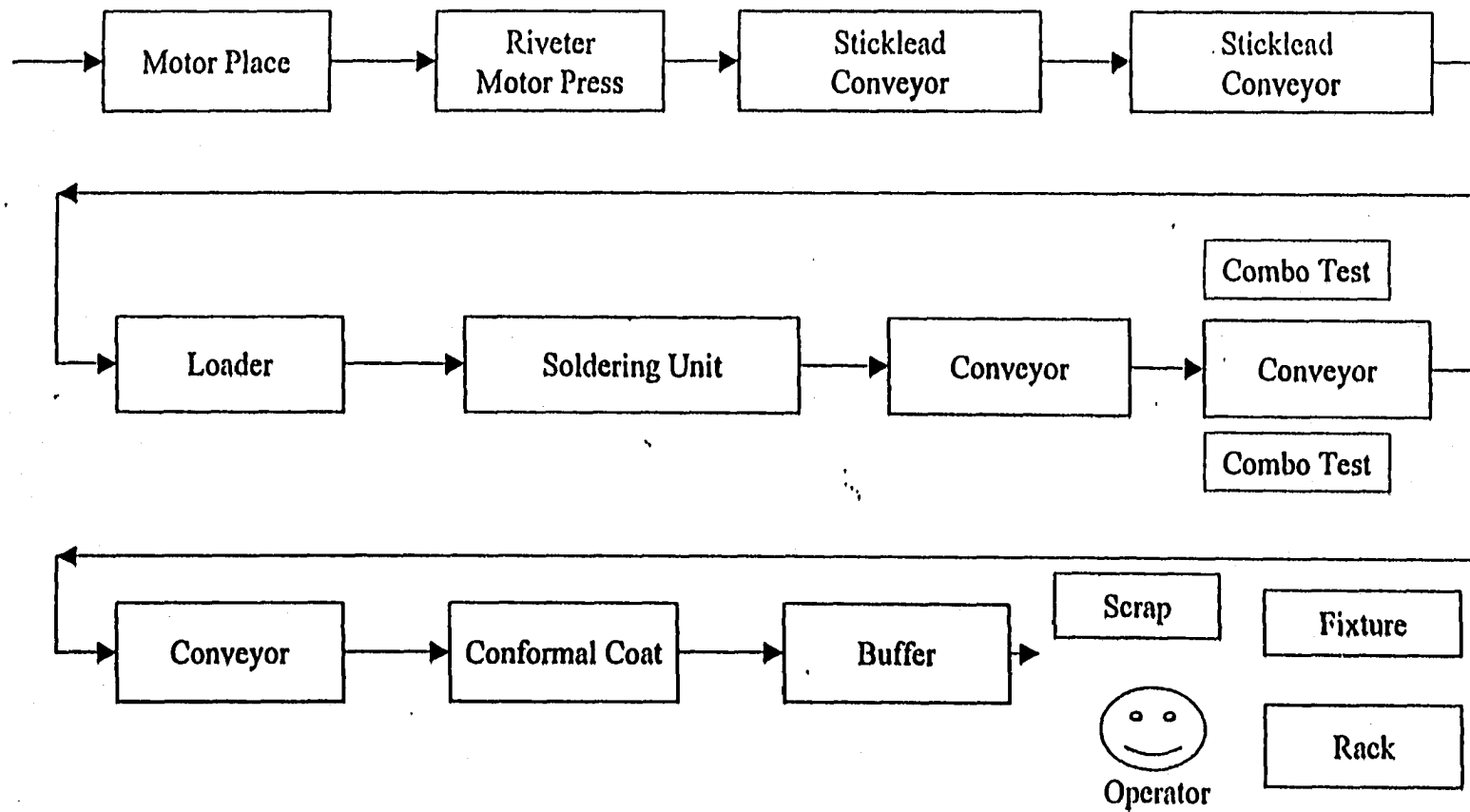


Figure 8: Schematic View of the Circuit Board Soldering Process

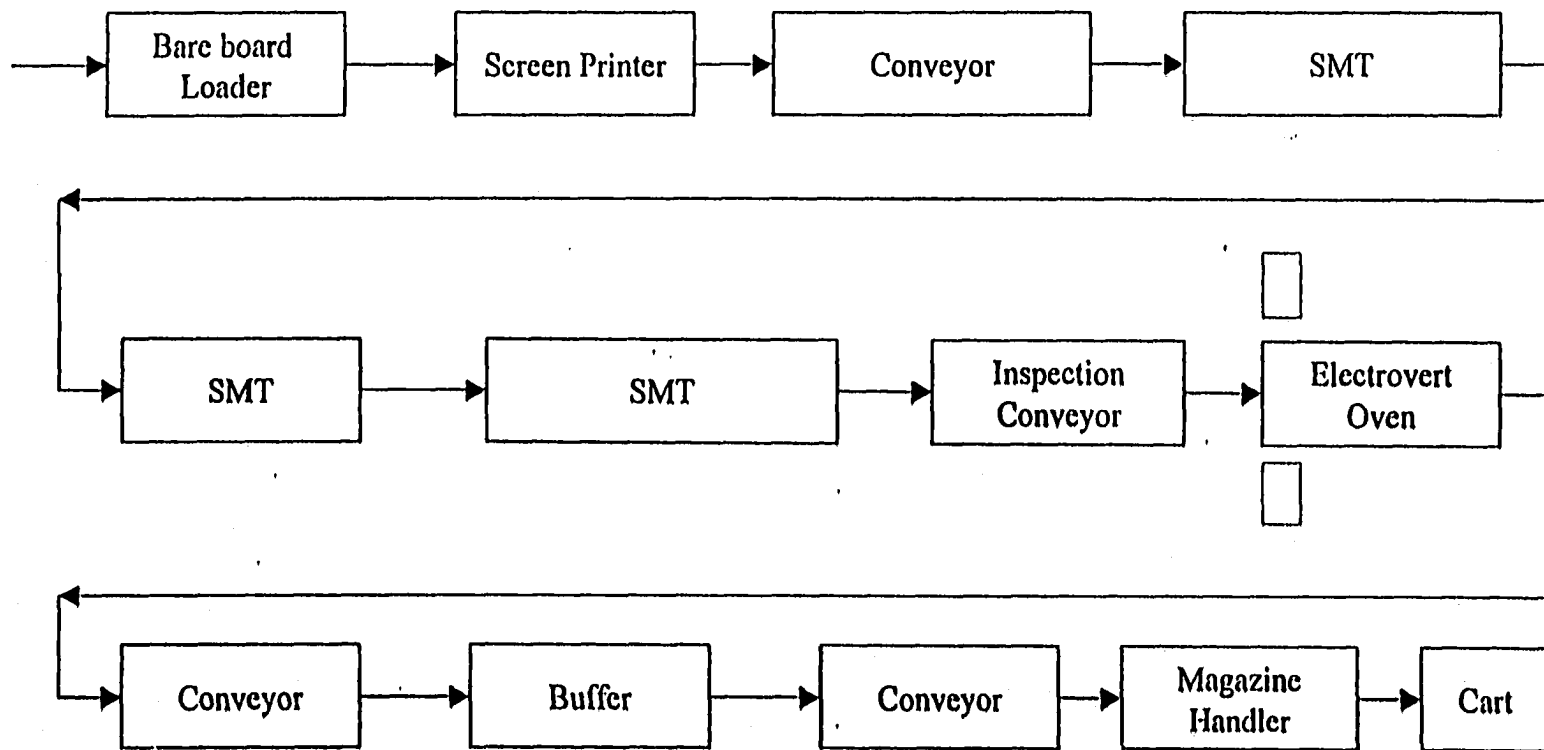


Figure 9: Schematic View of the Circuit Board Surface Mount Process



Table 1: Data Summary Sheet

MACHINE/PROCESS	QUANTITY	% FALLOUT	CYCLE TIME	CYCLE TIME	DOWN TIME	REPAIR TIME	MEAN TIME	MEAN TIME	RANDOM #
		AT MACHINE	MIN	SECONDS	%	MINUTES	OF FAILURE.	ACTUAL	STREAM
							FIRST		
BOARD LOAD	1	0.0%	0.08	5	1.0%	30	3000	9000	66
VCD 1	1	0.25%	0.6533	39	10.0%	20	200	210	49
RADIAL	1	0.25%	0.2933	18	10.0%	20	200	210	37
CLIP ROBOT	1	0.3%	0.603	36	2.0%	6.5	325	240	39
STICLEAD	2	0.6%	0.3	18	2.0%	6.5	325	200	34
SOLDER	1	1.60%	0.333	20	14.0%	15	107	100	49
BULB ROBOT	1	0.01%	0.4766	29	2.0%	6.5	325	240	59
COMBO TESTER	1	0.18%	0.25	15	12.0%	15	125	140	49
ARRAY BREAK	1	0.25%	0.2833	17	2.0%	5	250	200	49

Information regarding the input and output rules on the key elements and a summary of the actions language that needs to be included in the elements to give the necessary degree of logical control were selected. Table 2 shows the summary of the simulation model.

Items such as cycle times of machines and the capacities of buffers were also incorporated into the plan. The details of the successive simulation models are shown in Figure 10, 11, and 12.

### **Sensitivity Analysis**

The Witness analysis pointed out bottlenecks in the system. Machines with high downtime had large amount of WIP in front of them. This led to perform a sensitivity analysis to examine the effects of improvement in the machine characteristics on the system performance. The result of the sensitivity analysis is shown in Table 3 and Figure 13.

### **Detailed analysis**

This phase of analysis was performed to get more insight into the operations of critical production stages. Automatic component insertion and manual component insertion were the key areas of the system. A model to simulate these production stages was created using the Witness program. Final capacity planning was done on the basis of the simulation results.

### **Layout Design**

Using simulation results, a number of layouts were generated. PowerPoint was used to draw the different layouts. Figure 14 and Figure 15 show the layout of the process cells.

Table 2: Simulation Model Summary

Items	Type	How Used	Where Used	Type
board	part		vcd1	Machine
b1	Buffer	input rule	redell	Machine
		output rule	vcd1	Machine
OB	Buffer	Output rule	ODDFRM	Machine
		input rule	STILEAD1	Machine
b3	Buffer	input rule	clip	Machine
		output rule	redell	Machine
ODDFRM	Machine	Piechart scdio	addform	Pie Chart
b5	Buffer	input rule	ODDFRM	Machine
		output rule	clip	Machine
b6	Buffer	input rule	STILEAD2	Machine
		output rule	solder	Machine
b7	Buffer	input rule	BULLR	Machine
		output rule	TEST	Machine
b8	Buffer	input rule	TEST	Machine
		output rule	CONCOAT	Machine
STILEAD2	Machine	input rule	STILEAD1	Machine
conveyor	Conveyor	output rule	solder	Machine
		input rule	BULLR	Machine
clip	Machine	Piechart scdio/newcib		Pie Chart
		Piechart scdio/pycd		Pie Chart
		Piechart scdio/platform		Pie Chart
		Piechart scdio/radi		Pie Chart
		Piechart scdio/solder		Pie Chart
		Piechart scdio/bub		Pie Chart
		Piechart scdio/pleat		Pie Chart
		Piechart scdio/conform		Pie Chart
redell	Machine	Piechart scdio/redell		Pie Chart
vcd1	Machine	Piechart scdio/pycd		Pie Chart
STILEAD1	Machine	input rule	STILEAD2	Machine
solder	Machine	Piechart scdio/solder		Pie Chart
BULLR	Machine	Piechart scdio/bub		Pie Chart
TEST	Machine	Piechart scdio/pleat		Pie Chart
		Piechart scdio/conform		Pie Chart

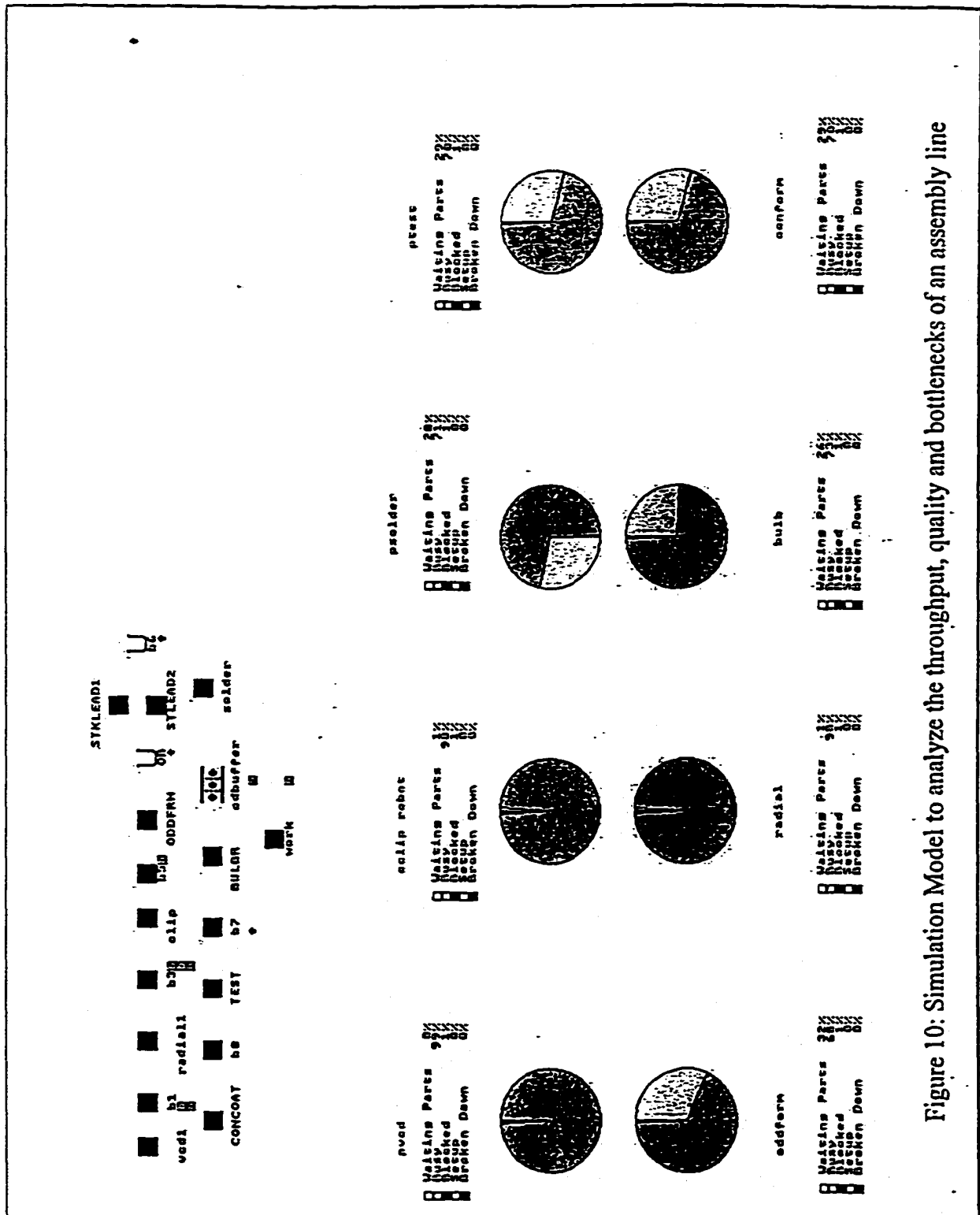
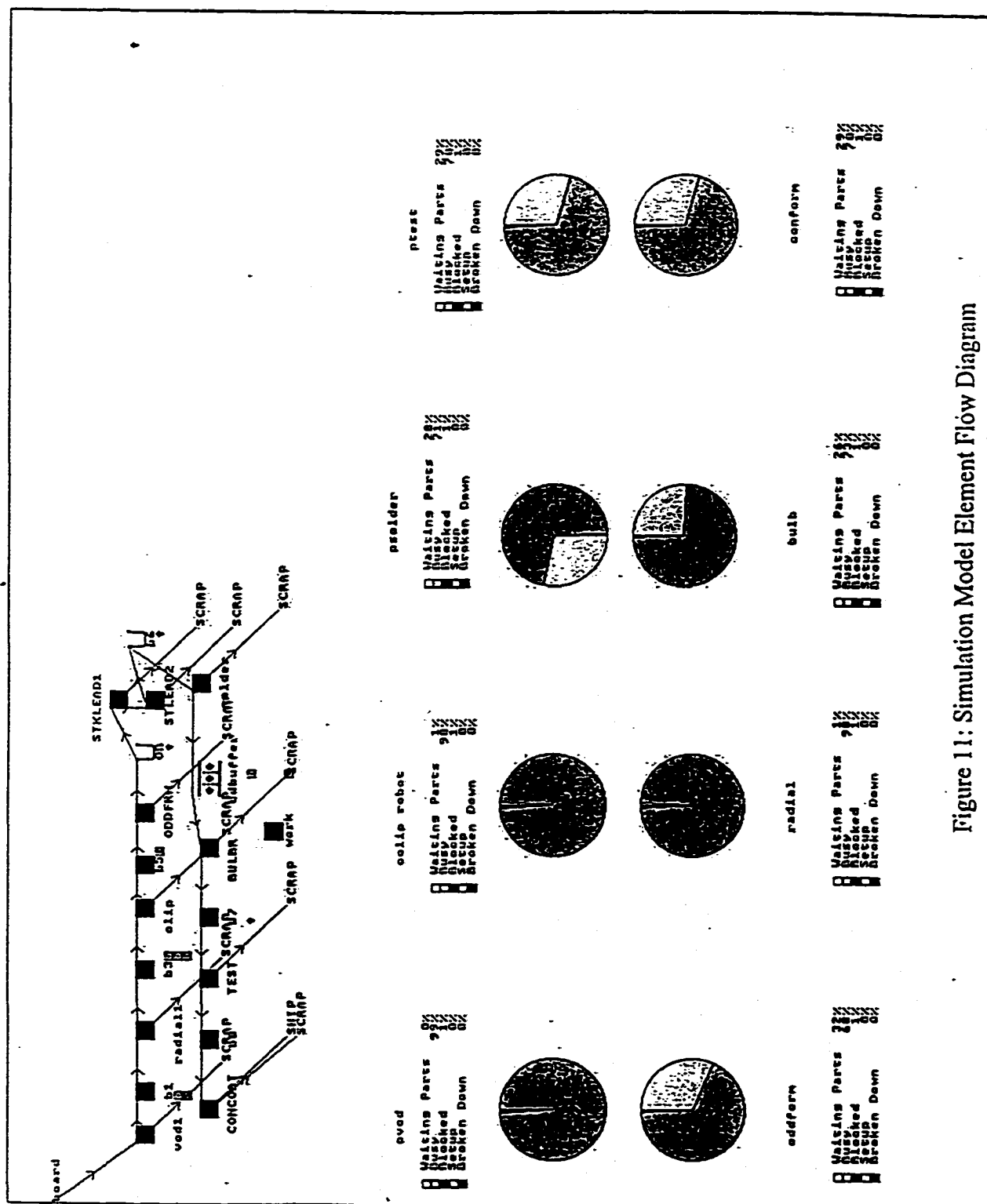


Figure 10: Simulation Model to analyze the throughput, quality and bottlenecks of an assembly line



**Figure 11: Simulation Model Element Flow Diagram**



Table 3: Sensitivity Analysis

SCENARIOS	CAPACITY	AVG.	AVG.	% GAIN		
THEORETICAL	PER LINE	W.I.P.	T.P.T			
PASS THRU	354k	24	22	N/A		
W/TAG RELIEF						
CURRENT D.T.	438k	21	18	24		
NO TAG RELIEF						
	401k	14	11	15		
TAG RELIEF	510k	113	108	44	Recommendation	
SMALL BUFFERS						
TAG RELIEF						
LARGE BUFFERS	522k	170	118	47		
GT 500 PIECES						

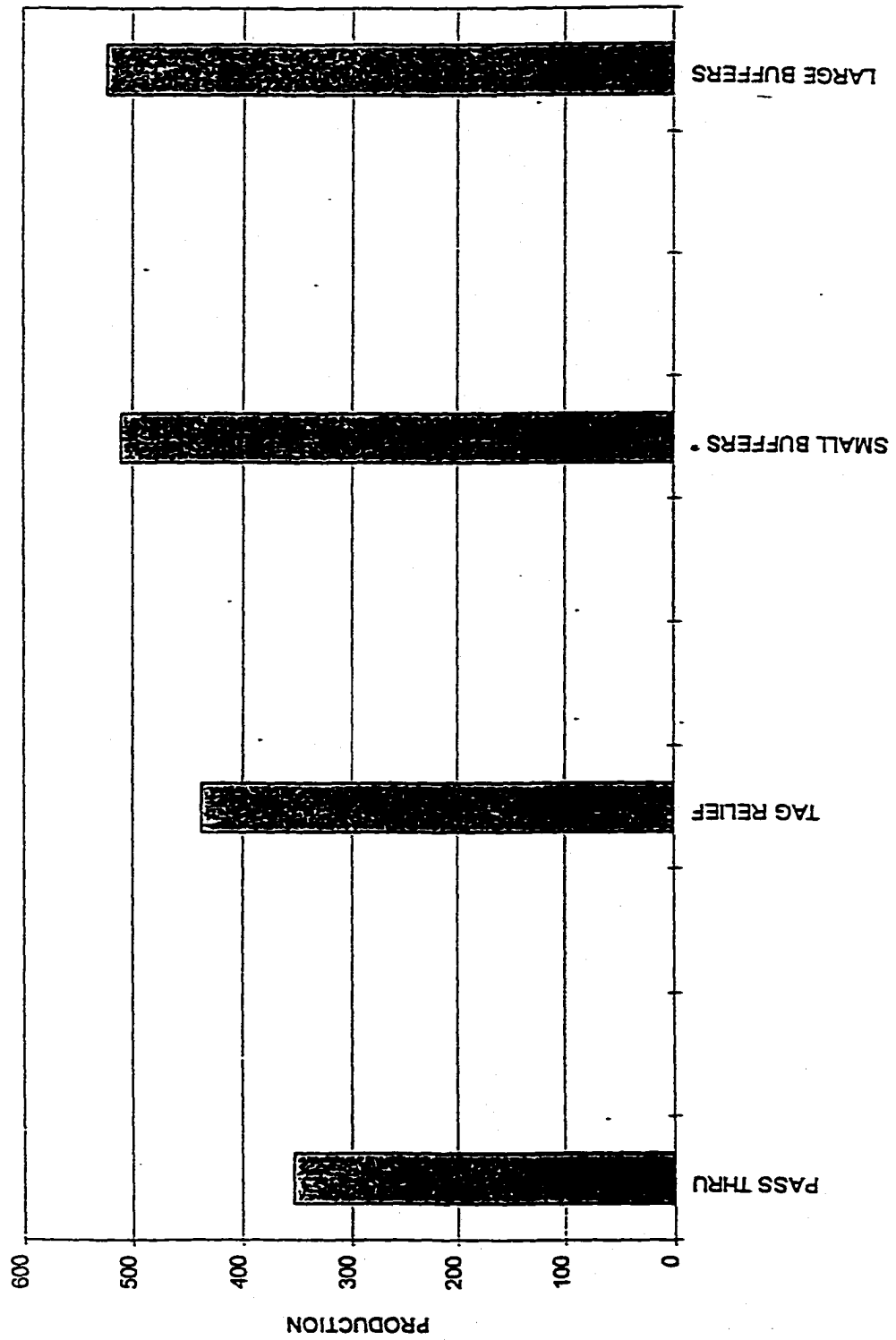


Figure 13: Sensitivity analysis



Axial Insertion (2)	Axial Insertion (2)	Axial Insertion (2)	Axial Insertion (2)
Radial Insertion (1)	Radial Insertion (1)	Radial Insertion (1)	Radial Insertion (1)
Clip Insertion (2)	Clip Insertion (2)	Clip Insertion (2)	Clip Insertion (2)
Hand Assembly (2)	Hand Assembly (2)	Hand Assembly (2)	Hand Assembly (2)
Wave Solder (1)	Wave Solder (1)	Wave Solder (1)	Wave Solder (1)
Bulb Insertion (2)	Bulb Insertion (2)	Bulb Insertion (2)	Bulb Insertion (2)
Test (1)	Test (1)	Test (1)	Test (1)
C.C (1)	C.C (1)	C.C (1)	C.C (1)

Figure 14: Process Cells

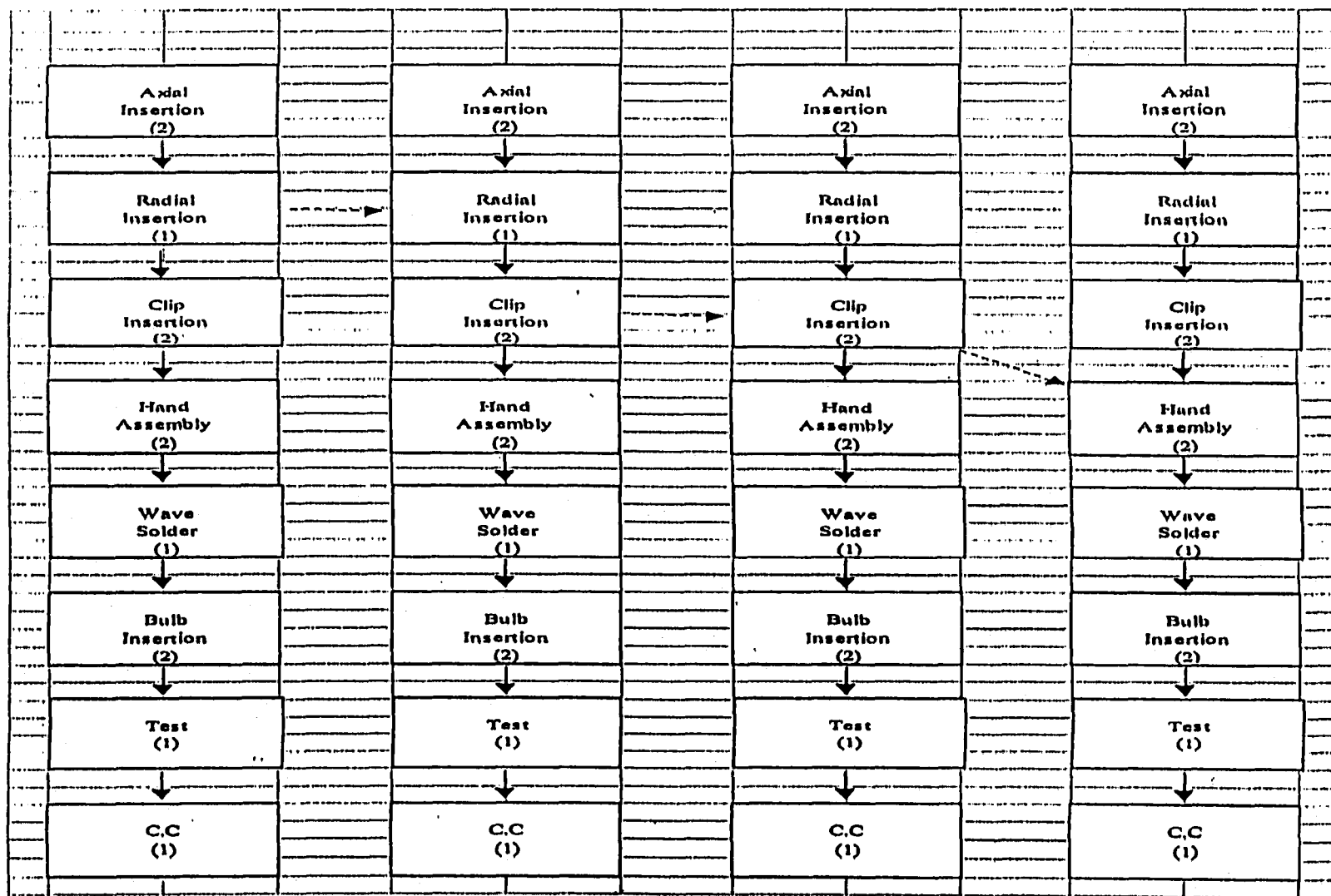


Figure 15: Integration/Consolidation of Cells

## **Analysis and Recommendations**

From the earlier design stages many qualitative factors had to be considered to evaluate alternatives. Finally the recommendations were made on the main aspects of manufacturing system operation to make JIT implementation successful.

The Witness models give steady state estimates of machine utilization; WIP and flow time based on the dynamic analysis of the system. The rough-cut analysis pointed out the bottlenecks in the system. The necessity to analyze the effect of machine performance improvements on the system parameters was felt. This led to perform a sensitivity analysis to examine the effects of improvement in the machine characteristics on the system performance. The what-if analysis was performed by making the changes in machine parameters. Detailed analysis using simulation helped to get more insight into the system performance. Rough cut analysis results showed that flow time and WIP were very sensitive to machine reliability and repair time.

The objective of the simulation analysis was to optimize buffer size between unlinked stations, in order to meet the required output. The simulation analysis progressed from confirming the rough-cut analysis results to final capacity planning based on physical constraints. The sensitivity analysis was the basis for most of the recommendations on the choice of buffer size. Small buffer size with tag relief was recommended based on the average WIP, capacity, average throughput and % gain numbers.

The different phases of the approach applied to the design of a circuit board manufacturing cell were detailed and the results are documented. Tables 4, 5, and 6 show the sample statistical summary reports.

Table 4: Statistical Summary Report

Item	Number Entered	Number Shipped	Number Scraped	Number Assembled	Number Rejected	W.I.P.	Av. W.I.P.	Av. Time
Board	34584	33112	1438	0	0	14	14.57	0.48

Item	Total In	Total Out	Now In	Now	Min	Average Size	Average Time	Aug. Delay Churn	Aug. Delay Time
B1	34273	34271	2	2	0	1.84	1.28		
OP	34170	34170	0	4	0	0.07	0.04		
B3	34378	34376	3	3	0	2.84	1.84		
B5	34274	34273	1	4	0	0.04	0.03		
B6	33272	33272	0	4	0	0.08	0.04		
B7	33287	33287	0	15	0	0.04	0.02		
B9	33204	33204	0	8	0	0.00	0.00		

Table 5: Statistical Summary Report

Name	%Idle	%Cycle - Busy	%Cycle - Filling	%Cycle - Emptying	%Stopped - Blocked	%Waking - Setup	%Stopped - Down	%Waking - Cycle	%Stopped - Setup	%Waking - Repair	Number of Ops.
COOFRM	45.61	53.22	0.00	0.00	1.01	0.00	0.00	0.00	0.00	0.00	34272
STLEAD2	47.40	51.36	0.00	0.00	0.75	0.00	0.48	0.00	0.00	0.00	33997
chp	1.12	97.79	0.00	0.00	0.78	0.00	0.21	0.00	0.00	0.00	34375
radcal	0.99	52.09	0.00	0.00	46.57	0.00	0.35	0.00	0.00	0.00	34471
vccl	0.00	86.03	0.00	0.00	13.10	0.00	0.87	0.00	0.00	0.00	34562
STLEAD01	49.98	48.40	0.00	0.00	1.34	0.00	0.19	0.00	0.00	0.00	34169
solder	39.00	60.13	0.00	0.00	0.40	0.00	0.47	0.00	0.00	0.00	33621
BLLR	34.51	65.05	0.00	0.00	0.12	0.00	0.22	0.00	0.00	0.00	33250
TEST	40.68	59.13	0.00	0.00	0.02	0.00	0.19	0.00	0.00	0.00	33350
CONCOAT	56.12	41.81	0.00	0.00	0.00	0.00	0.07	0.00	0.00	0.00	33704

Name	%Empty	%Access	%Miss	%Hit	%Queue	%Error	Name On	Total On	Max. Size	Max. Time
Emulator	2.44	88.68	0.88	0.88	0.00	0.01	3	33345	1.83	1.77

Table 6: Simulation Model Statistics

Item	%Idle	%Cycle - Busy	%Cycle - Filling	%Cycle - Emptying	%Stopped - Blocked	%Waiting - Setup	%Stopped - Down	%Waiting - Cycle	%Stopped - Setup	%Waiting - Repair	Number of Ops.
COOFFRM	12.90	64.50	0.00	0.00	21.54	0.00	1.06	0.00	0.00	0.00	3688
STLEAD2	14.89	63.92	0.00	0.00	17.55	0.00	3.84	0.00	0.00	0.00	3658
clip	13.43	64.78	0.00	0.00	20.71	0.00	1.09	0.00	0.00	0.00	3704
rediel	6.68	64.95	0.00	0.00	21.02	0.00	7.34	0.00	0.00	0.00	3714
vccl	0.00	59.73	0.00	0.00	33.28	0.00	6.99	0.00	0.00	0.00	3727
STLEAD1	12.70	64.36	0.00	0.00	21.13	0.00	1.81	0.00	0.00	0.00	3690
solder..	15.69	63.71	0.00	0.00	13.77	0.00	8.83	0.00	0.00	0.00	3643
DJLBR	20.89	62.67	0.00	0.00	14.67	0.00	1.98	0.00	0.00	0.00	3584
TEST	17.16	60.04	0.00	0.00	12.78	0.00	10.05	0.00	0.00	0.00	3583
CONCOAT	34.48	36.92	0.00	0.00	27.73	0.00	0.88	0.00	0.00	0.00	3578
CBCASE	16.88	66.94	0.00	0.00	14.17	0.00	0.21	0.00	0.00	0.00	3571
COL	11.31	65.66	0.00	0.00	0.00	0.00	3.03	0.00	0.00	0.00	3563
MARRAGE	21.08	73.86	0.00	0.00	1.73	0.00	3.25	0.00	0.00	0.00	3384
STAHER	20.96	76.74	0.00	0.00	0.00	0.00	2.30	0.00	0.00	0.00	3371

## **CHAPTER 4**

### **SYSTEM ARCHITECTURE, INTEGRATION AND OPERATION**

System integration is a difficult problem. Often the integration effort surpasses that of developing the intermediate subsystems. While the number of subsystems grows linearly with system size, the number of possible interfaces increases quadratically (every subsystem can potentially interface with each of the remaining ones). To alleviate this problem, it is important to carefully select both architecture and interfaces.

To be manageable, large systems need to be built around a solid architecture or framework. The advantages of modular system design are: simplicity (a large system is broken into smaller, simpler components which can be understood and developed in relative isolation), maintainability (failures can be isolated into the originating modules where they can be more easily addressed), reliability (redundancy can be introduced with replicated/redundant modules) [Pardo-Castellote, 1995]. However, modularity does not come for free. The design of the module interfaces and the interconnection of the different modules (i.e. the systems integration) becomes a large part of the overall development effort. This chapter presents the system architecture, interfaces and system operation.

#### **Factory Information System Overview**

The **Factory Information System (FIS)** is a tool designed to help enable the company to achieve Global Electronics Manufacturing Metrics performance in the manufacturing and assembly of electronic products. The intent of FIS is for the

company to have the capability to deploy a corporate system to share information across the globe, duplicate best practices, and drive improved performance levels across all manufacturing operations. Global Electronics Manufacturing best practices have been established and metrics are being employed to measure on-going performance. Some of the key Global Electronics Manufacturing best practices are: Mission and Vision, Resource Utilization, Equipment Maintenance, Factory Information System, Quality Improvement System, Supply Management, Uniform Performance Metrics, Learning Organization, Employee Participation, and Visual Management. The prominent electronics manufacturing companies, like Motorola and Honda, use the following metrics in their process to measure their performance: Part Number with Product Cost less than Target Cost, Equipment Utilization, Equipment Uptime, First Time Quality, Critical Processes with  $CpK > 1.67$ , Customer Returns in Parts Per Million (PPM), Delivery Performance, Manufacturing Cycle Time, Fully Integrated Focused Factories, Workplace Organization.

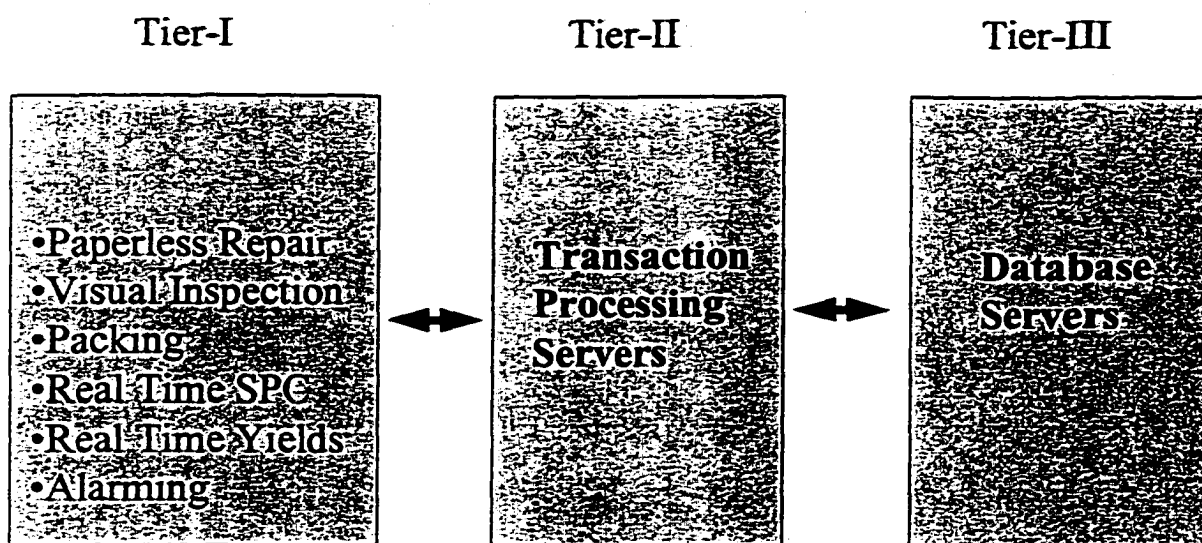
FIS represents the plan that ties the strategic business interests, manufacturing operations, and re-engineering efforts together in order to enable competitiveness and world leadership in the supply of automotive electronics.

### **System Functionality**

FIS is three-tier client server architecture. The first tier is the client interface function. This interface could be communications to factory equipment or a communications with a human operator. The main function of the Tier I application programming interface is to communicate with the client, sending requests from the client to the Tier II layer as well as presenting the results from Tier II back to the



client. Tier II is responsible for taking action on client requests. The application is configured during implementation to handle all site-specific information, such as what sequence of processes will occur on an assembly of a particular product. The business rules for a factory are managed within the Tier II application. Tier III is the Data Base Management system and will be responsible for either retrieving or storing pertinent data for Tier II.



### The Base System

The base system includes the initial system hardware configuration, software, and initial FIS set up. As components of the system are added, the hardware configuration is enhanced to meet additional requirements. The base system includes the real-time Product Flow Tracking system, which is the cornerstone of FIS. Product Tracking records the 5 W's: What, When, Where, Who, and Why of a unit of material through production. It tracks the product while it is being built from start of line through end of line.

The database is included in the base system. The DBMS chosen for FIS is Oracle. The base system is configured to retain three (3) months of data on-line. There are utilities included in the base system to archive and de-archive data from and to the DBMS. There are also several Quality and operational reports that are included in the base system. Reports included are Uniform Report, Daily Trend Charts, and 2-hour reports. These reports are generated via Excel.

Application programming interface (API)'s are responsible for communicating with the client and collecting the information a client wants to process. This includes configuration of a Windows NT server, where all API's resides as well as testing and integration of factory equipment or manual input. The information collected by the API will provide input to the real time product flow tracking function. Figure 16 shown below depicts a logical view of the FIS configuration.

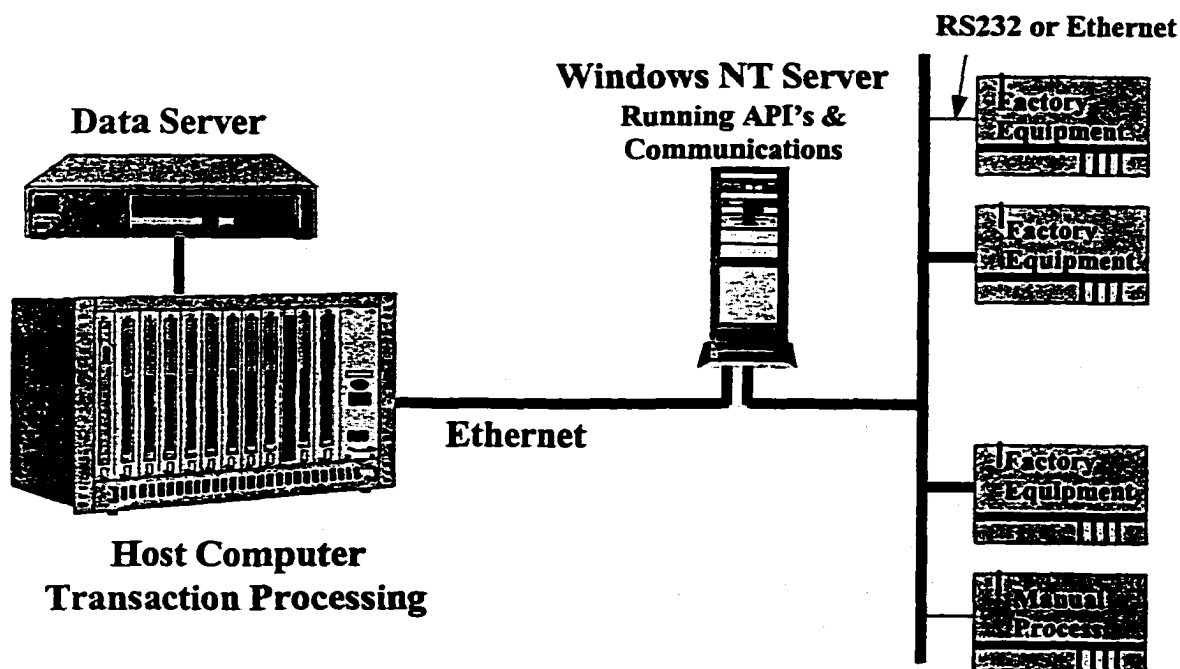


Figure 16: Logical view of the FIS configuration

Some of the significant features of the base FIS system are described below:

### **Modeling a factory**

The base system provides the ability to model the factory and the process flow of in-process material through the factory. A graphical model of the factory layout including machines and operations in the factory can be created and the flow of material within the factory can be animated. The base system is modular, flexible and scaleable. It is easy to modify an existing factory layout by adding, changing or removing operations and machines. Process flow routing can also be changed very easily.

### **Modeling in-process material**

In-process material (or work-in-process - WIP for short) can be modeled as lots or boards in the base system. Data pertaining to both types of material can be stored and retrieved easily and efficiently from the tier III database. Instead, the smallest unit is a lot, which is a group of several similar items. Board based tracking is used when a factory processes material in terms of individual serialized units. The smallest unit is a board in this type of tracking.

### **Tracking in-process material**

The base system tracks the complete path of units (whether lots or boards) through the factory - from the creation of a unit to its completion and stores the associated information in the tier III database.

New units can be created based on new material received in the factory. The type of product for each unit and, optionally, the order with which it is associated can

be tracked as well. The progress of each unit can be tracked at each operation and machine in the factory.

Critical data such as unit identification (id), unit status (whether the unit was successfully processed at the machine), cycle times, shifts associated with each unit at each step in the manufacturing are stored in the tier III database. Once a unit is completed, its status is changed from being in process to being complete and all information pertaining to it is stored in the tier III database.

#### **Viewing history of in-process material (WIP history)**

The base system provides the ability to view the WIP history of units in a variety of ways:

Specifying the id of a unit displays the WIP history for that particular unit.

Specifying the id of an operation displays the WIP history of all units that have been processed at that operation and are not complete yet. Selecting a unit being animated on the factory layout displays the WIP history of that particular unit. All units located on a conveyor connecting machines on the factory layout can also be seen.

The overall factory WIP can also be obtained. This provides factory-wide information indicating the number of units (lots and/or boards) that are currently in process, that have been completed and those that have been scrapped.

#### **Tracking parametric data**

The base system permits the tracking of parametric data reported by machines on the factory floor as well. This information is used to create statistical process control (SPC) charts and reports. New parameters can be dynamically created when

machines report them. These parameters can be configured so that various SPC charts can be generated.

### **Quarantining and retrieving in-process material**

A mechanism to quarantine in-process material and retrieve material that has been quarantined is included in the base system. There are two ways to quarantine in-process material (and retrieve it): By specifying the id of a unit and by specifying a machine and a time interval i.e., all units processed at a given machine in the time interval between two specified times.

### **Statistical Process Control (SPC)**

Statistical process control (SPC) is a capability available in the system to assist engineers in reducing process variability. SPC is a systematic application of statistical methods to uncover and understand the source of process variation. This is based on real time values or parameters collected from test machines throughout the manufacturing process. These data can also be collected manually if machines are not capable of sending pertinent information. SPC monitors process characteristics to determine process capability and stability.

When Key control characteristics (KCC's) are identified for a process, an engineer can monitor these KCC's to make improvements to the process. Of course, if these data are to be collected from the factory equipment, the equipment must be capable of sending the data to FIS. If the factory equipment is capable of sending the data, information can be collected by FIS. Once the engineer in FIS defines the process specification and control limits, the process can be monitored automatically without human intervention. When a violation occurs, a feedback loop is initiated in

accordance with QS9000 guidelines to achieve a process log of assignable causes and irreversible corrective actions with responsibilities and implementation dates. When the control limits or specification limits are outside of the parameters, alarms are initiated. The alarm can be anything from a light flashing up to FIS stopping the factory equipment.

SPC has the following functionality:

### **Data Collection**

Each machine can send process and product related parameters to FIS. If the SPC flag is set to true for a parameter, then the data are sent to all the SPC charts that are configured for this parameter. Graphical User Interface is provided to configure parameter's spec limits, subgroup size, alarm generation rules and alarm acknowledge mechanism.

### **Charting**

Available charts are:

Xbar/R

Xbar/Sigma

Individual X/Moving R

Median/R

u

c

p

np

Parato

Histogram

Gage R&R

### **Alarming**

Configurable rules are used for alarm generation for each SPC parameter. When a violation occurs, an alarm is generated and users are notified by the color appearance of that equipment (in red). Alarms can be cleared by completing acknowledgement procedures, which is configured for each parameter and can be dynamically reconfigured.

### **Paperless Repair**

This system dramatically reduces the overall time of repair, as operators no longer have to use hardcopy schematics of components.

### **Packing**

The packing system is a graphical aid for the packing operation of the factory. It shows operators where to pack the next unit for shipping while collecting traceability information for cartons and packed units.

### **Automatic Change Over**

Automatic change over (ACO) allows automatic download of product recipes or instructions. This flexibility reduces change over and set up time for new products. The machine instructions are automatically obtained from storage and presented to the machine. At any given time, one or more types of product can be manufactured in any electronic manufacturing or assembly plant. Consequently, each unit undergoing processing in a factory could be of a different type of product and each machine in the factory could have instructions to process a different kind of product. This sets up the

potential for a major problem during manufacturing. ACO synchronizes the machine software with the type of product of each unit to prepare the machine to process the unit. Features of the automatic change over function are described below:

Each time a unit is to be processed at a machine, a check is performed to see if the machine has the correct software (or instructions) to process that type of product. If it is determined that the software loaded on the machine is not the correct program to process a given unit, the ACO function issues an error indicating that a software change over is needed.

If the machine in turn requests the correct software to process a given unit, ACO provides the correct software to the machine based on the product type of the unit about to be manufactured.

### **Equipment Monitoring**

This function provides automatic calculation of real time yields, equipment uptime and utilization, and preventive maintenance. Some of the features of the FIS Equipment Monitoring function are:

The FIS has a wide array of machine parameters that it tracks in real time to monitor factory equipment.

First time yield

Total yield

Maximum cycle time

Minimum cycle time

Last cycle time

Average cycle time



Total uptime

Total downtime

Utilization

Production per shift

Machine status (Idle, Running, and Process Out of Control etc.)

Current product being manufactured on the machine

Machine software and its version

Preventive Maintenance

The ability to configure a new scheduled preventive maintenance for any equipment based on either product unit or time.

Generates preventive maintenance warning when a scheduled PM is due

Allows the user to perform any non-scheduled PM activities

Logs both scheduled and non-scheduled PM activities in the database

Logs equipment PM downtime for future pareto charting

### **Component Traceability**

Component Level Traceability (CLT) will enable the company to better monitor vendor component and lot quality. In a real-time environment, CLT will trace, alarm, quarantine and report on an assembly and its constituent materials in a module. CLT will allow for human, machine or other database interfaces

CLT has the following functionality:

**Alarming:** The ability to generate an alarm either when the material/or component is quarantined by vendor lot or per user's request via API's. The ability to generate an alarm either when the material prediction warning time is reached. The

ability to visually notify operations/engineers which equipment that the alarm is associated with. The ability to acknowledge and reset an alarm condition.

**Quarantine:** The ability to quarantine material/or component by vendor lot. The ability to create an alarm for the equipment where this material is used / or this component is placed when the material/or component is quarantined. The ability to notify operation/engineers where and when the quarantine is done. The ability to acknowledge quarantine condition via alarm acknowledgment.

**Interface:** Graphical User Interface to input Component Data and Material Storage Time (MST) data. Graphical User Interface to view vendor lot usage information for material and component. Graphical User Interface to quarantine material/or component by vendor lot.

**Component Verification:** The ability to verify the shelf life / expiration date for a given material vendor lot. The ability to verify the max running time on machine for a given material vendor lot. The ability to notify the user and generate an alarm when the material prediction warning time is reached. The ability to notify the user when the material/or component has expired, is not ready to use, quarantined or exceeded the max running time on the equipment. The ability to verify component product information.

### **Prior Step Verification**

At any given time, one or more types of product can be manufactured in any electronic manufacturing or assembly plant. Consequently, each unit undergoing processing in a factory could be of a different type of product. Depending upon the type of product, each unit has a specific process sequence or route associated with it,

which prescribes what operations the unit will undergo during its manufacture and in what order these operations will be performed.

Prior step verification (PSV) is a tool that ensures that each unit being manufactured is being processed according to the process plan or sequence prescribed for the type of product that the unit belongs to. To do this, PSV first finds the previous operation that the unit has undergone. Next, it verifies that the previous operation was indeed a valid operation for the manufacture of a unit of the specified product type and that the unit has a valid status (Pass/Fail) from the previous operation.

Each operation modeled in FIS has a limit associated with it to indicate the maximum number of times a unit can be processed through the same operation. PSV verifies to make sure that this limit is not exceeded for a unit and flags the unit if it exceeds this limit.

### **Scheduling Execution**

This function enforces product running based on work orders. A work order has to be created with the total quantity and the final product information before the first process can begin. Each time a product goes through the first process, the built quantity of the current work order is decrement by one. Work orders are considered fulfilled when the built quantity is equal to the total quantity and no more products can be built against this work order. Whenever a product finishes the final process, complete quantity of that work order is incremented by one. A work order is successfully completed when the complete quantity is the same, as the total needed quantity. At any single time, there can only be one active work order. Every work

order has a process priority. The system can automatically activate a work order when the current work order is fulfilled based on its priority.

### **Alarm Management**

An alarm management module manages all different alarm creations, alarm acknowledgements, and alarm displays. The SPC module, Component Traceability module, Paperless Repair and any Application Programming Interface (API) can generate alarms through Alarm Management. All alarms are associated with a specific equipment (or human station) in the system. Alarms can be acknowledged by completing a configurable serial of procedures. Each equipment has a 'Display Alarm' menu choice to launch alarm display screen. Users will be notified by the color appearance of the equipment whenever there is an alarm generated for that equipment.

### **Operational Report**

FIS provides many operational reports. These reports are generated via Excel.

#### **1. Yield Report:**

Yield by tester by shift

Yield by tester by day

Yield by product by shift

Yield by product by day

Yield by fixture by shift

Yield by fixture by day

#### **2. Tally Report:**

Defect Tally Reports by Summary

Defect Tally Reports by user-defined (i.e. parts, process, workmanship) groups

Repair Tally Reports by Summary

Repair Tally Reports by user-defined (i.e. parts, process, workmanship) groups

### 3. Packing Report:

Units Packed by Shift

Units Packed by Product by Shift

Units Packed by Day

Units Packed by Product by Day

Location and Number of Duplicate Labels

Serial Numbers by Carton

Carton Numbers by Date and Assembly Plant

Serial Numbers by Date and Assembly Plant

### 4. Test Result Report:

Test Results by Summary by Product

Test Results by Test by Product

### 5. History Report:

Event History for Unit Ids and Alias Ids

Event History for Work Order

Units built by Work Order

Units built by Work Order by Time

Maintenance History Summary

Maintenance History by Machine

**Alarm and SPC Related Report:**

**Amount of Scrap by Time**

**Cost of Scrap by Time**

**Descriptive statistics by SPC parameter**

**Alarm Causes and Corrective Actions by Alarm by Machine by Operation**

**6. Uniform Report**

**7. Daily Trend Chart**

## **CHAPTER 5**

### **PACKOUT VERIFICATION SYSTEM**

#### **New Packout Process**

Significant attention was directed to first-time-through quality and focused on reducing the number of parts rejected by the company's customers (measured by the number of Problem resolution requests (PRRs). The company used for this study identified that, at their manufacturing facility, over 25 percent of all PRR's were due to mis-labeling of products. These mis-labeling accidents at the company's facility in turn were causing quality accidents at company's customer facilities. The reoccurrence of these errors resulted in a serious financial penalty from customer facilities (estimated at over \$600,000 per year) and the strong potential for losing business. Mis-labeling errors at the parts plant can have serious impacts on assembly plants productivity and can cause considerable confusion in their material handling systems. A new Packout Verification Process was implemented to reduce and eventually eliminate labeling errors.

There were two steps to implement the new packout process. The first step was to get part label printers on every final assembly line to start printing their own part labels. The bottom bar code on the part label is the portion that is scanned into the new pack-out units. The assembly plants scan the upper bar code. The typical part bar code label is shown in Figure 17. The second step was the implementation of the new packout verification scanning process. The new process takes a different pull signal from shipping. The shipping scan sheet is shown in Figure 18.



Figure 17: Typical Part Bar Code Label



SUPPLIER NAME:

CUSTOMER REQ'D DATE 0623 1

PART NO. (01)

16175440



QUANTITY NO (02)

250



PULL SIGNAL (03)

0063IR



ENG CHG (04)



Figure 18: Typical Shipping Scan Sheet

Shipping used to send shipping labels as the pull signal to the final assembly lines. This caused confusion at the final assembly lines and also was one of the primary reasons for mislabeling errors. Shipping now prints what is called a shipping scan sheet. It has four bar code fields that are scanned into a packout unit at the start of a new shipping container. These fields are the part number, quantity, control code, and engineering change level.

Each part is scanned into the unit, as it is packed to the shipping container. The unit verifies that the part label matches the number scanned by the shipping sheet. It also verifies to see if the part had previously been scanned into the unit. If a part label passes those two tests, the unit increments the part quantity by 1. If a test is failed, the unit puts up an error message and beeps. Two shipping labels are printed when the number of parts scanned matches the quantity scanned in at the beginning. The process then starts all over. The shipping label format is shown in Figure 19 and the pack-out process flow diagram is shown in Figure 20.

#### Barcode Labeling Process Flow:

- Print Pull signal in Shipping
- Put Pull Signals in Racks
- Deliver Pull Signal to Each Department
- Place Pull Signal in Rack by Part Number
- Get Pull Signal before Scanning First Piece
- Scan Pull Signal (P/N, Quantity, Pull Signal)
- Put Pull Signal on Top of Printer






PART NO. (P) 29503600	
	
QUANTITY (Q) 36	PULL SIGNAL (K) 032556
	
SUPPLIER (V) 041456	GA 286
	DOORS MANUAL-OWNER MAINT
SERIAL (S) 101180	DELIVER TO: DR FINAL
	DRY: T21
XYZ MFG COMPANY INDIANAPOLIS, IN 46224	

Figure 19: Typical Shipping Bar Code Label

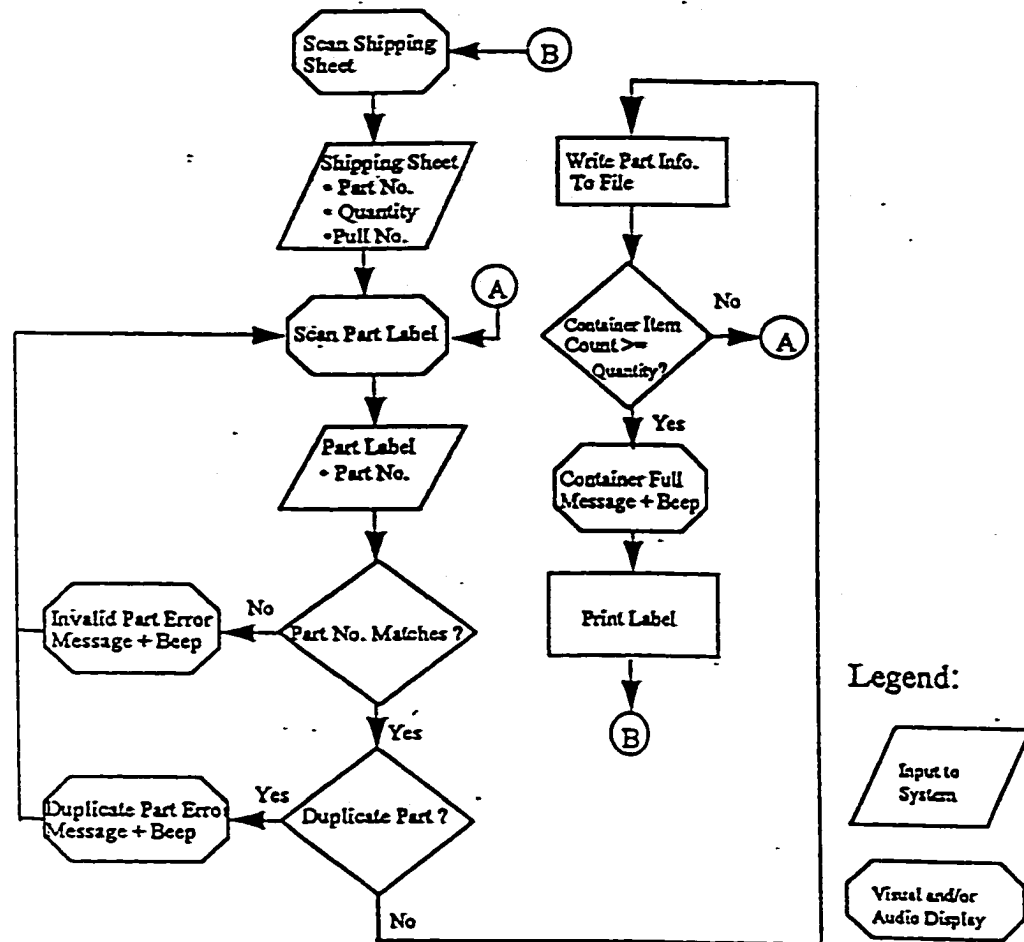


Figure 20: Packout Process Flow

- Scan Product Until Pull Signal Quantity is met
- Print Bar code Container Labels (automatically)
- Place Labels on Plastic Holders
- Put Labels on Adjacent Corners of Shipping Container
- Attach Pull Signal to last Cluster (i.e. on bag or Spot Stick)

Mislabeled is defined as a process of attaching an incorrect piece of marked paper to a container of parts. There are many reasons for mislabeling errors and usually the opportunities result from parts without labels generally being in the form of partial containers at the end of the manufacturing line. This sometimes occurs as a result of unanticipated scrap along with giving the lead operator a set of quantity of pieces to be produced.

Labels without parts can be seen on returnable containers, which are allowed to arrive at the packer's workstation. Pre-printed labels are the result of accepted practice at the individual work sites and the fear of running out of labels.

Some of the label error categories and their definitions are listed below:

**Duplicate Serial Numbers:** Each set of bar code shipping labels (two labels per container) has a unique serial number. Duplicate serial numbers is a label error where two or more shipping containers have shipping labels with the same serial number.

**Incorrect Part Label:** Label error where individual label on the part is incorrect. Label could be a bar code label, a broadcast code, a color code, etc.

**Incorrect Shipping Label:** Label error where shipping label does not identify the parts in the container. The parts in the container are not the ones referenced on the shipping

label. Typically refers to the label on a basket, a pallet, a master pack, etc.

**Wrong Quantity on Label:** Quantity referenced on label is not the quantity contained in the container.

**Missing Part Label:** Label error where label, which was supposed to be on a part, is missing. Label could be a bar code label, a broadcast code, a color code, etc.

**Missing Shipping Label:** Label error where the shipping label for container is missing. Typically refers to the label on a basket, a pallet, a master pack, etc.

**Missing Information on the Shipping Label:** Label error where the shipping label is not filled out completely; it is missing information such as date, serial number, etc.

**Mixed Parts:** Label error where different part numbers are mixed within one container. Two or more part numbers are mixed in the same container and identified as one of those part numbers. If parts have a part label, the mixed parts label error could occur if both part numbers had correct part labels (label error = mixed parts) or if one or both part numbers had incorrect part labels (label error = mixed parts and incorrect part label).

**Incorrect Label on Individual Containers:** Label error, which occurs when parts shipped in small cartons have the wrong label on one or more of the small cartons. The cartons are shipped together a "master pack". This label error refers to the labels on the individual cartons being wrong. If the shipping label were not correct also, the label error would be both incorrect label on individual containers and incorrect shipping label.

**Mis-Sequenced-Container:** Label error where sequence in the container is not correct.

Unable to read bar code label: Label error where, for some reason, the customer is unable to read the bar code label.

Mixed Containers: Label error where there are two or more part numbers on a “master pack”, skid, pallet, etc, when all cartons should have been the same part number. In this case, the individual containers are correctly labeled.

Major contributors:

Two conditions must exist for a container of parts to be mislabeled:

1. Opportunity - Parts in the assembly area without associated labels attached and/or unattached labels available in the assembly area create the opportunity for mislabeling and thus increase the chance of labeling errors.
2. Misapplication - Not only must the opportunity for the labeling error exist, but the wrong label must then be placed on the part container. Attaching the label takes a matter of seconds.

Mislabeled errors have a significant impact at the assembly plants. Listed below are the areas of impact the assembly plant has to contend with:

- \* Variation in the assembly plant quality
- \* Point of use misses
- \* Line Stoppages
- \* Incorrect parts assembled to the vehicle
- \* Material sorts
- \* Floor count of inventory
- \* Schedule adjustments
- \* Expedited material

- \* Disruption of material flow
- \* Confusion to the assembly line operator
- \* Confusion to the material pull system
- \* Premium overtime expenses

### **Introduction to Bar Coding Systems**

Bar Coding Systems is one of the most active of all the identification technologies. Some of the major considerations for implementing a bar code system are: cost of bad information, direct labor improvements, increased output of capital equipment, reduced inventory and improved control, improved quality reporting and traceability, higher level of customer service. Bar code systems are not just an automation of the data collection function; they are a rapid, accurate and cost effective method of capturing data about any individual item's identity, location and status.

Bar coding is becoming increasingly important for a number of reasons. Data processing happens quickly and effectively, but data can be processed only after it is entered. Bar coding greatly speeds input time allowing processing to occur more efficiently. Secondly, distributed data processing (in which the data may be derived from many sources) requires distributed data entry systems; multiple scanning stations easily accommodate this.

Perhaps, most importantly, business organizations require technologies that enhance productivity and allow them to compete better in the world's market place. Bar coding technology has become the dominant technology for automatic identification.



Material handling, manufacturing, warehousing, transportation and distribution can represent anywhere from 30 to 90 percent of the cost to bring a product to the market place. Poor material handling practices can result in the inefficient utilization of space, labor and energy. Firms that minimize handling costs will have a competitive advantage in their industry - let the rule be - "the system that handles the best handles the least." There is one common requirement for all material moving through the various stages of the processing pipeline - it must be identified before appropriate action can be initiated. Bar code readers can identify products moving at speeds that blur human vision with accuracy and consistency that defies human involvement.

Potential savings / benefits of bar code technology include real-time on-line control of people, process and the product.

1. Faster data entry
2. Enhance data accuracy
3. Reduction of material handling labor
4. Fixed asset / property accountability
5. Minimize on-hand inventory
6. Monitor labor efficiency
7. Improve customer service
8. Reduce product recall
9. Verify orders at receiving and shipping
10. Reduce work-in-process idle time
11. Monitor and control shop floor activity

12. Improve production scheduling
13. Optimize floor space
14. Improve product yield/reduce scrap

Bar code applications comprise a system of three elements - the symbol, the reader and the print process. In a bar code system, all elements are interdependent; each with equal importance determines how well the system operates.

### **Bar Code Symbols**

A bar code symbol is a group of parallel dark bars of varying widths separated by light spaces and arranged in a predetermined pattern. Variations in bar and space widths represent numeric or alphanumeric (combination of letters and numbers) information in machine-readable form. A bar code symbol is the representation in actual bars and spaces of a specific group of numbers or characters. Each bar code symbology is a code type, with its own standards or rules about how bar and space widths are arranged to uniquely identify each number or character. As a language relates to a word, a bar code symbology relates to a bar code symbol.

All bar code symbols are made up of a sequence of individual characters. A character is a combination of bars and spaces that represent the individual number, letter, punctuation mark or other graphic font. Symbologies in widespread use today include all of the following: Code 39, Code 2 of 5, Interleaved 2 of 5, UPC (Universal Product Code), EAN, Codabar, Code 93, Code 128.

Code 39 is the most widely used alphanumeric symbology. Its 43-character set consists of all the upper letters, numerals from zero through nine, and seven special

characters. Code 3 of 9 or code 39 is so named because three of nine elements in each character are wide; the remaining six are narrow.

Code 128 is a recently developed high-density symbology, so named because it encodes the complete set of 128 ASCII characters without adding extra symbol elements. Its structure is similar to that of UPC, facilitating edge to similar edge measurement techniques. This symbology offers the option of variable length symbols. Since it can be encoded with all ASCII characters, Code 128 can encode any character found on a CRT keyboard, including control characters.

### **Printing Methods**

There are many different methods presently available for printing bar code symbols. Every successful bar code printing method satisfies two fundamental requirements. First, it must be able to create an image of light spaces and dark bars; the widths of these bars must vary according to a standard scheme and must be held within specified tolerances. Secondly, the method must be able to create an image with sufficient optical contrast between the light spaces and dark bars. Available printing methods can be grouped under two general classifications off-site and on-site printing. When selecting an on-site printing capability, the following factors were considered:

Volume - How many labels are needed and how often?

Flexibility - Can the printer produce the desired format?

Symbol Quality - Do the inherent limitations of a printer type rule it out for use in a particular application?

Cost

Type of reader to be used

Intermec 3400 direct thermal printer was used.

### **Scanner**

The bar code reader mainly extracts width-encoded data from the printed symbol. To do so, it illuminates the symbol with a moving spot of light and monitors the reflected light level. Reflected light from a symbol space converts into a logic one electrical signal; reflected light from a bar converts into a logic zero electrical signals. The result is binary data that can be entered into a computer.

The LS 9100 projection scanner features a highly sophisticated programming capability which allows not only to modify its basic scanner operating characteristics, but to modify the content of bar code labels to enhance the capabilities of POS data capture application system/software. The LS 9100 is compact, fully integrated, reads all popular retail symbologies, and interfaces to all popular POS terminals. The LS 9100 Scanner can be mounted on a counter, on top or on the side of the POS terminal, or on the wall. The LS 9100 is an Omni-directional scanner which emits a 20-line scan pattern designed to allow to read bar codes, even very long and narrow ones, easily from almost any angle. The scan pattern is projected directly out of the scan window creating the three-dimensional "scan zone".

### **TimePoint computer**

The TimePoint computer may boot and operate locally by using a PCMCIA SRAM card. The SRAM card must be formatted and programmed using a PCMCIA reader/writer or a computer capable of handling SRAM cards. The 68-pin connector for the PCMCIA card may be found on the left side of the main circuit board. The

PCMCIA connector will accept type I SRAM cards. First, cards must be formatted using a PCMCIA reader/writer. Bootable cards must be formatted as system disks. The TimePoint SRAM card support emulates a floppy disk. Therefore, once formatted, the SRAM cards appear to be a floppy disk to the operating system and application systems. The five-pin header is provided for connection to an XT keyboard. This port adds full keyboard capability for file management, debugging, and system maintenance. The TimePoint computer provides two ports for bar code inputs. The main port is a DE9 connector with locking tabs. The secondary connector is a 5-pin header. The built-in bar code slot or magnetic stripe connects to the 5-pin header. TimePoint computer hardware is shown in Figure 21 and 22.

### **System Overview**

The purpose of the Packout Scanning System is to improve the accuracy of the packout process by automating the process of verifying whether parts are being packed in the right containers. This system consists of a bar code scanner, a data collection terminal, and a bar code printer. The system checks for duplicate and invalid parts, and also generates container labels once the required number of parts have been packed in a container. Figure 23 and 24 shows the hardware components of the Packout Verification System.

### **Hardware Overview**

The Packout Scanning System consists of the following hardware components.

- Symbol LS9100, Omni-Directional scanner. This scanner features hands-free operation. The LS9100 also has multiple mounting options and an adjustable scan range.

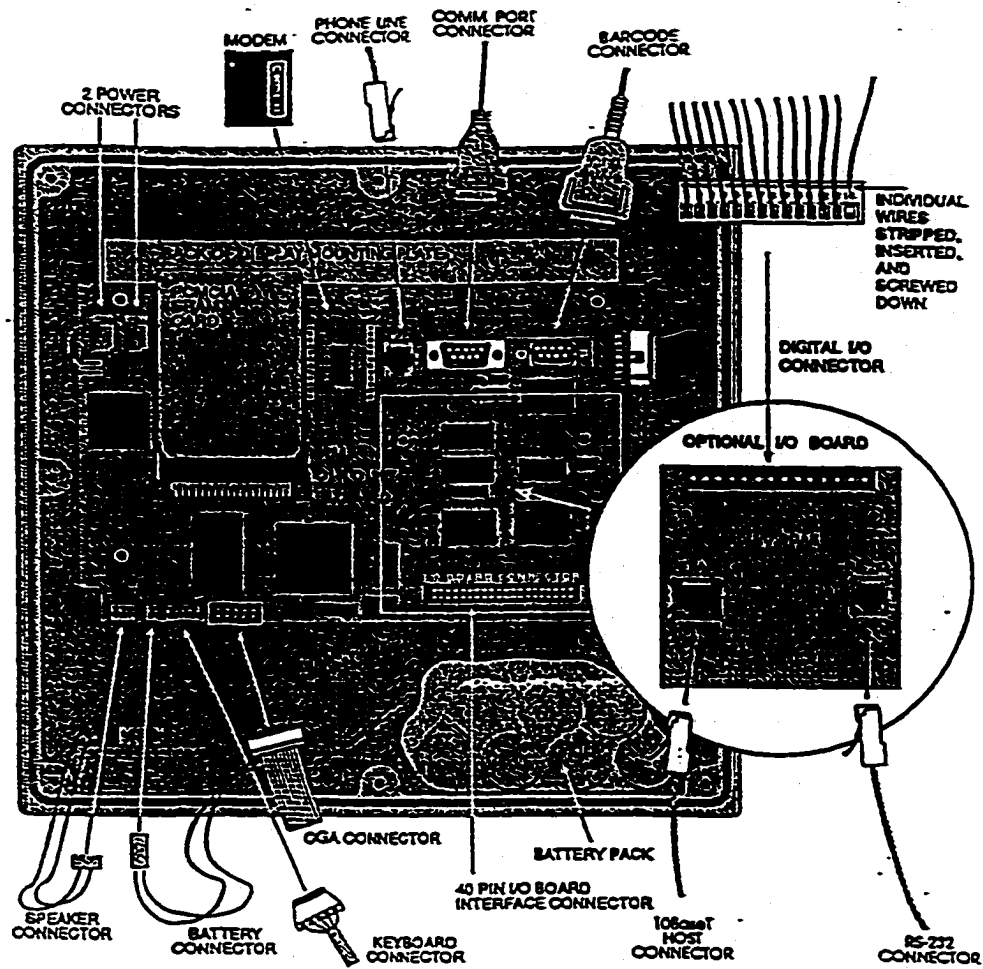


Figure 21: TimePoint Computer Showing Circuit Board and Connectors

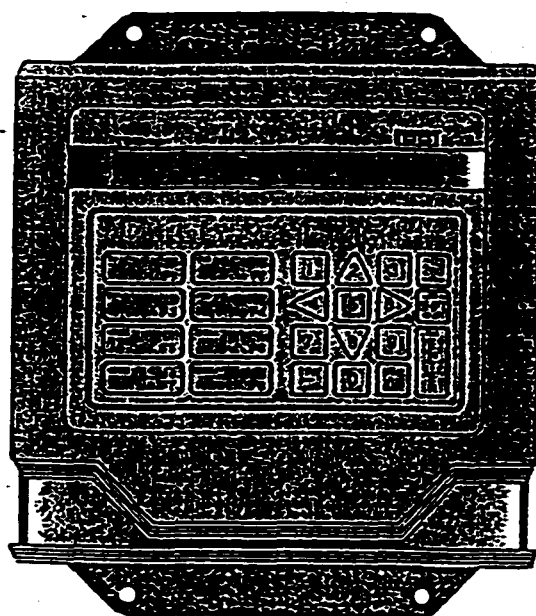


Figure 22: TimePoint Computer with Standard Mounting Plate

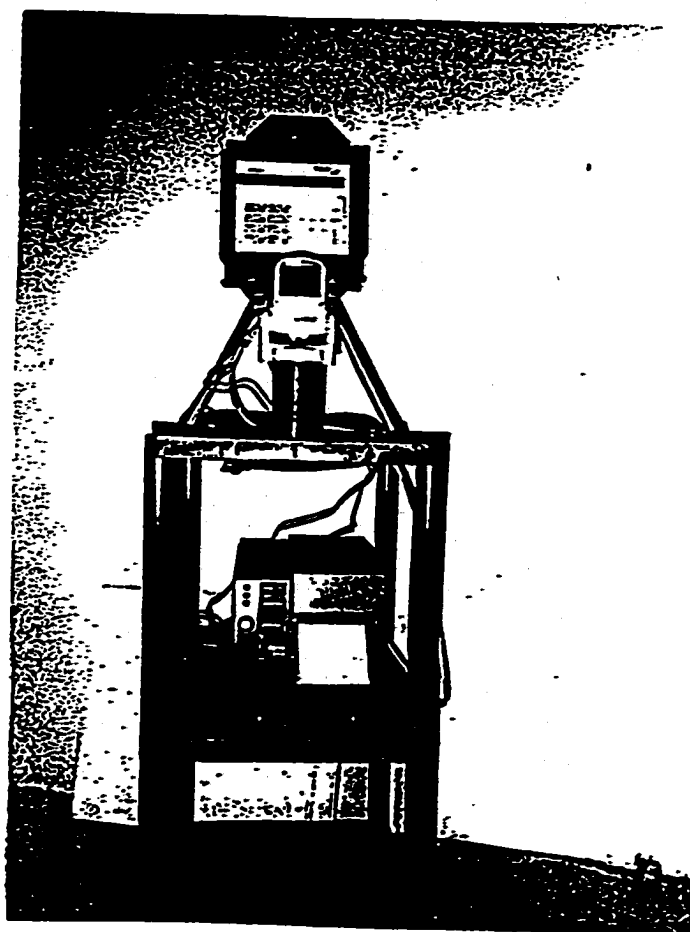


Figure 23: Packout Verification System



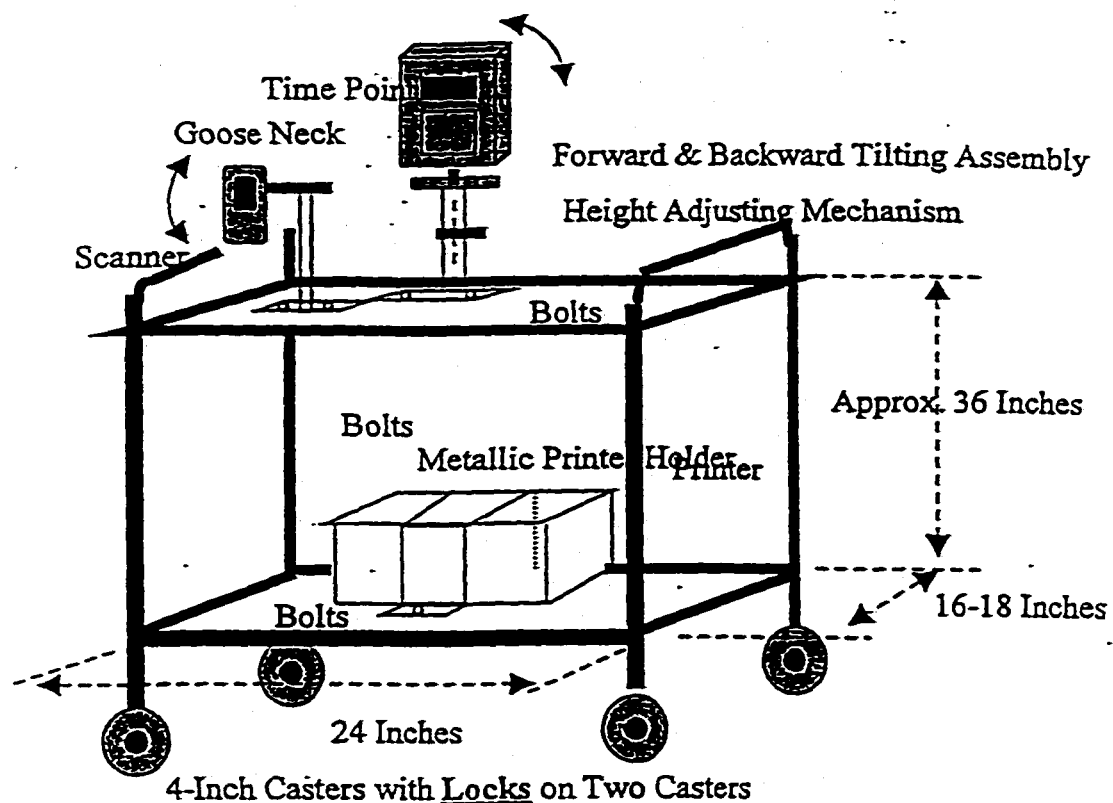


Figure 24: Data Collection Terminal and Printer Mounting Scheme

- Intelligent Instrumentation TimePoint. The TimePoint runs the DOS operating system.
- Intermec 3400 bar code printer. Thermal transfer bar code printer.

The Figure 25 shown below shows the different hardware components and their inter-connection.

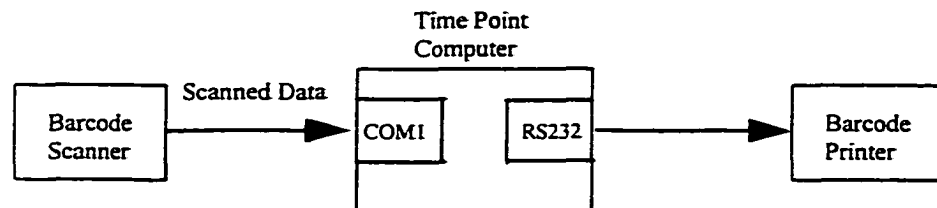


Figure 25: Hardware Configuration for the Packout Scanning System.

The Intermec 3400 printer is connected to RS232 port of the TimePoint. Two types of connectors are required to do this: RJ126 pin to DB25 interface, and RS-232 Null Modem cable for the PC (25-pin socket to 25-pin plug). The Intermec printer is configured as follows:

Baud rate:	9600
Parity:	Even
Stop bits:	1
Data bits:	7
Protocol:	XON/XOFF
XON/XOFF Selection:	No Status Response
DT/TTR:	Thermal Transfer

The dipswitches at the back of the printer are used to configure the printer.

The scanner is connected to COM1 of the TimePoint computer using the 9-pin connector that comes with the scanner. The scanner comes with a default configuration. Following parameters are changed to the new values.

Baud rate:	9600
Parity:	Even
Stop bits:	1
Data bits:	7
Code 39:	Enable, Bar code length between 1 - 20
I 2/5:	Enable, Bar code length between 1 - 20
Data Prefix:	None
Data Suffix:	CR/LF

The scanner can be configured by scanning bar codes. The TimePoint computer can be configured during the system boot up. When the TimePoint is powered up, it displays a message asking the user to press ENTER if they want to make changes in the CMOS setup. The CMOS is configured as follows:

Memory:	A (640K Conv., 256K Ext.)
PCMCIA:	Enabled
Full Screen Display:	CGA
Display Tracking	: Disable
Shift State Tracking:	ON
Cursor:	ON
Cursor Style:	Line

Enter Setup:           Enable  
COM1 Usage:           RS-232  
Boot ROM:             FFRPL  
Boot Priority:          Local

### **Software Overview**

The Packout Scanning System software consists of the following components.

- autoexec.bat.
- packout.exe.
- de\_aiag3.dax               Shipping Label Data File
- de\_aiag3.tax               Shipping Label Format File

The application runs automatically once power is applied to the system. The packout.exe command is included in the autoexec.bat. Two other files that must be present in the root directory along with the packout.exe file are de\_aiag3.tax and de\_aiag3.dax. The de\_aiag3.tax file is a label format file. This file is copied to the printer once every time the system is powered up. Data is substituted into the de\_aiag3.dax file and copied to the printer every time a label has to be printed. The software is developed using the C programming language.

### **Software Architecture**

The software for the data collection terminal was developed using Microsoft MS-DOS, Microsoft C and the Printer Software.

The Intelligent Instrument - Timepoint Computer uses DOS as its operating system, and can run applications developed for the Intel-8086 CPU. The software development was carried out on a DOS machine and the executables were downloaded to PCMCIA card. The TimePoint will run the application from the PCMCIA card.

Bar code label printing was carried out using the Loftware label design and printing software. This software enables users to generate 'Compiled Files'. These files contain fields for variable data, format information, and printer control characters. Bar code labels can be printed by substituting data into the variable data fields and then sending these files to the printer.

The Symbol LS9100 scanner was configured for Code 39 and interleave 2 of 5 formats. The decoded output of the LS 9100 is stored into the keyboard buffer of the Timepoint and is read by application as a keyboard input.

#### Data Design:

Two types of files are maintained by the system. These files are described in the following subsections.

#### Part Duplication File:

The Part identification numbers for the last 100 parts scanned are stored in this file. This information in the file is used for checking for duplicate parts. This file is temporary and is overwritten as new part label information is scanned. No backup for this file is required.

**Label Print Files:** Two types of label print files are maintained by the system. One of the files contains label format information (.DWG), and the other contains variable

data (.DAX). The format file is copied once to the printer. Data is substituted in the data file every time a different shipping label is to be printed and the file copied to the printer. The Software print software generates these files.

### **General Operation**

The different events involved with the operation of this system are given below:

After the system is powered-up and all the hardware modules are working, the system indicates to the user that it is ready to scan the shipping sheet. The shipping sheet contains the Quantity (number of parts that should be packed in the container), Part No, and Pull No. While this screen is displayed, the user also has the option of changing the current date and time, or scanning an Engineering Change, or changing the department number. Any of these functions can be accomplished by pressing the appropriate function key.

After the information is scanned from the shipping sheet, the system displays the scanned information. If the operator notices that scanned data are incorrect, he can clear this data by pressing the Clear key, and then confirming the clear operation. Next the system indicates to the user that it is ready to scan.

The operator then scans each part before putting it in the container. The process continues until the system detects an incorrect or duplicate part, or the quantity in the container equals the quantity indicated on the shipping sheet.

If an incorrect or duplicate part is scanned the system warns the user by displaying a warning message and sounding three beeps. The count will not be

incremented for incorrect or duplicate parts. The user must acknowledge the warning message for the system to continue its operation. The system sounds a beep and displays a message once the required number of parts has been packed in the container. The system automatically prints out two barcode labels for the shipping container. A user can cancel the scanning operation anytime before the quantity is complete by pressing the Clear (F2) key.

The system displays a printer-error message if the printer is not ready or if the printer is not connected properly to the TimePoint computer.

### **User Interface**

This section addresses topics such as screen formats, field entry and validation, and function keys. Each of these items is addressed in the subsections below.

#### **Screen Formats:**

The TimePoint display consists of a 2 x 40 LCD display window. The keyboard is a 23 key Numeric type.

#### **Field Entry & Function Keys:**

The normal operation of the system does not require keypad use except when the user wants to cancel the current scanning operation and start with a new setup sheet. The Clear key (F2) will clear all counters and part number. The Clear key is enabled for all data entry screens. The ENTER key is used as the Acknowledge key.

**Screen Flow:** The Product and Packout Scanning is a standalone application; once power is applied to the TimePoint Computer, it will automatically execute the

application. No login/password is required. The screen flow diagram for the system is shown in Figure 26.

The application does not require any data input from the keypad for normal operation; the application takes the user from one screen to another in response to the scanner input. The first screen to be displayed is the 'SCAN SHIPPING SHEET' screen. The next screen displayed by the system is the 'SHIPPING SHEET DATA ENTRY' screen. This screen is displayed only when all the required setup information (quantity, part no., and pull no.) has been scanned from the shipping sheet. This screen is displayed for 5 seconds.

The next screen displayed by the system is the 'SCAN PART LABEL' screen. In response to this screen, the system expects the user scan a part label.

The process of scanning part labels continues until an invalid or a duplicate part error occurs, or the container is full.

The system displays 'INVALID PART', 'DUPLICATE PART', or 'CONTAINER FULL' screen depending on the condition that is true. The sections below describe the screens in detail.

#### **Scan Shipping Sheet Label**

Layout:

Scan Shipping Sheet		
F4:Date&Time	F6:Dept.	F8:Engg Change



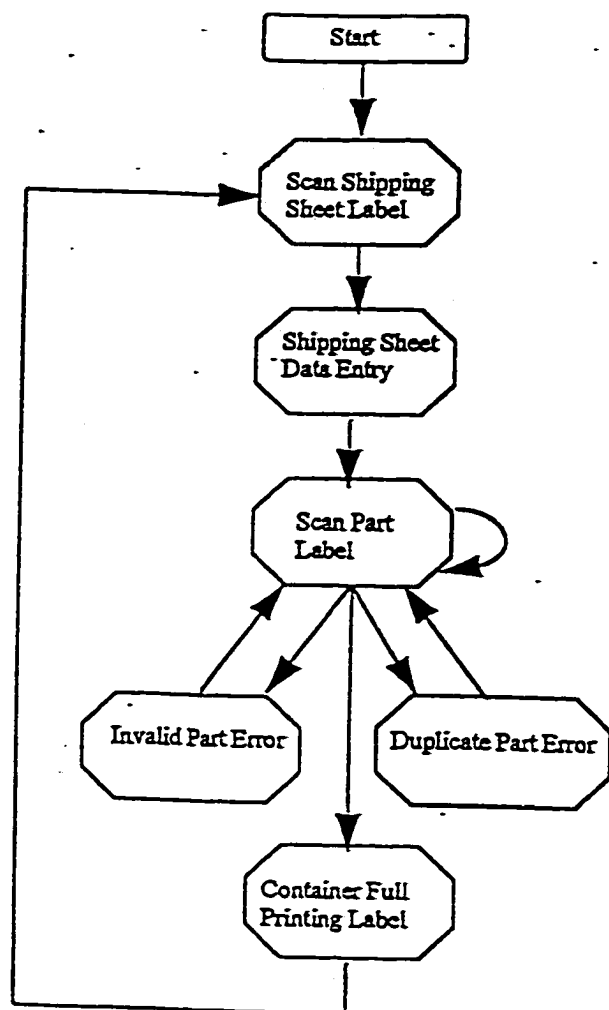


Figure 26: Screen Flow Diagram

**Description:**

This is the first screen that will be displayed by the application. In response to this screen, the application expects the user to scan the quantity, the part identification number, and the pull signal number from the setup sheet. The user can also change the current date and time, or department number by pressing the appropriate function key. The user can also scan an Engineering change. The SHIPPING SHEET DATA ENTRY screen is not displayed until all the setup information has been successfully scanned.

**DATE & TIME:****Layout:**

Current Date: xx/xx/xx	Time: xx:xx:xx
F1:Date	F3:Time F5:Exit

**Description:**

This screen is displayed if the user presses the F4 key during the SCAN SHIPPING SHEET screen. The current date and time are displayed. The user can either press the F1 or the F3 key to change dates or time. The F5 key exits this screen and takes the user back to the previous screen. Note that the time is kept using a 24 hour clock format.

**CHANGE DATE:****Layout:**

Enter New Date: xx/xx/xx (mm/dd/yy)
-------------------------------------

**Description:**

This screen is displayed if the user presses the F1 key during the DATE&TIME screen. A new date is accepted only if it is valid. The screen exits as soon as the date field has been completely filled in or the if the user presses the ENTER key. If the user leaves the date field blank then the date is left unchanged.

**CHANGE TIME:****Layout:**

Enter New Time: xx:xx:xx (hh:mm:ss, 24Hr)
---

**Description:**

This screen is displayed if the user presses the F3 key during the DATE&TIME screen. A new time is accepted only if it is valid. The screen exits as soon as the time field has been completely filled in or the if the user presses the ENTER key. In both cases the application will try to set the new time. Note that a 24-hour clock is used for maintaining time, and the user must enter time using the 24-hour clock format. If the user leaves the time filled blank then the time is left unchanged.

**CHANGE DEPT:****Layout:**

Current Department: xxxx
F3: New Dept F5 Exit

**Description:**

This screen is displayed if the user presses the F6 key during the SCAN SHIPPING SHEET screen. The current (at most 4 digit, numeric) department is displayed in this screen. The user can either exit this screen by pressing the F5 key or change department value by pressing the F3 key.

#### NEW DEPT:

Layout:

Enter New Department:   xxxx
------------------------------

Description:

This screen is displayed if the user presses the F3 key in the NEW DEPT. screen. The user can enter a department number that is at most 4 digits long and is numeric. The screen exits if the user enters all four digits, or if the user presses the enter key at anytime during the screen. The entered data is displayed in the CHANGE DEPT. screen as the new value of the department.

#### ENGG CHANGE:

Layout:

Scan Engineering Change:   xx
-------------------------------

Description:

This screen is displayed if the user presses the F8 key in the SCAN SHIPPING SHEET screen. The user can scan a 2 digit barcode prefixed by "04". The scanned data is displayed for 3 seconds and the TimePoint beeps reflecting a successful scan.

### Shipping Sheet Data Entry

Layout:

Part No.:xxxxxxxx	Quantity:xxx
Pull Signal No.:xxxxxxx	F2:Clear

Description:

This screen displays back to the user the information scanned from the shipping sheet. This screen is displayed for approximately 10 seconds. The user can check to see if correct information has been scanned. If the information is incorrect then the user can rescan the setup sheet by pressing CLEAR key. The system will erase only the setup information and prompt the user to scan the setup sheet again.

Field descriptions for shipping data entry screen is shown in Table 7.

Table 7: Field Descriptions for Shipping Data Entry Screen.

Field Name	Start Position	Length	Data Format	Description
Part No.	1, 10	8	Numeric	Part Identification No. read from the setup sheet
Quantity	1, 33	3	Numeric	The number of parts to be packed in a container
Pull Signal No.	2, 17	6	Alphanumeric	Pull Signal No. read from the setup sheet

CONFIRM1:

Layout:

All Setup Information will be lost Continue Y/N:
---

**Description:**

This screen is displayed if the user presses the Clear key (F2) in the SHIPPING SHEET DATA ENTRY screen. Pressing a 'Y' will clear all setup information and the SCAN SHIPPING SHEET screen is displayed. If the user presses 'N' then the setup information will not be cleared and the process will continue with the setup information scanned.

**Scan Part Label****Layout:**

Scan Part Label	Total:xxx	Scanned:xxx
Part No.:xxxxxxxxxxxxxxxxxxxx	F2:Clear	

**Description:**

Once the setup information has been scanned, and the scanned data displayed back to the user, the system then prompts the user to start scanning the part labels. This screen displays the total number of parts that must be packed in a container, the number of parts that have already been scanned, and the identification number for the part that was last scanned. In case the user wants to terminate the current scanning operation, they can use the Clear key (F2). The clear operation erases all information. This includes part numbers scanned during the current scanning operation. Field descriptions for scan part label screen is shown in Table 8.

Table 8: Field Descriptions for Scan Part Label Screen.

Field Name	Start Position	Length	Data Format	Description
Total	1, 25	3	Numeric	The total number of parts to be packed in a container
Scanned	1, 38	3	Numeric	Number of parts scanned
Part No.	2, 10	18	Numeric	Part Identification number read from the part label

## CONFIRM2:

Layout:

All Shipping Information will be lost Continue: Y/N
--

Description:

This confirmation screen is displayed once the user presses the Clear key. Pressing 'N' will cause the current scanning operation to continue. If the Clear key was pressed in the 'SHIPPING SHEET DATA ENTRY' screen then pressing 'Y' will only clear the setup information. However, if the Clear key is pressed during any other screen then pressing 'Y' will cause all information to be lost, counters will be reset, and the system will return to 'SCAN SHIPPING SHEET' screen.

## INVALID PART:

Layout:

Invalid Part - Press ENTER to Continue Part No.:xxxxxxxxxxxxxxxxxxxxxx
---

**Description:**

If the user scans a part number that does not match the number read from the setup shipping sheet, the system generates this error message. The part number is also displayed. The user presses ENTER to continue.

**DUPLICATE PART:****Layout:**

Duplicate Part - Press ENTER to Continue Part No.:xxxxxxxxxxxxxxxxxxxx
---

**Description:**

This message is displayed if the user scans a duplicate part. Every time a user scans a part label, the system searches for a part with the same serial number from a list of 100 previously scanned parts. If a part with matching serial number is found, then the above error message is displayed. The part number is also displayed. The user presses ENTER to continue.

**CONTAINER FULL:****Layout:**

Container Full - Printing Label
---------------------------------

**Description:**

This message is displayed when the number of parts scanned equals the number indicated on the setup-shipping sheet. This message also indicates to the user that the shipping label for the container is currently being printed. When the printing



is complete, the system resets its counters, and displays the 'SCAN SHIPPING SHEET LABEL' screen.

#### PRINTER ERROR:

Layout:

Printer Error - Press ENTER to Continue
---

#### Description:

This message is displayed when the printer is not ready. This message will be displayed whenever the application tries to write to the printer and cannot find it ready. In response to this screen, the user presses ENTER. Pressing the ENTER key will cause the application to check printer status again. The user may have to press the ENTER key more than once before the communication link is established between the printer and the TimePoint computer.

#### Data Dictionary:

The data dictionary (Table 9) provided below lists all data elements and their description, data types, length, and possible values.

#### Physical Data Design:

Physical data design defines all data tables (files) used by the system. The following files are maintained for this system and stored on the PCMCIA card:

- Part number file
- Pull Signal number file.

Part Number File (Table 10): This file contains the last 100 part numbers scanned. It is updated by the scan part function on the Timepoint computer.

**Pull Signal Number File (Table 11):**

This file contains the last 100 pull signal numbers scanned. It is updated by the scan shipping sheet function on the TimePoint computer.

**Table 9: Data Dictionary**

<b>Element Name</b>	<b>Data Type</b>	<b>Length (Bytes)</b>	<b>Possible Values</b>	<b>Description</b>
Part Number	Integer	17	Any part number	unique identifier for each instrument cluster
Container Quantity	Integer	4	1-32,767	Quantity of assemblies in a shipping container
Date	Character	6	MMDDYY	Date assembled
Pull signal number	Alphanumeric	6	any alphanumeric combination	Pull number that generate the part number
Shipping serial number	Character	9	last 8 numbers of the part number	Serial number on shipping label
Sequence number	Integer	4	0000-6999	Last 4 digits of part no
Supplier number	Integer	5	66031	Dunn & broadcast no
Dept. No	Integer	4	0000-9999	Dept the part was made
Engineering change	Integer	2	00-99	Engineering change associated with the part

Table 10: Part Number File

Element Name	Data Type	Length	Possible Values	Description
Part number	Integer	17	any 17 digit numeric combination	part number used to uniquely identify part

Table 11: Pull Signal Number File

Element Name	Data Type	Length	Possible Values	Description
Pull signal no	Alphanumeric	8	any 8 alphanumeric combination	Pull signal number used to uniquely identify part

## **CHAPTER 6**

### **RESULTS AND DISCUSSION**

This research helped the company improve their financial picture by reducing the expenses associated with mis-labeled parts. This activity enabled the company to reduce their PRRs by 25% and was a cost avoidance of approximately \$600,000 annually. The impact of this research was directly related to the company's customer satisfaction (their #1 corporate goal – to improve the customer satisfaction index) and to reducing their expenses. The research also helped the company to restore customer confidence in their ability to provide a quality product. In addition, the impact on quality was felt by all of their customer operations. Since the implementation of the system, the company has received zero PRRs for mis-labeled parts.

## **CHAPTER 7**

### **SUMMARY, CONCLUSION, AND SUGGESTIONS FOR FUTURE RESEARCH**

The motivation for this research comes from a field study with a Printed Circuit Board Assembly and Instrumentation Cluster manufacturer. The company made substantial investments in setting up elaborate systems for on-line data collection and monitoring of process status. While these modern quality control systems provided a rich database, their application in quality management was rather limited. This is partially due to the lack of appropriate methodology for quality decisions. The objective of this dissertation is to develop an integrative approach to process control and packout verification of products.

A significant amount of attention was placed on first time through quality and reducing the number of parts rejected by the company's customers (measured by the number of Problem Resolution Requests (PRRs)). The company used for this study identified that at their manufacturing facility, over 25 percent of all PRR's were due to mis-labeling of products. These mis-labeling accidents at the company's facility in turn were causing quality accidents at company's customer facilities. The reoccurrence of these errors has resulted in a serious financial penalty from customer facilities (estimated at over \$600,000 per year) and the strong potential for losing business. Mis-labeling errors can have serious impacts on assembly plants productivity and can cause considerable confusion in their material handling systems.

The objective of this study was to learn from, understand, and apply the use of Intelligent Information System to a workcell to control the processes. The underlying

goal of this study was to develop a generic process control methodology for use with various sensor-assisted, computer-controlled automated industrial processes in the effort to understand, control, and improve their performance and also to implement a process to reduce and eventually eliminate labeling error occurrences.

The results of this research imply several directions for future research. These include research and analysis on a model for justification of technology practices and a study relating these topics to the financial characteristics of companies. The direction for future research suggested by this study is to further investigate the relationships between the usage of intelligent manufacturing systems to control processes and their benefits on quality and total financial picture of the company. More extensive research is needed in these areas to establish these relationships and propose any models for the justification of technology.

The main contributions of this study lie in understanding parts manufacturing plant and assembly plant business processes looking from the big picture perspective and coming up with methodology to implement intelligent manufacturing systems and packout verification system to control metrics to improve quality of products and also to eliminate the occurrence of labeling errors. This research describes the design of a high volume circuit board manufacturing cell using a set of PC-based modeling and simulation tools. The simulation was performed using Witness, a visual interactive simulation tool. In addition, this study has led to valuable insights into how manufacturing industries can use some of the recent technologies and integrate their processes to proactively control and avoid quality accidents and reap huge benefits.

## REFERENCES

Atherton Robert, "Detailed Simulation for Semiconductor Manufacturing," In Proceedings of the 1990 Winter Simulation Conference, Osman Balci, Randall P. Sadowski, Richard E. Nance, Eds., IEEE, New Orleans, LA, 659-663.

Black J.T, "Leaning into Industrial Revolution III." Manufacturing Engineering. April 1993.

Brandyberry Alan, Justification Techniques for Computer Integrated Manufacturing Systems and Advanced Manufacturing Technology; An Empirical Study, Dissertation, Southern Illinois University at Carbondale, April 1995.

Bravoco, R. R. and Kasper, G. M., An Overview of how Information Flow and Architectures Support Computer-Integrated Manufacturing, Proceedings of CIMCON '85, Anaheim, CA, April 15-18, 1985.

CASA/SME Board of Advisors, 1992. The CASA/SME Strategic plan for the 1990s. (Society of Manufacturing Engineers: Dearborn, MI, 1992).

Cohen, P.A., Trends in Flexible Manufacturing Systems, Proceedings of CIMCOM '85, Anaheim, CA, April 1985.

Coleman, J.R, Editorial: Where is the Real Cutting Edge or R&D. Manufacturing Engineering January, page 4 (1993).

Daniel, R. W., Hewit, J.R, Editorial, Mechatronics 1(1), I-II (1992).

Dupont-Gatelmand, C., A Survey of Flexible Manufacturing Systems, Journal of Manufacturing Systems, Vol. 1. No. 1, 1982, pp 1-16.

Gausemeier, J., Gehnen, G. "Integrated Network for Decentral Intelligent Manufacturing Control and Automation", Intelligent Manufacturing Systems 1997, A proceedings volume from the IFAC workshop Seoul, Korea, 21-23 July 1997.

Gupta, D. and Buzacott, J. A., A Framework for Understanding Flexibility of Manufacturing Systems, Journal of Manufacturing Systems, Vol. 8. No. 2, 1988, pp. 89-97.

Hatami Steve, "Using Experimental Design to Improve the Efficiency of Simulation Modeling - A Manufacturing Perspective," In Proceedings of the 1990 Winter Simulation Conference, Osman Balci, Randall P. Sadowski, Richard E. Nance, Eds., IEEE, New Orleans, LA, 310-313.

Hirschfeld Robert, "Distributed Control in an Intelligent Manufacturing Cell", A Thesis, August 1993.

Hong Y. Xu, "Intelligent Control and Identification via Expert Systems, Adaptive Techniques, and Neural Networks", A Thesis, Technical University of Nova Scotia, 1992

Hunter, J.S., "The Exponentially Weighted Moving Average", Journal of Quality Technology, vol. 18, 1986.

Joshi, S. and Smith, J; "Intelligent Control of Manufacturing Systems" In Intelligent Design and Manufacturing, John Wiley & Sons, Inc. 1992.

Keys, L.Ken. and Parks, C.M., Mechatronics, Systems, Elements, and Technology: A Perspective. IEEE Transactions on Components, Hybrids, and Manufacturing Technology 14(3), 457-461 (1991).

Keys, L.Ken, Systems Life Cycle Engineering and DF"X". IEEE Transactions on CHMT 13(1), 83-93 (1990).

Keys, L. Ken, Robert A. Hirschfeld, and T. Warren Liao, The Intelligent Manufacturing Systems Initiative at Louisiana State University, IEEE/CHMT International Electronics Manufacturing Technology Symposium, 1993.

Kimenia, J.G. and Gershwin, S.B., An Algorithm for the Computer Control of a Flexible Manufacturing System, IIE Transactions, Vol. 15, No. 4, 1983.

Layden John, "Computerizing The American Factory" Industry Week. August 17, 1992.

Law Averill, "Simulation of Manufacturing Systems" In Proceedings of the 1998 Winter Simulation Conference, D.J. Medeiros, E.F. Watson, J.S. Carson and M.S. Manivannan, Eds, Washington, D.C. 13-16 December 1998.

Meieran Eugene "Intelligent Manufacturing Systems" IEEE/CHMT International Electronics Manufacturing Technology Symposium, 1993

Mitchell, J.D., The Coalition for Intelligent Manufacture. IEMTS '93 Session. Page, E.S., "Cumulative Sum Charts", Technometrics, vol. 3, 1961.

Pardo-Castellote Gerardo, "Experiments in the Integration and Control of an Intelligent Manufacturing Workcell", A Dissertation, Stanford University, 1995.

Pierce, C.M., Chairman, NRC Committee, "Committee Report on Analysis of



Research Directions and Needs in U.S. Manufacturing," National Research Council, National Academy Press, Washington, D.C., pp. 1-172, (1991).

Rawles, Ian., "The Witness Toolbox – A Tutorial" In Proceedings of the 1998 Winter Simulation Conference, D.J. Medeiros, E.F. Watson, J.S. Carson and M.S. Manivannan, Eds, Washington, D.C. 13-16 December 1998.

Shewhart, W. A., Economic Control of the Quality of Manufactured Product, Von Nostrand: New York, 1931.

Smith, Jeffrey Scott, "A formal design and development methodology for shop floor control in computer integrated manufacturing, A dissertation, The Pennsylvania State University, 1992

Smith, J.S., Development of a Hierarchical Control Model for a Flexible Manufacturing System, Master's thesis, The Pennsylvania State University, 1990.

Suri, R. and M. Tomsicek "Modeling Manufacturing Systems using Manuplan and Simstarter – A Tutorial," In Proceedings of the 1990 Winter Simulation Conference, Osman Balci, Randall P. Sadowski, Richard E. Nance, Eds., IEEE, New Orleans, LA, 168-176.

Tayyari, F, Kroll, D. E., "Total Cost Analysis of Modern Automated Systems," Justification Methods for Computer Integrated Manufacturing Systems: Planning, Design Justification, and Costing, H.R. Parsaei, T.L Ward, W. Karwowski, Eds, Elsevier, Amsterdam, 1990

The Competitive Edge: Research Priorities for U.S. Manufacturing. (National Academy Press: Washington DC, 1991). p 25-53.

Underwood Lynn, Intelligent Manufacturing. Addison-Wesley Publishing Company, 1994.

Western Electric Co., Inc., Statistical Quality Control Handbook, Delmar Printing Co, Charlotte, North Carolina, 1956.

Whitney, C.K., Control Principals in Flexible Manufacturing, Journal of Manufacturing Systems, Vol. 4, No. 2, 1985, pp. 157-166.

Wodley, H.N.G., and Eckhart, W.E. Jr., The Intelligent Processing of Materials for Design and Manufacture, Journal of Materials, October 10-16 (1989).

Yarling Susan, "Time series Modeling as an Approach to Automatic Feedback Control of Robotic Positioning Errors" IEEE/CHMT Int'l Electronics Manufacturing Technology Symposium, 1993.

Yolken, H.T., Mordfin, L., eds, Intelligent Processing of Materials—Report of an Industrial Workshop conducted by the NIST. Report No. NISTIR 789-4024, NIST, Gaithersburg, MD, 50 pages (1989).

**APPENDIX**  
**SOURCE CODE**

**scanning sheet format: (scanfrmt.dat)**


---

```

<STX><ESC>P<ETX>
<STX>E5<EXT>
<<STX>R<ETX>
STX>E5;F5,FLTSIG;<ETX>
<STX>B1,PARTNO;o460,50;f3;c0,0;h40;w1;i2;d0,14;p01;<ETX>
<STX>B3,QTY;o360,50;f3;c0,0;h40;w1;i2;d0,6;p02;<ETX>
<STX>H5,SUPPLY;o0,0;f3;c0,h1;w1;d0,11;<ETX>
<STX>H7,SERIAL;o0,0;f3;c0,h1;w1;d0,9;<ETX>
<STX>H9,ADDR;o0,0;f3;c0;d0,47;h1;w1;<ETX>
<STX>B11,SIGNAL;o260,50;f3;c0,0;h40;w1;i2;d1,6;p03;<ETX>
<STX>H20,NAME;o0,0;f3;c0;d1,20;h1;w1;<ETX>
<STX>H22,PLTDOC;o0,0;f3;c0;d1,7;h1;w1;<ETX>
<STX>H24,ROUTE;o560,150;f3;c0;d0,20;h2;w1;<ETX>
<STX>H26,DESCRI;o0,0;f3;c0;d1,20;h1;w1;<ETX>
<STX>H28,RQDATE;o530,0;f3;c0;d1,19;h1;w1;<ETX>
<STX>H30,RQDATD;o530,170;f3;c0;d1,7;h2;w1;<ETX>
<STX>H32,ENGCGL;o0,0;f3;c0;d1,8;h1;w1;<ETX>
<STX>B34,ENGCGD;o160,50;f3;c0,0;i2;d0,2;h40;w1;p04;<ETX>
<STX>H50;o560,0;f3;c0;d3,SUPPLIER NAME;;h1;w1;<ETX>
<STX>H52;o500,0;f3;c0;d3,PART NO. (01);h1;w1;<ETX>
<STX>H54;o400,0;f3;c0;d3,QUANTITY NO (02);h1;w1;<ETX>
<STX>H56;o300,0;f3;c0;d3,PULL SIGNAL (03);h1;w1;<ETX>
<STX>H58;o200,0;f3;c0;d3,ENG CHG (04);h1;w1;<ETX>
<STX>I1;o485,50;f3;c1;h2;w2;<ETX>
<STX>I3;o385,50;f3;c1;h2;w2;<ETX>
<STX>I11;o285,50;f3;c1;h2;w2;<ETX>
<STX>I34;o185,50;f3;c1;h2;w2;<ETX>
<STX>D0;R<ETX>
<STX>R<ETX>
<STX><ESC>E5<CAN>16231172<CR>02<CR><ETX>
<STX>SUPPLY <CR>999999<CR><ETX>
<STX>ADDRESS <CR><ETX>
<STX>999999<CR>NAME <CR><ETX>
<STX>PLTDOC <CR>ROUTE <CR>DESCRIPTION <CR><ETX>
<STX>CUSTOMER REQ'D DATE<CR><ETX>
<STX>REQDATE<CR>ENG CHG:<CR>00<ETX>
<STX><RS>1<US>1<ETB><ETX>

```

---

### Format for printing the 100 parts: (100part.tax)

---

```

<STX><ESC>P;E19;F19<ETX>
<STX>L39;D0;<ETX>
<STX>H0,H0;o0,25;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>D39;<ETX>
<STX>H1,H1;o0,50;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H2,H2;o0,75;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H3,H3;o0,100;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H4,H4;o0,125;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H5,H5;o0,150;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H6,H6;o0,175;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H7,H7;o0,200;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H8,H8;o0,225;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H9,H9;o0,250;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H10,H10;o0,275;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H11,H11;o0,300;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H12,H12;o0,325;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H13,H13;o0,350;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H14,H14;o0,375;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H15,H15;o0,400;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H16,H16;o0,425;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H17,H17;o0,450;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H18,H18;o0,475;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H19,H19;o0,500;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H20,H20;o250,25;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H21,H21;o250,50;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H22,H22;o250,75;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H23,H23;o250,100;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H24,H24;o250,125;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H25,H25;o250,150;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H26,H26;o250,175;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H27,H27;o250,200;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H28,H28;o250,225;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H29,H29;o250,250;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H30,H30;o250,275;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H31,H31;o250,300;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H32,H32;o250,325;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H33,H33;o250,350;f0;r0;c2;h1;w1;b0;d0,30;<ETX>
<STX>H34,H34;o250,375;f0;r0;c2;h1;w1;b0;d0,30;<ETX>

```

<STX>H35,H35;o250,400;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H36,H36;o250,425;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H37,H37;o250,450;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H38,H38;o250,475;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H39,H39;o250,500;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H40,H40;o475,25;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H41,H41;o475,50;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H42,H42;o475,75;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H43,H43;o475,100;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H44,H44;o475,125;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H45,H45;o475,150;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H46,H46;o475,175;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H47,H47;o475,200;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H48,H48;o475,225;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H49,H49;o475,250;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H50,H50;o475,275;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H51,H51;o475,300;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H52,H52;o475,325;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H53,H53;o475,350;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H54,H54;o475,375;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H55,H55;o475,400;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H56,H56;o475,425;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H57,H57;o475,450;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H58,H58;o475,475;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H59,H59;o475,500;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H60,H60;o700,25;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H61,H61;o700,50;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H62,H62;o700,75;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H63,H63;o700,100;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H64,H64;o700,125;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H65,H65;o700,150;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H66,H66;o700,175;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H67,H67;o700,200;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H68,H68;o700,225;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H69,H69;o700,250;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H70,H70;o700,275;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H71,H71;o700,300;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H72,H72;o700,325;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H73,H73;o700,350;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H74,H74;o700,375;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H75,H75;o700,400;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H76,H76;o700,425;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H77,H77;o700,450;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H78,H78;o700,475;f0;r0;c2;h1;w1;b0;d0,30;<ETX>  
 <STX>H79,H79;o700,500;f0;r0;c2;h1;w1;b0;d0,30;<ETX>

<STX>H80,H80;o925,25;f0;r0;c2;h1;w1;b0;d0,30;<ETX>

<STX>H81,H81;o925,50;f0;r0;c2;h1;w1;b0;d0,30;<ETX>

<STX>H82,H82;o925,75;f0;r0;c2;h1;w1;b0;d0,30;<ETX>

<STX>H83,H83;o925,100;f0;r0;c2;h1;w1;b0;d0,30;<ETX>

<STX>H84,H84;o925,125;f0;r0;c2;h1;w1;b0;d0,30;<ETX>

<STX>H85,H85;o925,150;f0;r0;c2;h1;w1;b0;d0,30;<ETX>

<STX>H86,H86;o925,175;f0;r0;c2;h1;w1;b0;d0,30;<ETX>

<STX>H87,H87;o925,200;f0;r0;c2;h1;w1;b0;d0,30;<ETX>

<STX>H88,H88;o925,225;f0;r0;c2;h1;w1;b0;d0,30;<ETX>

<STX>H89,H89;o925,250;f0;r0;c2;h1;w1;b0;d0,30;<ETX>

<STX>H90,H90;o925,275;f0;r0;c2;h1;w1;b0;d0,30;<ETX>

<STX>H91,H91;o925,300;f0;r0;c2;h1;w1;b0;d0,30;<ETX>

<STX>H92,H92;o925,325;f0;r0;c2;h1;w1;b0;d0,30;<ETX>

<STX>H93,H93;o925,350;f0;r0;c2;h1;w1;b0;d0,30;<ETX>

<STX>H94,H94;o925,375;f0;r0;c2;h1;w1;b0;d0,30;<ETX>

<STX>H95,H95;o925,400;f0;r0;c2;h1;w1;b0;d0,30;<ETX>

<STX>H96,H96;o925,425;f0;r0;c2;h1;w1;b0;d0,30;<ETX>

<STX>H97,H97;o925,450;f0;r0;c2;h1;w1;b0;d0,30;<ETX>

<STX>H98,H98;o925,475;f0;r0;c2;h1;w1;b0;d0,30;<ETX>

<STX>H99,H99;o925,500;f0;r0;c2;h1;w1;b0;d0,30;<ETX>

#### **Label format: de\_aiag3.dax**

<STX>R<ETX>

<STX><SI>F20<ETX>

<STX><SI>t0<ETX>

<STX><SI>f0<ETX>

<STX><SI>S20<ETX>

<STX><SI>d-5<ETX>

<STX><SI>c0<ETX>

<STX><SI>g1,420<ETX>

<STX><SI>R0<ETX>

<STX><SI>r000<ETX>

<STX><SI>T1<ETX>

<STX><SI>D0000<ETX>

<STX><SI>W831<ETX>

<STX><ESC>E1,1<CAN><ETX>

<STX><ESC>F"PRTNUM"<LF>P12345678<ETX>

<STX><ESC>F"QUANTITY"<LF>Q123<ETX>

<STX><ESC>F"SUPPL "<LF>V66031<ETX>

<STX><ESC>F"SERIAL"<LF>S123456789<ETX>

<STX><ESC>F"PONUMB"<LF>D123456<ETX>

```

<STX><ESC>F"PRTTXT"<LF>12345678<ETX>

<STX><ESC>F"QTYTXT"<LF>123<ETX>
<STX><ESC>F"POTXT "<LF>123456<ETX>
<STX><ESC>F"SUPTXT"<LF>66031<ETX>
<STX><ESC>F"SERTXT"<LF>123456789<ETX>
<STX><ESC>F"DEPT "<LF>1234<ETX>
<STX><ESC>F"DATE "<LF>123456<ETX>
<STX><ESC>F"ENGR "<LF>12<ETX>
<STX><RS>*QUANT,04<ETX>
<STX><US>2<ETX>
<STX><ETB><ETX>

```

---

**Label format: de\_aiag3.tax**

```

<STX><ESC>P;<ETX>
<STX>E1,1;A1,DE_AIAG3;<ETX>
<STX>L39;D0;<ETX>
<STX>B0,PRTNUM;o64,129;f0;h100;w3;rl;i0;c0,0;d0,9;D39;<ETX>
<STX>B1,QUANTY;o103,344;f0;h100;w4;rl;i0;c0,0;d0,4;<ETX>
<STX>B2,SUPPL ;o98,517;f0;h100;w4;rl;i0;c0,0;d0,6;<ETX>
<STX>B3,SERIAL;o97,674;f0;h100;w3;rl;i0;c0,0;d0,10;<ETX>
<STX>B4,PONUMB;o667,336;f0;h100;w3;rl;i0;c0,0;d0,7;<ETX>
<STX>H5,TXT002;o7,6;f0;h1;w1;c2;b0;d3,PART NO.;<ETX>
<STX>H6,TXT003;o7,248;f0;h1;w1;c2;b0;d3,QUANTITY;<ETX>
<STX>H7,TXT009;o0,469;f0;h1;w1;c2;b0;d3,SUPPLIER;<ETX>
<STX>H8,TXT010;o0,633;f0;h1;w1;c2;b0;d3,SERIAL;<ETX>
<STX>H9,TXT013;o621,249;f0;h1;w1;c2;b0;d3,PULL NO.;<ETX>
<STX>H10,TXT015;o4,24;f0;h1;w1;c2;b0;d3,(P);<ETX>
<STX>H11,TXT016;o6,266;f0;h1;w1;c2;b0;d3,(Q);<ETX>
<STX>H12,TXT017;o10,488;f0;h1;w1;c2;b0;d3,(V);<ETX>
<STX>H13,TXT018;o8,652;f0;h1;w1;c2;b0;d3,(S);<ETX>
<STX>H14,TXT019;o628,272;f0;h1;w1;c2;b0;d3,(D);<ETX>
<STX>H15,TXT020;o8,782;f0;h1;w1;c2;b0;d3,COMPANY NAME;<ETX>
<STX>H16,PRTTXT;o54,36;f0;h6;w6;c2;b0;d0,8;<ETX>
<STX>H17,QTYTXT;o90,264;f0;h5;w6;c2;b0;d0,3;<ETX>
<STX>H18,POTXT ;o678,288;f0;h2;w3;c2;b0;d0,6;<ETX>
<STX>H19,SUPTXT;o100,480;f0;h2;w3;c2;b0;d0,5;<ETX>
<STX>H20,SERTXT;o88,638;f0;h2;w3;c2;b0;d0,9;<ETX>
<STX>H21,TXT026;o726,474;f0;h2;w2;c2;b0;d3,COMPANY NAME;<ETX>
<STX>H22,TXT029;o716,642;f0;h1;w1;c2;b0;d3,DEPT. NO.;<ETX>
<STX>H23,TXT030;o716,716;f0;h1;w1;c2;b0;d3,DATE MANF.;<ETX>
<STX>H24,TXT031;o956,634;f0;h2;w1;c2;b0;d3,ENG. CHANGE;<ETX>
<STX>H25,DEPT ;o716,670;f0;h2;w2;c2;b0;d0,4;<ETX>

```



```

<STX>H26,DATE ;o716,742;f0;h2;w2;c2;b0;d0,6;<ETX>

<STX>H27,ENGR ;o954,676;f0;h2;w2;c2;b0;d0,2;<ETX>
<STX>L28;o5,237;f0;l1188;w3;<ETX>
<STX>L29;o608,238;f3;l223;w3;<ETX>
<STX>L30;o5,458;f0;l1188;w3;<ETX>
<STX>L31;o3,625;f0;l1188;w3;<ETX>
<STX>L32;o711,460;f3;l327;w3;<ETX>
<STX>L33;o950,624;f3;l154;w2;<ETX>
<STX>L34;o708,702;f0;l240;w2;<ETX>
<STX>R<ETX>

```

---

### **Packout.h**

```

void Display( char *dispbuf);
void SetCursor( int row, int col );
void Sleep( clock_t wait );
void Beep( int frequency, int duration );
void GetEnterKey( void );
void WriteToCom2( char *outbuf);
void ClearPort( int port );
int InitLabelData( void );
int InputScannerData( int port, char *scanbuf);
int GetFkey( void );
int GetF2( void );
int GetYesNo( void );
void GetStatus(char status[]);
int GetFkeyOrEnter( void );
void SupervisorScreenRoutine( int fromMain );

/* The next three "FLAGS" Need to be defined if you want this program to compile
correctly
    - RAP
*/
#define PACKSDEBUG    0
#define TP            1 /* TP compiles for TimePoint */
#define PC            0 /* PC compiles for Personal Computer */
#define SKIPINIT    0 /* SKIPINIT skips the printer initialization so that the
program
                        can be tested for scanning, etc.*/

/*
    Generic application of filenames on the PCMIA card.

```

```

*/

#if PC
#define CARDREADER_PATH      "d:\\\"
#else
#define CARDREADER_PATH      "a:\\\"
#endif

#define LABELFOR100PARTS_TAX  "100part.tax"
#define DE_AIAG3_TAX          "de_aiag3.tax"
#define DE_AIAG3_DAX          "de_aiag3.dax"

#define SCANSHIPPINGSHEET    1
#define SETUPDATAENTRY       2
#define SCANPARTLABEL        3
#define CONFIRM               4
#define INVALIDPART           5
#define DUPLICATEPART         6
#define CONTAINERFULL        7
#define DUPLICATEPULLSIG      8
#define BADSEQUENCENUMBER     9
#define SUPERVISORSCREEN      10
#define PRINT100PARTS         11
#define SHIPPINGINFOSCRN      12
#define DISPHIGHESTSEQNUM     13
#define PRINTERERROR          99

#define PULLSIG_LEN           8
#define PART_LEN              20
#define PART_QTY_LEN          4
#define SEQ_NUM_LEN           4

#define MAXLENBUF              5220
#define DAXFILESIZE  800      /* Size of buffer into which the DAX file is read in.
>= actual DAX file size. */
#define TAXFILESIZE  2000     /* Size of buffer into which the DAX & TAX files
are read in for processing purposes. > actual TAX file size. */
#define TAX100FILESIZE 6000

#define F1_KEY      1
#define F3_KEY      3
#define F5_KEY      5
#define F7_KEY      7

#define F2_KEY      2

```

```

#define F4_KEY      4
#define F6_KEY      6
#define F8_KEY      8
#define ENTER_KEY   9

#define PARTOK      1
#define QTYOK       1
#define PULLOK      1
#define DLCOK       1
#define ENGROK      1
#define PARTNOREAD  1

#define IS_YES      0
#define IS_NO       1
#define INVALID_KEY  2
#define IS_F2       3

#define TRUE        1
#define FALSE       0

#define COMM1 0x3F8
#define COMM2 0x2F8

#define PART_INVALID  1
#define PART_DUPLICATE 2
#define PART_OK       0

#define PULLSIG_OK      0
#define PULLSIG_DUPLICATE 1
#define PULLSIG_LEN     8

#define SYSTEMERROR    0

/* Offsets to variables in the list2 buffer. These offsets are found by counting all the
characters that come before the point where substitution starts. */
#define PARTOFFSET  230
#define QTYOFFSET   267

#define PULLOFFSET  371
#define SERIALOFFSET 333
#define DATEOFFSET  608
#define R_PARTOFFSET 405
#define R_QTYOFFSET  441
#define R_PULLOFFSET 472

```

```
#define R_SERIALOFFSET 539
#define LIST2_DEPTOFFSET 576
#define LIST2_ENGROFFSET 642
```

```
/* Offsets to the file on disk, opened in binary mode. These variables are substituted
directly into the DAX file on disk. */
```

```
#define ENGROFFSET 689 // Was 729
#define DEPTOFFSET 620 // Was 659
```

### PACUTILS.OBJ:

```

/*****
***
*      FILE:          PACUTILS.C                      *
*                                                           *
*      FUNCTIONS: Sleep                                *
*          Beep                                           *
*          Display                                         *
*          SetCursor                                       *
*          WriteToComPort                                   *
*          ClearPort                                       *
*              OpenComPort                                 *
*          InputScannerData                               *
*          GetStatus                                       *
*          GetYesNo                                         *
*          GetFkeyOrEnter                                   *
*                                                           *
*                                                           *
*****
*****/

/*****
*****
* FUNCTION:          Sleep                                *
*                                                           *
* DESCRIPTION:      *
*          This function is used to introduce a wait for the duration of *
*          time specified as the argument.                        *
*                                                           *
* INPUT PARAMETERS: *
*          Wait - The clock function used in this routine returns the *

```

```

*          product of time in seconds and the constant CLOCKS_PER_SEC *
*          for the calling process. The value of CLOCKS_PER_SEC is equal *
*          to 1000. If the user wants a wait of 2 seconds, then the      *
*          input argument (wait) should be set to 2*CLOCKS_PER_SEC =2000*

* GLOBALS USED:
*          None
*
* EXTERNAL FUNCTION:
*          clock() - C Runtime function
*
* RETURN VALUE
*          None
*****
/*****
*****
* FUNCTION:          Beep
*
* DESCRIPTION:
*          This function is used to control the internal beeper of the *
*          TimePoint computer
*
* INPUT PARAMETERS:
*          frequency - pitch of beep, values between 0, 1000
*
*          duration - duration of beep as product of Time in seconds and *
*          CLOCKS_PER_SEC (1000).
*
* GLOBALS USED:
*          None
*
* EXTERNAL FUNCTION:
*          _outp() - C Runtime function
*
*          inp() - C Runtime function
*
* RETURN VALUE:
*          None
*****
/*****
***
* FUNCTION:          Display

```

```

*
*
* DESCRIPTION:
*
* This function display a buffer either on the PC screen or the
* TimePoint. Interrupt21 service 9 is used to display buffer on
* the PC. Interrupt17 service 12 is used for display on
* TimePoint. Notice that TimePoint BIOS is not the same as PC.
*
* Special processing must be done.
*
* INPUT PARAMETERS:
*
* dispbuf - 80 character long buffer to be displayed.
*
* GLOBALS USED:
*
* None
*
* EXTERNAL FUNCTION:
*
* int86() - C Runtime function
*
* RETURN VALUE:
*
* None
*****
/*****
/*****
* FUNCTION: SetCursor
*
* DESCRIPTION:
*
* This function places the cursor on the specified row, col on the
* PC or TimePoint screen
*
* INPUT PARAMETERS:
*
* row - row number where cursor must be displayed
*
* col - column number where cursor must be displayed
*
* Notice that due to differences in BIOS special processing must
* be done for TimePoint. Use interrupt 10, service 2 for PC. Use
* interrupt 17, service 17 for TimePoint.
*
* GLOBALS USED:
*
* None
*
* EXTERNAL FUNCTION:
*
* int86() - C Runtime function

```

```

*
* RETURN VALUE:
* None
*****
***

/*****
***
* FUNCTION: WriteToComPort
*
* DESCRIPTION:
* This routine writes data to the SMARTPORT of the TimePoint.
* The SMARTPORT is controlled through the I/O Coprocessor.
* The communication between I/O Coprocessor and main board is
* accomplished through COM2 of the main board. Since the I/O
* Coprocessor is being used in the pass-through mode,
* reading/writing SMARTPORT is similar to reading/writing COM2.
* This routine has delays embedded in it. These delays are
* necessary so that neither the I/O coprocessor or the Printer
* input buffer is overrun with data.
*
* INPUT PARAMETERS:
* outbuf - input string to be written to COM2
*
* port - 1, 2 for COM1 or COM2 respectively
*
* delay1 - delay after every byte of data
* delay2 - delay after every PRINTER_BUF_LEN bytes of data
* if do not want delays then set a high value such as 10000
*
* GLOBALS USED:
* None
*
* EXTERNAL FUNCTION:
* None
*
* RETURN VALUE:
* None
*****
***

/*****
***
* FUNCTION: ClearPort
*

```

```

* DESCRIPTION:
*      This function reads a single character from the COM port passed *
*      to this function. Purpose is to clear the port. It was      *
*      observed on the PC that if a device sends data and it is not *
*      read within a given time then characters data overruns, and the *
*      last character received is the one that is still in the input *
*      buffer. The purpose of this routine is to read that last *
*      character.
*
* INPUT PARAMETERS:
*      port - 1,2 for COM1 or COM2 respectively
*
* GLOBALS USED:
*      None
* EXTERNAL FUNCTION:
*      None
* RETURN VALUE:
*      None
*****
***

/*****
***
* FUNCTION:                                OpenComPort
*
* DESCRIPTION:
*      This function configures a COM port.
*
* INPUT PARAMETERS:
*      port - 1,2 for COM1 or COM2 respectively
*      port_config - configuration for port
*      Bit meanings for port_config
*      bit 7 - 0, Normal access to ports 3F8h
*      1, Use ports 3F8h/3F9h to specify baud rate divisor
*
*      bit 6 - 1, Transmit break condition
*
*      bit 5 - 0, Parity held at value in bit 4
*      1, Parity operates normally
*
*      bit 4 - 0, Odd parity
*      1, Even parity
*
*      bit 3 - 0, Parity disabled

```



```

*          1, Parity enabled                                     *
*                                                                 *
*          bit 2 - 0, 1 Stop bit                                 *
*          1, 2 Stop bits (1.5 if bits 0-1 are clear)          *
*                                                                 *
*          bits 1&0 - 00, 5-bit data length                     *
*          01, 6-bit data length                                *
*          10, 7-bit data length                                *
*          11, 8-bit data length                                *
*                                                                 *
* GLOBALS USED:                                                 *
*          None                                                 *
*                                                                 *
* EXTERNAL FUNCTION:                                           *
*          None                                                 *
*                                                                 *
* RETURN VALUE:                                                *
*          None                                                 *
*                                                                 *
*****
***

/*****
***
* FUNCTION:          InputScannerData                           *
*                                                                 *
* DESCRIPTION:       *
*          This routine inputs scanner data. It also checks for F1 through *
*          F8 keys. If a function key is pressed the function exits and *
*          an appropriate code is returned.                             *
*                                                                 *
* INPUT PARAMETERS:  *
*          port - Port to which scanner is connected, 0,1 for COM1 or COM2 *
*          respectively                                     *
*          scanbuf - Buffer containing data on return         *
*                                                                 *
* GLOBALS USED:      *
*          None                                              *
*                                                                 *
* EXTERNAL FUNCTION: *
*          None                                              *
*                                                                 *

```

```

* RETURN VALUE:
*           0 - if scanner data received
*           function key code - if function key pressed
*****
***

/*****
***
* FUNCTION:           GetStatus
*
* DESCRIPTION:
*           This routine checks the printer status by writing "<ENQ>"
*           command to the printer. If the printer is ready then it returns
*           "<DC2>".
*
* INPUT PARAMETERS:
*
*           status - status of printer, if printer ready then the string
*           "<DC2>" is returned.
*
*           port - COM port to get status from, 1, 2 for COM1 or COM2
*
*
* GLOBALS USED:
*           None
*
* EXTERNAL FUNCTION:
*           None
*
* RETURN VALUE:
*           None
*****
***

/*****
***
* FUNCTION:           GetYesNo
*
* DESCRIPTION:
*           This function waits for YES (1) or NO (3) key on the
TimePoint. *
*           The TimePoint does not have <Y> or <N> keys; instead <1> and
*           <3> are used.
*
*

```

```

* INPUT PARAMETERS:                                     *
*           None                                         *
*                                                         *
* GLOBALS USED:                                         *
*           None                                         *
*                                                         *
* EXTERNAL FUNCTION:                                    *
*   getch() - C Runtime function                        *
*                                                         *
* RETURN VALUE:                                         *
*   IS_YES - if <1> is pressed                          *
*   IS_NO - if <3> is pressed                          *
*   INVALID_KEY - else                                  *
*****
***

/*****
***
* FUNCTION:           GetFkeyOrEnter                    *
*                                                         *
* DESCRIPTION:        *
*   This function waits for a function or ENTER key. If another key *
*   is pressed then invalid key code is returned          *
*                                                         *
* INPUT PARAMETERS:   *
*           None      *
*                                                         *
* GLOBALS USED:       *
*                                                         *
* EXTERNAL FUNCTION:  *
*   getch() - C Runtime function                        *
*                                                         *
* RETURN VALUE:       *
*   F1 - F8: for function keys between F1 and F8        *
*   ENTER_KEY - for <ENTER>                             *
*****
***

```

---

### install.bat:

```

echo off
cls

```

```

if "%1" == "" goto NEEDDRIVE
if %1 == c: goto COPYFILES
if %1 == C: goto COPYFILES
if %1 == d: goto COPYFILES
if %1 == D: goto COPYFILES
if %1 == e: goto COPYFILES
if %1 == E: goto COPYFILES
if %1 == f: goto COPYFILES
if %1 == F: goto COPYFILES
if %1 == g: goto COPYFILES
if %1 == G: goto COPYFILES
if %1 == h: goto COPYFILES
if %1 == H: goto COPYFILES
if %1 == i: goto COPYFILES
if %1 == I: goto COPYFILES

```

#### :INVALID

```

echo _ "%1" is an invalid drive name. Please enter your drive's
echo letter followed by a colon":"
echo For example:
goto Example

```

#### :NEEDDRIVE

```

echo _ Please specify your drive to install, for example:
goto Example

```

#### :EXAMPLE

```

echo _
echo _ install c:
goto End

```

#### :COPYFILES

```

if not exist a:\packout.exe goto packout
if not exist a:\de_aiag3.dax goto dax
if not exist a:\de_aiag3.tax goto tax
if not exist a:\autoexec.bat goto autoex

```

```

echo Copying files to %1...
copy a:\packout.exe %1\>nul
copy a:\de_aiag3.dax %1\>nul
copy a:\de_aiag3.tax %1\>nul
copy a:\autoexec.bat %1\>nul

```

```

if not exist %1\packout.exe goto InstallError
if not exist %1\de_aiag3.dax goto InstallError

```



```

#include <dos.h>
#include <stdlib.h>
#include <conio.h>
#include <ctype.h>
#include <time.h>
#include <conio.h>
#include <memory.h>
#include <packout.h>

// GLOBAL variables
char lastsetupbuf[6][20];
char list2[MAXLENBUF]; /* Global buffer containing the label data */

char partbuf100[100][20]; /* Global part buffer array */
char highest_sequence[10][5];
char lastsetupbuf[6][20];
char setupbuf[6][20];

/* This routine contains most of the messages that are displayed by the program.
   Use the appropriate identifier for a screen and simply send the identifier to this
   routine.
   All the parameters passes to this routine may not be actually used. */

void DispMessage( int screenid, char partpullsig[], char partpartno[],
                  char partqty[], char parttotal[], char partscanned[] )
{
    char dispbuf1[81] = "Scan Shipping Sheet   F3:Supvr Scrn  F4:Date&Time
F6:Dept. F8:Eng Change ";
    char dispbuf2[81] = "Part No.    Quantity    Pull No.    F2:Clear";
    // char dispbuf3[81] = "Scan Part Label  Total   Scanned   Part No.
F2:Clear ";
    char dispbuf3[81] = "Scan Part Label  Total   Scanned   Part: F3:Spvr F2:Clr ";
    char dispbuf4[81] = "All Shipping Information will be lost  Continue Y/N
";
    char dispbuf5[81] = "Invalid Part-Press ENTER to Continue  Part No.
";
    char dispbuf6[81] = "Duplicate Part-Press ENTER to Continue  Part No.
";
    char dispbuf7[81] = "Container Full - Printing Label
";
    char dispbuf8[81] = "Duplicate Pull Signal - Press <ENTER>  Pullsig:
";
    char dispbuf9[81] = "Bad Sequence Number  - Press <ENTER>  Part No.
";

```

```

char dispbuf10[81] = "F1:Print Parts Scanned F2:Exit      F3:View High Seqs
F4:View Ship Sheet ";
char dispbuf11[81] = "Do You Wish to Print the Last 100    Parts Scanned? Y/N
F2:Exit ";
char dispbuf12[81] = "PartNum:XXXXXXXXXXXXXXXXXXXXXXX Qty:XXXXXX
Pull Sig#:XXXXXXXXX      F2:Exit ";
char dispbuf13[81] = "XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX ";
//
123456789012345678901234567890123456789012345678901234567890123456789
01234567890
//
           1      2      3      4      5      6      7      8
int ii = 0, jj = 0, kk = 0, nn = 0;

switch(screenid)
{
    case SCANSHIPPINGSHHEET:
#if PC
        printf("\n%s\n",dispbuf1);
#else
        Display(dispbuf1);
#endif
        break;

    case SETUPDATAENTRY:
        ii = strlen(setupbuf[0]);
        jj = strlen(setupbuf[1]);
        kk = strlen(setupbuf[2]);

        strncpy(&dispbuf2[9], setupbuf[0], ii);
        strncpy(&dispbuf2[27], setupbuf[1], jj);
        strncpy(&dispbuf2[49], setupbuf[2], kk);

#if PC
        printf("\n%s\n",dispbuf2);
#else
        Display(dispbuf2);
#endif
        break;

    case SCANPARTLABEL:
        ii = strlen(parttotal);
        jj = strlen(partscanned);
        kk = strlen(partpartno);
        strncpy(&dispbuf3[24], parttotal, ii);

```

```

        strncpy(&dispbuf3[36], partscanned, jj);
//      strncpy(&dispbuf3[49], partpartno, kk);
        strncpy(&dispbuf3[45], partpartno, kk);
    #if PC
        printf("\n%s\n", dispbuf3);
    #else
        Display(dispbuf3);
    #endif
        break;

    case CONFIRM:
    #if PC
        printf("\n%s\n", dispbuf4);
    #else
        Display(dispbuf4);
    #endif
        break;

    case INVALIDPART:
        ii = strlen(partpartno);
        strncpy(&dispbuf5[49], partpartno, ii);

    #if PC
        printf("\n%s\n", dispbuf5);
    #else
        Display(dispbuf5);
    #endif
        break;

    case DUPLICATEPART:
        ii = strlen(partpartno);
        strncpy(&dispbuf6[49], partpartno, ii);

    #if PC
        printf("\n%s\n", dispbuf6);
    #else
        Display(dispbuf6);
    #endif
        break;

    case DUPLICATEPULLSIG:
        ii = strlen(partpullsig);
        strncpy(&dispbuf8[49], partpullsig, ii);

    #if PC
        printf("\n%s\n", dispbuf8);
    #else

```



```

        Display(disdbuf8);
    #endif
        break;

    case BADSEQUENCENUMBER:
        ii = strlen(partpartno);
        strncpy(&disdbuf9[49], partpartno, ii);
    #if PC
        printf("\n%s\n", disdbuf9);
    #else
        Display(disdbuf9);
    #endif
        break;

    case SUPERVISORSCREEN:
    #if PC
        printf("\n%s\n", disdbuf10);
    #else
        Display(disdbuf10);
    #endif
        break;

    case PRINT100PARTS:
    #if PC
        printf("\n%s\n", disdbuf11);
    #else
        Display(disdbuf11);
    #endif
        break;

    case SHIPPINGINFOSCRN:
    #if PC
        printf("\n%s\n", disdbuf12);
    #else
        Display(disdbuf12);
    #endif
        break;

    case DISPHIGHESTSEQNUM:
    #if PC
        printf("\n%s\n", disdbuf13);
    #else
        Display(disdbuf13);
    #endif
        break;

```

```

        case CONTAINERFULL:
        #if PC
                printf("\n%s\n", dispbuf7);
        #else
                Display(dispbuf7);
        #endif
                break;

        default:
                break;
    }
}

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

/*  Print100Parts:
*/
/*      This routine allows the Supervisor to print the "last" 100 part numbers read.
*/
/*      Assumptions:  1) This function will not be used very often.
*/
/*
/*                      2) The buffer with the one hundred parts is intialized to
something */
/*                      NULLS are okay - Zero filling is done at output time.
*/
/*  Memory conservation is in effect. So the form for the printer will be read from */
/*  the PCMCIA card everytime. The Internec Printer is not trusted to faithfully */
/*  store the form, so it needs to be downloaded everytime.
*/
/*****
*****/
void Print100Parts(int number_to_print )
{
    char list1[MAXLENBUF];
    char labelfor100parts[MAXLENBUF];
    char scrap_buffer[60];
    char fname[50];
    char status[20];
    char missbuf[81];
    char dispbuf8[81] = "Printing last 100 parts scanned. Takes approx. 2 minutes to
complete ";

```

```

int ii, jj, numread;

FILE *stream;

memset( fname, '\0', sizeof( fname ) );
strcpy( fname, CARDREADER_PATH );
strcat( fname, LABELFOR100PARTS_TAX );

memset( labelfor100parts, '\0', sizeof( labelfor100parts ) );

#if PC
    printf( "\n%s\n", dispbuf8 );
#else
    Display( dispbuf8 );
#endif

if( ( stream = fopen( fname, "r+t" ) ) != NULL )
{
    numread = fread( list1, sizeof( char ), TAX100FILESIZE, stream );

    // Strip out all linefeeds
    for( ii = 0, jj = 0; ii < numread; ii++ )
    {
        if ( list1[ii] != '\n' )
            labelfor100parts[jj++] = list1[ii];
    }
    fclose( stream );
}
else
{
    memset( missbuf, ' ', sizeof( missbuf ) - 1 );
    missbuf[ sizeof( missbuf ) - 1 ] = '\0';
    memcpy( missbuf, "Was not able to open ", 21 );
    memcpy( &missbuf[21], fname, strlen( fname ) );

#if PC
    printf( "\n%s\n", missbuf );
#else
    Display( missbuf );
#endif
    // SHOULD probably return or set an error value here!
}

// Erase AIAG label format to free up printer RAM

```

```

WriteToCom2( "<STX><ESC>P;<ETX>" ); // Program Mode
WriteToCom2( "<STX>E1;<ETX>" ); // Erase label format 1

// Send label format (from .TAX file) to printer
WriteToCom2( labelfor100parts );
do
{
    GetStatus( status );
} while( strcmp( status, "<DC2>" ) != 0 );

//
// This is the control string for printing the data to the label printer
//
memset( list1, '\0', sizeof( list1 ) );
strcpy( list1, "<STX>R<ETX>" ); // Place printer in print mode
strcat( list1, "<STX><ESC>G0<ETX><STX><ESC>E19<ETX><STX><CAN>
<ETX>" ); // Use and clear form 19

//
for( ii=number_to_print - 1, jj = 0; ii>=0; ii--, jj++)
{
    sprintf( scrap_buffer, "<STX><ESC>F%03d<DEL>%018s<ETX>", jj,
partbuf100[ii] );
    strcat( list1, scrap_buffer );
}
//
// This is the "tail" of the instructions - basically says print what is in the
// printer's buffer.
//
strcat( list1, "<STX><RS>1<US>1<ETB><ETX>" );
//
// Send the buffered data to the printer
//
WriteToCom2( list1 );
do
{
    GetStatus( status );
} while( strcmp( status, "<DC2>" ) != 0 );

// Call InitLabelData(). This function will erase the 100 part label format, 19,
// and will send the AIAG label format, 1, to the printer
InitLabelData();
}
/*****/

```

```

/* Initialize label data. Copy the label format file to the printer. This routine also uses
*/
/* the Display routine to display messages. Also read in the label data file to a buffer.
*/ /* This buffer is later substituted with data and sent to printer. */
/*****

```

```

int InitLabelData( void )
{
    char dispbuf9[81] = "Initializing Printer .....Wait";
    char missbuf1[81] = "Could not open de_aiag3.tax";
    char missbuf2[81] = "Could not open de_aiag3.dax";
    //
    123456789012345678901234567890123456789012345678901234567890123456789
    01234567890
    //          1      2      3      4      5      6      7      8
    char fname[50];

    FILE *stream;
    int ii, jj, numread, stat;

    char list1[MAXLENBUF];
    char status[20];

    memset( fname, '\0', sizeof( fname ) );
    strcat( fname, CARDREADER_PATH );
    strcat( fname, DE_AIAG3_TAX );

    memset( list1, '\0', sizeof( list1 ) );
    memset( list2, '\0', sizeof( list2 ) );

    stat = 1;

    if((stream = fopen( fname, "r+t" )) != NULL )
    {
        numread = fread( list1, sizeof( char ), TAXFILESIZE, stream );

        for( ii=0, jj=0; ii < numread; ii++ )
        {
            if( list1[ii] != '\n')
                list2[jj++] = list1[ii];
        }
        fclose( stream );
    }
}

```

```

    //ifndef SKIPINIT /* Want to skip printer initialization for PC test programming
    purposes */
    #if !SKIPINIT
        WriteToCom2("<STX><DLE><ETX>"); /* Clear data in printer buffers */

        // Erase format 19 (100 part label). This frees up needed printer RAM
        WriteToCom2( "<STX><ESC>P;<ETX>" ); // Program Mode
        WriteToCom2( "<STX>E19;<ETX>" ); // Erase label format 19
        do
        {
            GetStatus( status );
        } while( strcmp( status, "<DC2>" ) != 0 );

        Display( dispbuf9 );
        WriteToCom2( list2 ); /* Copy label format to Printer */
    #endif

    memset( fname, '\0', sizeof( fname ) );
    strcat( fname, CARDREADER_PATH );
    strcat( fname, DE_AIAG3_DAX );
    memset( list1, '\0', sizeof( list1 ) );
    memset( list2, '\0', sizeof( list2 ) );

    if((stream = fopen( fname, "r+t" )) != NULL )
    {
        numread = fread( list1, sizeof( char ), DAXFILESIZE, stream );

        for( ii=0, jj=0; ii < numread; ii++ )
        {
            if ( list1[ii] != '\n')
                list2[jj++] = list1[ii];
        }
        fclose( stream );
    }
    else
    {
        Display( missbuf2 );
        stat = 0;
    }

}
else
{
    printf( "Was not able to open de_aiag3.tax\n" );
    stat = 0;
}

```

```

    return( stat );
}

/*****
/*
/* This routine substitutes data into the label data (DAX) */
/* file. The offsets are fixed and predetermined.          */
/*
*****/
void PrintLabel(char partno[], char qty[], char pullsig[], char serial[],
                char dept[], char engr[])
{
    char dispbuf8[81] = "Printer Error - Press ENTER to Retry
";
    char dispbuf7[81] = "Container Full - Printing Label
";
    char status[20], tmpbuf[12], date1[10];

    /* In order that this substitution program works correctly, the length of the place
holder
    in the DAX file should only be as long as the variable actually is.
    extra characters from the DAX file should be deleted. */

    memcpy(&list2[PARTOFFSET], partno, 8);

    memset( &list2[QTYOFFSET], ' ', 3); /* copy spaces to qty field */
    memcpy( &list2[QTYOFFSET], qty, strlen( qty ) );

    memcpy( &list2[PULLOFFSET], pullsig, 6);
    memcpy( &list2[SERIALOFFSET], serial, 9);
    memcpy( &list2[R_PARTOFFSET], partno, 8);

    memset( &list2[R_QTYOFFSET], ' ', 3); /* copy spaces to qty-text field */
    memcpy( &list2[R_QTYOFFSET], qty, strlen( qty ) );

    memcpy( &list2[R_PULLOFFSET], pullsig, 6);
    memcpy( &list2[R_SERIALOFFSET], serial, 9);

    _strdate( tmpbuf ); /* Get current date */
    date1[0] = tmpbuf[0];
    date1[1] = tmpbuf[1];
    date1[2] = tmpbuf[3];
    date1[3] = tmpbuf[4];

```

```

    date1[4] = tmpbuf[6];
    date1[5] = tmpbuf[7]; /* Get rid of '/' in date */
    date1[6] = '\0';
    strncpy( &list2[DATEOFFSET], date1, 6); /* copy date */

    WriteToCom2("<STX><DLE><ETX>");
    do
    {
        GetStatus( status );
    } while( strcmp( status, "<DC2>" ) != 0 );

    Display( dispbuf7 ); /* Display Initializing data message again */
    WriteToCom2( list2 ); /* Print label to Printer */
}

/*****
/*
/* Configure system.
/*
*****/
void ConfigSystem( void )
{
    union _REGS inregs, outregs;

    char dispbuf9[81] = "Initializing Printer .....Wait";
    char status[20];
    /* Hardware configuration:
        1. Scanner connected to COM1 for both PC and TP
        2. Printer connected to COM2 for PC and RS232 (SMARTPORT) for
TP.
    */
    /* Default Setup for the I/O Coprocessor host port is 9600,8,1 Stop Bit,No Parity.
The COM2 parameters are
    set automatically at startup time by the BIOS. */

    //#ifdef PC /* Configure for PC. Printer on COM2 */
    #if PC
        inregs.h.al = 0xFA; /* 11111010, Configuration COM2 for Printer */
        inregs.h.ah = 0;
        inregs.x.dx = 1;
        _int86( 0x14, &inregs, &outregs);

    //#elif TP /* Configure for TimePoint, Printer SMARTPORT(RS232) */
    #else

```



```

/* Setup I/O Coprocessor SMARTPORT for communication with the Intermec
Printer. 9600, Even, 7, XON, XOFF */
WriteToCom2("SBA5\r\n"); /* Baud rate 9600 */
WriteToCom2("SDA0\r\n"); /* 7 bits, even parity */
WriteToCom2("SSA0\r\n"); /* One stop bit */
WriteToCom2("SHA1\r\n"); /* XON, XOFF protocol */
WriteToCom2("SRA1\r\n"); /* Input Data sent to COM2 */
WriteToCom2("SPA1\r\n"); /* Passthrough mode */
WriteToCom2("<STX><DLE><ETX>"); /* Clear printer buffer */

/* Initialize LCD display */
inregs.x.dx = 00;
inregs.h.ah = 0x10;
_int86(0x17, &inregs, &outregs);
    Display( dispbu9 ); /* Display wait message */
#endif

#if !SKIPINIT
    /*
        These are the intial codes used to set up the printer. These should not
        be over written by any of the label files used in this code. */

    WriteToCom2("<STX>R<ETX>"); //
    WriteToCom2("<STX><SI>T1<ETX>"); //
    Select label stock with gaps between labels

    WriteToCom2("<STX><SI>R1<ETX>"); // Enable Label Retract Option
    WriteToCom2("<STX><SI>D108<ETX>"); // End of printer
    WriteToCom2("<STX><BS><ETX>"); // Do reset to enable new functions
do
{
    GetStatus( status );
} while( strcmp( status, "<DC2>" ) != 0 );
#endif

/* Initialize Scanner for COM1. 9600, Even, 1 Stop Bit, 7 */
inregs.h.al = 0xFA; /* 11111010, Configuration for Scanner */
inregs.h.ah = 0;
inregs.x.dx = 0;
_int86( 0x14, &inregs, &outregs);
}
/*****
/* This routine inputs setup sheet data. */
/*The prefixes for part, quantity, and pull no. must */

```

```

/* be changed if a different prefix is used. */
/*****
int InputSetupData( int commport )
{
    char scanbuf[20], tempbuf1[20], tempbuf2[20];
    int ii = 0, jj = 0, setupcount = 0, part = 0, qty = 0, pull = 0, len = 0;
    char partprefix[5], qtyprefix[5], pullprefix[5], retval = 0;

    ClearPort(1);
    strcpy(partprefix, "01");
    strcpy(qtyprefix, "02");
    strcpy(pullprefix, "03");

    /* While all the 3 quantities have not been read */

    while (setupcount < 3)
    {
        retval = InputScannerData(commport, scanbuf);
        /* If retval is non-zero then it is a function key, return with Fkey code
*/
        if (retval)
            return retval;
        ii = 0;
        jj = 0;
        while (scanbuf[ii] != '\0')
        {
            /* Strip off CR and LF */

            if (scanbuf[ii] != '\r' && scanbuf[ii] != '\n' && scanbuf[ii] != ' ')
            {
                tempbuf1[jj] = scanbuf[ii];
                jj++;
            }
            ii++;
        }
        tempbuf1[jj] = '\0';

        /* Two digit long prefix */

        for (ii = 0; ii < 2; ii++)
            tempbuf2[ii] = tempbuf1[ii];
        tempbuf2[2] = '\0';

        /* Check prefix */
        if(strcmp(tempbuf2, partprefix) == 0)

```

```

{
    /* Copy part data */
    strcpy(setupbuf[0], &tempbuf1[2]);
    if (strlen(setupbuf[0]) == 8)
    {
        if (part != PARTOK)
        {
            part = PARTOK;
            setupcount++;
        }
    }
}

else if (strcmp(tempbuf2, qtyprefix) == 0)
{ /* Check prefix */
    strcpy(setupbuf[1], &tempbuf1[2]); /* Copy qty data */
    if (qty != QTYOK)
    {
        qty = QTYOK;
        setupcount++;
    }
}

else if (strcmp(tempbuf2, pullprefix) == 0)
{ /* Check prefix */
    strcpy(setupbuf[2], &tempbuf1[2]); /* Copy pull no. data */
    if (strlen(setupbuf[2]) == 6)
    {
        if (pull != PULLOK)
        {
            pull = PULLOK;
            setupcount++;
        }
    }
}

} /* end while (setupcount < 4) */

return 0;
}

/*****
/* This routine has been hardcoded to accept only
/* 18 digit long Part No.
*****/

```

```

int InputPartData( int commport, char partbuf[20] )
{
    char scanbuf[25], tempbuf1[25];
    int ii, jj, count = 0, len, retcode;

    ClearPort(1);

    /* Setup program to read only a single quantity from the label */
    while (count < 1)
    {
        ii = 0;
        jj = 0;
        retcode = InputScannerData( commport, scanbuf );

        if( retcode == F2_KEY || retcode == F3_KEY )
            count++;
        else if ( (len = strlen(scanbuf)) == 20 )
        { /* Only accept 18(20 including CR and LF) digit long I 2/5 code */
            strcpy(tempbuf1, scanbuf, 18); /* Strip off suffix(CR/LF) */
            tempbuf1[18] = '\0';
            strcpy(partbuf[0], tempbuf1);
            count++;
            retcode = 0;
        }
    }
    /* end while (count < 1) */

    return( retcode );
}

/*
****
*/
int CheckForDuplicate( char partbuf[20] )
{
    int status = 0, ii;

    status = PART_OK;
    for (ii = 0; ii < 100; ii++)
    {
        if ( strcmp(partbuf[0], partbuf100[ii]) == 0 )
            status = PART_DUPLICATE;
    }
}

```

```

    return status;
}

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

int CheckForDuplicatePullSig(char *pullsig, char pullsigbuf100[][PULLSIG_LEN])
{
    int status = 0, ii;

    status = PULLSIG_OK;
    for (ii = 0; ii < 100; ii++)
    {
        if (strcmp(pullsig, pullsigbuf100[ii]) == 0)
            status = PULLSIG_DUPLICATE;
    }
    return status;
}

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

int CheckForInvalid( char partbuf[][20] )
{
    int status = 0;
    char tempbuf[13];

    status = PART_OK;
    strncpy(tempbuf, partbuf[0], 8); /* Adjusted for a part no. that is 8 long */
    tempbuf[8] = '\0';
    if (strcmp(setupbuf[0], tempbuf) != 0)
    {
        status = PART_INVALID;
    }
    return status;
}

/*****
/* Clear Setup sheet data.
*****/

void ClearSetupData( void )
{
    int ii, jj;

```

```

    for (ii = 0; ii < 3; ii++)    /* Clear part no., pull no., and qty */
        for (jj = 0; jj < 20; jj++)
            setupbuf[ii][jj] = '\0';

}
/*****
**/
/* ClearSequenceData:
*/
/*      This was added to support version 2.0 - where the three highest */
/*      sequence numbers are stored per device used. There is a limit */
/*      of four devices. Clearing in this case means resetting to all */
/*      zeroes.
*/
/*****
**/

void ClearSequenceData( void )
{
    int ii;

    for( ii=0; ii < 10; ii++)
        strcpy( highest_sequence[ ii ], "0000" );
}
/*****
/* Clear all data
*/
/* setupbuf[0] : 8 digit Part No. from setup sheet
*/
/* setupbuf[1] : Quantity
*/
/* setupbuf[2] : Pull Signal No.
*/
/* setupbuf[3] : Engineering Change
*/

/* setupbuf[4] : Department
*/
/*
*/
/* partbuf[0] : 18 digit Part No. from part label
*/
/*
*/
/* partbuf100[] : List of last 100 parts scanned
*/
/*
*/
/*****

void ClearData( char partbuf[][20],
                char partpullsig[], char partpullsigbuf100[][8],
                char partpartno[], char partqty[] )
{
    int ii, jj;

```

```

    for (ii = 0; ii < 3; ii++)
        for (jj = 0; jj < 20; jj++)
        {
            setupbuf[ii][jj] = '\0';
            partbuf[ii][jj] = '\0';
        }

strcpy(setupbuf[3], " ");
strcpy(setupbuf[4], " ");

for (ii = 0; ii < PULLSIG_LEN; ii++)
    partpullsig[ii] = '\0';
for (ii = 0; ii < PART_LEN; ii++)
    partpartno[ii] = '\0';
for (ii = 0; ii < PART_QTY_LEN; ii++)
    partqty[ii] = '\0';

for (ii = 0; ii < 100; ii++)
    for (jj = 0; jj < PART_LEN; jj++)
        partbuf100[ii][jj] = '\0';
for (ii = 0; ii < 100; ii++)
    for (jj = 0; jj < PULLSIG_LEN; jj++)
        partpullsigbuf100[ii][jj] = '\0';
}

/*****
/*  SetEngChg: */
*****/
void SetEngChg( void )
{
    char dispbuf[81] = "Scan Engineering Change                F2:Exit";
    char scanbuf[25], tempbuf1[25], engrchgprefix[4], tempbuf2[6];
    char missbuf[81] = "de_aiag3.dax not found                ";

    int ii = 0, jj = 0, count = 0, len = 0, engr = 0, duration = 0, result;
    int retval=0, exit = FALSE, skip=FALSE;
    long kk = 0;
    int commport, frequency = 0;
    FILE *stream;
    strcpy(engrchgprefix, "04");
    ClearPort(1);
    Display(dispbuf);

```

```

    /* read a single quantity */
while (count < 1)
{
    /* Wait until get a scan from the user.
       A 0 is returned when valid data is scanned,
       ignore Fkeys in case */

    skip=FALSE;
    retval = InputSetupData( commport = 1 );
    if(retval == F2_KEY) return; //

    if(!skip)
    {
        ii = 0;
        jj = 0;

        /* Strip off suffix */
        while (scanbuf[ii] != '\0')
        {
            if (scanbuf[ii] != '\r' && scanbuf[ii] != '\n')
            {
                tempbuf1[jj] = scanbuf[ii];
                jj++;
            }
            ii++;
        }
        tempbuf1[jj] = '\0';

        /* Get prefix: Two digit long prefix */
        for (ii = 0; ii < 2; ii++)
            tempbuf2[ii] = tempbuf1[ii];
        tempbuf2[2] = '\0';
        /* Check prefix */
        if(strcmp(tempbuf2, engrchgprefix) == 0)
        {
            strcpy(setupbuf[3], &tempbuf1[2]); /* Copy data */
            if (strlen(setupbuf[3]) == 2)
            {
                strncpy(&dispbuff[25], setupbuf[3], 2); /* Display two digit long
engr. change */

                if (engr != ENGROK)
                {
                    engr = ENGROK;
                    count++; /* Exit loop */
                }
            }
        }
    }
}

```



```

        }
    }

    } /* end if(!skip)*/

} /* end while (count < 1) */

/* This is a logic to check for an exit (PF-2) from the Engineering Screen.
*/
if(count >= 1)
{
    Display(disbuf);
    Beep(frequency = 1000, duration = 600);
    duration = 2000;
    Sleep((clock_t) duration);

    /* Substitute value in label data buffer */

    strncpy(&list2[LIST2_ENGROFFSET], setupbuf[3], 2);

    /* Copy Engr. change to disk file, so it is not lost if power is turned off
*/

    if((stream = fopen( "a:\\de_aiag3.dax", "r+b" )) != NULL)
    {

        result = fseek( stream, ENGROFFSET, SEEK_SET);
        if(result)
            perror("Fseek failed");
        else
        {
            fputs(setupbuf[3], stream);
            fclose(stream);
        }
    }
    else
        Display(missbuf);
//    printf("Was not able to open de_aiag3.dax\n");
}

}

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

```

```

void SetCurrentDate(char tmbuf[])
{
    int day, month, year;
    char cday[5], cmonth[5], cyear[8];
    union _REGS inregs, outregs;
    struct _SREGS segregs;

    stncpy(cmonth, tmbuf, 2);
    cmonth[2] = '\0';
    stncpy(cday, &tmbuf[3], 2);
    cday[2] = '\0';
    stncpy(cyear, &tmbuf[6], 2);
    cyear[2] = '\0';

    void SetCurrentTime(char tmbuf[])
    {
        int hrs, min, sec;
        char chrs[5], cmin[5], csec[5];
        union _REGS inregs, outregs;
        struct _SREGS segregs;

        stncpy(chrs, tmbuf, 2);
        chrs[2] = '\0';
        stncpy(cmin, &tmbuf[3], 2);
        cmin[2] = '\0';
        stncpy(csec, &tmbuf[6], 2);
        csec[2] = '\0';
    }
}

/*
*****
*/
void SetCurrentTime(char tmbuf[])
{
    _int8x(0x21, &inregs, &outregs, &segregs);
    inregs.h.ah = 0x2b; /* Service 2Bh */
    inregs.x.cx = year;
    inregs.h.dh = month;
    inregs.h.dl = day;
    _int8x(0x21, &inregs, &outregs, &segregs);
    month = atoi(cmonth);
    day = atoi(cday);
    year = atoi(cyear);
    year = year + 1900;
    stncpy(cmonth, tmbuf, 2);
    cmonth[2] = '\0';
    stncpy(cday, &tmbuf[3], 2);
    cday[2] = '\0';
    stncpy(cyear, &tmbuf[6], 2);
    cyear[2] = '\0';
}

```

```

hrs = atoi(chrs);
min = atoi(cmin);
sec = atoi(csec);

/* If user did not enter any data then do not change time */
if(hrs != 0)
{
    inregs.h.ah = 0x2d; /* Service 2Bh */
    inregs.h.ch = hrs;
    inregs.h.cl = min;
    inregs.h.dh = sec;
    inregs.h.di = 00;
    _int86x(0x21, &inregs, &outregs, &segregs);
}

}

/*****
*/
/*****
void SetDateTime()
{
    char disbuf1[81] = "Current Date:      Time:      F1:Date F3:Time
F5:Exit ";
    char disbuf2[81] = "Enter New Date:      (mm/dd/yy)
";
    char disbuf3[81] = "Enter New Time:      (hh:mm:ss, 24Hr)
";
    char tmpbuf[24];
    int ii = 0, fkey, len = 0;

/* Display MS-DOS-style date and time. */
    _strtime(tmpbuf);
    strcpy(&disbuf1[30], tmpbuf, 8);
    _strdate(tmpbuf);
    strcpy(&disbuf1[14], tmpbuf, 8);
    Display(disbuf1);
    while((fkey = GetKey()) != F5_KEY)
    {
        if (fkey == F1_KEY)
        {
            strcpy(tmpbuf, " ");
            strcpy(&disbuf1[14], tmpbuf, 8);
            strcpy(tmpbuf, " / / ");
        }
    }
}

```

```

    strncpy(&dispbuf2[16], tmpbuf, 8);
    Display(dispbuf2);
    SetCursor(0, 16);
    ii = 0;
    while (ii < 8)
    {
        /* skip / character */
        if (ii == 2 || ii == 5)
        {
            ii++;
            SetCursor(0, ii+16);
        }
        tmpbuf[ii] = getch();
        if (tmpbuf[ii] != '\r')
        {
            strncpy(&dispbuf2[16], tmpbuf, 8);
            Display(dispbuf2);
            ii++;
            SetCursor(0, ii+16);
        }
        else
        {
            tmpbuf[ii] = '\0';
            ii = 8; /* Exit loop */
        }
    } /* End while (ii < 8) */

    tmpbuf[ii] = '\0';
    SetCurrentDate(tmpbuf); /* Reset system date */
    _strdate( tmpbuf ); /* Read new system date and display */
    strncpy(&dispbuf1[14], tmpbuf, 8);
    Display(dispbuf1);
} /* End IF(fkey == F1_KEY) */
else if (fkey == F3_KEY)
{
    strcpy(tmpbuf, "      ");
    strncpy(&dispbuf1[30], tmpbuf, 8);
    strcpy(tmpbuf, " : : ");
    strncpy(&dispbuf3[16], tmpbuf, 8);
    Display(dispbuf3);
    SetCursor(0, 16);
    ii = 0;
    while (ii < 8)
    {

```

```

/* skip / character */
if (ii == 2 || ii == 5)
{
    ii++;
    SetCursor(0,ii+16);
}
tmpbuf[ii] = getch();
if (tmpbuf[ii] != '\r')
{
    strncpy(&dispbuf3[16], tmpbuf, 8);
    Display(dispbuf3);
    ii++;
    SetCursor(0,ii+16);
}
else
{
    tmpbuf[ii] = '\0';
    ii = 8; /* Exit loop */
}
} /* End while (ii < 8) */

tmpbuf[ii] = '\0';
SetCurrentTime(tmpbuf); /* Reset system time */
_strtime( tmpbuf); /* Read new system time and display */
strcpy(&dispbuf1[30], tmpbuf, 8);
Display(dispbuf1);
} /* END else if (fkey == F3_KEY) */

} /* END while((fkey = GetFkey()) != F5_KEY) */
}

/*****
/*
*****/
void SetDept( void )
{
    FILE *stream;
    char dispbuf1[81] = "Current Department:
F5: Exit";
    char dispbuf2[81] = "Enter New Department:
char missbuf1[81] = "Was not able to open de_aiag3.dax
";

    int fkey, ii = 0, len = 0, result;
    char ch, tmpbuf[5];
    F3: New Dept
    ";

```

```

if( (stream = fopen( "a:\\de_aiag3.dax", "r" )) != NULL )
{
    result = fseek( stream, DEPTOFFSET, SEEK_SET);
    if(result)
        perror("Fseek failed");
    else
    {
        fgets(setupbuf[4], 5, stream);
        fclose(stream);
    }
}
else
    Display(missbuf1);

strncpy(&dispbuf1[23], setupbuf[4], 4);
Display(dispbuf1);
while((fkey = GetFkey()) != F5_KEY)
{
    if (fkey == F3_KEY)
    {
        strcpy(tmpbuf, " ");
        strcpy(setupbuf[4], " ");
        strncpy(&dispbuf2[23], tmpbuf, 4);
        strncpy(&dispbuf1[23], tmpbuf, 4);
        Display(dispbuf2);
        SetCursor(0, 23);

        ii = 0;
        while (ii < 4)
        {
            tmpbuf[ii] = getch();
            if ((ch = tmpbuf[ii]) != '\r')
            {
                strncpy(&dispbuf2[23], tmpbuf, 4);
                Display(dispbuf2);
                ii++;
                SetCursor(0, ii+23); /*Cursor ahead of data */
            }
            else
            {
                tmpbuf[ii] = '\0';
                ii = 4; /* Exit loop */
            }
        }
    }
}

```

```

        tmpbuf[ii] = '\0';
        len = strlen(tmpbuf);
        strncpy(&dispbuf1[23], tmpbuf, len);
        strncpy(setupbuf[4], tmpbuf, len); /* Copy dept. to setup buffer. */
        Display(dispbuf1);
    }
} /* END while((fkey = GetFkey()) != F5_KEY) */

/* copy spaces to dept-text field */

strncpy(&list2[LIST2_DEPTOFFSET], " ", 4);
strncpy(&list2[LIST2_DEPTOFFSET], setupbuf[4], len);

/* Copy dept. to disk file, so it is not lost if power is turned off */

if((stream = fopen("a:\\de_aiag3.dax", "r+b")) != NULL)
{
    result = fseek( stream, DEPTOFFSET, SEEK_SET);
    if(result)
        perror("Fseek failed");
    else
    {
        fputs(setupbuf[4], stream);
        fclose(stream);
    }
}
else
    Display(missbuf1);
}

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

void ScanPartLabel( char partbuf[20],
                    char partpullsig[], char partpullsigbuf100[8],
                    char partpartno[], char partqty[], char parttotal[],
                    char partscanned[] )
{
    int count, retcode, clear = 0, commport, ii = 0, skip = FALSE, jj = 0;
    char serial[10], tmpbuf[20], tmpbuf1[25], tempseq[5];
    long kk = 0;
    int frequency = 0, duration = 0;

```

```

int device_num;

strcpy(partqty, setupbuf[1]);
strcpy(partpullsig, setupbuf[2]);
count = atoi( partqty );
strcpy(parttotal, partqty);
strcpy(partscanned, itoa(ii, tempbuf, 10));
DispMessage(SCANPARTLABEL, partpullsig, partpartno, partqty,
            parttotal, partscanned);

/* Prompt user to scan part labels. */
/* Input part label data */

while( count )
{
    retcode = InputPartData( commport = 1, partbuf );

    if( retcode == F2_KEY )
    {
        DispMessage(CONFIRM, partpullsig, partpartno, partqty,
                    parttotal, partscanned);

        /* Wait for a 'Y' or a 'N' */
        while ((retcode = GetYesNo()) == INVALID_KEY)
            ;
        if (retcode == IS_YES)
        {
            clear = TRUE;
            count = 0; /* Exit while loop */
        }
        if (retcode == IS_NO)
        {
            skip = TRUE; /* User pressed F2 key and does not want to continue -
Exit while loop */

            clear = FALSE;
        }
    }
    else if( retcode == F3_KEY )
    {
        SupervisorScreenRoutine( FALSE );
        skip = TRUE;
        clear = FALSE;
    }

    if( clear )

```



```

{
    ClearData( partbuf, partpullsig, partpullsigbuf100,
               partpartno, partqty);
    ClearSequenceData();
}

else if( skip )
{
    DispMessage(SCANPARTLABEL, partpullsig, partpartno, partqty,
               parttotal, partscanned);
    skip = FALSE;
}
else
{
    if( CheckForInvalid( partbuf ) == PART_INVALID )
    {
        strcpy(partpartno, partbuf[0]);
        DispMessage(INVALIDPART, partpullsig, partpartno, partqty,
                   parttotal, partscanned);
        for (jj = 0; jj < 3; jj++)
        {
            Beep(frequency = 1000, duration = 400);
            Sleep((clock_t) duration);
        }
        GetEnterKey(); /* Wait for ENTER key */
        strcpy(partpartno, "          ");
        DispMessage(SCANPARTLABEL, partpullsig, partpartno, partqty,
                   parttotal, partscanned);
    }
    else if( CheckForDuplicate( partbuf ) == PART_DUPLICATE )
    {
        strcpy(partpartno, partbuf[0]);
        DispMessage(DUPLICATEPART, partpullsig, partpartno, partqty,
                   parttotal, partscanned);
        for (jj = 0; jj < 3; jj++)
        {
            Beep(frequency = 1000, duration = 400);
            Sleep((clock_t) duration);
        }
        GetEnterKey(); /* Wait for ENTER key */
        strcpy(partpartno, "          ");
        DispMessage(SCANPARTLABEL, partpullsig, partpartno, partqty,
                   parttotal, partscanned);
    }
}

```

```

    }
    else
    { /* Valid part */
        ii++;
        count--;
        for (kk = 0; kk < 99; kk++)
            strcpy(partbuf100[kk], partbuf100[kk+1]); /* Adjust part no. buffer
*/
        strcpy(partbuf100[99], partbuf[0]);
        strcpy(partscanned, itoa(ii, tempbuf, 10));
        strcpy(partpartno, partbuf[0]);
        if (ii == 1)
        { /* First part no, copy serial no. */
            strcpy(tempbuf1, partbuf[0]);
            strncpy(serial, &tempbuf1[9], 9);
        }

        /* Call subroutine to check to see if serial number (last four of the partno) is */
        /* one of the highest for that device. If it is then save it and continue */

        strcpy(tempbuf1, partbuf[0]);
        memset( tempseq, '\0', sizeof( tempseq ) );
        memcpy( tempseq, &tempbuf1[14], 4 );

        // Take the sequence number and figureout which device needs updating.
        // This will be based on the first digit of the number.
        device_num = (int)tempseq[0] - (int)'0';
        // Make sure the device_num is between 0 and 9.

        if( device_num >= 0 && device_num <= 9 )
        {
            if( strcmp( tempseq, highest_sequence[ device_num ] ) > 0 )
                strcpy( highest_sequence[ device_num ],
tempseq );
        }
        // The first character of the sequence number was not a
        //digit between 0 and 9. Display an error message. //
        else
        {
            DispMessage(BADSEQUENCENUMBER, partpullsig, partpartno,
partqty,
parttotal, partscanned);
            for (jj = 0; jj < 3; jj++)
            {
                Beep(frequency = 1000, duration = 400);
            }
        }
    }
}

```

```

        Sleep((clock_t) duration);
    }
}

Beep(frequency = 1000, duration = 600 );
DispMessage(SCANPARTLABEL, partpullsig, partpartno,
partqty,
            parttotal, partscanned);
    } // end else if CheckForDuplicate partno
}
/* Display Container full message and print label */
if (!clear)
{
    /* Print Part data for last part in the batch */
//    DispMessage(SCANPARTLABEL, partpullsig, partpartno, partqty,
//                parttotal, partscanned);
//    for (kk = 0; kk < 64000; kk++);

    DispMessage(CONTAINERFULL, partpullsig, partpartno, partqty,
                parttotal, partscanned);
    /* Wait loop so that user can see the message */
    for (kk = 0; kk < 64000; kk++);

    PrintLabel(setupbuf[0], setupbuf[1], setupbuf[2], serial, setupbuf[4],
setupbuf[3]);
}
for (ii = 0; ii < 3; ii++)
    for (jj = 0; jj < 20; jj++)
        partbuf[ii][jj] = '\0'; /* Clear part data for next scanning operation. */
}

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

void SetUpDataEntry(char partbuf[][20],
                    char partpullsig[], char partpullsig100[][8],
                    char partpartno[], char partqty[], char parttotal[],
                    char partscanned[] )
{
    long ii = 0, jj=0, kk=0;
    int clear = 0, retcode = 0;

```

```

    char dispbuf10[81] = "All Setup Information will be lost   Continue Y/N
";
    int frequency, duration;
/*
Got a good Part Label Scan, Now is the time to make sure it is not a repeat.
*/
    strcpy(partpullsig, setupbuf[2]);

    if (CheckForDuplicatePullSig(partpullsig, partpullsig100)
        == PULLSIG_DUPLICATE)
    {
        DispMessage(DUPLICATEPULLSIG, partpullsig,
                    partpartno, partqty, parttotal,
partscanned);
        for (jj = 0; jj < 3; jj++)
        {
            Beep(frequency = 1000, duration = 400);
            Sleep((clock_t) duration);
        }
    }
    else
    {
        for (kk = 0; kk < 99; kk++)
            strcpy(partpullsig100[kk], partpullsig100[kk+1]); /* Adjust pullsig100.
buffer */
        strcpy(partpullsig100[99], partpullsig);

        DispMessage(SETUPDATAENTRY, partpullsig, partpartno, partqty,
                    parttotal, partscanned);

        /* Give user a little time to view data, and clear it if they want to */
        for (ii = 0; ii < 26000; ii++)
        {
            if (kbhit())
            {
                if ((retcode = GetF2()) == IS_F2)
                {
                    Display(dispbuf10);
                    /* Wait for valid user input */
                    while ((retcode = GetYesNo()) == INVALID_KEY);
                    if (retcode == IS_YES)
                    {
                        clear = TRUE;

```

```

        ii = 26000; /* Exit loop */
    }
    else if (retcode == IS_NO)
        ii = 26000; /* Exit loop */
    }
}

if (clear)
{
    ClearSetupData();
    ClearSequenceData();
}
else
    ScanPartLabel( partbuf, partpullsig, partpullsig100, partpartno, partqty,
                  parttotal, partscanned );
} // end if dup pullsig
}

/*****
*/
/* SupervisorScreenRoutine: */
/*
/* This routine is called by the "main" menu as in the scanLabelRoutine */
/* Its sole purpose is to display the version 2 options and allow their */
/* execution. */
*****/
void SupervisorScreenRoutine( int fromMain )
{
    //
    // Buffer names and contents are derived from the DispMessage routine.
    // Eventually all of these buffers can migrate into a ".h" file.
    //
    //char dispbuf99[81] = "Function Not Enabled          F2:Exit
";
    char dispbuf2[81] = "Part No.      Quantity      Pull No.
F2:Exit ";
    char dispbuf10[81] = "F1:Print Parts Scanned F2:Exit      F3:View High Seqs
F4:View Ship Sheet ";
    char dispbuf11[81] = "Do You Wish to Print the Last 100      Parts Scanned? Y/N
";
    char dispbuf12[81] = "PartNum:XXXXXXXXXXXXXXXXXXXXX Qty:XXXXXX
Pull Sig#:XXXXXXXXXX      F2:Exit ";
    char dispbuf13[81];

```

```

//
123456789012345678901234567890123456789012345678901234567890123456789
01234567890

```

```

//      1      2      3      4      5      6      7

```

```

int ret_val=0, retcode=0;
int exit_flag = FALSE;
int number_to_print;

```

```

while(!exit_flag)
{

```

```

    Display(disdbuf10);

```

```

    //

```

```

    // F2 or <Enter> will cause the user to go "up" one menu structure.

```

```

    //

```

```

    ret_val = (int) GetFkey();

```

```

    switch(ret_val)

```

```

    {

```

```

        case F1_KEY: // F1:Print Last 100 Parts Scanned

```

```

            Display(disdbuf11);

```

```

    while ((retcode = GetYesNo()) == INVALID_KEY);

```

```

    if (retcode == IS_YES)

```

```

    {

```

```

        // there is the potential to setup a dialogue to allow

```

```

        // user to choose how many to print.

```

```

        //

```

```

        number_to_print=100;

```

```

        Print100Parts( number_to_print );

```

```

    }

```

```

        break;

```

```

    case F2_KEY: // F2:Exit

```

```

        exit_flag=TRUE;

```

```

        break;

```

```

    case F3_KEY: // F3:View High Seqs

```

```

        memset( disdbuf13, '\0', sizeof( disdbuf13 ) - 1 );

```

```

        disdbuf13[ sizeof( disdbuf13 ) - 1 ] = '\0';

```

```

        memcpy(&disdbuf13[0], highest_sequence[0],4);

```

```

        memcpy(&disdbuf13[6], highest_sequence[1],4);

```

```

        memcpy(&disdbuf13[12], highest_sequence[2],4);

```

```

        memcpy(&disdbuf13[18], highest_sequence[3],4);

```

```

        memcpy(&disdbuf13[24], highest_sequence[4],4);

```

the

```

        memcpy(&dispbufl3[40], highest_sequence[5],4);
        memcpy(&dispbufl3[46], highest_sequence[6],4);
        memcpy(&dispbufl3[52], highest_sequence[7],4);
        memcpy(&dispbufl3[58], highest_sequence[8],4);
        memcpy(&dispbufl3[64], highest_sequence[9],4);

        Display(dispbufl3);
        while(GetFkey() != F2_KEY);
        break;

    case F4_KEY: // F4:View Ship Sheet - taken from
DispMessage Routine

        if( fromMain )
        {
            memcpy( &dispbufl2[9], lastsetupbuf[ 0 ], strlen( lastsetupbuf[ 0 ]));
            memcpy( &dispbufl2[27], lastsetupbuf[ 1 ], strlen( lastsetupbuf[ 1 ] ) );
            memcpy( &dispbufl2[49], lastsetupbuf[ 2 ], strlen( lastsetupbuf[ 2 ] ) );
        }
        else
        {
            memcpy( &dispbufl2[9], setupbuf[ 0 ], strlen( setupbuf[ 0 ] )
);
            memcpy( &dispbufl2[27], setupbuf[ 1 ], strlen( setupbuf[ 1 ] )
);
            memcpy( &dispbufl2[49], setupbuf[ 2 ], strlen( setupbuf[ 2 ] )
);
        }

        Display( dispbufl2 );

        while(GetFkey() != F2_KEY);
        break;

    default: // Bad Key Press or ??
        //Display(dispbufl9);
        while(GetFkey() != F2_KEY);
        break;
}

} //end while (!exit_flag)
}
/*****
/*   Main Routine:
/*   Main Driver routine for C standards.
*/

```

```

/*****/

void main( void )
{
    char partbuf[3][20];
    char partpullsigbuf100[100][8];
    char partpullsig[8], partpartno[20] = "          ";
    char partqty[4], parttotal[4], partscanned[4];
    char unrecovbuf[81] = "Unrecoverable Error - Exiting
";
    int commport = 0, count = 0, jj = 0, ii = 0;
    int clear = 0, retcode = 0;
    int frequency = 0, duration = 0, retval = 0, skip = FALSE;

/*****/
/* Configure scanner port (COM1) and printer port (COM2) */
/*****/
    ConfigSystem();

    if ( InitLabelData() == SYSTEMERROR )
    {
        Display( unrecovbuf );
        exit(1);
    }

    /* Init buffers */
    ClearData( partbuf, partpullsig, partpullsigbuf100,
               partpartno, partqty);
    memset( lastsetupbuf, '\0', sizeof( lastsetupbuf ) );
    ClearSequenceData();

    /* Main loop */
    while(1)
    {
        /* Prompt user to scan setup sheet.
        All parameters passed may not be needed for this screen. */

        DispMessage(SCANSHIPPING SHEET, partpullsig, partpartno, partqty,
                    parttotal, partscanned);

        /* Prompt user to scan setup data, input scanned data. */
        retval = (int) InputSetupData( commport = 1 );
        skip = FALSE;
        /* InputSetupData return the function key code or a 0 if scanned data */

```



```

switch(retval)
{

    case F3_KEY:
        skip = TRUE;
        SupervisorScreenRoutine( TRUE );
        break;

    case F2_KEY:
        skip = TRUE; /* no action on F2 key */
        break;

    case F4_KEY:
        SetDateTime();
        skip = TRUE;
        break;

    case F6_KEY:
        SetDept();
        skip = TRUE;
        break;

    case F8_KEY:
        SetEngChg();
        skip = TRUE;
        break;

    default:
        break;

}

/* Set up data received, do not skip */
if( !skip )
{
    /* Clear Setup data for next scanning operation */
    Beep(frequency = 1000, duration = 600 );
    SetUpDataEntry( partbuf,
                    partpullsig, partpullsigbuf100,
                    partpartno, partqty,
                    parttotal, partscanned );

    for( ii=0; ii<6; ii++ )
        strcpy( lastsetupbuf[ii], setupbuf[ii] );
    ClearSetupData();
    strcpy(partpartno, " ");
}

```

```
    }  
  } /* END while(1) */  
}
```

---

## VITA

Sethumadhava.T.R, son of Rama Rao and Padmavathi, was born on January 19, 1960, in Sakrepatna, India. After graduation from Siddaganga High School in Tumkur, he entered Mysore University in Mysore, India. He received a bachelor of engineering degree with a major in mechanical engineering from Mysore University in January, 1985. After graduating from Mysore University, he worked for Indian Design Center in Bangalore, India, as a product engineer until June of 1986. From June of 1986 until May 1989, Sethu managed a small scale industry, Kanakapura Engineering Industries, as a proprietor. In June of 1989, Sethu entered the graduate program in industrial engineering at Louisiana State University, Baton Rouge. He received his master of science degree in Industrial Engineering in December of 1991. During August, 1989 - December 1989, he worked as a research assistant at the Department of Mechanical Engineering at L.S.U., working on an Aluminum Corporation of America funded research project. During January 1990 - September 1991, he was working at Department of Kinesiology, L.S.U., as a research assistant working on a Caffeine Institute of England funded research project. After graduating from L.S.U. with a master of science degree in Industrial Engineering, he worked for The School of Veterinary Medicine, L.S.U., as an information scientist. While at this job, he entered the Engineering Science graduate program in 1991. He received his master's degree in Engineering Science in the Spring of 1994. He married Suparna in January 1991. They have one daughter Spandana. The author entered the doctoral program in 1991, in engineering science and expects to receive the degree of Doctor of Philosophy in December 1999.

# DOCTORAL EXAMINATION AND DISSERTATION REPORT

**Candidate:** Turuvekere R. Sethumadhava

**Major Field:** Engineering Science

**Title of Dissertation:** Integrative Approach to Online Quality Management:  
Process Control and Packout Verification in an  
Intelligent Manufacturing Workcell

**Approved:**

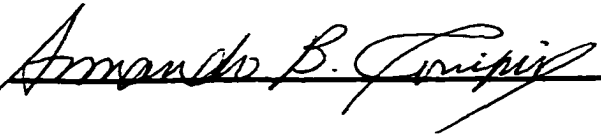


Major Professor and Chairman



Dean of the Graduate School

## EXAMINING COMMITTEE:



**Date of Examination:**

8 March 1999